International Journal of Computer Engineering and Data Science (ISSN: 2737-8543)
Volume 2– Issue 1, January - March 2022

Portfolio Selection Using Random Forest Algorithm

Daname KOLANI[1]
[1]Mohammed V University-FSJES-Agdal, Department of Management, Morocco
E-mail : danamekol@gmail.com

Article history

Received March 15, 2022
Revised March 29, 2022
Accepted March 30, 2022
Published April 01, 2022

ABSTRACT

Portfolio selection has long been a main topic in finance. What stocks should one invest in? How much should one allocate to each stock to maximize gain and minimize risk? These are the questions we aim to answer by demonstrating the possibility of obtaining abnormal returns above those offered by the benchmark by constructing a portfolio through a rule-based algorithm called Random Forest with Decision Tree as the base model. The use of Random Forest addresses the problem of over-fitting in the learning process and permits the prediction of a robust portfolio based on financial ratios. This approach has proven to outperform the S&P 500 Index and the Equal-Weight portfolio from 2013 to 2020.

Keywords: Portfolio Selection, Asset Allocation, Decision Tree, Random Forest, Value Investing.

I. INTRODUCTION

The financial market is a black box from which an immeasurable amount of incoming and outgoing information flows at high frequency and where a plethora of agents intervene. As a result, there are various paradigms and points of view in the literature and financial practice to identify the right stocks to invest in. Thus the predictability of stock return has been one of the most researched subjects in empirical asset pricing literature (see, [1], [2], [3], [4], [5]). Researchers have identified a variety of factors driving stock market prices. The use of financial ratios (e.g., Book-to-Market, Equity ratio, Dividends, etc.) as predictors of stock returns is a recurrent theme in research; most of the cases concern predicting stock returns by using a time-series/panel regression model.

Our focus here is not to complete the literature on the predictability of stock returns but rather to understand how a new approach could help in the problem of portfolio selection. Markowitz's Modern Portfolio Theory [6] is the seminal work of research which aims to solve this problem: How can an investor allocate the limited capital at his disposal in a finite set of stocks from many available alternatives? In his work, Markowitz approaches the problem with the well-known Mean-Variance (MV) model. According to the MV model, the investor has to make a trade-off between the maximization of the return (estimated as the expected value of stocks returns) and the minimization of the risk (represented by the variance of stocks returns). This theory has known its share of critics; the model does not take into account many real-world issues regarding investment such as the limitation of variance to represent risk, trading constraints, ignoring many fundamental factors ([5], [7]).etc. Since then, the theory has evolved with many approaches used for its robustification, as the authors in [8] summarized.

The progress of modern computing and the accessibility of data of any kind (structured or unstructured), especially in the finance field, has paved the way for Machine Learning algorithms to tackle portfolio selection problems leveraging the tremendous amount of available data. Machine Learning is a set of advanced techniques, models, or algorithms (see [9]) used in various areas to mine available information in data. Several pieces of literature (see [10], [11], [12], [13], [14]) have shown how one can pick stocks with Machine Learning (ML) procedures in the portfolio selection field. In most of the studies, ML algorithms were used for stock returns prediction and then the final portfolio was constructed by selecting the top performers. Models used included the Artificial Neuronal Network in [10] and [13] and Random Forest algorithm in [14].

We approach the same portfolio selection problem by using Random Forest as more than a mere prediction tool. Instead, we present the Decision Tree as a conceptual model of investor behavior. This type of investor is the one who relies on companies' fundamentals to select stocks to invest in (also known as value investor). Then, we improve the Decision Tree

model by including resampling methods through exploration of the Random Forest algorithm for portfolio selection.

II. Model Construction

To build this model, we first examine the motivations behind our approach. For this reason, we first discuss the decision tree algorithm to understand its implications before finally presenting the random forest algorithm as an efficient solution for the robustification of the portfolio selection and prediction model based on financial fundamentals.

A. Motivations

Binary decision trees, or simply Decision Trees, are one the most intuitive algorithms in data science. Depending on whether the response variable is qualitative or continuous, they are referred to as the classification tree or regression tree. As a Machine Learning algorithm, Decision Trees are better known through the famous CART (Classification And Regression Trees) algorithm developed and introduced in [15], an algorithm that is also used for both classification and regression problems. It is therefore a supervised learning model, which has the merit of being non-parametric, making no a priori on the distribution of response variables.

1) Decision Tree, CART Algorithm: The construction of a Decision Tree[1] is still defined by the general formulation of a supervised learning problem, such as:

$$Y = f(X) + \epsilon \tag{1}$$

We have at the base, a matrix X with n observations and p variables (continuous quantitative and/or qualitative), associated with a vector Y of length n (quantitative or qualitative variable to predict or explain). To make it possible, the observations n are partitioned into regions $R_1, R_2, \cdots, R_n$. These regions are referred to as terminal nodes or leaves of the tree. CART is a specific decision tree algorithm that grows a tree by splitting nodes into two child nodes repeatedly until a stopping criterion is satisfied (i.e., maximum depth is reached, or sample size in each child node is less than the minimum samples, or the sample size is equal to one). It follows a greedy approach known as recursive binary splitting. The binary split procedure which is carried out recursively until the stopping criterion is satisfied is as follows: Given a feature x_i for $i = 1, 2, \ldots p$, and a target variable y, a decision tree partitions the data D_t (with N_t samples) at note t into $D_t^{left}(\theta)$ and $D_t^{right}(\theta)$ subsets by considering each split candidate θ such that :

$$\begin{aligned} D_t^{left}(\theta) &= \{(x, y) \mid x\_j \leq S_t\} \\ D_t^{right}(\theta) &= D_t \setminus D_t^{left}(\theta) \end{aligned} \tag{2}$$

A split candidate $\theta = (j, S_t)$, consists of a feature j and threshold S_t. To select the best split candidate, there is a loss function H( ) to consider as it is frequent in supervised ML problems. In the context of decision trees, H is referred to as an impurity function, which measures the quality of the candidate split of note t. To determine the best $\theta^\star$ one must minimize the impurity so that:

$$\theta^\star = \min_\theta G(D_t, \theta)$$

where G, the weighted impurity function, is expressed as follows:

$$G(D_t, \theta) = \frac{N_t^{left}}{N_t} H\left(D_t^{left}(\theta)\right) + \frac{N_t^{right}}{N_t} H\left(D_t^{right}(\theta)\right) \tag{3}$$

2) Decision Tree as conceptual model of value investor behavior: The paradigm of value investing involves selecting stocks which seem to be priced for less than their fair value. Graham & Dodd, present a rigorous way of conducting fundamental analysis in their book *[16]*, which is considered foundational to the value investing paradigm. Fundamental analysis is a framework by which one can estimate a company's fair value and then compare it with its current price in the market. If the fair value is more significant than its price, the stock is considered underpriced and will ultimately grow to its fair value. A value investor seeks such stock to buy at the current price (which is underappreciated compared with its fair value), to sell in the future when it reaches the level of target fair value. That is the value investing view of one of the oldest rules in the stock market: "buy low and sell high."

Furthermore, in a second work *[17]* titled *Intelligent Investor*, Graham presents an intelligent investor in contradistinction to a speculator. The latter follows the market sentiment and buys or sells according to market changes. In contrast, the former realizes what Graham calls "Investment operation," which he defines as follows “An investment operation is one which, upon thorough analysis, promises safety of the principal and an adequate return. Operations not meeting these requirements are speculative.”

We can define an "Intelligent Investor" as an investor who only buys a stock which he has a rational knowledge about, through deep analysis in order to protect his investment (principal) and also to make a profit (return). He has a target value concerning any stock he invests in; he does not change his holdings because of market sentiment but instead sticks to his objectives. A value investor goes through fundamental analysis to select the best stocks, which requires diligence, expertise, and

[1] We focus on CART algorithm, but the approach developed here could be used with any other Decision Tree algorithm as long as rules can be extracted.

patience. Piotrosky *[7]* proved that the estimation of the fair value is not essential. Instead, factors that drive the value (consequently the market price) and help distinguish between winning and losing stock are directly taken from company financial statements (e.g., cash flow, net income, etc.) or indirectly after transformation as a ratio (e.g., return on asset, debt to equity, etc.). In practice, it is impossible to make use of all of those factors; rather, value investors follow these steps:

- They gather a list or set of determinant factors which may vary from one professional to another.
- Then from the most essential factor to the least, the value investor discriminates a given population of stocks to select the final portfolio by setting a threshold for each factor.

Piotrosky followed the same logic using only nine factors in his work to prove how efficient fundamentals can be when it comes to pick winning stocks. Moreover, as we can notice, this procedure is heuristic and very similar to the Decision Tree algorithm described above. Decision Tree algorithm offers the same possibilities to select the most important features (factors) recursively, to decide on the threshold value to split a parent node, and finally, when the tree is fully grown, to extract rules than can be generalized.

3) From Decision Tree to Portfolios: *Assuming CART as* the conceptual model of the investor *behavior, how does it* select a portfolio? *With the CART algorithm we arrive at* the terminal branches of the tree *which are* pure (that is to say homogeneous with respect to the selection variables and the variable to be predicted) sub-populations or sets *of* fundamentally similar *stocks*. *In our model we define these final nodes or sets as* a portfolio -- a collection or set of investments made on financial instruments or products. *Because d*e facto leaves or terminal nodes resulting from *the* CART decision tree algorithm are portfolios, *we have:*

$$Y_{(t)} = H\left(X_{(t-1)}\right) + \epsilon \qquad (4)$$

where:

- $Y_{(t)}$, represents the long-term expected returns at period t. Similar to the investor depending on the fundamentals to make these decisions, we assume no change of strategy prior to publication of the financial statements, all other things being equal. Long-term refers to the frequency of the financial data updates (i.e., quarterly, bi-annually, and annually).
- $X_{(t-1)}$, is the matrix of fundamentals or factors describing stocks under consideration from the previous period (t-1). There is a lag between the target variable and features matrix to allow for prediction and represent the time it takes for the market price to reach its target fundamental value. It also represents the minimum time frame needed by the investor to adjust his portfolio base on new information.

To find the best model, the equation (4) must be trained to imitate the expertise of value investor. Therefore, Decision Tree model is a good candidate for H as it offers the characteristics that portray value investor reasoning:

- **Selection**: there is an implicit variable or features selection, because not all fundamental indicators or ratios are relevant. Decision Tree is able to select relevant variables by measuring their importance.
- **Prediction**: Being a supervised ML method, Decision Trees allow predictions to be made after learning from a training sample, that is acquiring experience. It is moreover this functionality which allows the generation of portfolio by anticipation.
- **Interpretation**: Binary Decision Trees are more easily understood because it is not only possible to deduce decision rules but also a readable directional graph.

Nevertheless, Decision Tree algorithm has its share of drawbacks; the most critical is over-learning or overfitting. The goal of the model, intended to be predictive, is to allow a good generalization of the patterns and rules learned from training data to another sample without significantly reducing its performance.

4) Problem of Decision Trees Model: Given its predictive nature, it is important to highlight the generalization risk components of a ML model in order to control the causal factors. Over-fitting is a characteristic of models which describe and make good predictions on training data but which excessively extrapolate or poorly predict on data not included in the training process. Viewed from this angle, all models are not equal; some are more subject to overfitting compared to others, even more so if these models are of greater complexity. To better understand this phenomenon of overfitting and its causes, *[9]* examines a case of regression hypothesis function, considering (1) where $\varepsilon \sim N(0,1)$, the expected error expressed in the case of regression model $\hat{f}(X)$ at point $X = x_0$ as follows:

$$\begin{aligned} Err(x_0) &= E\left[\left(Y - \hat{f}(X)\right)^2 \middle| X = x_0\right] \\ &= \sigma_\epsilon^2 + \left[E\hat{f}(x_0) - f(x_0)\right]^2 \\ &\quad + E\left[\hat{f}(x_0) - E\hat{f}(x_0)\right]^2 \\ &= \sigma_\epsilon^2 + Bias^2\left(\hat{f}(x_0)\right) + Var\left(\hat{f}(x_0)\right) \\ &= Bias^2 + Variance + Irreducible\ Error \end{aligned} \qquad (5)$$

In the expression ()**Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.Error! Reference source not found.**, the first term Irreducible Error represents the variance of the data to be learned around its mean $f(x_0)$. This is inevitable regardless of the quality of the model $f(x_0)$ unless σ_ϵ^2 is really zero. Thus, we focus on the other two terms: bias and variance.

- The bias is the error coming from the hypothesis function retained $\hat{f}$ with respect to f. The more $\hat{f}$ diverges from f the greater the bias. In general, a low bias model suggests that few assumptions are made about the function f. As an example, we have the decision trees, the K-Nearest Neighbors' algorithm and the machine vector support algorithm. In contrast a high biased model suggests that more assumptions are required to learn the data. In this category, we have models that require a lot of hypotheses such as Linear Regression, Logistic Regression, etc.
- The variance is the error related to the sensitivity of the model if we vary the training sample. $\hat{f}$ is learned and estimated on a different sample than a sample on which it will be tested, validated, or used to make predictions. The variance measures how much the training model $\hat{f}$ deviates from its mean. Thus, this results in variances, but ideally the model would not change much from one data sample to another, meaning the model is efficient. When a model is significantly influenced by the training sample, there is a greater chance that the variance is high. However, if the model including its parameters are not weakly influenced by the training sample, the variance is low. In general, very complex machine learning algorithms have a greater variance (e.g., Decision Tree, K-Nearest Neighbors, etc.). Conversely, algorithms with low variance include linear regression, logistic regression, etc.

The goal of any supervised ML algorithm is to achieve low bias and low variance, and in return, the algorithm should achieve good predictive performance. In general, linear ML algorithms often have high bias but low variance, and nonlinear ML algorithms often have low bias but high variance. The tuning of ML algorithms is often a bias and variance tradeoff to find a balanced model. Knowing this, we sought to reduce the variance component to increase the bias in our approach.

B. Ensemble Learning methods

In this section, we will discuss Ensemble Learning (EL) methods which can be used to achieve this objective of variance reduction and thus make the model robust. EL methods are a technique of combining prediction of multiple learners (learning models) to achieve a more robust optimal predictor that takes advantage of the collaboration of these diverse learners. In EL methods, our approach focuses on resampling techniques.

1) Bagging: The Bagging technique (**B**ootstrap **Agg**regat**ing**), is an EL meta-algorithm proposed by *[18]*, one of the most popular and simple techniques. This technique consists of aggregating the individual predictions obtained with models (called basic model or weak learner) trained on training samples randomly drawn from the population. Bootstrap is a resampling technique used to reduce the variance / sensitivity of statistical estimators, developed by *[19]* and which was the subject of the work of *[20]* to robustify Markowitz's Mean-Variance model *[6]*. We consider a training sample D of size N. We generate B training samples by performing n uniform random draws with replacement. For each sample D_i , we will learn and build B model $\hat{f}_i$ to try to approximate f. Particularly in the case of a regression, Bagging consists of aggregating the predictions of the B learners such that:

$$\hat{f}_{bagging}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}_i(x) \tag{6}$$

If we consider decision trees as a basic model, Bagging consists of the construction of B decision trees as follows:

- Draw randomly and with replacement B samples on the pair (X, Y), denoted by D
- Train a Decision Tree on each D subset of (X, Y)
- Finally, if Y is categorical, use the majority class as prediction class. If Y is numerical, use the average of the predictions of each tree.

In comparison to the expression (), we can estimate the error made by the aggregate predictor $\hat{f}_{bagging}$ at a point x_0 as follows:

$$\begin{aligned} Err(x_0) = \ & \sigma_\epsilon^2 + Bias^2\left(\hat{f}(x_0)_{bagging}\right) \\ & + Var\left(\hat{f}(x_0)_{bagging}\right) \\ = \ & Irreducible\ Error + Bias^2 + \frac{1}{B} Variance \end{aligned} \tag{7}$$

Hence, the larger B, the smaller the variance will be in the composition of the error, thus reducing the overall error. The Bagging technique was precisely designed to reduce the variance of the model or of the basic learner (especially decision trees). Bagging is an algorithm which correctly mitigates overfitting especially by using complex models such as decision trees.

2) Random Forest(RF): The Bootstrap is particularly effective if B samples obtained are relatively uncorrelated with each other. In other words if there is only little disturbance in the resampled data, we will not have learners or models diversified enough for the variance reduction sought. As an example, if we consider X_1 , X_2, $X_3, \cdots, X_q$ of the same law, such as the variance $Var(X_l) = \sigma^2$ and the correlation $Corr(X_l, X_l') = \rho$, $\forall l \neq l'$, we can say of the following variance:

$$Var\left(\frac{1}{q}\sum_{l=1}^{q} X_q\right) = \rho\sigma^2 + \frac{1-\rho}{q}\sigma^2 \tag{8}$$

The resulting variance becomes smaller when q approaches infinity and approaches zero if the correlation ρ is smaller. In other words, the reduction in variance sought in the Bootstrap is limited by the existence of a strong correlation between the variables.

This observation, noticed by Breiman is the reason he proposes another technique called the Random Forest [21]

whose goal is to optimize the Bagging simply by adding a second dimension of resampling to decorrelate the learners or the collection of models trained on the different samples. What does it mean to add a second dimension? As seen previously, we already have a resampling in Bagging, the bootstrap carried out on the population. It is a random draw on the observations or individuals of the population. The second dimension would be to introduce random drawing on the variables used in the learning of the different samples. The Random Forest would therefore be a combination of Bagging and features sampling (drawing on the variables)

Thus, the advantage of Random Forest lies in the creation of sufficiently diversified decision trees (from this plurality and the diversity of trees, this technique takes its name Random tree forest or more commonly Random Forest) in order to reduce the correlation mentioned in the expression (8).

In summary, we can distinguish two main families of EL methods Bagging and Boosting. The two types of procedures or meta-algorithms use the same base model Decision Tree. In the first approach, the base model is fitted to samples of equiponderated observations drawn with replacement. However, in the second approach the observations are only equal-weighted at the initialization of the algorithm later the weights are gradually readjusted. Bagging is parallel learning while Boosting proceeds sequentially with each new model taking advantage of the previous one. Concerning the prediction, they also use the mean/majority in the case of regression/classification. This average or majority is equally weighted in the case of Bagging while in Boosting, the best performing models are those which are the more weighted.

While Bagging participates in the reduction of variance to resolve over-learning, Boosting reduces the bias which leads to a risk of over-learning. In the end, the set models are those that we need not only to overcome the weaknesses of the basic model (the weak learner) – here the decision tree model, i.e. to improve the performance of generalization – but also bring a consensual dimension to the model that will be defined in the following section.

C. Using Random Forest for Portfolio Construction

This section describes the approach of the model developed in this research. The conclusion lies in a question of understanding how everything, that is to say, Machine Learning tools already available, answers the question of the selection of stocks to forge portfolios that satisfy the a priori previously defined.

1) Learning phase or induction phase: To proceed in the formulation of the model, we will define some terms:

- y_{it}, a performance metric calculated on the market prices of the stock i over the period t such that Y_t is the performance vector over the period t of a data sample.
- The vector x_{it} of p variables describing the title i and consequently the matrix X_t of dimension $n \times p$, to describe the n stocks. These variables are essentially financial fundamentals. Similarly, we have the matrix $X_{(t-1)}, X_{(t-2)}, \cdots, X_0$ variables for the previous periods $(t-1), (t-2), \cdots, 0$.
- h_m, a decision tree model obtained or estimated at iteration m over the M iterations of a set algorithm. It is a question of learning a set algorithm with as basic or weak model a decision tree on the training data $X_{(t-1)}$ such as:

$$Y_{(t)} = H\left(X_{(t-1)}\right) + \epsilon \tag{9}$$

$$\hat{Y}_{(t)} = H\left(X_{(t-1)}\right) = \sum_{m=0}^{M} \alpha_m h_m(x) \tag{10}$$

where H, is a meta-algorithm or a set algorithm of M decision tree estimators. From each tree h_m we can extract decision rules which are in fact an abstraction of the analysis of a single investor; the idea being to induce generalizable decision rules and a significant relation between a set of indicators (or features in ML jargon) chosen by a single investor in period $(t-1)$and the performance over the period t of stock i.

2) Generation of portfolios or predictive deduction phase: A portfolio is a set of stocks to which are associated proportions that represent their individual shares in an investment of a monetary unit.

A portfolio can therefore be represented as a set of proportions $w_1, w_2, w_3, \cdots, w_i, \cdots, w_n$ given a universe of n stocks, which respect the budget constraint:

$$\sum_{i=1}^{n} w_i = 1 \tag{11}$$

Given h_m, a decision tree which is an estimator at iteration m of the ensemble model H, each sub-population or partition at terminal branches or leaves are in fact homogeneous or pure sets of individuals; in this case individuals are stocks. Thus, to make prediction, the average of the performance measure y_{it} of each stock in the leaf (terminal sub-population or partitions) is used to determine the future performance of a given features vector $x_{.t}$ of any stock, which ends in the corresponding leaf according to the decision rule extracted from h_m.

Thus, we can deduce $L_1, L_2, \cdots, L_k, \cdots, L_K$, the subset or partition leaves of the total population (Y_t, X_t) which are at the terminal nodes, and consequently deduce K portfolio such that for each portfolio P_k we have:

$$\sum_{i=1}^{n} w_i^k = \sum_{k=1}^{K} \frac{1}{|L_k|} = 1 \tag{12}$$

The portfolio k is said to be equally weighted because the stocks in L_k contribute equally to the prediction. T_m, is the set of leaves $L_1, L_2, \cdots, L_k, \cdots, L_K$ in an equivalent way $P_1, P_2, \cdots, P_k, \cdots, P_K$ obtained from the prediction $h_m(X_t)$.

3) Formulation of a selectivity criterion or scoring phase: The determining point of this model is to find out how to select the best leaf out of T_m. We define L_k, any leaf of

T_m *which* have K leaves or portfolios, each represented by the pair (Y_t^k, X_t^k) sub-population of size $S = |L_k|$ of the set of the total population (Y_t, X_t).

To do this, we will measure the selectivity of a leaf with a utility function $U(x)$ that the investor will define. This utility function could be defined as the sum or the mean or a quartile, etc. As a result, the score for each given leaf or portfolio will make it possible to choose portfolio $L_m^\star / P_m^\star$ as being the vote or the selection of T_m.

$$L_m^\star / P_m^\star = \arg\max_{L/P} U(T_m) \tag{13}$$

For example, with the maximum of the average return $\bar{r}$, we can express the function as follows:

$$U(L_k) = \frac{\sum_{i=1}^{S} (\bar{r}_{it}^k)}{|L_k|} \tag{14}$$

where S is the size of the subpopulation L_k. The score function may or may not depend, directly or indirectly, on transformation of the variables used to induce the decision rules.

4) Portfolio design or aggregation phase: The final portfolio is built with the collection or set $T_m^\star$ of leaves or portfolios $L_1^\star / P_1^\star, L_2^\star / P_2^\star, \cdots, L_k^\star / P_k^\star, \cdots, L_K^\star / P_K^\star$, having maximum utility for M iterations. In other words, each estimator votes on a leaf or portfolio that is most useful. The final portfolio $P^\star$ is obtained by aggregating the M leaves or portfolios $L_1^\star / P_1^\star, L_2^\star / P_2^\star, \cdots, L_k^\star / P_k^\star, \cdots, L_K^\star / P_K^\star$ proportional to their factor $\alpha_1, \alpha_2, \cdots, \alpha_m, \cdots, \alpha_M$ in the global learning model overall. This aggregation process propagates down to the level of individual stock. That is, the weight or the proportion w_i of the stock or stock i in the sought portfolio $P^\star$ is determined as follows:

$$w_i = \sum_{m=1}^{M} \frac{\alpha_m \times \theta_m^i}{\alpha_m \cdot S_m} \tag{15}$$

Where $\theta_m^i = 1$ if the stock $i \in L_m^\star / P_m^\star$ otherwise $\theta_m^i = 0$, and S_m the size of the sub-population of the leaf or portfolio $L_m^\star / P_m^\star$.

To sum up, this approach differs from other previous work in that the portfolio selection is made on the basis of in two steps rather than on the selection of n best predictions of the Random Forest algorithm presented in [14]. First, each decision tree votes one portfolio at a time. This vote is made on the basis of a utility function that may or may not be the average of y. It is predicted from the application of the rules which result from the learning phase. Secondly, established on the logic behind Random Forest, the portfolios voted by m trees are aggregated to form the final portfolio. In other words, using the conceptual model of value investor behavior, the final portfolio is the aggregation of multiple portfolios proposed by a group of experienced investors. Each investor made his selection on the basis of fundamental factors that are important to him, contributing his own view of the composition of the best portfolio.

Finally, even if the model already determines by default the weights of each stock in the portfolio, our approach can also be used in combination with classic portfolio optimization models to determine the weights.

III. Empirical Implementation and Analysis

As proposed, the detailed approach is essentially based on fundamental indicators. To do this, the algorithm needs the data for both training/learning and prediction purposes.

A. Data description and methodology

Data used in this study was collected annually from the USA Market specifically concerning S&P 500 listed companies from 2012 to 2020 (as described in table I). It is composed of essentially fundamental indicators and annual financial statement components on one side, and close historical prices collected from YahooFinance and MorningStar websites on the other.

TABLE I. Structure of data collected from 2013 to 2019

Year	Number of stocks	Number of features/variables
2011-12-31	460	59
2012-12-31	466	59
2013-12-31	476	59
2014-12-31	484	59
2015-12-31	489	59
2016-12-31	492	59
2017-12-31	492	59
2018-12-31	494	59
2019-12-31	494	59

As the financial data used here has an annual frequency, we estimated the target variable y as the average of the returns over a one-year interval. To determine the portfolio to select at the beginning of the year $(t+1)$, we used the hold-out as a validation method, especially the sliding window approach described as follow:

- The training data: the average of the returns for the year (t) of each security is determined to form the target variable y_t. The Random Forest algorithm is trained with the variables or fundamental factors or financial ratios extracted at the end of the period $(t-1)$, which are represented by $X_{(t-1)}$.
- The test data: $X_{(t)}$, a set of the fundamental factors or financial ratios published at the end of the period (t).

To test the strategy, that is **A**ggregating **E**qual-**W**eight **P**redicted **P**ortfolio (**AE-WPP**), three Random Forest algorithms were trained for different numbers of estimators respectively for 30, 300 and 3000 Regression Trees estimators. To make implementation possible, Python Tools Scikit-Learn package was used to carry on the procedure as it provides all necessary features. By default, the resampling numbers of variables equal $\sqrt{p}$, where p is the total number of variables. The strategy will be compared to S&P 500 Index and **E**qual-**W**eight **P**ortfolio (**E-WP**) of all the S&P listed companies.

B. Results

As seen in Table (2), each year in the period from 2013 to 2020, the procedure has proven to outperform not only the benchmark S&P 500 Index but also the naive strategy of Equal-Weight Portfolio (E-WP) for all the companies. On average AE-WPP generated at least two times the mean return of both S&P 500 Index and E-WP. However, as consequence it has higher mean volatility around 25% versus 15% for both S&P 500 Index and E-WP. However globally, it has better risk remuneration considering the average Sharpe Ratio of around 1.47 for AE-WPP against 1.44 for E-WP and 1.14 for S&P 500. The table below gives father details on the comparison of the strategies in each year.

TABLE II. BACKTESTING PREDICTED AGGREGATING EQUAL-WEIGHT PORTFOLIO (PAE-WP) FROM 2013 TO 2020

Designation	Year	Numbers of Estimators	Total Return	Sharpe Ratio	Mean Return	Volatility
AE-WPP	2013	30	0.5554	2.6242	0.4592	0.175
		300	0.4964	2.545	0.4185	0.1644
		3000	0.4786	2.4631	0.4065	0.1651
E-WP			0.3727	2.7352	0.3253	0.1189
SP 500			0.2639	2.2287	0.2411	0.1082
AE-WPP	2014	30	0.4498	1.5149	0.4097	0.2704
		300	0.4103	1.4614	0.379	0.2593
		3000	0.4159	1.4659	0.3836	0.2616
E-WP			0.2002	1.5957	0.1904	0.1193
SP 500			0.1239	1.0894	0.1237	0.1135
AE-WPP	2015	30	0.0763	0.474	0.093	0.1963
		300	0.116	0.6567	0.1296	0.1974
		3000	0.1028	0.6018	0.1172	0.1947
E-WP			0.04	0.3352	0.0509	0.1519
SP 500			-0.0069	0.0325	0.005	0.1552
AE-WPP	2016	30	0.1625	0.8812	0.1699	0.1928
		300	0.15	0.809	0.16	0.1978
		3000	0.1538	0.8202	0.1636	0.1995
E-WP			0.1854	1.2589	0.1811	0.1439
SP 500			0.1124	0.886	0.1154	0.1303
AE-WPP	2017	30	0.2447	1.7822	0.229	0.1285
		300	0.2004	1.6032	0.1913	0.1193
		3000	0.2174	1.7007	0.2057	0.1209
E-WP			0.2418	2.9979	0.2211	0.0738
SP 500			0.1842	2.5949	0.1726	0.0665
AE-WPP	2018	30	0.036	0.2668	0.0718	0.2692
		300	0.0756	0.4033	0.1116	0.2767
		3000	0.1065	0.5052	0.1407	0.2786
E-WP			-0.0518	-0.2542	-0.0408	0.1603
SP 500			-0.0701	-0.3439	-0.0587	0.1706
AE-WPP	2019	30	0.9403	2.3135	0.7139	0.3086
		300	1.0418	2.4039	0.7688	0.3198
		3000	1.0614	2.4096	0.7796	0.3235
E-WP			0.3304	2.3346	0.2948	0.1263
SP 500			0.2871	2.0914	0.2614	0.125
AE-WPP	2020	30	1.2595	1.9064	0.9376	0.4918
		300	1.1907	1.8807	0.9006	0.4788
		3000	1.2422	1.9174	0.9258	0.4829
E-WP			0.149	0.5663	0.2048	0.3617
SP 500			0.1529	0.5861	0.2022	0.3449
AE-WPP	**Mean**	**30**	**0.4656**	**1.4704**	**0.3855**	**0.2541**
		300	**0.4601**	**1.4704**	**0.3824**	**0.2517**
		3000	**0.4723**	**1.4855**	**0.3903**	**0.2534**
E-WP			**0.1835**	**1.4462**	**0.1785**	**0.157**
SP 500			**0.1309**	**1.1456**	**0.1328**	**0.1518**

Finally, we present the ten most important features in 2020 (TABLE III) among all 59 features available.

TABLE III. TOP TEN FEATURE IN 2020

Features	Importance
Revenue	0.096314
Operating income in pct 3-year average	0.064819
Operating cash flow	0.039483
Operating income pct 5-year average	0.038452
Operating income pct year over year	0.037944
Cap spending	0.030685
Debt to equity	0.027306
Payable's period	0.02726
Revenue pct year over year	0.022682
Operating income pct 10-year average	0.021999

IV. CONCLUSION

This paper examined a portfolio construction through training and rules-based learning procedures to predict stocks which fit to belong to the best performing portfolio. The empirical test makes use of daily data from the Standards and Poor's 500 listed companies and their financial statements data and financial ratios, with the Random Forest Algorithm and Decision Tree as base model.

Random Forest Algorithm is well-known to be a robust model when it comes to prediction as it applies a double resampling procedure on observations and features reduce overfitting and generally outperform Decision Trees. AE-WPP provides a means to take advantage of the understanding of how different features interact with the target variable (mean returns) but also classify them according to their ability to partition or properly split the population. That recursive splitting procedure leads to pure partitions or leaves from which the best leaf that maximizes the mean return is chosen to be equal-weight portfolio. Lastly, these individual equal-weight portfolio are aggregated to get the Aggregating Equal-Weight Predicted Portfolio (AE-WPP) as the final predicted portfolio.

As seen previously, the backtesting strategy has proven that the algorithm is promising and can systematically outperform the benchmark from 2013 to 2020. The practical challenge that one can face is the number of stocks or stocks that composed Aggregating Equal- Weight Predicted Portfolio (AE-WPP), as it can contain atomic weights and may demand a big size investment amount. Future improvements still remain such as improving the algorithm by fixing the minimum weights to avoid very small allocation in the final portfolio.

REFERENCES

[1] E. F. Fama and K. R. French, "Dividend yields and expected stock returns," *Journal of Financial Economics*, vol. 22, p. 325, 1988.

[2] R. J. Balvers, T. F. Cosimano and B. McDonald, "Predicting Stock Returns in an Efficient Market," *The Journal of Finance*, Vols. 109-1128, no. 4, p. 45, 1990.

[3] E. F. Fama and K. R. French, "The Cross-Section of Expected Stock Returns," *The Journal of Finance*, vol. 47, pp. 427-465, 1992.

[4] M. H. Pesaran and A. Timmermann, "Predictability of Stock Returns: Robustness and Economic Significance," *The Journal of Finance*, vol. 50, no. 4, pp. 1201-1228, 1995.

[5] J. Lewellen, "Predicting Returns with Financial Ratios," *Journal of Financial Economics*, vol. 74, p. 209–235, 2004.

[6] H. M. Markowitz, "Portfolio Selection," *Journal of Finance*, vol. 7, pp. 77-91, 1952.

[7] J. D. Piotroski, "Value Investing : The Use of Historical Financial Statatement Information to Separate Winners from Losers," *Jounal of Accounting Research*, vol. 38, pp. 1-41, 2000.

[8] P. N. Kolm, R. Tütüncü and F. J. Fabozzi, "60 Years of portfolio optimization: Practical challenges and current trends," *European Journal of Operational Research*, vol. 234, no. 2, pp. 356-371, 2014.

[9] T. Hastie, R. Tibshirani and J. Friedman, Elements of Statistical Learning, Springer ed., Springer, 2009.

[10] E. Guresen, G. Kayakutlu and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Systems with Applications*, vol. 38, pp. 10389-10397, 2011.

[11] S. Lee, D. Enke and Y. Kim, "A relative value trading system based on a correlation and rough set analysis for the foreign exchange futures market," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 47-56, 2017.

[12] N. El Karoui, G. Ban and A. E. B. Lim, "Machine Learning and Portfolio Optimization," *Management Science*, vol. 64, no. 3, pp. 1136-1154, 2018.

[13] T. Conlon, J. Cotter and I. Kynigakis, "Machine Learning and Factor-Based Portfolio Optimization," *Michael J. Brennan Irish Finance Working Paper Series*, vol. 21, no. 6, p. 89, 2021.

[14] T. Kaczmarek and K. Perez, "Building portfolios based on machine learning predictions," *Economic Research-Ekonomska Istraživanja,* pp. 1-20, 2021.

[15] L. Breiman, J. Friedman, R. Olshen and C. J. Stone, Classification and Regression Trees (2nd Ed.), Boca Raton: Chapman and Hall/CRC, 1984.

[16] B. Graham and D. Dodd, Security Analysis, Whittlesey House, McGraw-Hill Book Co., 1934.

[17] B. Graham, The Intelligent Investor, Harper & Brothers, 1949.

[18] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[19] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1-26, 1979.

[20] R. O. Michaud and R. O. Michaud, Efficient asset allocation: A practical guide to stock portfolio optimization and asset allocation, Harvard Business School Press, 1998.

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[22] S. A. Klarman, Margin of Safety: Risk-Averse Value Investing Strategies for the Thoughtful Investor, HarperCollins, 1991.

[23] J. Lintner, "he valuation of risk assets and the selection of risky investments in stock portfolio and capital budgets," *Review of Economics and Statistics*, pp. 47:13-37, 1965.

[24] W. F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk," *Journal of Finance*, pp. 19:425-442, 1964.