CHAPTER 6
Ensemble Methods

6.1 MOTIVATION

In this chapter we will discuss two of the most popular ML ensemble methods.[1] In the references and footnotes you will find books and articles that introduce these techniques. As everywhere else in this book, the assumption is that you have already used these approaches. The goal of this chapter is to explain what makes them effective, and how to avoid common errors that lead to their misuse in finance.

6.2 THE THREE SOURCES OF ERRORS

ML models generally suffer from three errors:[2]

1. **Bias:** This error is caused by unrealistic assumptions. When bias is high, the ML algorithm has failed to recognize important relations between features and outcomes. In this situation, the algorithm is said to be "underfit."
2. **Variance:** This error is caused by sensitivity to small changes in the training set. When variance is high, the algorithm has overfit the training set, and that is why even minimal changes in the training set can produce wildly different predictions. Rather than modelling the general patterns in the training set, the algorithm has mistaken noise with signal.
3. **Noise:** This error is caused by the variance of the observed values, like unpredictable changes or measurement errors. This is the irreducible error, which cannot be explained by any model.

Consider a training set of observations $\{x_i\}_{i=1,\ldots,n}$ and real-valued outcomes $\{y_i\}_{i=1,\ldots,n}$. Suppose a function $f[x]$ exists, such that $y = f[x] + \epsilon$, where ϵ is white noise with $E[\epsilon_i] = 0$ and $E[\epsilon^2_i] = \sigma_\epsilon^2$. We would like to estimate the function $\hat{f}[x]$ that best fits $f[x]$, in the sense of making the estimation error $E[(y_i - \hat{f}[x_i])^2]$ minimal (the mean squared error cannot be zero, because of the noise represented by σ^2_ϵ). This mean-squared error can be decomposed as

$$\mathrm{E}\left[(y_i - \hat{f}[x_i])^2\right] = \left(\underbrace{\mathrm{E}[\hat{f}[x_i] - f[x_i]]}_{bias}\right)^2 + \underbrace{\mathrm{V}[\hat{f}[x_i]]}_{variance} + \underbrace{\sigma_\varepsilon^2}_{noise}$$

An ensemble method is a method that combines a set of weak learners, all based on the same learning algorithm, in order to create a (stronger) learner that performs better than any of the individual ones. Ensemble methods help reduce bias and/or variance.

6.3 BOOTSTRAP AGGREGATION

Bootstrap aggregation (bagging) is an effective way of reducing the variance in forecasts. It works as follows: First, generate *N* training datasets by random sampling *with replacement.* Second, fit *N* estimators, one on each training set. These estimators are fit independently from each other, hence the models can be fit in parallel. Third, the ensemble forecast is the *simple* average of the individual forecasts from the *N* models. In the case of categorical variables, the probability that an observation belongs to a class is given by the proportion of estimators that classify that observation as a member of that class (majority voting). When the base estimator can make forecasts with a prediction probability, the bagging classifier may derive a mean of the probabilities.

If you use sklearn's `BaggingClassifier` class to compute the out-of-bag accuracy, you should be aware of this bug: https://github.com/scikit-learn/scikit-learn/issues/8933. One workaround is to rename the labels in integer sequential order.

6.3.1 Variance Reduction

Bagging's main advantage is that it reduces forecasts' variance, hence helping address overfitting. The variance of the bagged prediction ($\varphi_i[c]$) is a function of the number of bagged estimators (N), the average variance of a single estimator's prediction ($\bar{\sigma}$), and the average correlation among their forecasts ($\bar{\rho}$):

$$\mathrm{V}\left[\frac{1}{N}\sum_{i=1}^{N}\varphi_i[c]\right] = \frac{1}{N^2}\sum_{i=1}^{N}\left(\sum_{j=1}^{N}\sigma_{i,j}\right) = \frac{1}{N^2}\sum_{i=1}^{N}\left(\sigma_i^2 + \sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}\right)$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\left(\bar{\sigma}^2 + \underbrace{\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho}}_{\substack{=(N-1)\bar{\sigma}^2\bar{\rho} \\ \text{for a fixed } i}}\right) = \frac{\bar{\sigma}^2 + (N-1)\bar{\sigma}^2\bar{\rho}}{N}$$

$$= \bar{\sigma}^2\left(\bar{\rho} + \frac{1-\bar{\rho}}{N}\right)$$

where $\sigma_{i,j}$ is the covariance of predictions by estimators i,j; $\sum_{i=1}^{N}\bar{\sigma}^2 = \sum_{i=1}^{N}\sigma_i^2 \Leftrightarrow \bar{\sigma}^2 = N^{-1}\sum_{i=1}^{N}\sigma_i^2$; and $\sum_{j\neq i}^{N}\bar{\sigma}^2\bar{\rho} = \sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j} \Leftrightarrow \bar{\rho} = (\bar{\sigma}^2 N(N-1))^{-1}\sum_{j\neq i}^{N}\sigma_i\sigma_j\rho_{i,j}$.

The equation above shows that bagging is only effective to the extent that $\bar{\rho} < 1$; as $\bar{\rho} \to 1 \Rightarrow \mathrm{V}[\frac{1}{N}\sum_{i=1}^{N}\varphi_i[c]] \to \bar{\sigma}^2$. One of the goals of sequential bootstrapping (Chapter 4) is to produce samples as independent as possible, thereby reducing $\bar{\rho}$, which should lower the variance of bagging classifiers. Figure 6.1 plots the standard deviation of the bagged prediction as a function of $N \in [5, 30]$, $\bar{\rho} \in [0, 1]$ and $\bar{\sigma} = 1$.

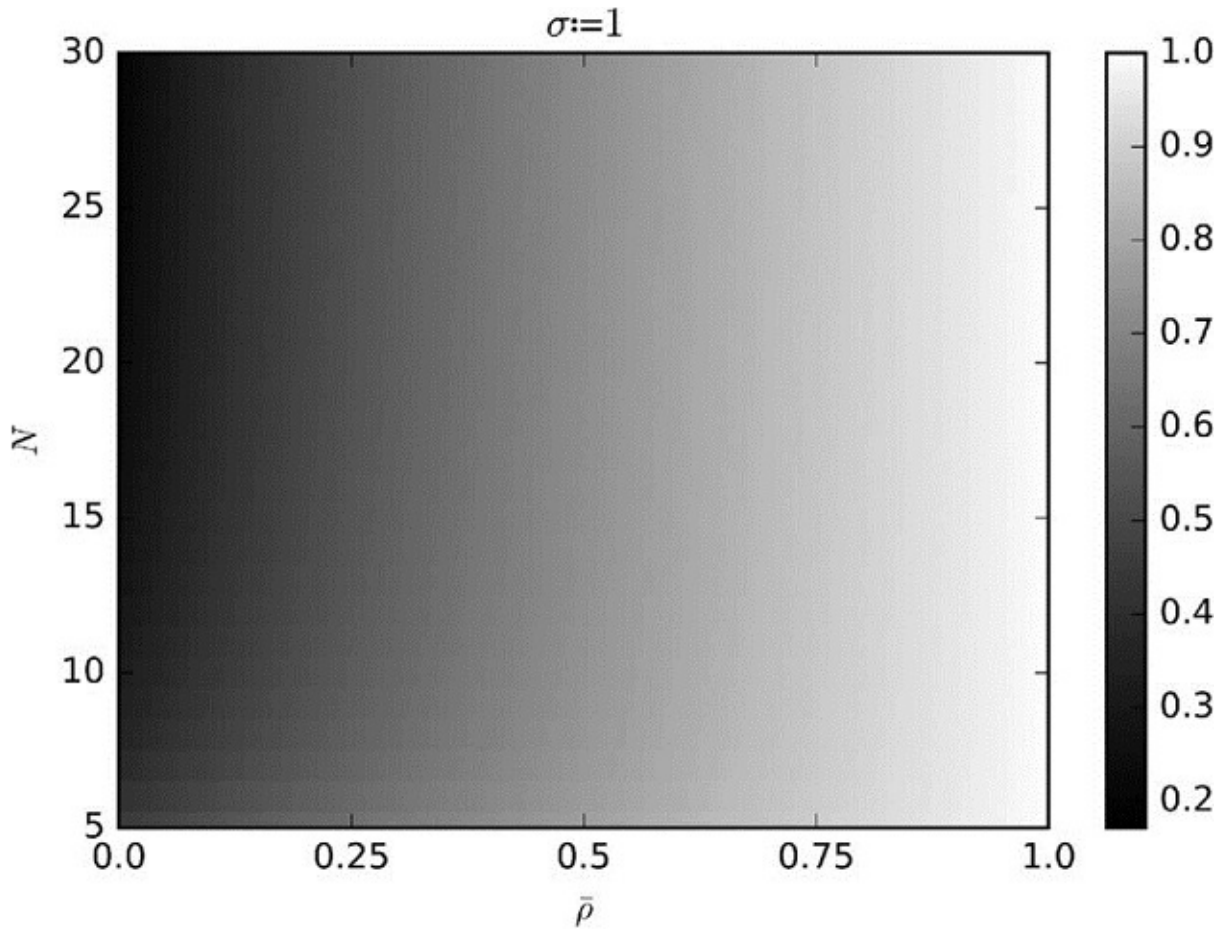

FIGURE 6.1 Standard deviation of the bagged prediction

6.3.2 Improved Accuracy

Consider a bagging classifier that makes a prediction on k classes by majority voting among N independent classifiers. We can label the predictions as {0, 1}, where 1 means a correct prediction. The accuracy of a classifier is the probability p of labeling a prediction as 1. On average we will get Np predictions labeled as 1, with variance $Np(1 - p)$. Majority voting makes the correct prediction when the most forecasted class is observed. For example, for $N = 10$ and $k = 3$, the bagging classifier made a correct prediction when class A was observed and the cast votes were $[A, B, C] = [4, 3, 3]$. However, the bagging classifier made an incorrect prediction when class A was observed and the cast votes were $[A, B, C] = [4, 1, 5]$. A sufficient condition is that the sum of these labels is $X > \frac{N}{2}$. A necessary (non-sufficient) condition is that $X > \frac{N}{k}$, which occurs with probability

$$\mathrm{P}\left[X > \frac{N}{k}\right] = 1 - \mathrm{P}\left[X \leq \frac{N}{k}\right] = 1 - \sum_{i=0}^{\lfloor N/k \rfloor} \binom{N}{i} p^i (1-p)^{N-i}$$

The implication is that for a sufficiently large N, say $N > p(p - 1/k)^{-2}$, then $p > \frac{1}{k} \Rightarrow \mathrm{P}[X > \frac{N}{k}] > p$, hence the bagging classifier's accuracy exceeds the average accuracy of the individual classifiers. Snippet 6.1 implements this calculation.

SNIPPET 6.1 ACCURACY OF THE BAGGING CLASSIFIER

```
from scipy.misc import comb
N,p,k=100,1./3,3.
p_=0
for i in xrange(0,int(N/k)+1):
    p_+=comb(N,i)*p**i*(1-p)**(N-i)
print p,1-p_
```

This is a strong argument in favor of bagging any classifier in general, when computational requirements permit it. However, unlike boosting, bagging cannot improve the accuracy of poor classifiers: If the individual learners are poor classifiers ($p \ll \frac{1}{k}$), majority voting will still perform poorly (although with lower variance). Figure 6.2 illustrates these facts. Because it is easier to achieve $\bar{\rho} \ll 1$ than $p > \frac{1}{k}$, bagging is more likely to be successful in reducing variance than in reducing bias.

For further analysis on this topic, the reader is directed to Condorcet's Jury Theorem. Although the theorem is derived for the purposes of majority voting in political science, the problem addressed by this theorem shares similarities with the above discussion.

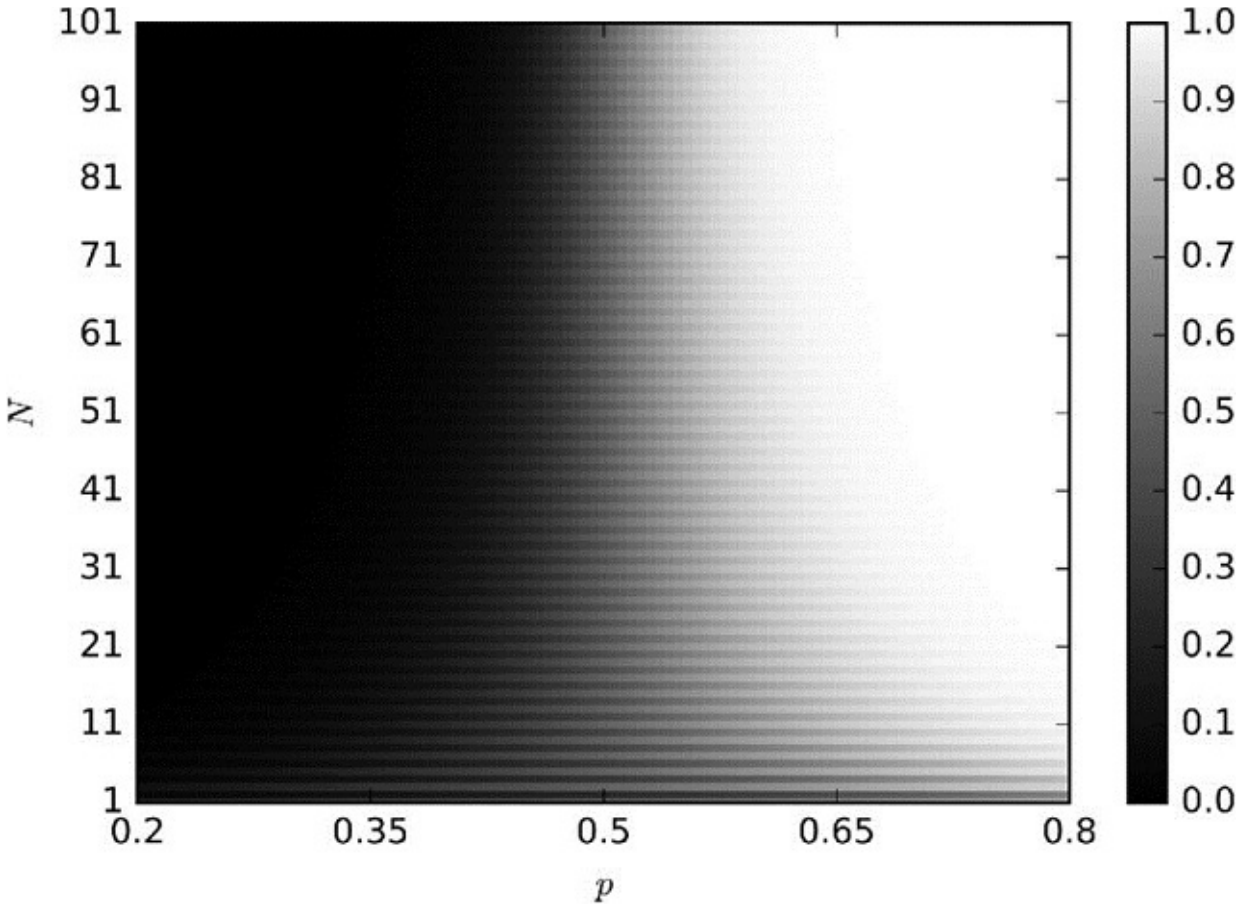

FIGURE 6.2 Accuracy of a bagging classifier as a function of the individual estimator's accuracy (P), the number of estimators (N), and $k = 2$

6.3.3 Observation Redundancy

In Chapter 4 we studied one reason why financial observations cannot be assumed to be IID. Redundant observations have two detrimental effects on bagging. First, the samples drawn with replacement are more likely to be virtually identical, even if they do not share the same observations. This makes $\bar{\rho} \approx 1$, and bagging will not reduce variance, regardless of N. For example, if each observation at t is labeled according to the return between t and $t + 100$, we should sample 1% of the observations per bagged estimator, but not more. Chapter 4, Section 4.5 recommended three alternative solutions, one of which consisted of setting `max_samples=out['tW'].mean()` in sklearn's implementation of the bagging classifier class. Another (better) solution was to apply the sequential bootstrap method.

The second detrimental effect from observation redundancy is that out-of-bag accuracy will be inflated. This happens because random sampling with

replacement places in the training set samples that are very similar to those out-of-bag. In such a case, a proper stratified k-fold cross-validation without shuffling before partitioning will show a much lower testing-set accuracy than the one estimated out-of-bag. For this reason, it is advisable to set `StratifiedKFold(n_splits=k, shuffle=False)` when using that sklearn class, cross-validate the bagging classifier, and ignore the out-of-bag accuracy results. A low number *k* is preferred to a high one, as excessive partitioning would again place in the testing set samples too similar to those used in the training set.

6.4 RANDOM FOREST

Decision trees are known to be prone to overfitting, which increases the variance of the forecasts.[3] In order to address this concern, the random forest (RF) method was designed to produce ensemble forecasts with lower variance.

RF shares some similarities with bagging, in the sense of training independently individual estimators over bootstrapped subsets of the data. The key difference with bagging is that random forests incorporate a second level of randomness: When optimizing each node split, only a random subsample (without replacement) of the attributes will be evaluated, with the purpose of further decorrelating the estimators.

Like bagging, RF reduces forecasts' variance without overfitting (remember, as long as $\bar{\rho} < 1$). A second advantage is that RF evaluates feature importance, which we will discuss in depth in Chapter 8. A third advantage is that RF provides out-of-bag accuracy estimates, however in financial applications they are likely to be inflated (as discussed in Section 6.3.3). But like bagging, RF will not necessarily exhibit lower bias than individual decision trees.

If a large number of samples are redundant (non-IID), overfitting will still take place: Sampling randomly with replacement will build a large number of essentially identical trees ($\bar{\rho} \approx 1$), where each decision tree is overfit (a flaw for which decision trees are notorious). Unlike bagging, RF always fixes the size of the bootstrapped samples to match the size of the training dataset. Let us review ways we can address this RF overfitting problem in sklearn. For illustration purposes, I will refer to sklearn's classes; however, these solutions can be applied to any implementation:

1. Set a parameter `max_features` to a lower value, as a way of forcing discrepancy between trees.

2. Early stopping: Set the regularization parameter `min_weight_fraction_leaf` to a sufficiently large value (e.g., 5%) such that out-of-bag accuracy converges to out-of-sample (k-fold) accuracy.
3. Use `BaggingClassifier` on `DecisionTreeClassifier` where `max_samples` is set to the average uniqueness (`avgU`) between samples.
   a. `clf=DecisionTreeClassifier(criterion='entropy',max_features='a`
   b. `bc=BaggingClassifier(base_estimator=clf,n_estimators=1000,max_`
4. Use `BaggingClassifier` on `RandomForestClassifier` where `max_samples` is set to the average uniqueness (`avgU`) between samples.
   a. `clf=RandomForestClassifier(n_estimators=1,criterion='entropy',`
   b. `bc=BaggingClassifier(base_estimator=clf,n_estimators=1000,max_`
5. Modify the RF class to replace standard bootstrapping with sequential bootstrapping.

In summary, Snippet 6.2 demonstrates three alternative ways of setting up an RF, using different classes.

SNIPPET 6.2 THREE WAYS OF SETTING UP AN RF

```
clf0=RandomForestClassifier(n_estimators=1000,class_weight='balanced_subsample',
    criterion='entropy')
clf1=DecisionTreeClassifier(criterion='entropy',max_features='auto',
    class_weight='balanced')
clf1=BaggingClassifier(base_estimator=clf1,n_estimators=1000,max_samples=avgU)
clf2=RandomForestClassifier(n_estimators=1,criterion='entropy',bootstrap=False,
    class_weight='balanced_subsample')
clf2=BaggingClassifier(base_estimator=clf2,n_estimators=1000,max_samples=avgU,
    max_features=1.)
```

When fitting decision trees, a rotation of the features space in a direction that aligns with the axes typically reduces the number of levels needed by the tree. For this reason, I suggest you fit RF on a PCA of the features, as that may speed up calculations and reduce some overfitting (more on this in Chapter 8). Also, as discussed in Chapter 4, Section 4.8, `class_weight='balanced_subsample'` will help you prevent the trees from misclassifying minority classes.

6.5 BOOSTING

Kearns and Valiant [1989] were among the first to ask whether one could combine weak estimators in order to achieve one with high accuracy. Shortly after, Schapire [1990] demonstrated that the answer to that question was affirmative, using the procedure we today call boosting. In general terms, it works as follows: First, generate one training set by random sampling with replacement, according to some sample weights (initialized with uniform weights). Second, fit one estimator using that training set. Third, if the single estimator achieves an accuracy greater than the acceptance threshold (e.g., 50% in a binary classifier, so that it performs better than chance), the estimator is kept, otherwise it is discarded. Fourth, give more weight to misclassified observations, and less weight to correctly classified observations. Fifth, repeat the previous steps until N estimators are produced. Sixth, the ensemble forecast is the *weighted* average of the individual forecasts from the N models, where the weights are determined by the accuracy of the individual estimators. There are many boosting algorithms, of which AdaBoost is one of the most popular (Geron [2017]). Figure 6.3 summarizes the decision flow of a standard AdaBoost implementation.

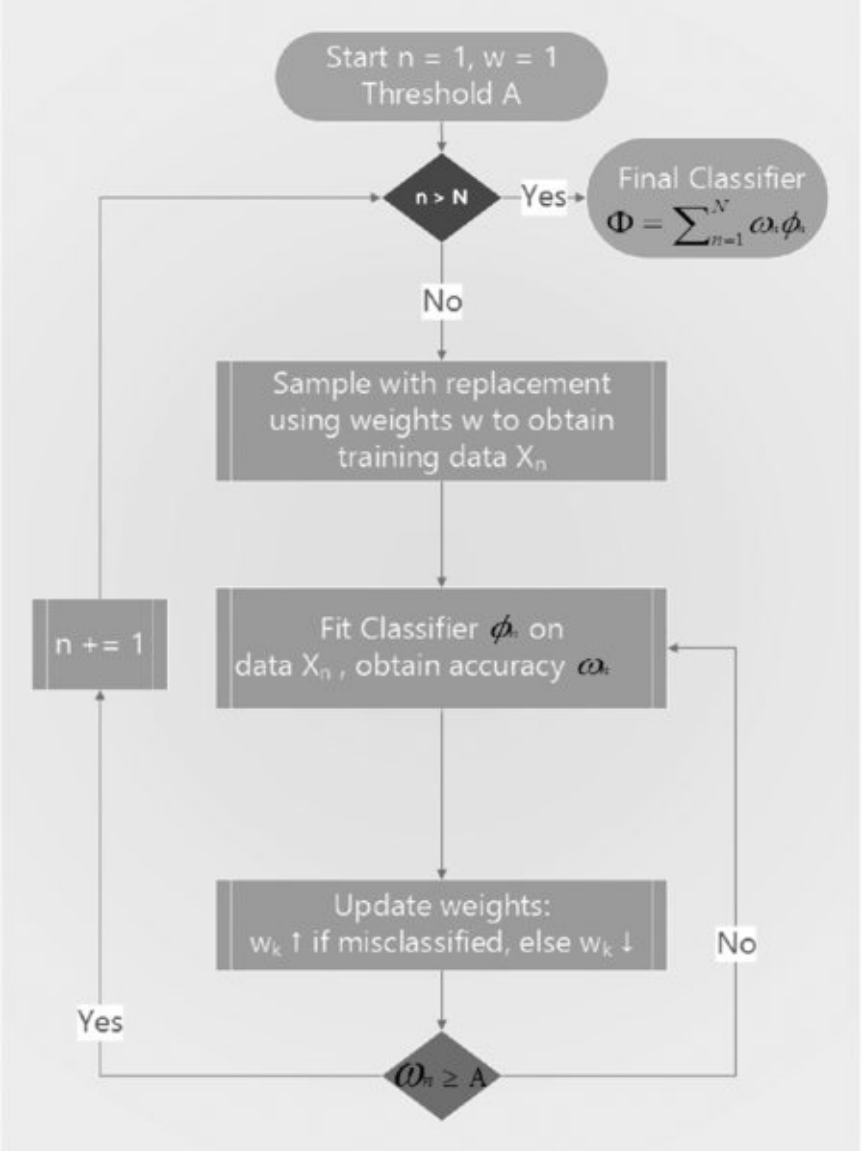

FIGURE 6.3 AdaBoost decision flow

6.6 BAGGING VS. BOOSTING IN FINANCE

From the above description, a few aspects make boosting quite different from bagging:[4]

- Individual classifiers are fit sequentially.
- Poor-performing classifiers are dismissed.
- Observations are weighted differently in each iteration.
- The ensemble forecast is a weighted average of the individual learners.

Boosting's main advantage is that it reduces both variance and bias in forecasts. However, correcting bias comes at the cost of greater risk of overfitting. It could be argued that in financial applications bagging is generally preferable to boosting. Bagging addresses overfitting, while boosting addresses underfitting. Overfitting is often a greater concern than underfitting, as it is not difficult to overfit an ML algorithm to financial data, because of the low signal-to-noise ratio. Furthermore, bagging can be parallelized, while generally boosting requires sequential running.

6.7 BAGGING FOR SCALABILITY

As you know, several popular ML algorithms do not scale well with the sample size. Support vector machines (SVMs) are a prime example. If you attempt to fit an SVM on a million observations, it may take a while until the algorithm converges. And even once it has converged, there is no guarantee that the solution is a global optimum, or that it is not overfit.

One practical approach is to build a bagging algorithm, where the base estimator belongs to a class that does not scale well with the sample size, like SVM. When defining that base estimator, we will impose a tight early stopping condition. For example, in sklearn's SVM implementation, you could set a low value for the `max_iter` parameter, say 1E5 iterations. The default value is `max_iter=-1`, which tells the estimator to continue performing iterations until errors fall below a tolerance level. Alternatively, you could raise the tolerance level through the parameter `tol`, which has a default value `tol=1E-3`. Either of these two parameters will force an early stop. You can stop other algorithms early with equivalent parameters, like the number of levels in an RF (`max_depth`), or the minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node (`min_weight_fraction_leaf`).

Given that bagging algorithms can be parallelized, we are transforming a large sequential task into many smaller ones that are run simultaneously. Of course, the early stopping will increase the variance of the outputs from the individual base estimators; however, that increase can be more than offset by the variance reduction associated with the bagging algorithm. You can control that reduction by adding more independent base estimators. Used in this way, bagging will allow you to achieve fast and robust estimates on extremely large datasets.

EXERCISES

6.1 Why is bagging based on random sampling with replacement? Would bagging still reduce a forecast's variance if sampling were without replacement?

6.2 Suppose that your training set is based on highly overlap labels (i.e., with low uniqueness, as defined in Chapter 4).

a. Does this make bagging prone to overfitting, or just ineffective? Why?

b. Is out-of-bag accuracy generally reliable in financial applications? Why?

6.3 Build an ensemble of estimators, where the base estimator is a decision tree.

a. How is this ensemble different from an RF?

b. Using sklearn, produce a bagging classifier that behaves like an RF. What parameters did you have to set up, and how?

6.4 Consider the relation between an RF, the number of trees it is composed of, and the number of features utilized:

a. Could you envision a relation between the minimum number of trees needed in an RF and the number of features utilized?

b. Could the number of trees be too small for the number of features used?

c. Could the number of trees be too high for the number of observations available?

6.5 How is out-of-bag accuracy different from stratified k-fold (with shuffling) cross-validation accuracy?

REFERENCES

Geron, A. (2017): *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st edition. O'Reilly Media.

Kearns, M. and L. Valiant (1989): "Cryptographic limitations on learning Boolean formulae and finite automata." In Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 433–444, New York. Association for Computing Machinery.

Schapire, R. (1990): "The strength of weak learnability." *Machine Learning*.

Kluwer Academic Publishers. Vol. 5 No. 2, pp. 197–227.

BIBLIOGRAPHY

Gareth, J., D. Witten, T. Hastie, and R. Tibshirani (2013): *An Introduction to Statistical Learning: With Applications in R*, 1st ed. Springer-Verlag.

Hackeling, G. (2014): *Mastering Machine Learning with Scikit-Learn*, 1st ed. Packt Publishing.

Hastie, T., R. Tibshirani and J. Friedman (2016): *The Elements of Statistical Learning*, 2nd ed. Springer-Verlag.

Hauck, T. (2014): *Scikit-Learn Cookbook*, 1st ed. Packt Publishing.

Raschka, S. (2015): *Python Machine Learning*, 1st ed. Packt Publishing.

NOTES

[1] For an introduction to ensemble methods, please visit: http://scikit-learn.org/stable/modules/ ensemble.html.

[2] I would not typically cite Wikipedia, however, on this subject the user may find some of the illustrations in this article useful: https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff.

[3] For an intuitive explanation of Random Forest, visit the following link: https://quantdare.com/random -forest-many-is-better-than-one/.

[4] For a visual explanation of the difference between bagging and boosting, visit: https://quantdare.com/ what-is-the-difference-between-bagging-and-boosting/.