A Work Project, presented as part of the requirements for the Award of a Master's Degree in Finance from the Nova School of Business and Economics.

Implementing Machine Learning in the Stock Picking Process of Nova Students Portfolio

Miguel Pardal Afonso

26208

Work project carried out under the supervision of:

Gonçalo Sommer Ribeiro

03-01-2020

Abstract

In a time when algorithmic trading accounts for over 50% of US equities' traded volume, this work project proposes a holistic approach to the implementation of Machine Learning in the Stock Picking process of the Nova Students Portfolio. The presented algorithms can help investors in the identification of the features that drive stock returns and results show that our predictive algorithm provides an edge in the selection of outperforming stocks. An investor using our method from 2006 to 2019 would have achieved an annualized return of 4.8% in excess of the S&P 500 and an Info Sharpe gain of 0.2.

Keywords: Machine Learning, Stock Picking, Portfolio Management

This work used infrastructure and resources funded by Fundação para a Ciência e a Tecnologia (UID/ECO/00124/2013, UID/ECO/00124/2019 and Social Sciences DataLab, Project 22209), POR Lisboa (LISBOA-01-0145-FEDER-007722 and Social Sciences DataLab, Project 22209) and POR Norte (Social Sciences DataLab, Project 22209).

1. Introduction

For many decades, well-known investors like Benjamin Graham, Warren Buffett, Peter Lynch, amongst others, were able to build-up their fortunes through the "*Art of Stock-Picking*"[1] (Munger 2008). Despite what has been accomplished by these successful investment managers, over the past few years there has been an increase in the belief that, going forward, humans will no longer be able to "beat the market" on their own due to the rise of machines, which are able to process a much larger amount of data in a much shorter period, giving them an edge over humans (The Economist 2019).

We acknowledge the many advantages that machines can bring to the stock picking process but also defend the need to exist humans alongside the machines to validate investment decisions. As such, we propose several ways by which Machine Learning algorithms can be introduced in the Nova Students Portfolio, a Master's in Finance course at Nova SBE where students manage a real portfolio worth 370,000 USD.

At Nova Students Portfolio (NSP), students' goal is to outperform the benchmark performance, which consists of a portfolio composed by 40% of the S&P 500 Index and 60% of the ICE US Treasury 3-7y Total Return Index. At the start of each academic year, the NSP portfolio starts with 40% invested in the SPY ETF, which tracks the S&P 500 performance, and 60% invested in the IEI ETF, which tracks the ICE US Treasury 3-7y Total Return Index. Throughout the year, students can outperform the benchmark in two ways: 1) by deviating from the original asset allocation, i.e., by investing a higher proportion of the portfolio in equities or bonds, depending on their assessment of the general macroeconomic conditions; 2) by replacing the exposure of the SPY ETF by exposure to stock picks which they believe will outperform the benchmark during the following months. Knowing that the academic year starts in September and finishes in May, students are advised to have a time-horizon of 6 to 12 months

[1] The "Art of Stock Picking" was advocated by Charlie Munger in 2008 as a very selective process by which investment managers provide a value-added to clients by picking companies that will be winners over a long-term period.

when making their stock picks. This work project aims to improve the stock picking process within the NSP by implementing Machine Learning algorithms that can help students make better decisions and improve the alpha[2] generated by their stock picks.

In Section 2, the Literature Review provides some background on Stock Picking, Machine Learning, and how the two can be integrated. Section 3 regards the Methodology used throughout the work, namely how data was curated, and Section 4 discusses the Machine Learning algorithms that were used. Section 5 presents the results of our work, as well as a backtesting of some strategies based on the algorithms' output. Finally, Section 6 contains the conclusion and practical applications of this work, as well as some limitations and guidelines for future research.

2. Literature Review

The Efficient Market Hypothesis (Fama 1970), dictates that the price of an asset should reflect all the available information at a given point in time, implying that it should not be possible to consistently earn abnormal excess returns through stock selection. Nonetheless, several well-known success stories, such as the case of Warren Buffett, Peter Lynch, or Benjamin Graham, are real-life case studies of individuals who defy the proposition of the Efficient Market Hypothesis, by consistently investing in outperforming stocks.

A lot of research has been conducted around strategies and factors capable of delivering abnormal excess return. Fama and French (1992) firstly argued that factors such as size premium (SMB) and value premium (HML) could be added to the traditional Capital Asset Pricing Model, since the exposure to these factors helps explaining individual stock returns. The Fama-French three-factor model was later expanded by Carhart (1997) with the addition of a momentum factor, which reflected the tendency of past winners to keep their

[2] Throughout this report, we denote by alpha the excess return of a stock over the benchmark (S&P 500) over a certain timespan.

outperformance and past losers to continue their underperformance. Other practitioners, such as Joel Greenblatt and William O'Neill, propose screening strategies for selecting stocks that are likely to outperform the broad market. Greenblatt (2005) proposed the use of a "magic formula" to select 30 stocks with high earnings yield and high return on capital, whereas O'Neill (2009) suggested a CAN SLIM investment strategy which consists of combining seven qualitative and quantitative investment criteria to select stocks based on value and momentum factors. Similarly, Joseph Piotrosky (2000) examined how an accounting-based fundamental analysis can shift the distribution of returns earned by an investor and developed a score that uses nine criteria to evaluate the financial strength of a firm. O'Shaugnessy (1997) followed a statistical approach to rank companies based on growth and value-focused criteria to select the best-ranked stocks, which are statistically more likely to outperform the others.

With the rise of computational power, some of the above cited rule-based strategies evolved to become more complex and difficult to implement by humans. As a result, academics and practitioners have turned to Machine Learning Algorithms (MLAs) instead of traditional statistical techniques like the Ordinary Least Squares (OLS) regression given that MLAs impose little structure to the analysis, enabling them to uncover complex nonlinear patterns, which are harder to explore with OLS (Rasekhschaffe and Jones 2019).

Machine Leaning (ML) can be defined as "*the field of study that gives computers the ability to learn without being explicitly programmed*" (Samuel 1959). Tom Mitchell (1997), computer scientist and professor at Carnegie Mellon University, defined the field in a more formal fashion: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E*". These definitions state two main characteristics of ML – the capability of detecting patterns without having been programmed with explicit rules and the ability to improve performance as the amount of available information increases.

Despite being a fairly new field of investigation, Machine Learning applications in financial markets have been well regarded both by academics and practitioners given the non-linearity and non-stationarity of the data, two issues that MLAs can address better than conventional linear regressions (Geron 2017). As a result, quantitative Hedge Funds such as D.E. Shaw, Citadel, or Bridgewater are increasingly employing ML techniques to make investment decisions (Patterson 2010). Tabb Group (2018) estimates that High-Frequency Trading, which is largely operated by algorithms, accounted for circa 52% of US equities daily volume in 2018.

A wide body of research has been conducted around the deployment of MLAs to forecast financial markets within the last two decades. Quah (2008) was one of the first to write about the use of Neural Networks for stock selection within the Dow Jones Industrial Average and Krauss, et al. (2017) showed how an equal-weighted ensemble of Deep Neural Networks, Gradient-Boosted Trees and Random Forests produced the best performance in an intraday trading statistical arbitrage context of the S&P 500. Gerlein et al. (2016) analyzed the role of simple ML models to achieve profitable trading in the FX market, whilst Patel, et al. (2015) focused on using ML techniques to foresee stock market indices. Takeuchi and Lee (2013) applied Deep Learning to enhance momentum trading strategies in stocks and Khaidem, Saha and Dey (2016) studied the use of Random Forests to forecast the direction of the stock market. Gu, Kelly and Xiu (2019) undertook a comparative analysis of Machine Learning methods for empirical asset pricing and identified decision trees and neural networks as the best performing methods, since these models allow nonlinear predictor interactions that are missed by other methods.

This paper builds upon the work of Rasekhschaffe and Jones (2019), who discuss how practitioners can use ML techniques to forecast the cross-section of stock returns while avoiding overfitting, the primary problem of these techniques. We follow the approach taken by Morgan

Stanley in their Global Factor Guide (2019), which identifies fundamental and technical factors with strong links to stock performance globally, and incorporate both value, growth, momentum and quality factors in our model to predict the performance of stocks in the United States.

3. Data and Methodology

The goal of this work project is to develop an algorithm capable of taking financial data as inputs, processing these data, and predicting if a given stock will outperform the broad market over a given time-frame. Although the models developed throughout this project can be applied in various markets and can be optimized for different holding periods, we focused on their application in the Nova Students Portfolio, where students can only invest in stocks quoted in the US market. As such, we conducted our research for stocks that are part of the Russell 1000, an index that comprises the 1000 largest companies in the US market, representing around 90% of the total US market capitalization. The Russell 1000 was used instead of the S&P 500 to have more data observations per year (1000 instead of 500). However, we did not choose a broader index like the Russell 3000 because several stocks in this index are small-caps with very limited liquidity, which makes them difficult to trade. Furthermore, the data available for this small, unknown companies is often scarce, which would jeopardize the quality of the model. All financial data used in this report was collected from a Bloomberg Terminal.

3.1. Data Curation

Throughout the process of gathering and cleaning the data, we followed the rules of Machine Learning outlined by Martin Zinkevich (2019), data scientist at Google, who talks about the best practices for Machine Learning Engineering.

We started by collecting the necessary data for the construction of the 51 features described in Appendix 1, which will be used as inputs for our model. These features were carefully selected in order to include characteristics related to Value, Growth, Momentum and

Quality factors. The following step involved conducting an Initial Data Analysis (IDA) to verify the structure and quality of the raw data. The IDA process revealed several issues in terms of missing values and poor data quality of some figures, which were solved in different ways. In the presence of missing values, practitioners usually opt for one of three options (Batista and Monard 2003): 1) dropping the whole line of data, which in our case implies dropping one stock from the analysis; 2) dropping the whole column of data, which in our case implies dropping one feature; or 3) replacing the missing value for a proxy. Whenever possible, we decided to fill the missing value by a proxy, as the major advantage of this solution is that we do not decrease our sample (Donders, et al. 2006). As an example, in the cases where certain years used to calculate sales growth rates were not available, we used the closest years for which information was available. Another example includes filling the missing value by the industry average. Nevertheless, given that we want our sample to have the most accurate possible picture of each company, we dropped the companies for which most of the values were missing. Initially, we had more than 70 features, but we decided to drop the ones for which less than 50% of companies had information available. After filling missing values with proxies and eliminating the stocks for which less than 75% of the features were available, we ended up with a sample of 51 features and over 10,000 stocks spread over a 14-year period.

After dropping data points and filling missing values, all data was scaled to a common benchmark. This was a three-step process which aimed to improve the comparability across features and reduce noise. The first step included winsorizing the data. As explained by Hellerstein (2008), the winsorization process is used to eliminate the effect of having outliers that can bias the sample. This process includes replacing the outliers by the number closest to the percentile interval we find adequate. In our case, we replaced every number above the 95^{th} percentile and below the 5^{th} percentile by the numbers that fall on these percentiles. This process

reduces the impact of having outliers without reducing the quality or meaningfulness of the data.

The second step included the standardization of the numbers. As outlined by Osborne (2013), Machine Learning algorithms are prone to be tilted by larger numbers. Our dataset contains figures that are represented in percentages, such as growth rates, but also ratios which are represented by integers, like the P/E Ratio. If one does not standardize the data, the model is likely to be more impacted by ratios than by percentages. The standardization process consisted of scaling each data point to unit variance through the calculation of its *z-score*.

The final step of the data curation process consisted of scaling the values relative to their industries averages. As discussed by Asness, Porter and Stevens (2000), different sectors can have very different characteristics. For instance, technological companies usually have higher growth rates and higher P/E ratio than Utilities, due to structural differences in their industries. For investors looking to compare companies across sectors, it is important to acknowledge these differences and find a way to make companies in different sectors comparable. As such, firstly we classified the companies according to their GICS[3] sector and then calculated the sector average by taking the average of the companies within that sector. After that, we divided the company's value by the sector's average, which gave us a ratio that compares how the company performs against its sector. This largely offsets the previously mentioned problem of not being able to compare growth rates between Technology companies and Utilities, since all numbers now reflect how companies perform relative to their industries. As an example, a value of 1.1x for 1Y Sales Growth Rate means the company has a 1Y Sales Growth Rate 1.1 times higher than the average of its sector.

[3] The Global Industry Classification Standard (GICS) is a classification system developed by MSCI and Standard & Poor's that classifies companies across 11 different sectors: Materials, Energy, Financials, Industrials, Consumer Discretionary, Consumer Staples, Real Estate, Health Care, Information Technology, Utilities, and Communication Services.

4. Models

This paper aims to develop an algorithm capable of predicting which stocks will outperform the benchmark (S&P 500), with the objective of implementing it in the NSP investment process through various forms. Throughout the process of developing the mentioned predictive algorithm, several questions arose, such as "Which years/features shall we take as inputs to train our model?".

To answer these questions, we had to recur to other Machine Learning algorithms. This section describes the steps taken and the algorithms used throughout the process. In summary, we used four different models with distinct purposes:

i. A clustering algorithm was used to check whether outperforming stocks have common characteristics that differentiate them from underperforming stocks, and to confirm if the disparity between the characteristics of outperforming and the characteristics of underperforming stocks was statistically significant;
ii. A cosine similarity algorithm was used to check which years were most similar to the year under analysis, and therefore should be used to train the predictive model;
iii. A feature elimination algorithm was used to select the right features to feed into the model;
iv. A predictive algorithm was applied to forecast which stocks would outperform the market.

Each of these models is further explained in the subsections below.

4.1. Clustering Algorithm

This thesis is built upon the hypothesis that stocks' returns can be predicted based on the features identified in Appendix 1. However, before developing an algorithm that take these features as inputs, we found it essential to perform a preliminary validation of our thesis. As such, we took a statistical approach to understand whether the characteristics of top-performing

stocks are statistically significant from the characteristics of the bottom-performing stocks. To do this analysis, we used the K-Means algorithm, a Python algorithm that is part of the Scikit-Learn library. The K-Means algorithm is an unsupervised[4] learning classification algorithm, whose function is to classify the data into a number of clusters defined by the user. The algorithm operates by looking at the characteristics of each stock and grouping stocks with similar characteristics, allowing the user to find hidden patterns in the unlabeled data. Figure 1 exemplifies how 23 observations are grouped into 4 clusters according to two features.

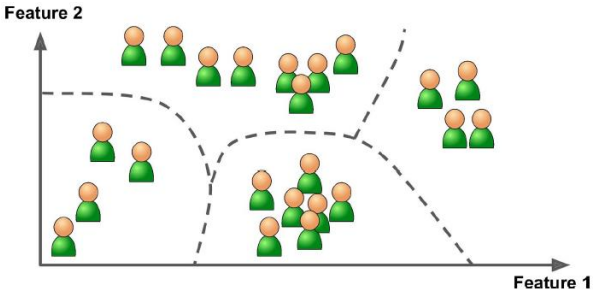

Figure 1 - Clustering representation (Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow)

Mathematically, the algorithm operates by randomly picking *n* points, called centroids, within the dataset, where *n* is the user-defined number of clusters. Then, it calculates the Euclidean distance between every point in the dataset and these *n* points (cluster centroids). The Euclidean distance between $p = (p_1, p_2)$ and $q = (q_1, q_2)$ is given by:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

For each cluster centroid c_i, a data point *x* is assigned to a given cluster based on:

$$\underset{c_i \in C}{argmin}\ d(c_i, x)^2$$

The algorithm then proceeds to find the new centroid from the clustered group of points:

$$c_i = \frac{1}{|S_i|} \sum_{x \in S_i} x_i$$

Where S_i is the group of points that was assigned to the i^{th} cluster. The process is iterated over all k points in the i^{th} cluster, for all n clusters, until the squared error function is minimized:

$$Min \sum_{i=1}^{k} \sum_{j=1}^{n} d(c_i, x)^2$$

[4] Contrary to supervised learning systems, where the data fed into the algorithm includes the desired solution (labels), in unsupervised learnings systems the data fed is unlabeled. The objective of an unsupervised system is, therefore, to label the data according to the algorithm selected by the user.

We selected three as the appropriate number of clusters after applying the reasoning that the first cluster should be constituted by the outperforming stocks, the second cluster should be made of underperforming stocks, and the third cluster should contain the "intermediate" stocks. The "intermediate" cluster is a way to statistically separate the top-performing cluster from the bottom-performing cluster by reducing the noise that would arise if only two clusters were applied.

The K-Means algorithm took the 51 aforementioned features and classified the stocks fed into the model into three clusters. We then proceeded by calculating the *t-statistic* of difference between the characteristics of the clusters. As section 5 outlines in detail, we found evidence of the characteristics and returns of the stocks in the top-performing cluster to be statistically significant from the characteristics and returns of the stocks that constitute the underperforming cluster. These findings allowed us to conclude that these features could be used as inputs for our predictive algorithm.

4.2. Cosine Similarity

After confirming that the difference of characteristics and returns between top-performing stocks and bottom-performing stocks was statistically significant, we were confident to start developing the algorithm that would predict which stocks would be the outperformers over a given timespan. Taking the standard practice amongst data scientists, we split the data into 75% of observations used to train the model and 25% of observations used to perform an out-of-sample evaluation of the model.

The problem now relied on how we should choose the most appropriate training dataset. Given that our goal was to predict the outperformance of stocks in a given year[5], we could only use past data to train the model, or otherwise we would be incurring into a forward-looking

[5] We define as a "year" the period that starts in October and ends in October of the following year, as this is the most relevant period for the NSP. For instance, the year of 2016 refers to the period from Oct-2016 to Oct-2017.

bias. But which years would be more appropriate to select as training years? We took two different approaches to tackle this issue.

The first approach was to simply take the three previous years as training years and then performing an out-of-sample evaluation of the model in the year we wanted to study. In other words, if we wanted to predict which stocks would outperform in year T, we would take all the data for the stocks present in the Russell 1000 in years T-1, T-2 and T-3 and feed their characteristics into the model. The model would then analyze the impact that each feature had on the return of the stock and make a prediction of which stocks would outperform in year T, based on the effects that were seen in the three previous years.

The second approach involved a more sophisticated way to select the three years used to train the model. We took the basic premise explored by Fama and French (1989), that similar years should see similar behaviors in terms of stock returns, and used the cosine similarity to check which years were most similar to the year under analysis in terms of stock characteristics.

The cosine similarity determines the similarity between two vectors by calculating the cosine of the angle between them, irrespective of the magnitude. Mathematically, it computes the product of two vectors divided by the product of the magnitude of each vector:

$$sim(A, B) = \cos(\theta) = \frac{A.B}{|A|.|B|}$$

As an example, if, in year T, stocks showed to have high valuation multiples, the cosine similarity will identify the three previous years with most similar characteristics, i.e., three years that also presented high valuation multiples. By selecting similar years to the one we wanted to predict, the model would be able to understand which features played a bigger role in the outperformance of equities during those years and make more accurate predictions for year T, given that the situation was similar. Empirical evidence shows that the same kind of stocks seems to continue to outperform in the same kind of situations (Kwag and Lee 2006). As an example, value investing has been showed to outperform in periods of crisis, when valuations

play a bigger role, whereas growth stocks outperform in the periods following the crisis, when stocks resurge and investors pay more attention to fast-growing businesses with higher Beta (Clare and Cosimano 2019).

4.3. Recursive Feature Elimination

After selecting the appropriate years to train the model, there was also the need to select the right features. Numerous data scientists point out how "garbage in, garbage out" can ruin a model, i.e., if one feeds inappropriate features into the model, one cannot expect its results to be trustworthy (Kira and Rendell 1992). As such, we took a two-step approach to select the right features, which were later used as independent variables in the predictive model.

The first step consisted in detecting multicollinearity to improve the robustness of the model. Multicollinearity can be defined as a state of very high intercorrelations among the independent variables, which is a disturbance that can affect the reliability of the data. Mansfield and Helms (2012) propose identifying features with high correlation coefficients amongst themselves and eliminating at least one of them. In this case, we checked every pair of features that had an absolute correlation coefficient above 0.5 and eliminated the feature that presented the highest *p-value* in the cluster analysis mentioned in section 4.1. This resulted in the elimination of 9 features that demonstrated a correlation with other features above the threshold, therefore reducing our final sample of features from 51 to 42.

In the second step, we used the Recursive Feature Elimination (RFE) method, which is a backward selection method part of Python Scikit-Learn library, to select the features that have the highest impact on the outcome we are trying to predict. In this method, an external estimator that assigns weights to each feature is firstly trained on the initial set of features and the importance of each feature is obtained through an attribute. Then, the RFE algorithm works by recursively considering smaller sets of features, eliminating the least important features (with lower attributes) from the existing set of features.

The years considered to apply the RFE method were the ones obtained through the cosine similarity algorithm presented in section 4.2, given that these were also the years used to train the main predictive algorithm, which is described in the next subsection (4.4). When putting together the step taken in subsection 4.2 with the step taken in this subsection, we are effectively selecting the years most similar to the year under analysis and considering the features that had the highest importance in those years, hoping that the same features may again have a higher importance in terms of stock returns.

We note that the features are not always the same throughout the analyzed period. This result goes towards the findings of Fidelity (2019), according to which different features play a bigger role during different stages of the business cycle. As a result, there are some years such as 2009 in which the most important features had a "value" tilt, whereas in a year like 2012, most of the features had a "momentum" tilt.

4.4. Predictive Model

After selecting the right years to train the data and the right features to feed the model, the last step of this work involved using an algorithm that could effectively predict which stocks would be able to outperform the benchmark over a given timeframe. Rather than using a linear regression model to predict how much a given stock would yield over a year, we took a nonlinear, binary approach, which simply indicates whether a stock is likely to outperform the benchmark over a given timeframe or not. The reason for having decided to follow a binary approach is because it contributes in a different manner to the existing body of research that aims to predict nominal stock returns. Furthermore, we found this method to be more suitable to the needs of the NSP. NSP students' goal is to beat the benchmark, and so they need a framework useful to select stocks that can fulfill this goal. Structuring the investment decision in a binary, simplified way facilitates the process of stock selection and improves the capacity of ML tools to support it.

Python's Scikit-Learn library[6] has numerous binary predictive algorithms that we could use in our task. However, as outlined by Lagoudakis, Littman and Parr (2001), data scientists often find themselves struggling to understand which model is best to use under which circumstances and why. Especially when the model used is a time-series model rather than a cross-sectional model, the algorithm that yields the best results in one year might not be the algorithm that yields the best results in the following year. Moreover, since these algorithms are not linear, there is a higher probability of overfitting to the training data, and subsequently having a poor performance on the testing set (Hawkins 2003). To avoid these problems, the approach taken was to use four different algorithms and then using a VotingClassifier to select the final output. The VotingClassifier is a meta-classifier for combining different Machine Learning classifiers via majority voting. It can operate through "hard-voting", in which case the final class label is the class label that has been predicted most frequently by the individual classification models, or through "soft-voting", where class labels are predicted by averaging the class-probabilities. We decided to use the "soft-voting" since this method considers not only the individual binary decisions of each classifier, but also the likelihood of their outcomes, which is not captured under "hard-voting".

In "soft-voting", the class label $\hat{y}$ is predicted based on the predicted probabilities p for each classifier:

$$\hat{y} = argmax \sum_{j=1}^{m} w_j p_{ij}$$

Where w_j is the weight assigned to the j^{th} classifier.

Since this is a binary classification task, the only two possible outcomes are 0, which means that the stock will not outperform the benchmark, and 1, which means the stock will outperform the benchmark.

[6] The oficial documentation for the Scikit-Learn library can be found in https://scikit-learn.org/stable/user_guide

One of the major drawbacks of Machine Leaning Algorithms is the uncertainty around how they work and issue predictions (Rudin 2019). To avoid the issue of not understanding the functioning of the model, we opted not to use more complex models like Deep Neural Networks, as these are often referred to as "black-box" algorithms, difficult to dissect. As such, we used four easily understandable models for which there is already a fair amount of research, as previously outlined in the Literature Review. In Appendices 3-6, we take a look at the mathematical explanation behind each of the four Scikit-Learn algorithms used: Logistic Regression, Random Forest, Support Vector Machines and Gradient Boosting.

5. Results

After having shared the reasoning, the literature support and the methodology that has been applied throughout this work, we present our results in this section. We start by presenting the results of the clustering algorithm in subsection 5.1, which demonstrates the statistical relevance of the features initially selected. We then proceed to show the features with higher predictive power over stock returns in subsection 5.2, and the results of our prediction algorithm are presented in subsection 5.3, as well as a backtesting of its performance for the Oct-2006 to Oct-2019 period.

5.1. Clustering

When analyzing the results of the clustering algorithm, we aim to answer the question "Are there any statistically significant differences between the characteristics of top-performing stocks vs. lower-performance stocks?". To answer this question, we compared the *p-value* of the difference between the average values for the top-performing cluster, i.e., the cluster with the best 12m average performance, and the bottom-performing cluster, i.e., the cluster with the worst 12m average performance. We did this analysis per year and per sector to understand if

results changed between time-periods and across sectors. A 5% threshold has been kept throughout the analysis of *p-values* in this section.

Appendix 2 shows the *p-values* as well as the averages of each cluster for three different time periods: 2006-2010, 2010-2014 and 2014-2018. 2006-2010 corresponds to the Financial Crisis Period, 2010-2014 corresponds to the period in the aftermath of the Financial Crisis, and 2014-2018 corresponds to the period since the inception of the NSP. Our results show that, out of the 51 analyzed features, 20 appear to be statistically significant across the three periods. As Figure 2 shows, the number of Growth, Quality and Value factors seemed to have been less relevant in the after-crisis period than in the other two periods, whilst there was a higher number of Momentum factors that were significant in the aftermath of the crisis than in the other two periods. We note that Quality factors were particularly significant throughout the crisis years, which shows that investors tend to focus more on robust businesses during tough periods.

	HC	Ind	Mat	RE	Tech	Uti	Comm	Disc	Stap	Energy	Fin	2006-2010	2010-2014	2014-2018
Performance 12 months*	3.3%	0.0%	0.7%	54.8%	0.2%	0.8%	0.3%	0.0%	0.1%	3.9%	0.0%	0.0%	0.0%	0.9%
No. of Growth Features**	7	11	6	7	13	6	11	11	8	13	5	12	9	13
No. of Quality Features	5	6	4	4	7	3	5	2	5	7	6	8	4	7
No. of Value Features	2	6	5	4	4	5	3	4	4	4	4	6	5	6
No. of Momentum Features	10	7	6	7	8	5	6	8	8	7	7	9	11	9
Total Number of Features	24	30	21	22	32	19	25	25	25	31	22	35	29	35

* P-value of the difference between the average return of stocs in the top-performing cluster and the returns of stocks in the bottom-performing cluster

** Number of features belonging to that factor that show a p-value below 5% when comparing the top-performing and bottom-performing clusters

Figure 2 - Significance of performance and features between clusters per sector and period

Looking at the performance of each cluster, measured by the average alpha of that cluster relative to the S&P 500, one can see that the differences in 9m and 12m performances are always statistically significant but the same doesn't hold true in the case of 1m/3m/6m performances. Since most of the features considered are related to Financial Statements, it is important to understand that these may take time until they are reflected into stock returns, which possibly explains why the return divergences are only significant in the longer term.

Turning to a sector analysis, Figure 2 also highlights the sectors for which each factor was most relevant. A complete table of results can be found in Appendix 2. If one excludes the Financial sector from the analysis, given the particular characteristics of this sector, it can be

noticed that Growth Factors were most significant for the Information Technology and Energy sectors, and less important for Utilities and Materials. Quality factors show higher significance for Information Technology and Energy and lower significance for Consumer Discretionary. A higher number of Value factors is significant for Industrials, but a lower number is significant for Health Care. Finally, Momentum seems to play a bigger role in Health Care, but the opposite is true for Utilities.

5.2. Selected Features

The top 10 most selected features by our Recursive Feature Elimination model is shown in Figure 3. When comparing these features with the results obtained in the previous subsection (5.1), we observe that the features with lower *p-values* in our cluster analysis (Appendix 2) are also the ones most chosen to explain the alpha of some securities. In an opposite way, one can also see that some of the least selected features are the same that present higher *p-values* across sectors and years.

Top 10 most selected features		Top 10 least selected features	
Feature	**Years selected***	**Feature**	**Years selected***
PE	13	Sales_Growth_1Y	4
EPS_Growth_3FW	12	FCF_Growth_3Y	5
SALES_SURPRISE_LAST_ANNUAL	12	FCF_Growth_1Y	5
Sales_Growth_1FW	11	EBITDA_Growth_1Y	6
Sales_Growth_3FW	11	FCF_Growth_1FW	6
Impl_Vol	11	ROE	6
Vol30_Vol260	11	EBITDA_Growth_1FW	7
ADX	11	Div growth	7
Beta	11	Div_Yld	7
Sales_Growth_3Y	10	Buyback Yield	7

* Number of years the feature was selected as having predictive power over stock's returns

Figure 3 - Top 10 most and least selected features for the period 10/2006 – 10/2019

These results corroborate our thesis that different features can have distinct levels of importance across time and sectors, and so it is essential to find out which features are most important and should therefore be used as inputs when building a predictive model.

5.3. Prediction algorithm

When evaluating the performance of a binary model like the one used in this work project, there are two metrics of utmost importance: accuracy and precision. Accuracy measures the ratio between correct prediction and total predictions. Formally:

$$Accuracy = \frac{True\ Positives + True\ Negative}{True\ Positives + True\ Negative + False\ Positives + False\ Negatives}$$

In other words, accuracy attempts to answer the question "What proportion of all predictions was actually correct?". Applying to this business case, accuracy will measure the percentage of times that the model was right in predicting the over/underperformance of stocks. As an example, an accuracy of 70% means that every time the model predicted that a stock was going to outperform/underperform the market, it was correct 70% of the times.

On the other hand, precision measures the ratio between correct positive prediction and all positive predictions. Formally:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

In other words, precision attempts to answer the question "What proportion of positive predictions was actually correct?". Applying to this business case, precision will measure the percentage of times that the model was right in predicting just the outperformance of stocks. As an example, a precision of 70% means that every time the model predicted that a stock was going to outperform, it was correct 70% of the times.

The main difference between accuracy and precision is that, whilst accuracy measures how often the model is right at predicting both the overperformance and underperformance of stocks, precision focuses only on how often the model is correct at predicting the overperformance. Given the NSP is a long-only portfolio and the goal of this model is to help in the identification of stocks which are likely to outperform the market, it is preferable to have a model with higher precision.

Figure 4 shows the precision and accuracy of our model when predicting stock return performances over a 6m, 9m and 12m period. One can see that average precision and average accuracy are higher than 50%, the "random guess" threshold. This implies that our model provides an edge at predicting whether a stock will outperform the broad market. Figure 5, which shows how good the model is at predicting if a stock is going to outperform its sector, also presents average results for precision/accuracy higher than 50%, implying that the model also provides an edge at identifying which stocks will be outperformers within a certain sector.

	6 months		9 months		12 months		
Year	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Alpha***
2006	60.8	59.6	52.7	52.6	47.1	48.5	3.0
2007	51.4	51.7	51.3	52.3	53.7	49.0	0.0
2008	49.1	52.9	52.5	51.2	53.2	51.6	2.6
2009	57.6	51.8	63.0	57.8	62.5	57.8	9.6
2010	58.7	54.1	58.6	56.5	48.6	55.0	0.3
2011	53.3	50.9	46.9	49.9	44.7	51.7	-1.3
2012	68.3	57.2	67.7	61.4	63.4	51.8	11.5
2013	51.1	55.0	50.0	52.9	41.8	55.9	-0.7
2014	63.4	57.5	58.6	56.4	68.0	56.0	8.4
2015	47.5	49.0	49.3	50.6	51.2	54.6	0.7
2016	46.1	49.5	54.0	55.2	48.9	54.3	2.5
2017	50.7	54.7	49.5	56.4	48.4	63.0	4.6
2018	60.2	55.5	66.4	59.2	68.5	59.1	8.9
Average	**55.2**	**53.8**	**55.4**	**54.8**	**53.8**	**54.5**	**3.9**

* Annualized excess return of the algorithms' predictions over the S&P500 during that year

Figure 4 - Precision and Accuracy per year across the period 10/2006 – 10/2019, for different holding periods

	6 months		9 months		12 months		
GICS Sector	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Alpha***
Communication Services	56.4	49.6	43.4	49.8	45.6	49.2	2.8
Consumer Discretionary	51.9	54.5	47.7	54.1	44.7	51.5	-1.8
Consumer Staples	54.6	53.0	55.6	51.4	50.0	50.2	1.9
Energy	48.9	51.5	48.4	51.9	51.2	53.3	1.5
Financials	51.3	48.8	52.7	50.5	49.9	50.6	1.5
Health Care	58.5	54.7	58.1	53.5	50.3	52.8	5.3
Industrials	56.8	53.4	59.7	56.9	58.3	53.3	4.0
Information Technology	53.6	50.2	56.0	52.1	47.0	53.0	1.2
Materials	54.3	54.1	53.3	52.0	48.4	49.3	1.5
Real Estate	49.7	49.1	57.1	54.7	57.8	55.5	2.5
Utilities	61.5	53.6	53.9	50.5	59.1	53.7	2.8
Average	**54.3**	**52.1**	**53.3**	**52.5**	**51.1**	**52.0**	**2.1**

* Annualized excess return algorithms' predictions over the respective sector performance

Figure 5 - Precision and Accuracy per sector across the period 10/2006 – 10/2019, for different holding periods

We took the predictions of the algorithm for the period Oct-2006 to Oct-2019 and backtested two long-only strategies that invested in the stocks predicted by the algorithm as

being outperformers over the following 12m. Strategy 1 predicts which stocks will outperform the S&P 500 over 12m by any amount, whilst Strategy 2 predicts which stocks will outperform the S&P 500 by at least 5% over the following 12m. At the end of 12m, we rebalance the portfolios by considering the new predictions of the algorithm.

	S&P 500	Strategy 1	Strategy 2
Average no. of stocks*	n/a	420	166
Cumulative return	193%	353%	460%
Annualized Return	7,9%	11,1%	12,7%
Annualized Volatility	18,9%	20,1%	20,4%
Maximum Drawdown	-55,2%	-57,0%	-56,4%
% Positive Months	67,9%	66,0%	66,0%
% Positive Years	85,7%	85,7%	92,9%
Info Sharpe	0,42	0,55	0,62
Info Sortino	0,49	0,66	0,75
Information Ratio	n/a	0,75	0,92
Skewness	-0,13	-0,32	-0,30
Kurtosis	11,76	9,06	7,89

* Average number of stocks invested per year

Figure 6 - Risk/Return profiles of S&P 500, Strategy 1 and Strategy 2, for the period 10/2006 – 10/2019

As one can see in Figure 6, both strategies outperform the S&P 500 over the 14-year timespan, although Strategy 2 presents a better performance. Strategy 1 presents an annualized return of 11.1% vs. 12.7% for Strategy 2 and 7.9% for the S&P 500. Strategy 2 presents a higher Info Sharpe (0.62) vs. Strategy 1 (0.55) and the S&P 500 (0.42). The Information Ratio, a measure of portfolio returns in excess of the returns of a benchmark, also highlights the outperformance of Strategy 2, which has a ratio of 0.92, compared to 0.75 for Strategy 1. The outperformance of Strategy 2 is explained by the fact that, by setting the threshold of outperformance to be higher than 5% rather than higher than 0%, the algorithm becomes much more selective at predicting outperformers, improving its precision. As such, the average number of stocks selected as outperformers per year dropped from 420 to 166, highlighting the selectiveness of the model.

The cumulative performance for the S&P 500, Strategy 1 and Strategy 2 is shown in Figure 7. The matrices of monthly returns for both strategies can be found in Appendices 7-8.

Figure 7 - Cumulative performance of Strategies 1 and 2 vs. S&P 500, for the period 10/2006 – 10/2019

We also tested the application of the model to the stock picks inside the NSP. Taking the characteristics of each stock pick at the time the NSP invested, we fed the algorithm with the respective features and considered the output of whether that stock would be an outperformer or not. The algorithm achieved an accuracy of 52% and a precision of 57% at predicting the stocks that would be outperformers. Figure 8 compares the dollar P&L for a strategy that invested in all securities picked in the NSP since inception for different holding periods versus two strategies that only invested in the securities that were identified by the two previously mentioned algorithms as being outperformers. The results show how the P&L of the NSP since inception could be increased by a cumulative amount of circa USD 19,000 by the implementation of such an algorithm.

	Total no. of stock picks	6 months*		9 months*		12 months*	
		Alpha***	P&L	Alpha***	P&L	Alpha***	P&L
NSP**	185	-1.4%	$ 20,176	-2.2%	$ 22,707	-0.8%	$ 48,913
Strategy 1	56	1.2%	$ 38,009	1.0%	$ 44,077	1.8%	$ 66,823
Strategy 2	30	1.3%	$ 38,285	1.3%	$ 46,066	1.9%	$ 67,490

* Holding period of the pick

** Considers the actual amount invested at the time of the pick

*** Calculated as the average alpha of each pick during the selected holding period

Figure 8 - Adjusted alpha and P&L for NSP actual portfolio, Strategy 1 and Strategy 2, for the period 10/2006 – 10/2019

Strategy 2, which only invests in the stocks likely to overperform the S&P 500 by at least 5%, continues to present the best performance in terms of P&L. Appendix 9 shows the precision and accuracy of Strategy 2 at predicting which past NSP picks were likely to outperform at the time.

6. Conclusion, Applications & Directions for further research

This work project introduces the use of Machine Learning algorithms to the stock picking process and shows how algorithms can help humans in stock selection, taking as example the case of NSP. We demonstrate the models presented can attain at least three different purposes: 1) indicate which past years are most similar to the current one, helping in the assessment of the current microeconomic conditions; 2) select the most relevant features/factors, in general and by sector, serving as a support to investors in the process of building stock screeners/scorecards; 3) identify which stocks are most likely going to be outperformers over a given timeframe, providing an edge over a random guess.

Beyond the theoretical framework outlined in this paper, which contributes to the previous research around the use of Machine Learning in stock selection, we made available to the NSP all of the tools used in our research and we encourage their use by the current team, who can start implementing some of the highlighted strategies straight away.

We view the results achieved by our work as a starting point for future research and implementation of other ideas within the NSP and recognize some of the limitations of the presented models. Firstly, we only considered microeconomic data for the feature selection. We note that the inclusion of macroeconomic data, such as measures of economic growth and interest rates, is likely to further boost the model by enlarging the scope of the analysis. Secondly, the strategies applied focused on a long-only investment approach, which does not allow for short selling. Given that the model achieves an accuracy above 50%, we denote that a strategy that is long the stocks predicted as outperformers and short the ones predicted as underperformers is also likely to achieve good risk-adjusted returns. Finally, we focused our analysis in the US market given the investment policy of the NSP. Nonetheless, we encourage similar studies to be conducted for other regions of the world, to understand to what extend Machine Learning can be used by investors on a global scale.

7. References

Asness, Clifford, R. Burt Porter, and Ross L. Stevens. 2000. “Predicting Stock Returns Using Industry-Relative Firm Characteristics.” Working Paper.

Batista, Gustavo, and Maria Monard. 2003. “An Analysis of Four Missing Data Treatment Methods for Supervised Learning.” *Applied Artifical Intelligence* 519-533.

Breiman, Leo. 1998. “Arcing Classifiers.” *The Annals of Statistics* Vol. 26, No. 3, pp. 801-824.

Carhart, Mark. 1997. “On Persistence in Mutual Fund Performance.” *The Journal of Finance* 57-82.

Clare, Andrew, and Thomas Cosimano. 2019. “Economics and Investment Markets.” In *2019 CFA Program Curriculum Level II Volume 6*, by CFA Institute, 379-469. CFA Institute.

Donders, Rogier, Geert Heijden, Theo Stijnen, and Karel Moons. 2006. “Review: A gentle introduction to imputation of missing values.” *Jornal of Clinical Epidemiology* 1087-1091.

Fama, Eugene. 1970. "Efficient Capital Markets: A Review of Theory and Empirical Work." *Journal of Finance* 383-417.

Fama, Eugene, and Kenneth French. 1989. “Business Conditions and Expected Returns on Stocks and Bonds.” *Journal of Financial Economics* 23-49.

Fama, Eugene, and Kenneth French. 1992. “The Cross-Section of Expected Stock Returns.” *Journal of Finance* 427-465.

Fidelity. 2019. *How to invest using the business cycle.* 26 June. Accessed December 20, 2019. https://www.fidelity.com/viewpoints/investing-ideas/business-cycle-investing.

Friedman, Jerome. 1999. "Greedy Function Approximation: A Gradient Boosting Machine."

Gerlein, Eduardo A., Martin McGinnity, Ammar Belatreche, and Sonya Coleman. 2016. “Evaluating machine learning classification for financial trading: An empirical approach.” *Expert Systems with Applications* 193-207.

Geron, Aurelien. 2017. *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* O'Reilly.

Greenblatt, Joel. 2005. *The Little Book That Beats the Market.* Wiley.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu. 2019. “Empirical Asset Pricing via Machine Learning.”

Hawkins, Douglas. 2003. "The Problem of Overfitting." *Chemical Information and Computer Sciences* 1-12.

Hellerstein, Joseph. 2008. "Quantitative Data Cleaning for Large Databases."

Khaidem, Luckyson, Snehanshu Saha, Suryoday Basak, and Saibal Kar. 2016. “Predicting the direction of stock market prices using random forest.” *Applied Mathematical Finance.*

Kira, Kenji, and Larry Rendell. 1992. “A Practical Approach to Feature Selection.” *Machine Learning Proceedings* 249-256.

Krauss, Cristopher, Xuan Do, and Nicolas Huck. 2017. “Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500.” *European Journal of Operational Research* 689-702.

Kwag, Seung, and San Lee. 2006. “Value Investing and the Business Cycle.” *Journal of Financial Planning* Article 7.

Lagoudakis, Michail, Michael Littman, and Ron Parr. 2001. “Selecting the Right Algorithm.” *American Association for Artificial Intelligence* 74-75.

Mansfield, Eduard, and Billy Helms. 2012. “Detecting Multicollinearity.” *The American Statistician* 158-160.

Mitchell, Tom. 1997. *Machine Learning.* McGraw-Hill.

Morgan Stanley. 2019. “Global Factor Guide.” Quantitative Research.

Munger, Charlie. 2008. "The Art of Stock Picking."

O’Shaughnessy, James. 1997. *What works on Wall Street.* McGraw-Hill.

O'Neill, William. 2009. *How to Make Money in Stocks.* McGraw-Hill.

Osborne, Jason. 2013. *Best Practices in Data Cleaning.* Sage.

Patel, Jigar, Sahil Shah, Priyank Thakkar, and Kim Kotecha. 2015. “Predicting stock market index using fusion of machine learning techniques.” *Expert Systems with Applications* 2162-2172.

Patterson, Scott. 2010. “Letting the Machines Decide.” *Wall Street Journal*, 13 July: 23-25.

Piotrosky, Joseph. 2000. "Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers." *Journal of Accounting Research* 1-41.

Quah, Tong-Seng. 2008. "DJIA stock selection assisted by neural network." *Expert Systems with Applications* 50-58.

Raschka, Sebastian, and Vahid Mirjalili. 2019. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition.* Packt Publishing.

Rasekhschaffe, Keywan, and Robert Jones. 2019. “Machine Learning for Stock Selection.” *Financial Analysts Journal.*

Rudin, Cynthia. 2019. "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead." *Nature Machine Intelligence* 206-2015.

Samuel, Arthur. 1959. "Some Studies in Machine Learning Using the Game of Checkers." *IBM Journal of Research and Development* 210-229.

Tabb Group. 2018. *Equities Research.* Accessed December 17, 2019. https://research.tabbgroup.com/research.

Takeuchi, Lawrence, and Yu Lee. 2013. “Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks.”

The Economist. 2019. "March of the machines." *Masters of the universe*, October 5: 18-20.

Vapnik, Vladimir. 1995. *The nature of statistical learning theory.* Springer.

Zinkevich, Martin. 2019. *Rules of Machine Learning: Best Practices for ML Engineering.* Accessed December 12, 2019. https://developers.google.com/machine-learning/guides/rules-of-ml.

8. Appendices

Appendix 1 - Features' description

Factor	Feature	Description
Growth	Sales_Growth_3Y	Annualzed Sales growth rate over the previous 3 years
	Sales_Growth_1Y	Sales growth rate over the previous year
	Sales_Growth_1FW	Sales growth rate over the following year (Bloomberg estimates)
	Sales_Growth_3FW	Annualzed Sales growth rate over the next 3 years (Bloomberg estimates)
	EBITDA_Growth_3Y	Annualzed EBITDA growth rate over the previous 3 years
	EBITDA_Growth_1Y	EBITDA growth rate over the previous year
	EBITDA_Growth_1FW	EBITDA growth rate over the following year (Bloomberg estimates)
	EBITDA_Growth_3FW	Annualzed EBITDA growth rate over the next 3 years (Bloomberg estimates)
	EPS_Growth_3Y	Annualzed EBITDA growth rate over the previous 3 years
	EPS_Growth_1Y	EPS growth rate over the previous year
	EPS_Growth_1FW	EPS growth rate over the following year (Bloomberg estimates)
	EPS_Growth_3FW	Annualzed EPS growth rate over the next 3 years (Bloomberg estimates)
	FCF_Growth_3Y	Annualzed FCF growth rate over the previous 3 years
	FCF_Growth_1Y	FCF growth rate over the previous year
	FCF_Growth_1FW	FCF growth rate over the following year (Bloomberg estimates)
	FCF_Growth_3FW	Annualzed FCF growth rate over the next 3 years (Bloomberg estimates)
	Div growth	Dividend growth rate over the previous year
Quality	EBITDA_Margin	EBITDA Margin
	Op_Margin	Operating Margin
	Prof_Margin	Profit Margin
	ROE	Return on Equity
	ROA	Return on Assets
	ROIC	Return on Invested Capital
	INTEREST_COVERAGE_RATIO	Interest Coverage Ratio
	TOT_DEBT_TO_TOT_EQY	Debt to Equity ratio
	CUR_RATIO	Current ratio
	NET_DEBT_TO_EBITDA	Current Return on Equity
	Accruals	Current Return on Equity
	Asset Turnover	Asset turnover
Value	PE	Price to Earnings Ratio
	PEG	Price to Earnings Growth ratio
	PB	Price to Book ratio
	FCF_YLD	FCF Yield
	EV_EBITDA	EV/EBITDA
	Div_Yld	Dividend Yield
	Buyback Yield	Buyback Yield
	Total Yield	Dividend Yield + Buyback Yield
Momentum	Impl_Vol	Implied Volatility based on options
	Vola30_Vola260	Volaility 30d vs Volatitility 260d
	Vol30_Vol260	Volume 30d vs Volume 260d
	MA50_MA200	Moving Average 30d vs Moving Average 200d
	TRR_3M	Total return over past 3 months
	TRR_6M	Total return over past 6 months
	TRR_12M	Total return over past 12 months
	ADX	Average Directional Index
	RSI	Relative Strength Index
	EPS_SURPRISE_LAST_QTR	% Actual vs Bloomberg Estimates EPS in the last available quarter
	SALES_SURPRISE_LAST_QTR	% Actual vs Bloomberg Estimates Sales in the last available quarter
	EPS_SURPRISE_LAST_ANNUAL	% Actual vs Bloomberg Estimates EPS in the last available year
	SALES_SURPRISE_LAST_ANNUAL	% Actual vs Bloomberg Estimates Sales in the last available year
	Beta	Market Beta

Appendix 2 – Features' significance per sector and period

		Sectors												Periods		
	Factor	**HC**	**Ind**	**Mat**	**RE**	**Tech**	**Uti**	**Comm**	**Disc**	**Stap**	**Energy**	**Fin**	**No. sectors***	**2006-2010**	**2010-2014**	**2014-2018**
Performance	Alpha_1m	2.9%	5.2%	88.3%	51.7%	71.7%	39.6%	90.7%	25.8%	69.7%	48.4%	1.6%	**2**	0.0%	22.0%	1.5%
	Alpha_3m	3.0%	2.3%	91.0%	80.4%	44.7%	1.0%	73.0%	0.6%	4.5%	0.0%	14.5%	**6**	0.0%	0.5%	5.7%
	Alpha_6m	63.3%	21.8%	91.6%	11.9%	18.2%	1.4%	6.9%	3.1%	2.9%	0.8%	3.0%	**5**	0.2%	0.1%	78.2%
	Alpha_9m	68.1%	0.0%	39.1%	56.8%	2.3%	13.1%	1.3%	0.0%	1.8%	36.2%	0.0%	**6**	0.0%	0.0%	2.8%
	Alpha_12m	3.3%	0.0%	0.7%	54.8%	0.2%	0.8%	0.3%	0.0%	0.1%	3.9%	0.0%	**10**	0.0%	0.0%	0.9%
Growth	Sales_Growth_3Y	0.0%	0.1%	9.6%	0.0%	0.0%	58.8%	0.0%	0.0%	0.5%	0.0%	0.8%	**9**	0.0%	0.6%	0.0%
	Sales_Growth_1Y	n/a	19.8%	17.8%	42.2%	0.0%	5.9%	30.0%	0.0%	6.1%	0.0%	n/a	**3**	n/a	4.4%	0.0%
	Sales_Growth_1FW	0.0%	0.1%	55.0%	90.7%	0.0%	0.6%	0.0%	0.0%	1.2%	0.6%	0.0%	**9**	0.0%	76.4%	0.2%
	Sales_Growth_3FW	0.5%	0.0%	24.2%	2.4%	0.0%	35.7%	0.0%	0.0%	0.1%	1.7%	60.7%	**8**	10.8%	0.0%	0.0%
	EBITDA_Growth_3Y	74.4%	6.4%	36.0%	1.9%	96.6%	0.7%	12.4%	0.4%	0.0%	9.3%	99.8%	**4**	0.0%	31.2%	0.0%
	EBITDA_Growth_1Y	21.8%	0.2%	2.6%	0.2%	0.0%	0.0%	0.0%	4.4%	41.8%	0.0%	68.8%	**8**	0.0%	30.1%	0.0%
	EBITDA_Growth_1FW	0.0%	1.6%	4.3%	84.2%	0.0%	40.7%	0.0%	32.9%	59.2%	0.1%	26.0%	**6**	17.9%	40.6%	0.0%
	EBITDA_Growth_3FW	23.7%	0.0%	74.4%	0.0%	0.0%	16.4%	2.1%	93.1%	n/a	0.9%	22.9%	**5**	13.0%	55.6%	0.0%
	EPS_Growth_3Y	28.9%	21.5%	0.9%	57.0%	49.2%	21.6%	0.0%	6.4%	0.0%	0.0%	2.2%	**5**	0.3%	0.0%	0.0%
	EPS_Growth_1Y	91.7%	0.5%	0.2%	78.8%	1.3%	n/a	94.6%	22.6%	68.0%	75.0%	18.3%	**3**	0.0%	60.4%	0.0%
	EPS_Growth_1FW	55.8%	1.7%	9.4%	3.1%	57.0%	44.8%	0.2%	1.9%	4.6%	35.1%	11.0%	**5**	0.0%	4.9%	3.2%
	EPS_Growth_3FW	71.4%	0.0%	0.0%	0.0%	4.0%	6.1%	0.0%	2.4%	22.6%	0.0%	7.8%	**7**	0.0%	0.0%	0.0%
	FCF_Growth_3Y	0.2%	5.9%	n/a	17.0%	0.0%	0.0%	0.0%	0.8%	1.4%	2.5%	n/a	**7**	0.0%	82.0%	0.8%
	FCF_Growth_1Y	91.7%	n/a	n/a	n/a	3.0%	n/a	n/a	10.3%	n/a	1.3%	n/a	**2**	0.0%	50.0%	16.6%
	FCF_Growth_1FW	3.5%	19.3%	68.7%	n/a	n/a	n/a	0.2%	66.3%	75.1%	2.1%	n/a	**3**	n/a	1.2%	29.0%
	FCF_Growth_3FW	5.3%	0.2%	0.9%	23.0%	4.4%	0.0%	71.8%	0.0%	0.4%	52.7%	0.1%	**7**	0.0%	0.0%	91.2%
	Div growth	0.0%	0.0%	15.0%	31.4%	0.0%	0.1%	5.3%	0.0%	8.7%	4.6%	0.0%	**7**	0.0%	0.0%	28.4%
	Number of Growth Factors**	**7**	**11**	**6**	**7**	**13**	**6**	**11**	**11**	**8**	**13**	**5**		**12**	**9**	**13**
Quality	EBITDA_Margin	0.0%	0.0%	23.9%	0.0%	0.0%	69.4%	0.0%	0.0%	0.0%	0.0%	0.0%	**9**	0.0%	0.1%	0.7%
	Op_Margin	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	68.7%	n/a	**0**	n/a	n/a	n/a
	Prof_Margin	n/a	n/a	n/a	n/a	n/a	n/a	13.1%	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	ROE	n/a	0.0%	0.0%	13.7%	n/a	n/a	n/a	n/a	n/a	0.0%	n/a	**3**	0.0%	0.0%	0.0%
	ROA	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	ROIC	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	INTEREST_COVERAGE_RATIO	0.0%	0.0%	n/a	17.2%	1.2%	7.4%	0.0%	0.7%	0.0%	0.0%	0.0%	**8**	0.0%	0.0%	0.0%
	TOT_DEBT_TO_TOT_EQY	0.0%	4.7%	0.0%	0.2%	0.0%	4.4%	32.6%	64.4%	0.0%	0.0%	0.0%	**9**	0.0%	28.2%	0.0%
	CUR_RATIO	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.3%	5.5%	0.2%	0.0%	0.0%	**10**	0.1%	34.5%	0.0%
	NET_DEBT_TO_EBITDA	n/a	5.7%	0.0%	0.0%	0.0%	n/a	n/a	n/a	0.0%	0.0%	2.9%	**6**	0.0%	12.9%	0.0%
	Accruals	61.5%	0.0%	10.5%	n/a	0.0%	0.9%	0.0%	57.1%	73.2%	3.7%	3.2%	**6**	0.4%	24.2%	0.0%
	Asset Turnover	0.0%	n/a	n/a	n/a	0.0%	n/a	0.1%	n/a	n/a	n/a	n/a	**3**	0.0%	0.0%	59.9%
	Number of Quality Factors	**5**	**6**	**4**	**4**	**7**	**3**	**5**	**2**	**5**	**7**	**6**		**8**	**4**	**7**
Value	PE	30.5%	0.0%	0.0%	79.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	**9**	0.0%	0.0%	0.0%
	PEG	17.3%	2.6%	7.9%	7.0%	79.8%	0.5%	48.7%	0.0%	8.9%	4.4%	0.0%	**5**	0.0%	0.2%	0.0%
	PB	n/a	0.0%	0.0%	3.4%	n/a	0.0%	0.0%	n/a	0.0%	9.7%	n/a	**6**	0.0%	0.0%	0.0%
	FCF_YLD	0.5%	0.0%	1.0%	1.8%	0.3%	25.5%	22.0%	0.0%	0.2%	13.9%	0.0%	**8**	0.0%	68.1%	0.0%
	EV_EBITDA	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	Div_Yld	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.2%	0.0%	0.0%	0.0%	0.0%	**11**	0.0%	0.0%	0.0%
	Buyback Yield	25.5%	0.0%	0.0%	0.0%	0.0%	0.0%	89.5%	16.0%	99.3%	0.0%	61.8%	**6**	0.0%	0.0%	0.0%
	Total Yield	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	Number of Value Factors	**2**	**6**	**5**	**4**	**4**	**5**	**3**	**4**	**4**	**4**	**4**		**6**	**5**	**6**
Momentum	Impl_Vol	0.0%	0.0%	n/a	17.0%	0.0%	n/a	0.0%	0.0%	0.0%	n/a	24.7%	**6**	0.0%	0.0%	0.4%
	Vola30_Vola260	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	**11**	0.0%	0.0%	0.5%
	Vol30_Vol260	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	n/a	n/a	4.4%	26.9%	0.0%	**8**	0.0%	0.0%	0.0%
	MA50_MA200	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	TRR_3M	0.0%	0.0%	53.0%	0.0%	18.5%	n/a	n/a	0.0%	0.0%	0.1%	0.0%	**7**	0.0%	0.0%	12.8%
	TRR_6M	0.0%	35.9%	0.0%	n/a	n/a	n/a	n/a	0.0%	19.8%	n/a	32.0%	**3**	0.2%	0.0%	0.2%
	TRR_12M	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	ADX	n/a	n/a	0.0%	0.0%	0.0%	0.0%	12.4%	0.0%	0.0%	0.7%	0.0%	**8**	0.1%	0.1%	0.0%
	RSI	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	**0**	n/a	n/a	n/a
	EPS_SURPRISE_LAST_QTR	0.0%	8.4%	0.3%	0.0%	0.0%	0.6%	0.0%	0.0%	0.0%	0.5%	1.7%	**10**	5.6%	2.9%	0.1%
	SALES_SURPRISE_LAST_QTR	0.0%	0.0%	0.0%	40.4%	13.4%	14.1%	34.8%	36.7%	57.2%	48.7%	2.7%	**4**	4.2%	0.0%	0.0%
	EPS_SURPRISE_LAST_ANNUAL	0.5%	7.6%	21.7%	0.0%	0.0%	35.5%	1.2%	0.3%	0.0%	0.0%	12.0%	**7**	0.1%	0.0%	0.2%
	SALES_SURPRISE_LAST_ANNUAL	0.0%	0.1%	17.6%	13.6%	0.0%	83.7%	0.1%	76.3%	22.5%	4.9%	99.0%	**5**	7.7%	3.0%	40.4%
	Beta	0.7%	0.0%	n/a	0.0%	0.0%	0.0%	2.3%	0.0%	0.0%	0.0%	0.0%	**10**	0.0%	0.0%	0.1%
	Number of Momentum Factors	**10**	**7**	**6**	**7**	**8**	**5**	**6**	**8**	**8**	**7**	**7**		**9**	**11**	**9**
	Total Number of Factors	**24**	**30**	**21**	**22**	**32**	**19**	**25**	**25**	**25**	**31**	**22**		**35**	**29**	**35**

* Represent the number of sectors in which we observe a p-value below 5%

** Represents the number of features that have a p-value below 5% in that sector / period

*** n/a means that feature was not considered in the analysis due to multicollinearity

Legend: HC = Health Care, Ind = Industrials, Mat = Materials, RE = Real Esatate, Tech = Information Technology, Uti = Utilities, Comm = Communication Services, Disc = Consumer Discretionary, Stap = Consumer Staples, Energy = Energy, Fin = Financials

Source: Author calculations

Appendix 3 - Logistic Regression

The Logistic Regression is a classification algorithm that predicts the probability of a binary outcome of belonging to a given class. The regression hypothesis is based on the Sigmoid Function $(\theta^T x)$:

$$H_\theta(x) = \sigma(\theta^T x) = \frac{1}{1 + e^{(-\theta^T x)}} \tag{3.1}$$

Results of 3.1 are rebased into the range [0, 1] (Figure 9), such that $H_\theta(x)$ is interpreted as a probability:

$$H_\theta(x) = P(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1 + e^{(-\theta^T x)}} \tag{3.2}$$

And

$$P(y = 0|x) = 1 - \sigma(\theta^T x) = 1 - \frac{1}{1 + e^{(-\theta^T x)}} \tag{3.3}$$

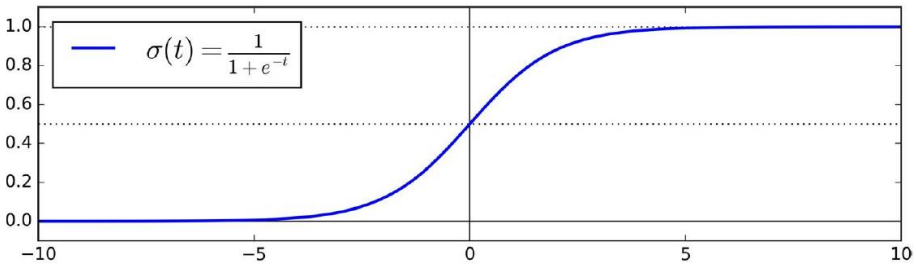

Figure 9 - Logistic Function (Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow)

The Logistic Regression Loss Function, which evaluates the goodness of the parameters' fit, is defined as:

$$\mathrm{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(y^i \ln(p^i)) + (1 - y^i)(1 - \ln(p^i)) \tag{3.4}$$

Where:

$$p^i = H_\theta(x^i) = \frac{1}{1 + e^{(-\theta^T x)}} \tag{3.5}$$

The Logistic Regression Loss Function is minimized throughout the training process and optimized in function of θ by using a gradient descendent function and deriving with respect to $\theta_j (\forall j \in [0, n])$:

$$\frac{\partial(J(\theta))}{\partial\theta_j} = --\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i \tag{3.6}$$

Iteratively repeating:

$$\theta_j = \theta_j - \alpha\left(-\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)x_j^i\right) \tag{3.7}$$

Where α is the gradient descendent function's learning rate.

Appendix 4 - Random Forest

The Random Forest Classifier is an Ensemble Algorithm. Ensemble algorithms combine more than one algorithm for classifying objects. As outlined in Figure 10, Random Forests work by training several Decision Trees on random subsets of the features, and then averaging out their predictions.

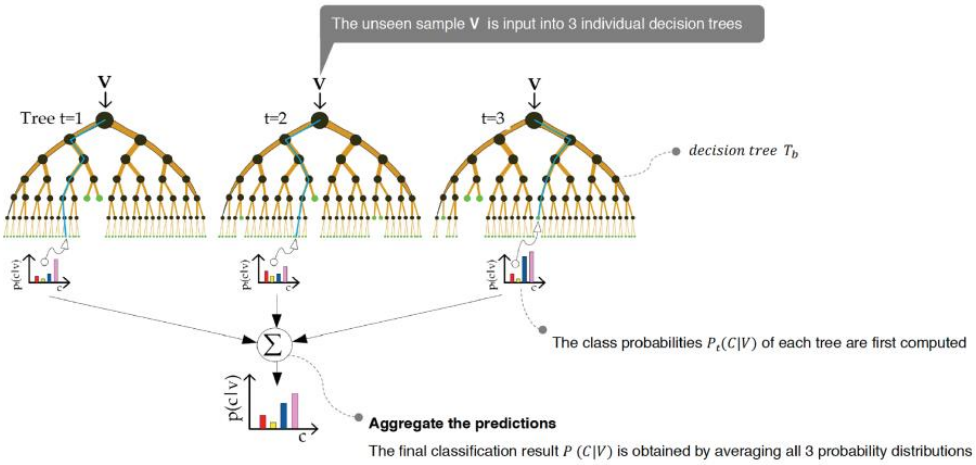

Figure 10 - Random Forest representation (Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow)

The Decision Trees inside the Random Forest Classifier use the CART algorithm, where the Gini index is used as a metric. The Gini index in a cost function defined as:

$$G_i = 1 - \sum_{k=1}^{n} {P_{i,k}}^2 \tag{4.1}$$

Where $P_{i,k}$ is the ratio of class k amongst the training instances in the i[th] node. The Gini score states how good a split is by showing how mixed the classes are in the two groups created by the split. A Gini score of 0 indicates perfect separation, whereas the worst split results in 50/50 classes. The process in repeated recursively for every row and the data is split accordingly in a binary tree.

The Gini Index is computed for every feature. Taking the average information entropy for the selected attribute:

$$H_\theta(S) = \sum -p(c) log_2 p(c) \tag{4.2}$$

Where $c = \{yes, no\}$, we are able to calculate the Gini gain. The best Gini gain attribute is picked, and the process repeated until the desired tree is reached. The CART algorithm splits the training set to minimize the MSE:

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right} \tag{4.3}$$

$$\text{where: } \begin{cases} MSE_{node} = \sum_{i \in node} (y_{node} - y^i)^2 \\ y_{node} = \frac{1}{m_{node}} \sum_{i \in node} y^i \end{cases} \tag{4.4}$$

A forecast f at a new point x can be made through:

$$f^{B}_{\substack{rf \\ b=1\ to\ B}} = \frac{1}{B} \sum_{b=1}^{B} Tb(x) \tag{4.5}$$

Considering $C_{b(x)}$ as the class prediction of the b[th] decision tree. Then:

$$C_{rf}^{B}(x) = majority\ vote\ \{C_b(x)\}_1^B \tag{4.6}$$

Appendix 5 - Support Vector Machines

Support Vector Machines (SVM) are a classification method, firstly introduced by Vapnik (1995), that works by drawing a straight line between two classes. The data points that fall on a given side of the line are labeled as one class and the points that fall on the other side of the line are labeled as a second class. The goal is to find a hyperplane that maximizes the separation between classes. In the illustration shown in Figure 11, it is possible to observe how the red line maximizes the separation between the blue datapoints and the red ones.

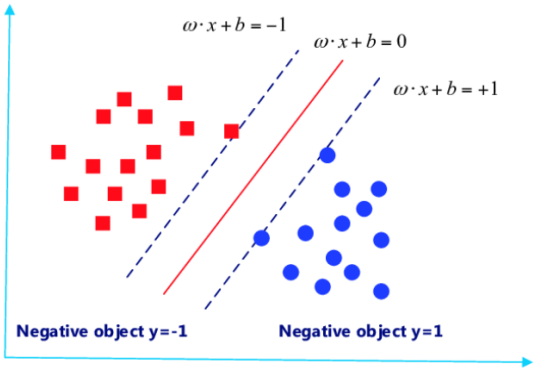

Figure 11 - Illustration of an SVM classifier (Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow)

Mathematically, given a dataset $\{x_i, y_i\}_{i=1}^{l}$, $x_i \in \mathbb{R}^n, y_i = \pm 1$, where x_i is an input vector and y_i is the class label, the SVM approximates a separating hyperplane through:

$$f(x) = w.\emptyset(x) + b \tag{5.1}$$

Where $\emptyset(x)$ is a high dimensional feature space.

The optimal hyperplane for equation 5.1 is found by setting the value of the margin to $\frac{2}{|w|}$. The maximum value for the margin is found though a minimization process to estimate *w* and *b*:

$$Min\ C \sum_{i=1}^{l} \varepsilon_i + \frac{1}{2} |w|^2 \tag{5.2}$$

$$s.t.\ y_i(w.\emptyset(x) + b \geq 1 - \varepsilon_i, i = 1, \dots, l$$

In equation 5.2, $C \sum_{i=1}^{l} \varepsilon_i$ sums the training errors. C is a user-defined parameter that determines the trade-off between training error and margin, and $\frac{1}{2} |w|^2$ maximizes the margin. If one applies the Lagrange multipliers α_i to exploit the optimality conditions, equation 5.1 takes the form:

$$f(x, \alpha_i) = \sum_{i=1}^{l} \alpha_i, y_i k(x_i, x) + b \tag{5.3}$$

Where α_i is obtained by optimizing equations 5.2 and 5.3:

$$R(\alpha_i) = \sum_{I=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j x_i x_j K(x_i, x_j) \tag{5.4}$$

$$s.t. \sum_{i=1}^{l} \alpha_i \alpha_j = 0, 0 \leq \alpha_i \leq C$$

$K(x_i, x_j)$ is known as the kernel function. The user determines which kernel function is appropriate. Given the non-linearity of our original problem, we followed the advice of Raschka and Mirjalili (2019) and opted to use the Radial Basis Function kernel, which is a non-linear kernel.

Appendix 6 - Gradient Boosting

Like the Random Forest, the Gradient Boosting is a ML technique used for regression and classification problems, which consists of an ensemble of weaker models. The boosting method consists in generalizing weaker models though the optimization of an arbitrary differentiable function. The "boosting" idea was first observed by Breiman (1998) and subsequently developed by Friedman (1999).

Using a training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ of inputs x and outputs y, the objective is to find an approximation $\hat{G}(x)$ to a function $G(x)$, minimizing the expected value of a loss function $L(Y, G(x))$:

$$\hat{G} = argmin E_{x,y}[L(y, G(X))] \tag{6.1}$$

The Gradient Boosting seeks to find an approximation $\hat{G}(x)$ in the form of a weighted sum of functions $h_i(x)$, which are the weak learners:

$$\hat{G}(x) = \sum_{i=1}^{m} \gamma_i h_i(x) + b \tag{6.2}$$

As per the empirical risk minimization principle (Vapnik 1995), the value of $\hat{G}(x)$ should be such that it minimizes the loss function on the training set. As shown in Figure 12, the algorithm starts with an initial model consisting of a constant function $G_0(x)$ and incrementally expands it by adding other weaker learners:

$$G_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma) \tag{6.3}$$

$$G_m(x) = G_{m-1}(x) + argmin \left[\sum_{i=1}^{n} L(y_i, G_{m-1}(x_i) + h_m(x_i)) \right] \tag{6.4}$$

Where $h_m \in H$ is a weaker learner function. The number of weaker learners should be pre-defined by the user in order to avoid the overfitting to the training set.

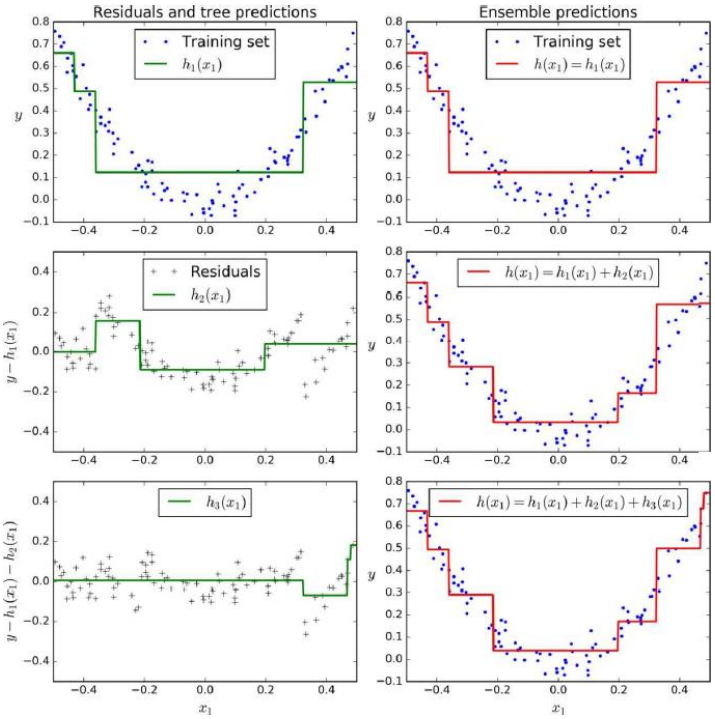

Figure 12 - Illustration of the Gradient Boosting expanding process (Source: Hands-On Machine Learning with Scikit-Learn & TensorFlow)

Appendix 7 - Strategy 1 monthly returns

Year	January	February	March	April	May	June	July	August	September	October	November	December	YTD
2006										3.9%	3.7%	-0.1%	**7.5%**
2007	3.7%	-0.3%	0.7%	3.8%	4.0%	-1.8%	-3.7%	0.5%	3.2%	2.3%	-4.7%	-1.4%	**6.3%**
2008	-6.4%	-2.0%	-1.2%	6.0%	3.6%	-9.0%	-1.9%	2.2%	-12.3%	-22.0%	-9.4%	3.2%	**-49.1%**
2009	-8.4%	-10.1%	8.6%	12.1%	3.4%	0.6%	8.6%	4.3%	4.7%	-3.1%	5.1%	5.2%	**30.9%**
2010	-3.8%	4.5%	6.5%	3.4%	-7.1%	-5.5%	6.8%	-4.3%	10.2%	3.9%	1.7%	6.4%	**22.4%**
2011	2.0%	4.2%	1.8%	3.4%	-0.8%	-1.7%	-3.1%	-6.7%	-9.6%	13.3%	-0.7%	-0.1%	**1.9%**
2012	6.3%	4.3%	2.5%	-0.6%	-7.6%	2.9%	-0.6%	3.5%	2.5%	0.1%	1.8%	1.9%	**16.9%**
2013	6.9%	1.8%	4.3%	1.0%	2.0%	-1.2%	5.9%	-2.7%	4.6%	4.3%	2.7%	3.2%	**32.7%**
2014	-2.0%	5.6%	0.2%	-1.0%	2.6%	3.0%	-2.5%	4.7%	-3.0%	4.3%	3.3%	0.3%	**15.6%**
2015	-1.3%	5.8%	0.3%	-1.0%	2.4%	-1.0%	2.6%	-5.6%	-2.7%	6.1%	1.2%	-2.1%	**4.6%**
2016	-6.5%	1.1%	7.2%	1.0%	2.3%	0.3%	3.9%	0.4%	0.0%	-2.7%	4.5%	1.2%	**12.8%**
2017	2.4%	3.5%	0.5%	1.8%	1.8%	0.7%	2.0%	0.7%	2.7%	2.7%	3.9%	0.8%	**23.4%**
2018	4.9%	-2.8%	0.3%	-0.1%	3.2%	1.0%	2.9%	4.1%	0.3%	-8.8%	2.4%	-8.2%	**-0.8%**
2019	9.3%	5.9%	2.1%	4.7%	-2.9%	6.4%	2.3%	-0.7%	-1.2%				**26.1%**

Source: Author calculations

Appendix 8 - Strategy 2 monthly returns

Year	January	February	March	April	May	June	July	August	September	October	November	December	YTD
2006										3.8%	3.9%	0.4%	**8.2%**
2007	4.4%	-0.6%	0.9%	3.8%	4.9%	-1.6%	-4.0%	0.3%	3.6%	2.5%	-6.1%	-1.8%	**6.2%**
2008	-6.3%	-2.2%	-0.5%	6.1%	3.0%	-10.6%	-1.9%	2.1%	-13.7%	-21.1%	-7.4%	3.4%	**-49.1%**
2009	-6.6%	-9.5%	8.4%	10.6%	3.4%	0.0%	7.8%	3.7%	4.5%	-4.9%	4.9%	6.6%	**28.9%**
2010	-3.6%	5.0%	6.5%	4.3%	-7.7%	-7.2%	7.1%	-5.3%	11.2%	4.1%	1.9%	5.7%	**21.9%**
2011	2.0%	4.0%	2.2%	3.2%	-0.7%	-1.1%	-3.0%	-5.3%	-8.9%	12.8%	0.3%	-0.4%	**5.0%**
2012	6.5%	4.5%	3.3%	0.2%	-6.6%	3.1%	-0.2%	3.7%	1.5%	-1.1%	3.0%	0.9%	**18.9%**
2013	6.8%	2.2%	5.6%	1.7%	2.9%	-1.3%	5.6%	-1.9%	5.2%	4.9%	2.8%	3.4%	**38.0%**
2014	-1.7%	5.7%	-0.1%	-2.2%	2.8%	3.5%	-2.9%	5.4%	-3.0%	2.4%	3.0%	-0.5%	**12.5%**
2015	-3.0%	6.9%	0.4%	-0.1%	2.2%	-1.1%	2.5%	-5.8%	-3.6%	6.9%	1.1%	-1.3%	**5.1%**
2016	-7.6%	1.2%	7.9%	1.1%	2.6%	1.0%	5.4%	-0.6%	-0.1%	-2.4%	5.8%	0.8%	**15.0%**
2017	2.5%	3.5%	0.8%	3.1%	3.4%	0.5%	2.5%	0.6%	3.5%	3.4%	4.5%	0.5%	**28.9%**
2018	6.3%	-2.4%	0.9%	0.1%	4.3%	1.4%	2.7%	6.1%	1.3%	-11.2%	2.7%	-7.7%	**4.4%**
2019	11.2%	7.1%	2.5%	4.8%	-2.7%	6.4%	2.9%	-1.1%	-2.7%				**28.5%**

Source: Author calculations

Appendix 9 – Precision and Accuracy of Strategy 2 applied to the NSP for different holding periods

	6 months		9 months		12 months	
Year	**Precision**	**Accuracy**	**Precision**	**Accuracy**	**Precision**	**Accuracy**
2014	55.6	67.4	40.0	62.8	100.0	76.7
2015	40.0	55.9	50.0	50.0	66.7	61.8
2016	100.0	47.2	75.0	27.8	80.0	38.9
2017	100.0	57.9	70.0	73.7	69.2	68.4
2018	0.0	50.0	100.0	64.7	100.0	73.5
Average	**59.1**	**55.7**	**67.0**	**55.8**	**83.2**	**63.9**

Source: Author calculations