UNIVERSIDAD TORCUATO DI TELLA

Master's in Management & Analytics
Business School

ML in Finance: Portfolio Management via Side & Size Prediction on the Bonds Market

Student: Nahuel Rodrigo Sanchez

Thesis Director: Pablo Roccatagliata

Date: 20/07/2020

ML in Finance: Portfolio Management via Side & Size Prediction on the Bonds Market

Abstract

The rise in automation and utilization of algorithms in the last decades had been meaningful in several areas, finance, and portfolio management included. The present study combines two approaches to reach an integral optimized model. The first one is the traditional approach, which starting from forecasting the future bond yield curve, generates a decision to take: establish a long position (expecting a rise on the price) or a short one (expecting a fall on the price). Therefore, the output of this first model will be to determine the position side. The second approach is the application of the bet-sizing technique to optimize the resulting decisions from the traditional model by assigning them a probability of being correct: decisions with a low probability of generating profits will have a lower size, while decisions with a high probability of generating returns will have a bigger size. The algorithms used were the ARIMA regression for the traditional model and random forest for the bet-sizing model. Cross-validation and out-of-sample backtests were conducted to evaluate how the model would have performed and results show that employing the integrated optimized model exhibits higher Sharpe ratios than using only the traditional approach. The work demonstrates that the modern techniques used along with the traditional ones reach better efficiency on returns than when only traditional models are employed. Additionally, generalizations to other areas inside finance, both on asset management as well as on credit risk are discussed.

UNIVERSIDAD TORCUATO DI TELLA

Master's in Management & Analytics
Business School

Aprendizaje Automático en Finanzas: Administración de Portfolios vía Predicción de Dirección y Tamaño en el Mercado de Bonos

<u>Resumen</u>

El aumento en la automatización y la utilización de algoritmos en las últimas décadas han sido significativos en varias áreas, incluidas las finanzas y la gestión de la cartera. El presente estudio combina dos enfoques con el objetivo de alcanzar un modelo integral optimizado. El primero es el enfoque tradicional, que a partir de pronosticar la curva de rendimiento de los bonos futuros, genera una decisión: establecer una posición larga (esperando un aumento en el precio) o una posición corta (esperando una caída en el precio). Por lo tanto, el resultado de este primer modelo será determinar el lado de la posición. El segundo enfoque es la aplicación de la técnica de bet-sizing para optimizar las decisiones resultantes del modelo tradicional al asignarles una probabilidad de ser correctas: las decisiones con una probabilidad baja de generar ganancias tendrán un tamaño menor, mientras que las decisiones con una probabilidad alta de generar retornos tendrán un tamaño mayor. Los algoritmos utilizados fueron la regresión *ARIMA* para el modelo tradicional y el *random forest* para el modelo de bet-sizing. Los *back-tests* en validación cruzada y fuera de la muestra se realizaron para evaluar cómo se habría desempeñado el modelo en esos escenarios y los resultados muestran que el empleo del modelo optimizado integrado tiene ratios de *Sharpe* más altos que el uso del enfoque tradicional exclusivamente. El trabajo demuestra que las técnicas modernas utilizadas junto con las tradicionales alcanzan una mayor eficiencia en los retornos que cuando solo se emplean modelos tradicionales. Además, se discuten las generalizaciones a otras áreas dentro de las finanzas, tanto en la gestión de activos como en el riesgo crediticio.

Index:

1. Introduction

1.1 Machine Learning in Finance

Over the last decades, we have witnessed the increase in algorithms being applied in finance in a lot of different environments: identification of market trends, risk management (on all of its dimensions: credit, market, reputation, and so on), and also for automated trading.

What once started with lineal regressions have evolved to building complex models to predict where prices will be in the near or long term future, classify companies into clusters to differentiate them by how risky they are, or directly estimate the probability of default of a given obligor. The quantitative approach has expanded to every interacting unit: large investment banks, hedge funds, corporations, even individuals and organizations without profit objectives.

This became possible because of the advances in the whole machine learning area, but also due to the great advances in the Data Engineering field: not only to store data but also on transfer and processing speed. On this same line, because information flows faster, the models developed must be flexible enough to adapt or recalibrate quickly to the market conditions. When a model generates positive returns, it is exploiting an open space on the market, but because of the information spreading speed, will eventually be discovered by other models and competitors, demanding, as it was stated, to adapt and search for the next opportunity.

On the present work, the focus will be on the bonds market, starting and taking advantage of the traditional approaches on forecasting and exploiting opportunities on this market, to afterward extend and reinforce the approach using an additional machine learning model. This first section will introduce the concepts which will be used throughout the entire work. The second section and third section will be taking on the traditional and modern approaches respectively, as well as explaining the datasets, engineering, modeling, and results. Finally, the fourth section will perform the backtesting, discuss the generalization of this two-model approach to other kinds of situations or products, and express the conclusions.

1.2 The bonds market and the traditional approaches on forecasting

Investors include bonds in their investment portfolios for a range of reasons including income generation, capital preservation, capital appreciation and as a hedge against economic slowdown.

Income generation: Bonds provide investors with a source of income in the form of coupon payments, which are typically paid quarterly, twice yearly or annually. The investor can use the income generated by their investments for spending or reinvestment. In comparison, shares also provide income in the form of dividends: however, such payments are less certain and tend to be less than bond coupons.

Capital preservation: Unlike stocks, the principal value of a bond is returned to the investor in full at maturity. This can make bonds attractive to risk-averse investors who are concerned about losing their capital.

Capital appreciation: Although bonds are often viewed as a capital preservation tool, they also offer opportunities for capital appreciation. This occurs when investors take advantage of rising bond prices by selling their holdings prior to maturity on the secondary market. This is often referred to as investing for total return and is one of the more popular bond investment strategies.

Hedge against economic slowdown: While investors in stocks typically do not welcome a slowdown in economic growth, it can be a good thing for bond investors. This is because a slower growth usually leads to lower inflation, which makes bond income more attractive. An economic slowdown may also be negative for company profits and stock market returns, adding to the attractiveness of bond income during such a time.

The bonds market has two main partitions: sovereign bonds and corporate bonds. While the bonds from the first kind are issued by government entities like countries, states, or municipalities, the second ones are issued by companies (this includes those which are controlled by the government as well). Additionally, the bonds can have a lot of characteristics, like coupon quantity and rates, maturity terms, and amortization schedules among others.

On the bonds markets, there is an inverse relationship between the yield of a given bond and its price: when the yield of a bond goes up, its price goes down. At each point of time, each bond type will have a yield and a maturity, therefore a "yield curve" can be created at each point of time showing the yield for each maturity.

The yield curve represents the relationship, at any particular point in time, between yields on bonds and their remaining maturities for the same issuing entity (i.e., the US government). Yield curves are typically presented for benchmark government bonds, so that the curve reflects "pure" interest rate factors and is not confused with credit risk, liquidity, and other factors. (Frank J. Fabozzi, 2012)

There are three main types of yield curve shapes: normal (upward sloping curve), inverted (downward sloping curve), and flat. The first yield curve type is called normal as this is the general expectation because of the relation between time and risk in lending: as the time horizon extends, the uncertainty of getting back the borrowed money or value is higher, and the interest rate demanded increases consequently. The second case is the inverted curve, which is exactly the opposite of the normal curve: long-term maturities fall below the short-term maturities. Finally, the third type is the flat curve, where short-term and long-term yields are on very similar levels.

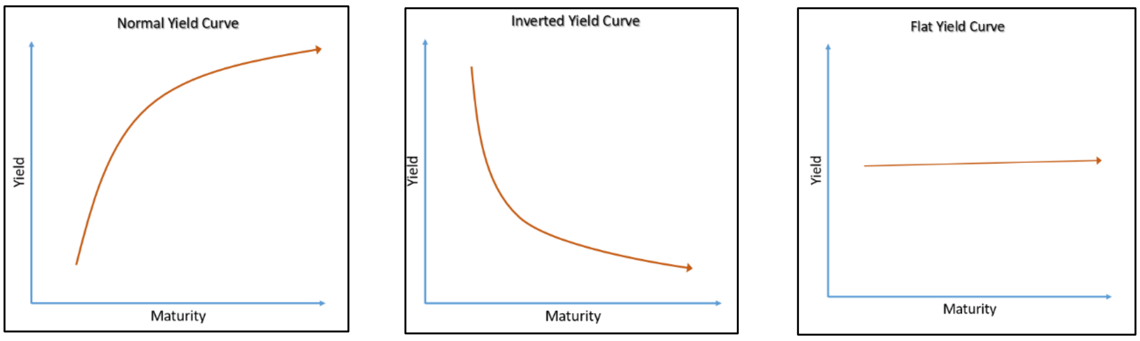

Figure n°1: Examples of the main types of yield curve.

On practice, the shape of the yield curve is determined by the following points (Frank J. Fabozzi, 2012):

- Expectations on the part of market participants as to future (short-term) interest rates.
- Expectations as to inflation over the period covered by the bond.

- Interactions between supply and demand factors pertaining to particular points (maturities) on the curve plus risk aversion.
- Interest rate volatility and bond convexity.

Each point in time will have its particular curve, and two given maturities can be compared to see if the yield rate increases, decreases, or remains at the same level. By identifying the change between two points in time for the same maturity, the price change sign can be identified: if the yield rate goes down, the price will rise, and if the yield rate increases, the bond price will fall. This happens because of the relationship between rate and the bond prices: if a zero-coupon bond is considered (for simplicity), the return is computed via calculating the difference between its actual price and its par value (the value upon redemption at maturity): as the price goes down, the difference will be bigger (meaning a higher yield rate), and vice versa.

As such, to be able to know for a given bond price will rise or decrease, a comparison must be made between two different curves: today's curve and tomorrow's curve. On the maturities where the yield goes down, a long position must be taken (buy the security). On the contrary, if a yield goes up, the price will decrease, and a short position (borrow the security and return it later) will be convenient to exploit the forecast.

The traditional approach to forecasting a yield curve is the equation developed by Nelson-Siegel (1987) and further expanded by Diebold and Li (2005). They separate the yield curve into three main elements: level, slope, and curvature. The first component (β_1), level, is considered the long term factor. The second component (β_2), slope, is the short term element, and curvature, the third component (β_3), is the medium-term factor. In addition to those elements, there is one more factor of λt, which governs the exponential decay rate along the curve. The equation can be written as below:

$$y_t(\tau) = \beta_{1t} + \beta_{2t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau}\right) + \beta_{3t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau} - e^{-\lambda_t \tau}\right).$$

Each factor will be associated with one specific maturity to represent the factor influence. The point of this work is to further improve the traditional approach, therefore, to apply the before mentioned equation to a dataset and learn the patterns behind the data, the association will be the same as the established by Diebold and Li on their paper. The long-term factor will be aligned with the 10-year maturity yield, the short term factor will be the difference between the 10-year maturity yield and the 3-month maturity yield, and finally, the medium-term factor will be 2 times the 2-year maturity yield minus the 10-year and 3-month maturity yields. Summarizing in equation form:

1. $\beta_1 = 10 year\ maturity\ yield$
2. $\beta_2 = 3 month\ maturity\ yield$
3. $\beta_3 = 2 * 2 year\ maturity\ yield - (10 year\ maturity\ yield + 3 month\ maturity\ yield)$

Finally, the model framework used to estimate the Nelson-Siegel equation is the ARIMA (Autoregressive integrated moving average) model (Box, Jenkins, 1991). Several elements participate on the ARIMA framework as the initials indicate: auto regression, moving average, integration.

In a multiple regression model, the variable of interest is forecasted using a linear combination of predictors. In an autoregression model, the variable of interest is forecasted using a linear combination of past values of the variable. Therefore, the term autoregression indicates that it is a regression of the variable against itself. An autoregressive model of order ***p*** can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Different than the autoregression term, which operates on the past values of the variable to predict itself, the moving average term uses the past forecast errors, also in a regression-like model, to forecast the future value of the variable of interest. A moving average model of order ***q*** can be written as:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

Finally, the integration component refers to the possibility and order of differencing required to make the time series stationary. A time series is stationary when its statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. A given time series may not be stationary on its original state but upon differentiation (calculating the difference of the variable of interest between its actual value and a given number of lags) the time series may become stationary and the ARIMA framework will be able to forecast appropriately.

In summary, the three parameters to optimize on this model are the following:

- ***p***, the autoregressive operator: defines how many of its past values the variable depends to forecast. (e.g. if p=2, the model will incorporate the past two values of the variable to perform the forecast)
- ***d***, the differencing order: defines how many times the time series must be differentiated to reach a stationary status.
- ***q***, the moving average operator: defines how many of the previous white noise error terms the variable depends to forecast. (e.g, if q=3, the model will take into account the past three error terms to perform the prediction)

The ultimate objective of employing the traditional approach as the first step on the portfolio management integrated model is to determine the *side* of the investments to be made at each point of time, which means if the trade to execute is going long (buy) or short (sell). In other words, the first model will be the *position-side* predictor.

1.3 The incorporation of Machine Learning to the traditional approach

Artificial intelligence, automation, and predictive analytics are transforming virtually every industry. It's imperative to incorporate technology into the investment process to provide managers with tailored information that enables faster decision-making, makes every step of the process more efficient, and provides a more complete picture of risk and opportunity. (Distenfeld, DiMaggio, Skoglund, Swtizer, 2018).

There are several contributions to the machine learning introduction into the finance world, and specifically to the fixed-income environment. On one of the relatively recent papers (Barnabás, Hollifield, 2018.), a study is carried to analyze and predict the price of 3-year, 10-year, and 30-year U.S. treasuries using different machine learning algorithms: random forests, support vector machines, and neural networks, using a lag method: given the previous n prices, predict the next price.

An early paper (Colin, 2006) to use machine learning for bond price prediction used a neural network to predict the 50-year U.S. Treasury bond price. Four variables were used as input: transaction settlement date, coupon rate, yield, and maturity date. The model output was the bond's quoted price.

Another case (Castellani, dos Santos, 2006) attempted to predict the monthly yield of 10-year treasury bonds using macroeconomic indicators such as the CPI (Consumer Price Index). However, being the indicators used as input for the model monthly-based, the results couldn't outperform ordinary regression.

A later study (Ganguli, Dunnmon, 2017) executed a more comprehensive analysis using a large corporate bond dataset which contained sixty-one attributes for each bond, with some of them describing the bond itself, while others came from the previous ten transactions, including the time for the trade to be reported, quantity, real price, etc. The authors used several classification methods, from linear regression models to more complex algorithms like random forests and neural networks.

A final study to mention took a different approach to predict the prices of Japanese corporate bonds. (Jotaki, Yamashita, Takahashi, 2017) Instead of the previous prices, the authors used fundamental and macroeconomic factors as inputs: some related to company-specific data, but others to management expectations. The machine learning algorithm they used was a support vector machine, and the output was if the bond would have a normal or abnormal negative return. They used six years of data for training and two years for testing, and they reached an accuracy of 65% of the negative abnormal returns and 62% of the normal returns.

The integration of technology and precisely machine learning into both modeling and quantitative research opens the possibility of efficiency gains along the whole spectrum of asset management via several points:

1. Lower transaction costs.
2. Faster investment of new cash flows.
3. Better executions. (Higher accuracy, rate of returns, lower risk)

The machine learning model developed in this study will mainly refer to the third point, achieving better results by enhancing the side-predictor model via side optimization.

Once the side-prediction model has been calibrated to choose on each scenario if the position to open must be a long or a short one, predictions can be made over a test period. However, this test period must have not been shown to the model on the training phase to avoid what is called "*data leakage*". This concept embodies essentially the introduction of information about the target meant to forecast, even if it was done unintentionally (Kaufman, Rosset, Perlich, 2011). The consequence of data leakage is that the model will have a higher accuracy on validation, but when the model goes live, the accuracy will be a lot lower.

As a result of running the position-side model over the test dataset, a decision vector will be generated, and as for every estimation, some of those decisions will be incorrect. If the model were to be applied as it is, having decent accuracy, it would generate profitable strategies.

To improve the results of just selecting the position side, a second variable will be incorporated into the portfolio management integrated model: the "*position size*". The framework which will be followed to optimize this second variable is called "*bet-sizing*" (Lopez de Prado, 2018): Basically, for every given scenario there is a given action to do (in this case, the position side: long or short), and the users of said recommendations can either take the *bet* of following it or do not take it. The *sizing* part of the approach comes on identifying which bets are more prone to generate profits and assigning them a bigger size, and which are more susceptible to generate losses, and assigning as little size as possible, always in function on how big or small is the probability of the bet being correct.

On the traditional models, where the position side is the only prediction, it can be assumed that the size is always 1 (full position), however, if the profitable decisions could be differentiated from the decisions which are not at some extent, a bet-sizing approach can be executed: the position size will not be fixed at 1, it will depend on the probability of the analyzed position size being correct.

The bet-sizing optimization is implemented via labeling the decisions and training a machine learning model to make the classification. This technique called "Meta-Labeling" (Lopez de Prado, 2018) has many uses and allows using additional information to calibrate the classification. For example, if the situation for the model were a portfolio manager or credit officer decisions, the model could account for physiological factors, being mental state or rest levels of the person making the decision; or it could incorporate broad market information, context factors or any other component which could influence the result of a given decision. The meta-labeling technique applied allows for any kind of classification algorithm to perform the calibration: support vector machines, logistic regressions, neural networks, etc.

To conclude, the result of this second model will be a probability vector of classifying a given decision to be of a certain class. The class specification will be simplified as good or bad: a binary problem. The predicted probability can be directly translated into size, the close this probability is to 1, the more chance there is that it will generate profits, so the position to take will be bigger. The same reasoning can be applied to low probability bets, where the position to take will be small. The advantage of this approach is that the probability vector can be directly translated into a size vector. From there, the user only has to determine the maximum position he would take, and multiple the size on each bet to know how much money invest in each position.

1.4 Objective: Portfolio Management

The proposed objective is to consolidate all the statements discussed so far: the rationale behind forecasting on the bonds market, the traditional approach used to learn the side of the new trade to execute, and the integration approach assessing the size of the new position. The union requires a trading logic behind to exploit every model output and generate the linkage between the first model and the second model.

As explained on the previous section, two datasets will be utilized for the two integrated models: the first one will be used to forecast the future yield curves and identify maturities where the rate differs, and a second dataset on the actual asset on which the investments will be done to take advantage of the initial forecasts and optimize them via bet-sizing. The particular asset which will be used for the optimization is an ETF.

A stock ETF, or exchange-traded fund, is an asset that tracks a particular set of equities, similar to an index. It trades just as a normal stock would on an exchange, but unlike a mutual fund, prices adjust throughout the day rather than at market close. ETFs can track stocks in a single industry, such as energy, or an entire index of equities like the S&P 500. Additionally, ETFs can use specific strategies (i.e. Momentum, which is looking to buy securities on the rise and sell them when their price are around their peaks), or follow non-equity related indexes. In this particular case, the ETF will follow an index which price varies depending on how the yield curve changes its form over time.

The logic behind the models will be first identifying when the yield curve is steepening or when is flattening, and then exploiting the same through said ETF. A steepening (meaning that the difference between the long term and the short term yields is increasing on a given segment of time) will be reflected on the ETF as a raise in the price, therefore, a long position will be opened. If, by the contrary, the yield curve is flattening (the difference between the maturities is decreasing), a short position will be created. (Fabozzi, 2012) With these criteria, we can extract the decision vector, indicating the position sides, from the first model estimations.

Once the side has been calculated, a random forest model will be trained using the resulting decision vector and the ETF market information to differentiate those decisions which generated a loss from those which generated profits. Both models will be trained on a one-step-forward basis (which means that for each prediction performed, it is considered that all past information is known and certain). This second step, using a pure machine learning model, will yield as a result the position size.

Once both models are trained and compared on performance against the test data, the integrated model will be completed, serving the purpose of the portfolio management objective: for each day on live trading the integrated models will return a position side, and a position size associated to the probability of that recommendation being profitable. The additions this model has compared to previous works on the subject are on two main points:

- An evaluation of the position size is implemented, instead of forecasting just the action to take on a given scenario.
- Strategies are implemented into the model, through the criteria around steepening/flattening of the yield curve.

Before running the model on production (that is, acting of live feeds from the market), a usual practice done on finance is to run backtests to see how the two integrated models would have behaved on given periods. Backtests themselves are simulations of trading strategies that use historical data to generate results and analyze risk and profitability before risking any actual capital. The usual method is to leave out a subset of data, which the model has never seen and is called out-of-sample, and which is located on the time axis after the training data.

There is one variant to use strictly historical data for backtests, and it is called purged k fold cross validation (Lopez de Prado, 2018). The main limitation to use cross validation on time series datasets is that the observations themselves are correlated over time, so if the dataset is divided over 4 folds, and from fold 1 to fold 4 the time axis advances, upon selecting fold 1, 2 o 3 for validation purpose, there will be folds assigned to the training data that will be located on the future of the data left for validation.

As such, the purged k fold cross validation method consists on purging the correlation between folds to eliminate any bias by eliminating the necessary observations between the validation fold and previous training folds. Using this method is actually a complement to also employing the out-of-sample validation, adding more instances to perform backtests as well as hyper-parameter tuning.

It is to be noted that backtests are hypothetical and by no means an experiment, and as such, a backtest resulting in positive feedback does not guarantee good performance on reality, because of factors like survivorship bias, look-ahead bias, outliers, between others. The backtests performed on the data will compare Sharpe Ratios and Return Ratios over different periods, on three possible configurations:

1. Following a Buy & Hold strategy: buying the security at the start of the evaluation period and selling it in the end.
2. Using only the first model: side prediction. At each point where a trade is executed, the same size will be employed: 1. (full position)
3. Using the two models: both sides and sizes sprediction. Long and short positions will be weighted based on the size assigned by the second model.

2. The First Model

2.1 The Yield Curve Dataset

The dataset for the first model is sourced from the U.S. Department of the Treasury page[1]. It covers information on several rates (real yield rates, bill rates, historical treasury rates, among others) from 1990 to today's date, but on the realization of this study, the cut-off date was January 2nd, 2020. The selection was "Daily Treasury Yield Curve Rates" for all periods.

Regarding structure, it contains one row for each business date (so it excludes holidays and weekends), stating the yield rate at each of the following maturities for said date:

- Date (Index)
- 1 Month
- 2 Months
- 3 Months
- 6 Months
- 1 Year
- 2 Years
- 3 Years
- 5 Years
- 7 Years
- 10 Years
- 20 Years
- 30 Years

The data is not continuous along with the whole dataset. For example from 1990 to August 2001, there is no data on the 1-month maturity. 2-months maturity information only becomes available after October 2018. Finally, from 2002 to 2006 there is no information on the 30-year maturity.

[1] Data Source: https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=yieldAll

Therefore, for any two dates selected, the curve comparison like figure n°1 between them can be visualized. On the below figure, the horizontal axis shows maturity in months, while the vertical axis shows the yield (in nominal percentages) at that point on time. This would be and historical analysis, but in this situation, it can be seen that the red curve (which is the future curve, from a standing point on the blue line) is flattening. The general advice would be to set short positions on the longer maturities of the curve.

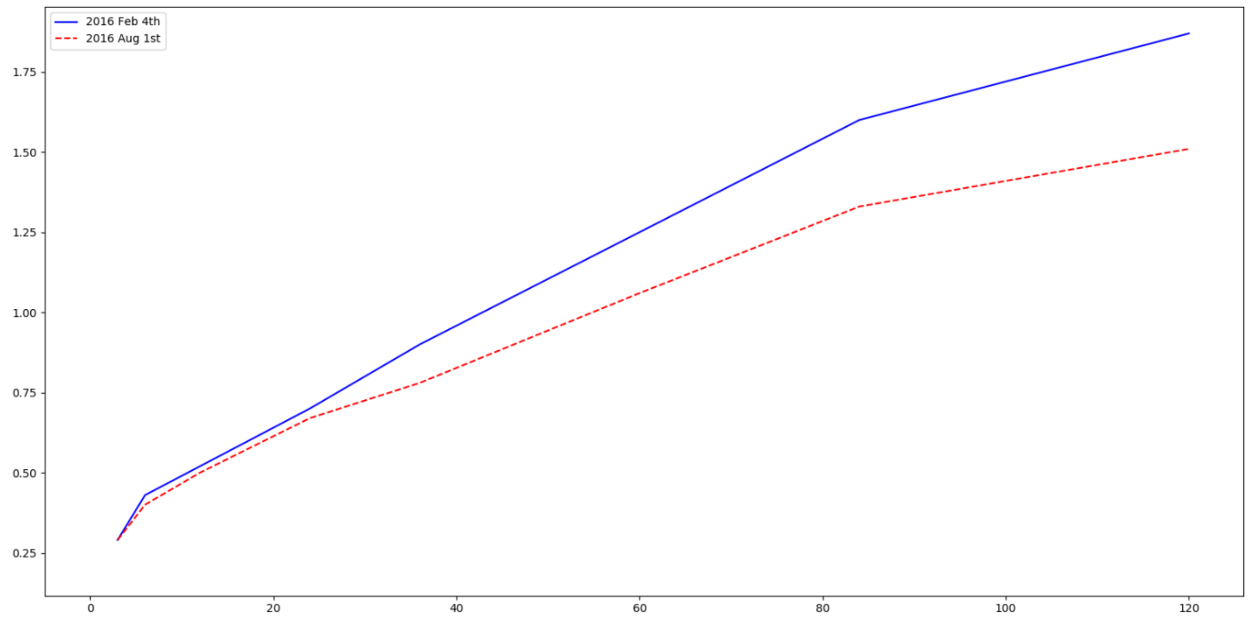

Figure n°2: Plot of the yield curves as of February 4th, 2016; and August 1st, 2016. The horizontal axis shows the maturity in months from 1 month to 10 years (120 months), while the vertical axis shows the yield rate, which covers a range between 0.25% and 1.75%.

In the current state, the dataset cannot be directly utilized in a model, as every yield for a given maturity could be the objective of the following model. If the estimation were to be performed on a single spread (for example, the spread between the 5y yield and the 1y yield), the estimation would cover several data points, but the scope for the second model would be drastically reduced to only those maturities.

As the objective is to forecast every single yield per maturity and be able to choose any desired maturity, one alternative would be to create a specific model for each maturity. However, in that case, the rate structure wouldn't be exploited as each maturity would be forecasted independently of the rest. The missing data points discussed earlier would also influence the outcome of this approach.

The alternative approach, as discussed in the introduction, will be to approximate the yield curve using an equation to describe the behavior of the yields over different maturities and subsequently, being able to recreate any curve required. By identifying the components and making a forecast on them, the model will be flexible enough to predict any maturity while not requiring a model for each maturity. However, this approach was meant to be used on zero-coupon yields (Nelson and Siegel, 1987), so an additional step (transforming the yields into an acceptable form) will be required.

2.2 Data Engineering: Zero-Coupon yields for longer term maturities

Three financial assets that participate in the source of the original dataset to build the yields which are being shown. The treasury bills, which are for maturities on the very short term (less than a year) and do not have any intermediate payments until maturity (coupons); the treasure notes, which cover maturities from 1 year to 10 years (not including the 10 years); and the treasury bonds, which cover maturities from 10 years onwards. Those last two possess coupons, and their existence and influence on the yield rate will create distortions on the construction of the forecasted yield curves via using the Nelson-Siegel model.

To address this issue, there is a technique called "*bootstrapping*" that will accomplish the objective of homogenizing the yield rates. The information required for the method is the prices of the shorter term securities (which do not possess coupons) and the price and coupon rates for the longer maturities whose rates are meant to be calculated. If the yield for the shorter maturities is not known, it can be calculated from their price with this formula:

$$r_{maturity} = \left(\frac{100}{price} - 1\right) * \frac{12}{maturity}$$

Once the zero-coupon yield for the shorter maturities is obtained, to be able to calculate coupon rates for the longer maturities, the next step is to subtract from their price the present value of the cash flow from maturities shorter than the one intended to be calculated, to compute the present value. To exemplify, for the 3-year maturity yield:

$$\boldsymbol{PV(ZC_{3y})} = Actual\ Price - PV(ZC_{2y}) - PV(ZC_{1y})$$

Once the present value is computed, the last step is to calculate at which rate that present value would have to be placed to reach the actual price after 3 years:

$$PV\left(ZC_{3y}\right) * (1 + \boldsymbol{i})^3 = Actual\ Price$$

This process is designed to iteratively build the zero-coupon rates from lower to higher. If the last known zero-coupon rate is 1-year, from that information the 2-year can be built, from 1-year & 2-year information the 3-year maturity can be computed consequently, and so on.

On the present work, the zero-coupon data for maturities equal or longer than 1 year were imported from a dataset on Quandl[2] where they were already calculated. While the yields for 1, 2, 3, and 6 months come from the U.S. Department of the Treasury data, the maturities from 1 year to 30 years will be sourced off the Quandl data.

2.3 Data Engineering: Decomposing the Yield Curve

To build the Nelson-Siegel equation, the pre-processing step to be performed on the dataset is to compute three components. The specifications used will be the same ones which were stated by Diebold and Li (2005):

$$y_t(\tau) = \beta_{1t} + \beta_{2t}\left(\frac{1-e^{-\lambda_t \tau}}{\lambda_t \tau}\right) + \beta_{3t}\left(\frac{1-e^{-\lambda_t \tau}}{\lambda_t \tau} - e^{-\lambda_t \tau}\right).$$

1. The long term factor (β_1): **Level**

$$\boldsymbol{\beta_1 = (10\ year\ yield)}$$

2. The short term factor (β_2): **Slope**

$$\boldsymbol{\beta_2 = (10\ years\ yield) - (3\ months\ yield)}$$

3. The medium term factor (β_3): **Curvature**

$$\boldsymbol{\beta_3 = 2 * (2\ years\ yield) - (10\ years\ yield) - (3\ months\ yield)}$$

[2] Source: https://www.quandl.com/data/FED/SVENY-US-Treasury-Zero-Coupon-Yield-Curve

Additionally, there is one more parameter to define, which influences the component weight: **λt**. As this factor governs the exponential decay rate, low values will make the curve fit better to long maturities while high values will make the curve decay faster improving the fit of the curve to short maturities. It also governs where the medium-term factor (Curvature) will achieve its maximum. This factor will be set to 0.0609, which is the value where the loading on the medium-term factor is maximized. Once computed, the time series for the components can be visualized through the whole period (1990 – 2020 Jan). In Fig n°2 each component is visualized separately with their yield rate range but sharing the time axis. The long term factor had a consistent downtrend over time, from levels of 8% to near 2% on the present.

The short term factor is very volatile and happens to have negative values for some periods (around 2000, 2008, and recently in 2019). When the short term factor falls into negative terrain, the yield curve will become inverted, meaning that the higher yield rates are on short maturities in contrast with the long maturities. On both, the first two periods where the short term factor was consistently negative were the ones with a crisis: the .com crisis in 2000 and the subprime crisis in 2007. Finally, the medium-term factor is the most volatile out of the three, but it is consistently negative through the majority of the period.

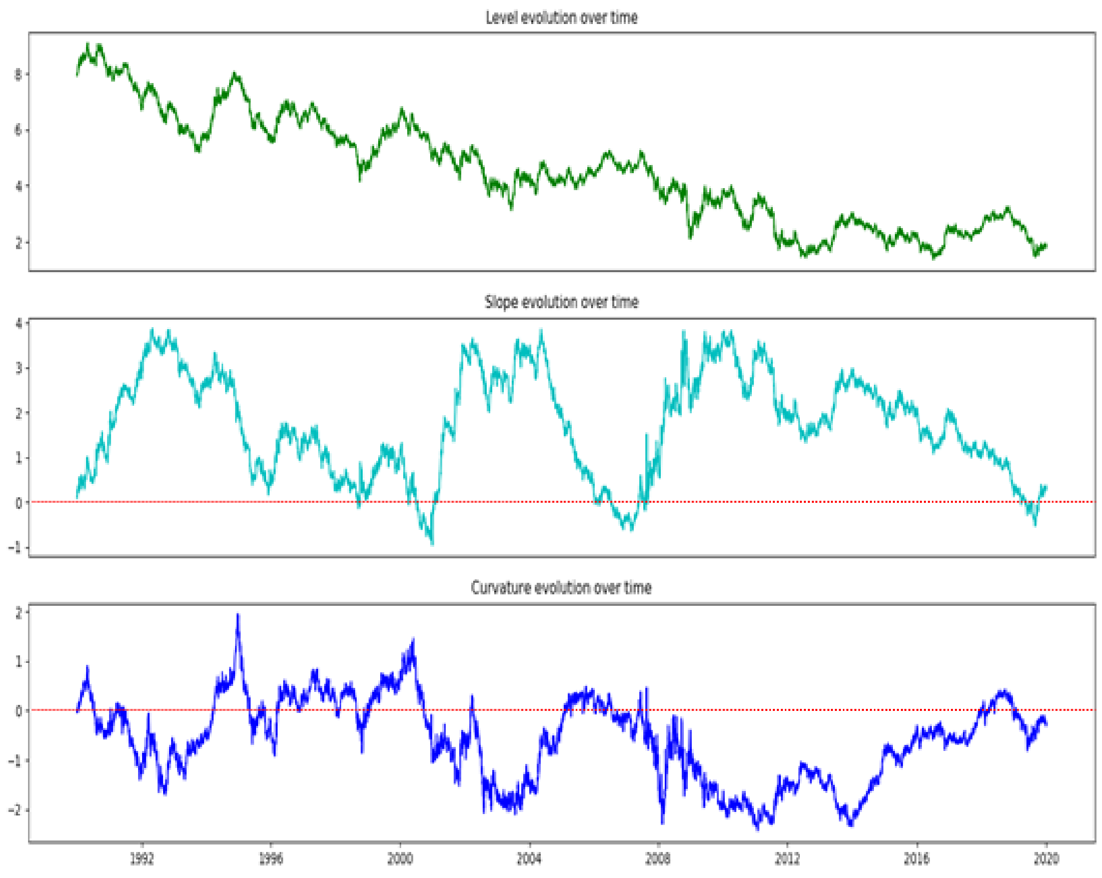

Fig n°3: Plot of the three components: level, slope and curvature across the whole period covered on the yield curve dataset.

2.4 Modelling: Forecasting the Nelson-Siegel equation

Before starting to forecast the curves, the comparison in Fig. n°3 serves as an example of the similarity between the actual yield curves and the Nelson Siegel curves.

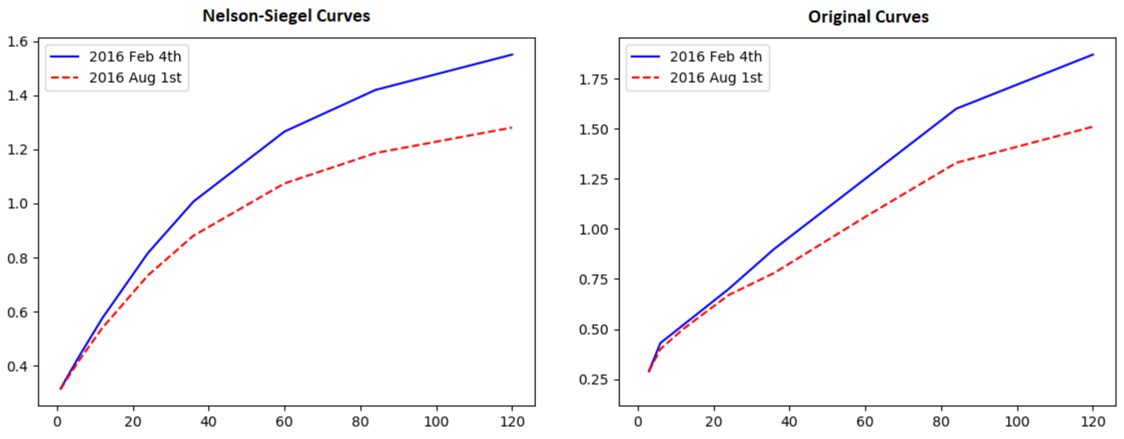

Fig n°4: Comparison between plots for yield curves from the Nelson Siegel equation vs the source data.

To forecast the future yield curve, every component will be forecasted using a simple AR (1) model on a *one-step-forward* basis for each row on the dataset. This means that each point in time, all the information available excluding future data is being used (i.e. for forecasting February 4th, 2016, the data used will be everything from 1990 until February 3rd. However, once going to predict February 5th, the real data for February 4th is incorporated into the training set).

Once the components ($\hat{\beta}_{i,t+1}$) have been computed, the curve on t+1 can be calculated for any given maturity **t,** and evaluate the results. An example of this is shown in fig n° 4.

$$\hat{y}_{t+1}(\boldsymbol{t}) = \hat{\beta}_{1,t+1} + \hat{\beta}_{2,t+1}\left(\frac{1- e^{-\lambda_t t}}{-\boldsymbol{\lambda_t t}}\right) + \hat{\beta}_{3,t+1}\left(\frac{1- e^{-\lambda_t t}}{-\boldsymbol{\lambda_t t}} - e^{-\boldsymbol{\lambda_t t}}\right).$$

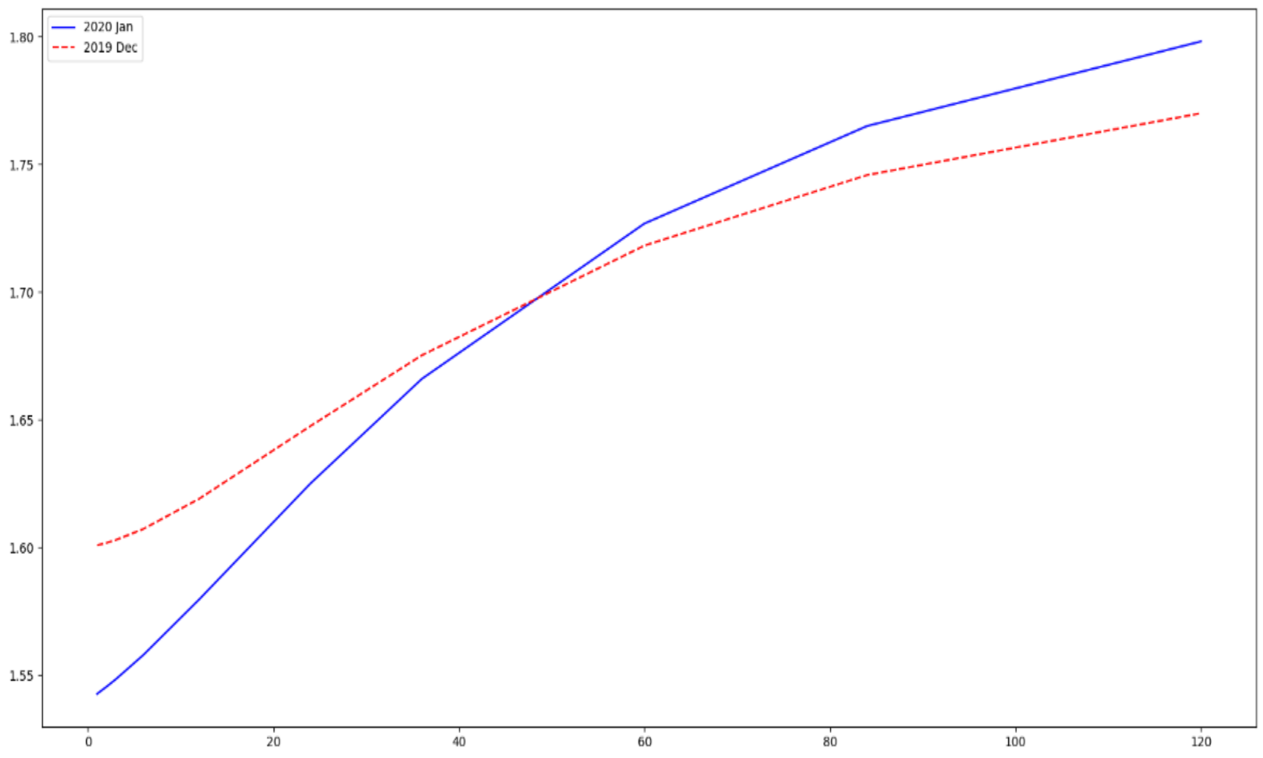

Fig n°5: Comparison between yield curves on the start of December 2019 and January 2020.

The three series did show to do not be stationary, (the probability distribution, and therefore, statistical moments like mean and variance vary over time) so differentiation of grade 1 was applied to generate the forecasts.

On the optimization side of the primary model, tests were run on diverse sets of ARIMA (p,d,q) with no meaningful advantage but generating exceedingly high processing times for the one-step-forward forecasts. Additionally, there are no antecedents of metrics to evaluate the curve consistency and accuracy as a whole, as the forecast accuracy must be measured on the yield per maturity, meaning that different models and approaches will concentrate on specific maturity horizons. In the present study, the focus will be on the 10 Year – 2 Year yield spread.

2.5 Results: Building the decision vector: Position side

The decision vector is determined out of calculating from the forecasted yield curves one-step-forward, the difference between the 10-year and 2-year yields (spread) for the present work. (However, any time horizon could be calculated using this method) Comparing the existing spread (out of the actual data) against the forecasted spread will determine if the prediction overall is that the curve is going to steepen (the spread is increasing) or flatten (the spread is decreasing).

The decision vector will be filled with one possible value out of two (-1, 1) for each date. A "1" will be labeled if the curve is steepening, meaning that the position side to take is a long one. Symmetrically, if the curve is flattening, a "-1" will be labeled to indicate a short position has to be assumed.

As it is for every model forecasts, the resulting vector won't have all correct decisions. If the model were to be applied to reality directly at this stage, a moderate size would need to be selected to appropriately consider the possibility of wrong decisions. However, as it was explained on the introduction, performing an adequate bet-sizing on the decision vector labels would allow to invest more capital on the bets which are more likely to yield profits.

The next model employed on this study will take over instrumenting the bet-sizing technique and optimizing the probabilities of each decision of being correct, to be able to translate those to position sizes afterwards. The more a given decision probability of generating profits approximates to 1, the bigger the position size for that decision will be. By the contrary, the closer a probability is to 0, the smaller the position size will be.

3. The Second Model

3.1 The ETF dataset

The ETF (iPath US Treasury Steepener ETN {STPP}) dataset was extracted using as source yahoo finance[3], which is a site that collects stock market data on all kinds of financial assets that are publicly traded. The *pandas_datareader* library was used to retrieve the data. The information is structured as a time series, available from 2010 until today, only for business days (same as the yield curve dataset), and it possesses the following fields for 2367 rows:

- Date (Index)
- High: The maximum price the ETF reached during the given date.
- Low: The minimum price the ETF reached during the given date.
- Open: The starting price of the ETF when the market opened.
- Close: The ending price of the ETF when the market closed.
- Adj. Close: The ending price of the ETF when the market closed after considering any corporate actions. (Like stock splits, dividends, or distributions)
- Signal: The field populated from the primary model, indicating if the position to take on this date is a long or a short one.

In Fig. n°5 the evolution of the price and the volume can be visualized. The figure correlates with the historical trend that the long term yield (10-year) has been decreasing and the curve has become flatter over time.

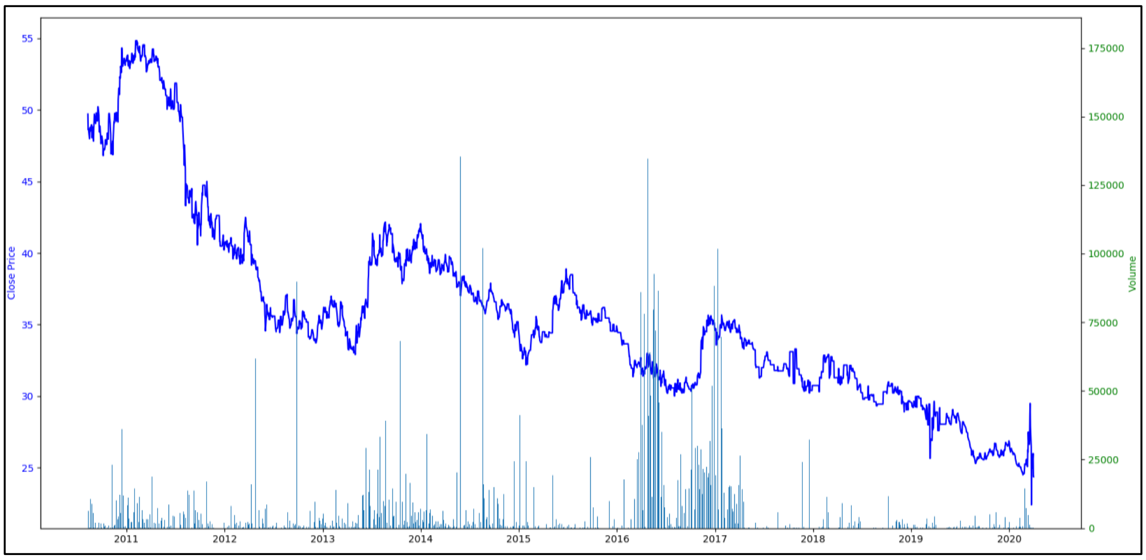

Fig n°6: Historical closing price (left axis) and volume (right axis) on the STPP across time, from 2010 to 2020.

[3] Source: https://finance.yahoo.com/quote/STPP/profile?p=STPP

3.2 Data Engineering: Generating the Meta-labels

The Meta-Label column will be created from verifying if the signals provided by the first model were good decisions or not. As the very last observation cannot be verified, this observation will be removed from the dataset.

Before computing the meta-labels, all observations in which a side prediction couldn't be computed (before the date had no information on the yield dataset) or specific days were market information for the ETF was missing were removed. Also, if the price didn't move, the decision was by default classified as correct.

After the calculation, the proportion of good decisions, performed by the first model following the above criteria, Fig. n°6 displays the actual proportions, and as it can be seen, there is a certain unbalance between correct and incorrect decisions. Problems like this must be taken into consideration when a classification model is being used, as it could generate distortions on the predictions.

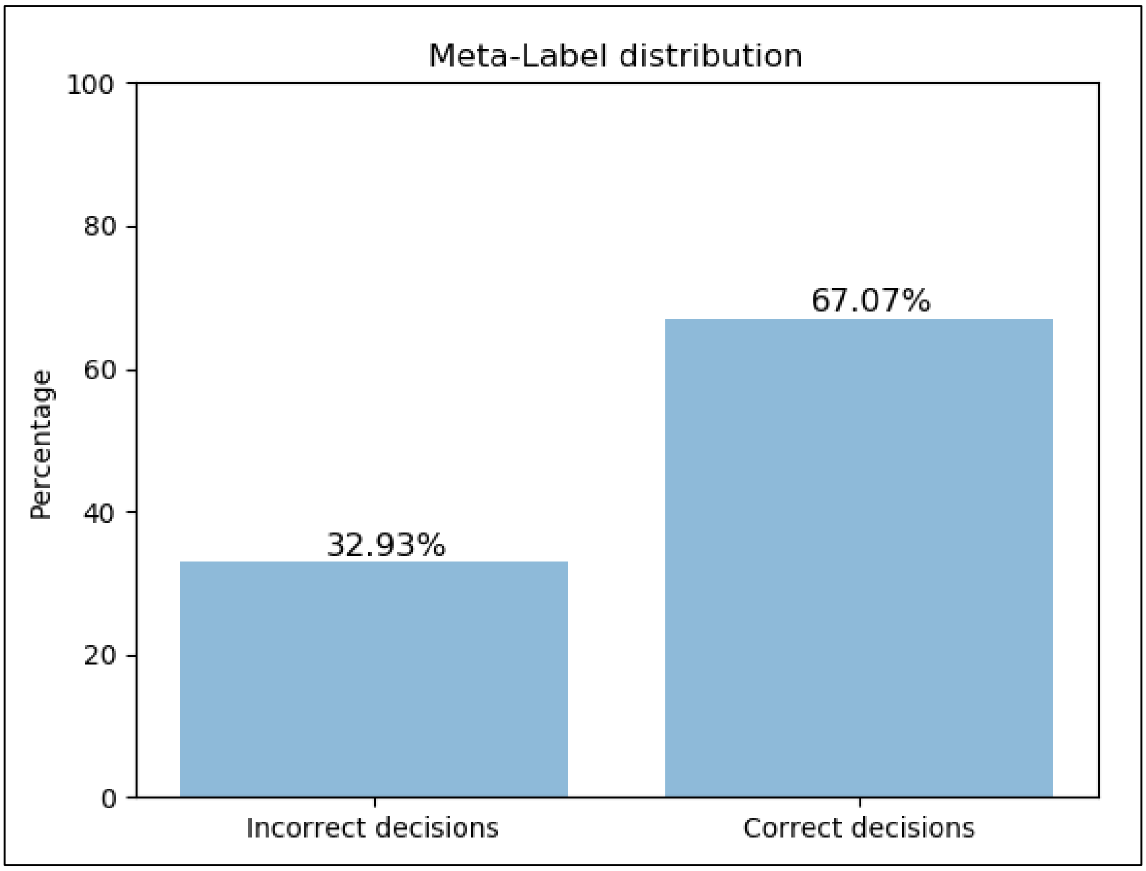

Fig n°7: Meta-Labels proportions after computing right and wrong decisions on the ETF dataset.

3.3 Data Engineering: Sample Weights

The purpose of the classification model is not only to bring and improvement on accuracy, but it is also to generate a profitable trend where the investment strategy applied can generate sustainable winnings and avoid big losses. While the algorithm accuracy will be fine-tuned on the optimization, the information intrinsic value is not the same at each data point and the model requires the specification to be able to weight differently the observations.

One dimension to weigh for is time. As the model is forecasting over a time series, it will predict new observations coming after the last date of training data, so then it is understandable that recent observations carry more importance than those in the past. The treatment for this "time decay weights" (Lopez de Prado, 2018) on the present study will be to assign a 0 weight to the older observation and progressively increase the weight until reaching the most recent observation, which will carry a 1 as weight.

The second dimension used is return-attributed weights (Lopez de Prado, 2018). The method has been adapted to work on a one-step-forward basis, so if at each time the return is computed and after all returns are computed, the one with the biggest absolute return has a return attributed weight of 1 while the one with minimal variation is labeled with a 0, the rest of the values will fall on the intermediate side of the spectrum [0, 1].

However, the fit function of the model will allow for only one vector of weights, so both time decay and return attribution weights must be combined into a simple measure of model weights. Adding them linearly is a bad option, it does not reflect the proper combination of them as on a case of very low return (say, weight = 0.1) but very near in time (say, weight = 0.9), the observation could receive the same weight as one observation with a moderate weight on both (0.5 on each).

To conclude, the combination criteria employed will be the product of the weights:

$$Model\ Weight = Time\ Decay\ Weight * Return\ Attribution\ Weight$$

That means that cases were one of the two factors is very close to 0, the observation importance for the model will be very low. Another advantage of this approach is that we keep the dimension [0, 1] along the whole weight vector.

3.4 Modelling: Random Forest

Random forest is a supervised learning algorithm that builds multiple decision trees and merges them (also known as ensemble models). As the name indicates, it is a collection of multiple regression trees where each tree is generated based on the values of an independent set of random vectors (Tan, Steinbach, and Kumar, 2006). The model will be trained to predict the probability of a given decision being a good one (1).

This ML algorithm has several parameters (called hyper-parameters) that can be fine-tuned to adjust better to the data and generate better predictions (both in-sample and out of sample). Each hyper-parameter is listed below:

- n_estimators: Represents the number of trees in the forest. Usually, the higher the number of trees the better to learn the data, but the more the processing time increases.
- max_depth: Indicates how deep a tree can be. The deeper the tree, the more splits it has, and it captures more information about the data. But deeper trees can lead to overfitting, which means over learning from the dataset, considering the noise. Overfitting tends to have high variance and low bias.
- min_samples_split: The minimum number of samples required to split an internal node. This can vary from considering at least one sample at each node to considering all of the samples at each node. By increasing this parameter, the forest is more constrained, and the deepness is limited.
- min_samples_leaf: The minimum number of samples required to be at a leaf node. Like min_samples_split, increasing the value of this parameter can cause underfitting, which can be defined as the failure to capture the underlying pattern of the data. An under-fitted model, in contrary to overfitting, has low variance and high bias.
- max_features: Represents the number of features to consider when looking for the best split.
- min_impurity_decrease: A node will be split if the split indicated a decrease of the impurity greater or equal to this value.
- min_weight_fraction_leaf: Defined as the minimum weighted fraction of the sum of weights required to be a leaf node.

3.5 Optimization: Cross validation with Embargo

To perform the fine-tuning (hyper-parameter value selection), all machine learning models must be subjected to cross-validation tests. At each test, there is a training set and a validation subset (both partitions of the original dataset), and the model is trained to take all the information from the training set. After training, the model receives as input all the independent variables (features) of the validation set and makes its predictions for the dependent variable. Consequently, the predictions are compared against the real answers to measure the accuracy of the model.

There are several methods of applying cross-validation, but in this study, the K-Fold cross validation will be performed. Following the logic of the previous explanation, the dataset is first divided between the cross-validation dataset and the test set. This "test set" will never be shown to the model until the very end, as it will measure the final accuracy after tuning the hyper parameters.

The cross-validation dataset is the one that will be used to perform the K-Fold method. First, a number (n) of folds must be defined; second, the dataset is divided by 1/n folds. For the first split, the validation set will be fold 1, while fold 2, 3, 4 & 5 will be the training set. Then the second split will be only fold 2 of the dataset, while the training set will be composed of folds 1, 3, 4, & 5. Fig n°7 summarizes the idea:

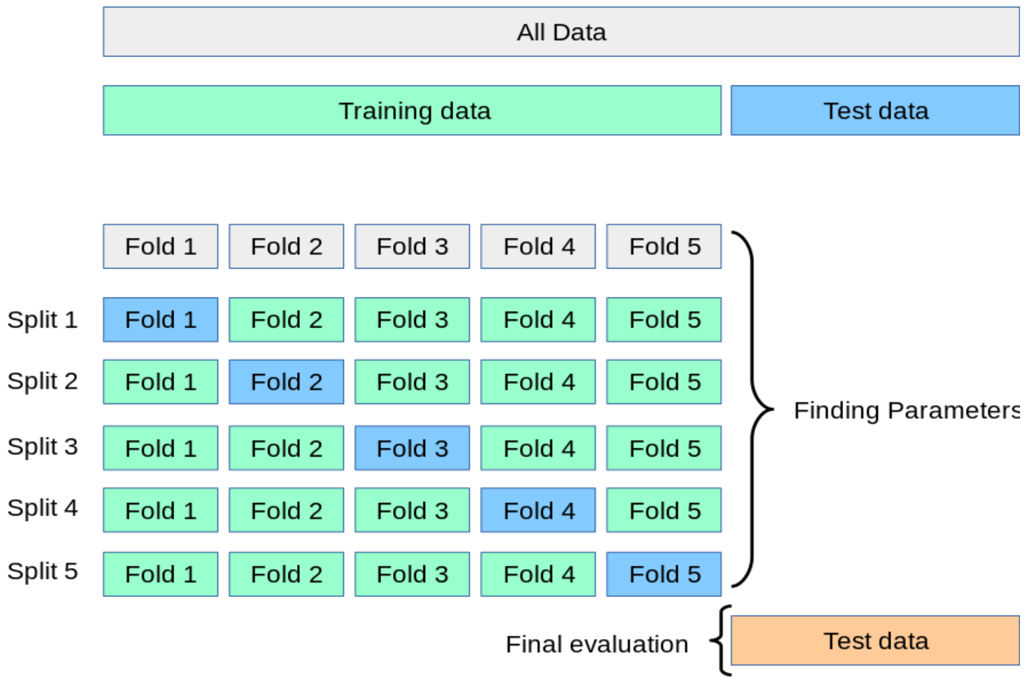

Fig n°8: Meta-Labels proportions after computing right and wrong decisions on the ETF dataset.

However, for the dataset employed in this study, there is one limitation to using kfold cross-validation: when the validation set is different from the last fold, the training set has observations on the future of the validation set. This issue can lead to Data Leakage issues.

Data Leakage happens when data related to the set to predict is directly fed into the training set. On this particular dataset, the leakage happens through the correlation along with the time series: By giving the model a data point next to the one it will predict, it is accessing information that drastically increases the accuracy, but via overfitting. (Over adjustment to the training set).

Therefore, a technique called Embargo (Lopez de Prado, 2018) will be applied to the folds: some of the observations of the training set which are next to the validation set will be removed. The number of observations excluded will be functional to the correlation level of the validation set with said observations.

On the same line, Bergmeir, Hyndman, and Koo (2015) argue the importance of employing cross validation procedures when the models are not heavily misspecified and to not only refer to out of sample testing procedures. In a theoretical proof, they showed that a normal K-fold cross-validation procedure can be used if the lag of the models is adequately specified, while in the experiments they performed, they showed empirically that even if the lag structure is not correct, as long as the data are fitted well by the model, cross-validation is a valid choice.

For the particular case of this study, 30% of the data was assigned to the test set, and from the remaining 70%, 4 folds were generated to cross-validate the data (each one accounting for 25% of the dataset). The embargo applied was for 5 business days (1 week).

3.6 Results: Classifying the labels and setting position size

After tuning the random forest model, the optimal parameters found and utilized on the final model were:

- n_estimators: 1000
- max_depth: 8
- min_samples_split: 40
- min_samples_leaf: 20
- max_features: 5
- min_impurity_decrease: 0.
- min_weight_fraction_leaf: 0.

The accuracy the model reached, measured by the f1-score, was 73% for the test set, which means an improvement of roughly 6% over the benchmark. Fig N°8 & 9 show the ROC curve and the Confusion matrix respectively.

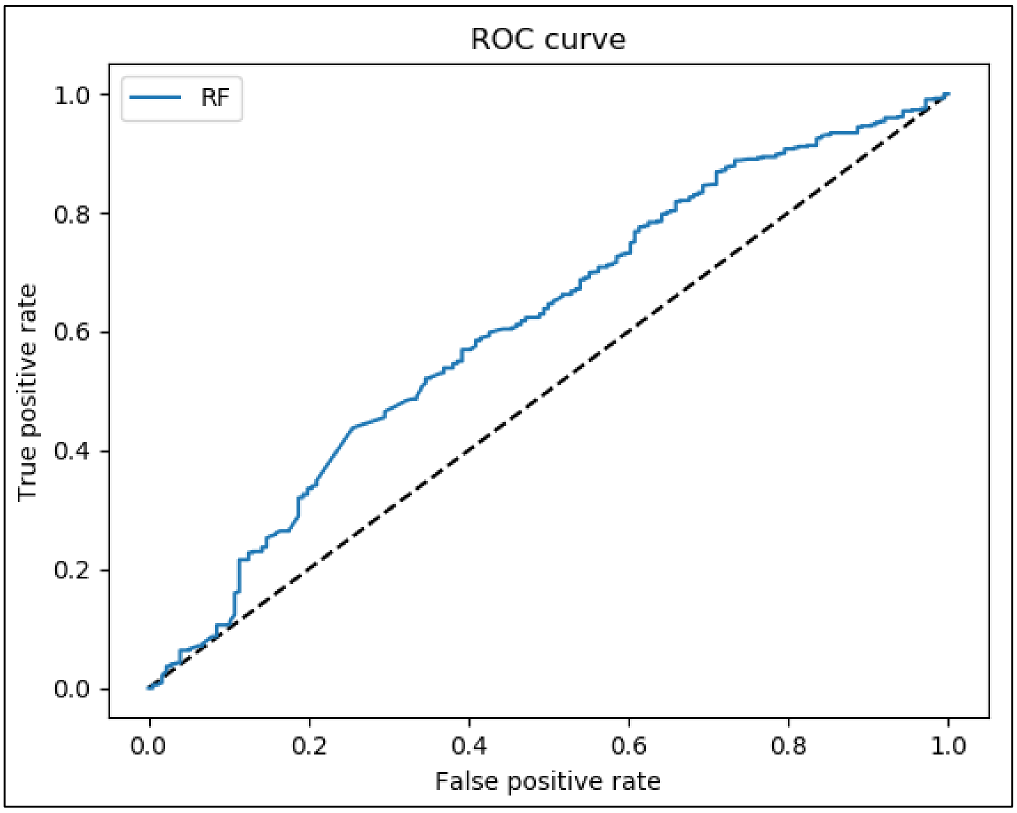

Fig n°9: ROC curve, built with the false positive rate on the horizontal axis, and the true positive rate on the vertical axis.

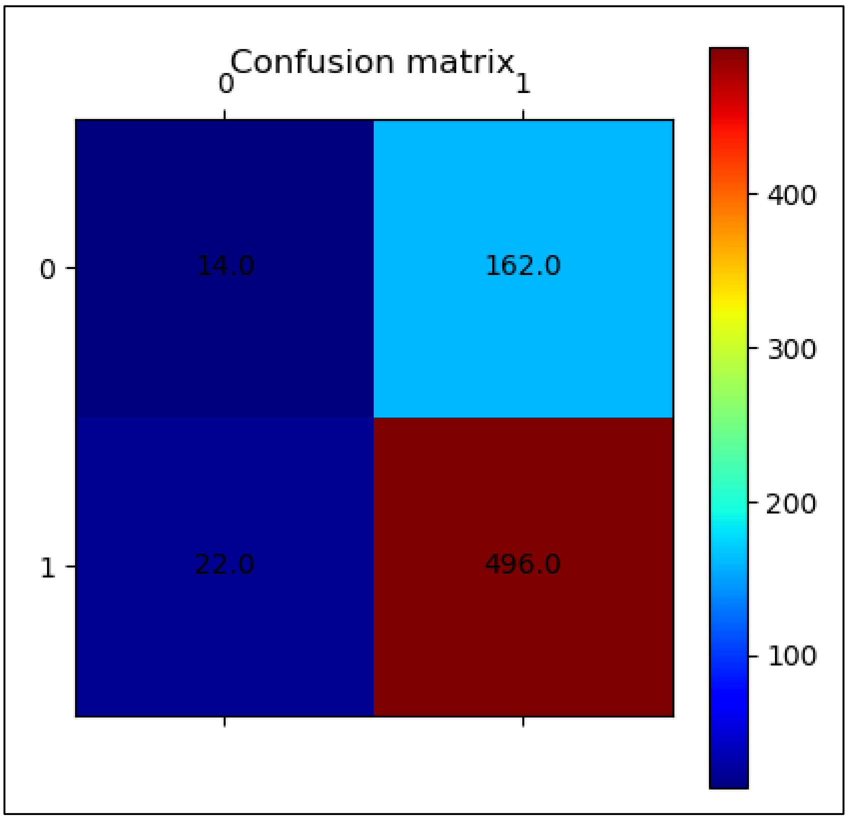

Fig n°10: Confusion matrix, the vertical options are the predicted ones, and the horizontal mean the actual labels. The diagonal therefore are the correctly predicted labels.

The 73% accuracy may seem low, but the context on the finance applications (and specifically, to the stock market) does provide background: the models will operate over small edges and the model has been trained (through the sample weights) to mainly focus in big recent bets.

4. Conclusions

4.1 Back testing

While the model accuracy, presented in the previous section, is an important metric, the key aspect is to corroborate is if the sizes predicted to produce a profitable strategy over time. To address this point, a back test will be performed over the two models.

Backtesting is the general method for seeing how well a strategy or model would have done ex-post. Backtesting assesses the viability of a trading strategy by discovering how it would play out using historical data. If backtesting works, traders and analysts may have the confidence to employ it going forward. The metrics selected for backtesting were the following:

- Return over the whole period: The cumulative sum of returns of the trades weighted by their size.
- Quantity of trades: Number of trades executed.
- The ratio of Longs: Proportion of the trades which were executed as long positions.
- Sharpe Ratio: A measure of the profit by risk assumed. The higher the value, the higher the return is being achieved after taking into account the risk the trade is exposed to. Mathematically is defined as (Total Return – Risk-Free Rate) / Returns Standard Deviation.

The scope for the testing will be three possibilities: the buy and hold strategy, where the comparison is made between the opening price and the closing price of the testing period; the only-size strategy, where the only application is the signal output from the first model, and all positions have size = 1; and finally the side and size model, where each position has a side determined by the first model, and the size the second model provided as output.

Two kinds of backtests were performed: one via cross-validation on the training data (similar to what was used on the hyper-parameter optimization) and one via the out-of-sample data. The cross-validation backtests, performed on four different folds, training the model only with the data of the remaining folds (no data from validation nor test sets), and applying embargo for the intermediate folds.

Table 1 summarizes the results for the cross-validation backtests. As can be seen, returns are usually higher on the only side strategy, as we are exposing completely to each bet the first model assigns. However, to adequately compare results, risk must be taken into account for comparability: Sharpe Ratio. The formula for this ratio is:

$$Sharpe\ Ratio = \frac{Asset\ Return - Risk\ Free\ Return}{Standard\ Deviation\ of\ Asset\ Returns}$$

The Sharpe ratio proves to be worse on every fold but the last one. The Sharpe ratio magnitude is significantly high on several folds because of the low value of the denominator (standard deviation of asset returns) on the Sharpe ratio computations.

	Buy and Hold	Only Side		Side & Size		
Fold	**Return**	**Return**	**Sharpe Ratio**	**Return**	**Sharpe Ratio**	**Spread on SR**
0	-20,71%	65,04%	41,731	47,21%	40,674	-1,057
1	3,68%	53,37%	35,414	31,96%	31,150	-4,264
2	-6,03%	45,50%	39,200	30,57%	36,960	-2,240
3	-7,66%	2,21%	1,623	2,74%	3,526	1,903

Table 1: Cross-validation backtest results per strategy and fold.

As it can be seen on table 2, the conclusion from the cross-validation results is further supported as this data was never seen at any point by the machine learning algorithm, not on the training phase, nor the parameter optimization. It is to be noted that the developed model weights do influence as well the performance because the model was instructed to focus more on recent data than on older observations.

	Buy and Hold	Only Side		Side & Size		
Fold	**Return**	**Return**	**Sharpe Ratio**	**Return**	**Sharpe Ratio**	**Spread on SR**
TEST	-23,51%	61,33%	38,442	43,79%	38,748	0,306

Table 2: Out of sample backtest results per strategy and fold.

An important item to consider is the influence that number of observations of the dataset after splitting the dataset (both for hyper-parameter optimization as well as for backtesting performance) has on both backtesting and parameter calibration. Machine learning models greatly depend on data availability, for stock market models it would be optimal to use data at the highest granularity possible (for example every 15 minutes refresh). In the case of this study, the same was limited by the yield curve dataset most granular frequency (daily), as yield curve rates are not computed on an intra-daily basis.

4.2 Generalization and applications outside the treasuries market

The complete portfolio management model built and evaluated in this study can be generalized to other environments as it is flexible enough to forecast other kinds of situations. However, there are changes to be made on the sources of data (side-selection dataset and size-selection dataset) and the way the models get connected to each other. On this section three environments will be examined for generalization:

I. Fixed-income portfolios, for bonds issued by another country, or by private companies.
II. Financial assets different that fixed-income related, for example on single equities, ETFs, commodities, etc.
III. Decision-based environments outside the stock market. For example credit score effectiveness.

For the fixed-income portfolios generalization case, to extend this model to bonds of another issuer but still public, the sources of data must be changed: first, the yield curve dataset now needs to be functional to the issuer meant to be forecasted, and second, the financial asset selected to exploit the yield curve changes over time must get directly impacted by changes on the curve. Second, the relation may be the same or not, if steepening/flattening is the approach to take advantage of movements on the curve, then there is no need for additional changes to what has been explained in this study.

If now the focus changes to bonds issued by private companies (i.e. corporate bonds), the main difference which will be found is to appropriately build a representative yield curve. Several considerations are necessary: the bonds on the yield curve should be of the same quality, the bonds used should cover a wide spectrum of maturities, but also should be directly aligned with the asset (e.g. ETFs, mutual funds, etc.) on which investments will be made and size will be computed by the second model.

The proposed integrated model can also be generalized to another kind of asset outside the fixed-income assets. The source of data for the size-selection model will be straightforward in this case: the historical market data of the asset determined for investment (as stated before, this could be a single equity, an ETF, or a commodity). Now, for the side-selection model data, there are several options, but one possible with the least differences possible to the case analyzed in this study is to use an index or a measure of performance of a given industry and identify rise or decrease trends and therefore long/short trades opportunities.

The other possibility is to use the same dataset for both side-selection and size-selection, but in this situation, the relation between models must be modified, and therefore the strategy to be applied to identify trade opportunities. In this category, momentum strategies would work, calculating from historical market data different kinds of moving averages (50-days and 200-days are the usual ones), and based on how they cross over time identify long/short signals.

Aside from changing the data source for the side-prediction model, the model itself could be entirely replaced by an investments expert recommendations, for example, a portfolio manager decisions: this dataset would contain on which moments the professional stated a trade was or should have been made and its direction (long or short). From there, using the financial asset historical data, the second model can be calibrated to identify the good decisions from the bad ones and effectively assign an optimum size to each of the portfolio management decisions.

To conclude this section, the third possible generalization field is decision-based environments, for example, credit risk management and score effectiveness. On several investment banks, the scoring procedure has multiple stages, and several credit offices study clients from different perspectives and using different metrics, causing the score to change from stage to stage (For example, initial grades, investment grades, client-obligor grades).

As it was while classifying the portfolio management decisions mentioned before, now the side-selection model is entirely performed by the credit officers, and the size-selection model must be trained to recognize which score is consistent and may endure over the next stages. Once trained, the model could allow to automate and approve the scoring of entities when the probability of being correct (size-prediction) is high enough to reduce excessive reviewing or identify the patterns behind drastic score changes, while leaving low probability scoring to manual processing.

4.3 Conclusions

The purpose of this study was to take the traditional approach of forecasting the position sides, that is when to execute a trade (and which type: long or short) and from there optimize its predictions by assigning them a probability of being correct, what is later traduced to position size. On top of fine-tuning traditional models, the case studied also adds value by linking the model itself to investment strategies (exploiting the evolution of the yield curves over time).

The study demonstrates the potential behind decision classification models and how it can be, in a relatively simple way, integrated with traditional models and theory to improve the strategies used. The accuracy increment of about 6% and the Sharpe Ratios increasing consistently over the most recent periods are solid measures of the improvement this kind of ensemble models can bring out to organizations to become more competitive.

Additionally, the model has been trained and tested using only public data and limited available data points, expanding this to bigger datasets will allow for even better-fine-tuned models and more consistent results. Even one more percent of improvement in accuracy can significantly augment the potential of generating profits.

Even though the scope of the study has been very narrow, the generalization to other areas within finance has been explored, and the proposed integrated model is flexible enough, both to taking quantitative information (like stock market data) and qualitative information (i.e., psychological data), and is specifically oriented to decision optimization with a focus on actual results.

As time passes, decision optimization will become more crucial to not only trading decisions on the stock market but to risk management and even outside the finance specific world. Having additional tools to validate or evaluate decisions will be an improvement on even non-profit oriented organizations. In addition to this, shorter-term horizons and increasingly volatile environments like the ones happening in the recent times will prove ideal to further expand the use of these optimization models.

5. References

- Alfaro, Becerra, Sagner, 2010. *"The Dynamic Nelson-Siegel model: empirical results for Chile and US"*
- Barnabás, Hollifield, 2018. *"Machine learning-aided modeling of fixed income instruments"*
- Bergmeir, Hyndman, Koo, 2015. *"A note on the validity of cross-validation for evaluating time series prediction"*
- Box, Jenkins, 1991. *"Time series analysis, forecasting and control"*
- Bianchi, Büchner, Tamoni, 2020. *"Bond Risk Premia with Machine Learning"*
- Breiman, 2001. *"Random forests."*
- Bruche, 2007. *"Estimating Structural Models of Corporate Bond Prices"*
- Castellani, Augusto dos Santos, 2006. *"Forecasting long-term government bond yields: an application of statistical and ai models. Technical report, ISEG–Departamento de Economia."*
- Colin, 2006. *"Fixed income attribution with minimum raw material. Journal of Performance Measurement"*
- De Pooter, 2007. *"Examining the Nelson-Siegel Class of Term Structure Models"*
- Diebold, Li, 2005. "Forecasting the term structure of government bond yields"
- Distenfeld, DiMaggio, Skoglund, Swtizer, 2018. *"The future of fixed income. How technology will revolutionize asset management"*

- Duffee, Stanton, 2004. *"Estimation of Dynamic Term Structure Models"*
- Ericsson, Reneby, 2005. *"Estimating Structural Bond Pricing Models"*
- Fabozzi, 2007. *"Fixed Income Analysis"*
- Fabozzi, 2012. *"The handbook of fixed income securities"*
- Ganguli, Dunnmon, 2017. *"Machine learning for better models for predicting bond prices."*
- Jotaki, Yamashita, Takahashi, 2017. *"Predicting corporate bond prices in japan using a support vector machine."*
- Kaufman, Rosset, Perlich, 2011. "*Leakage in Data Mining: Formulation, Detection and Avoidance*"
- Lee, Ingram, 1991. *"Simulation estimation of time-series models"*
- Lopez de Prado, 2018*. "Advances in Financial Machine Learning"*
- Nelson, Siegel, 1987. *"Parsimonious Modeling of Yield Curves"*
- Piazzesi, 2005. *"Handbook of Financial Econometrics"*
- Tan, Steinbach, and Kumar 2006. *"Introduction to Data Mining"*