Exploring Machine Learning Advances in Finance

Bachelor's degree thesis

Guillermo Creus Botella

Director:
Daniel P. Palomar (HKUST)

Tutor:
Jordi Castro Pérez (UPC)

Bachelor's degree in Mathematics
Bachelor's degree in Industrial Technology Engineering

December 2020

Exploring Machine Learning Advances in Finance

by Guillermo Creus Botella

Abstract

It is not a mystery that Machine Learning has revolutionized the way humans analyze data. However, when treating data as complex as the one found in the stock market, clear advances are yet to be accomplished. This work will be focused on three novel areas of Machine Learning applied to finance: meta-labeling, fractional differentiation and financial data parsing in the form of bars. Every aspect will be analyzed independently so as to determine its individual effectiveness.

Furthermore, as Machine Learning thrives in data, one should be very careful with the information provided to algorithms. That is why, whenever possible, the proposed solution will be tested with synthetic data, providing a controlled environment. When success is achieved under these conditions, the procedure will be put to use with real data, where these methods will be compared with standard techniques to ascertain if predictions deliver better forecasts or strategies with higher risk-adjusted returns.

Keywords: Machine Learning, quantitative finance, Meta-labeling, fractional differentiation, data parsing, time series analysis, feature engineering.

MSC Code: 91B84

Explorando avances del aprendizaje automático en las finanzas

por Guillermo Creus Botella

Resumen

No es un misterio que el aprendizaje automático ha revolucionado la forma en que los humanos analizamos datos. Sin embargo, al tratar datos tan complejos como los que se encuentran en el mercado de valores, aún no se han logrado avances claros. Este trabajo estará centrado en tres áreas novedosas del aprendizaje automático aplicado a las finanzas: Meta-labeling, diferenciación fraccional y *parsing* de datos en forma de barras. Cada aspecto será analizado de forma independiente para determinar su eficacia individual.

Además, dado que el aprendizaje automático se nutre de datos, se debe tener mucho cuidado con la información proporcionada a los algoritmos usados. Por eso, siempre que sea posible, la solución propuesta se probará con datos sintéticos, proporcionando un ambiente controlado. Cuando se logre el éxito en estas condiciones, se utilizará el procedimiento con datos reales, donde se compararán los métodos usados con técnicas estándar para determinar si las predicciones ofrecen mejores pronósticos o estrategias con mayores rendimientos ajustados al riesgo.

Palabras clave: Aprendizaje automático, matemática financiera, Meta-labeling, diferenciación fraccional, *parsing* de datos, análisis de series temporales, *feature engineering*.

Código MSC: 91B84

Explorant avenços d'aprenentatge automàtic a les finances

per Guillermo Creus Botella

Resum

No és un misteri que l'aprenentatge automàtic ha revolucionat la forma en que els humans analitzem dades. No obstant, a l'hora de tractar dades tan complexes com les trobades al mercat de valors, encara no s'han aconseguit avenços clars. Aquest treball estara centrat en tres àrees noves de l'aprenentatge automàtic aplicat a les finances: Meta-labeling, diferenciació fraccional i *parsing* de dades en forma de barres. Cada aspecte serà analitzat de forma independent per determinar la seva eficàcia individual.

A més, atès que l'aprenentatge automàtic es sustenta amb dades, s'ha d'anar amb molt de compte amb la informació proporcionada als algoritmes usats. Per això, sempre que sigui possible, la solució proposada es provarà amb dades sintètiques, proporcionant un ambient controlat. Un cop assolit l'èxit en aquestes condicions, s'utilitzarà el procediment amb dades reals, on es compararan els mètodes usats amb tècniques estàndard per determinar si les prediccions donen millors pronòstics o estratègies amb majors rendiments ajustats al risc.

Paraules clau: Aprenentatge automàtic, matemàtica financera, Meta-labeling, diferenciació fraccional, *parsing* de dades, anàlisi de sèries temporals, *feature engineering*.

Codi MSC: 91B84

Acknowledgements

First of all, I would like to thank Prof. Daniel P. Palomar for all of his help and guidance throughout the thesis. I sincerely hope this work will be the start of a fruitful relationship. Also, I am grateful for the work done by Jordi Castro, the link between Hong Kong and Barcelona.

Moreover, I want to express my gratitude to *Fundació Privada* Cellex and CFIS. Without their help I would certainly not be here. Special thanks to Toni Pascual, who is the man behind the curtain. Thank you for making everything possible. Additionally, I am really grateful of having crossed paths with Miguel Ángel Barja, a huge professional and excellent director. Thank you for your willingness to help students grow.

Finally, huge thanks to my family for their unconditional support throughout my education.

Guillermo Creus Botella
Barcelona
December 2020

Contents

Chapter 1

Introduction

Since the main goal of the thesis is to explore several advances of Machine Learning in Finance, three techniques firstly introduced by Marcos López de Prado [5] have been chosen. The methods are the following:

1. Meta-labeling
2. Fractional differentiation
3. Data parsing as bars

While meta-labeling is a finance-specific modification of binary classification, the last two methods deal mostly with data modification/pre-processing. Although the methods are different in essence, the objective is similar. That is, compare the methods to standard techniques so as to determine if they provide an edge to practitioners.

When dealing with financial data, where signal-to-noise ratio is low, establishing the validity of the previous statement is not clear. That is why the first two methods have been validated with synthetic data in what will be called "Toy Projects". These projects will be extremely important because they will provide a controlled environment where statistical parameters can be tuned to our liking, helping to determine the true performance of the proposed method.

1.1 Document overview

The work will be structured in the following chapters:

- Chapter 2 will introduce readers to essential financial concepts that will be extremely vital to understand the succeeding chapters.
- Chapter 3 explores the concept of Meta-labeling via a Toy Project to later apply it to financial data.
- Chapter 4 will present fractional differentiation of time series in order to obtain stationary time series without giving up memory. Again, before applying the method to financial time series, a Toy Project will be created.
- Chapter 5 will explain a novel way to sample high frequency data in finance. Also, it will illustrate how it can improve forecasts.
- Chapter 6 shows the impact these techniques could have in Finance, analyzes the standing of Machine Learning in Finance and reflects on future possibilities that could arise from this work.

Chapter 2

Primer in financial data

Although financial data comes in many shapes and forms, this work will be centered around the one that comes from stocks or indices (basket of financial instruments). To make matters easier, instead of thinking of financial instruments one can think about the underlying price time series.

That is why it is important to introduce the time series $\{p_t\}$, which is the price of the stock/index at the discrete time stamp t. Note that t will will depend on the sampling frequency used to gather data.

Frequency:

- Low Frequency Data (LF): Daily, Monthly, Quarterly.
- High Frequency Data (HF): Intraday (30 min., 5 min. ...)

Having introduced these concepts, it should be pointed out that modeling in finance is carried out with the natural logarithm of the price (log-prices) instead of regular prices. It will be referred to as $y_t := \log(p_t)$. To illustrate this, the simple model, yet widely used, of a random walk with drift will be introduced:

$$y_t = y_{t-1} + \mu + \epsilon_t$$

where $\epsilon_t \sim$ i.i.d. $N(0, \sigma^2)$

2.1 Asset returns

The next concept to introduce are the returns, which is a technique to normalize prices. To be specific, it allows comparison between different time series regardless of the price value. The two types that are going to be used are the linear and logarithmic returns.

- **Linear:** $R_t(1) \equiv R_t := \frac{p_t - p_{t-1}}{p_{t-1}} = \frac{p_t}{p_{t-1}} - 1$
- **Logarithmic:** $r_t(1) \equiv r_t := \log\left(\frac{p_t}{p_{t-1}}\right) = \log(p_t) - \log(p_{t-1}) = y_t - y_{t-1}$

Where log is the natural logarithm.

A few properties that can be highlighted:

1. $r_t = \log(1 + R_t)$
2. The Taylor series of $\log(1 + x) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{x^k}{k}$ yields $r_t \approx R_t$ whenever $R_t \approx 0$
3. Compounded linear returns:
   $R_t(k) := \frac{p_t - p_{t-k}}{p_{t-k}} = \frac{p_t}{p_{t-k}} - 1 = \frac{p_t}{p_{t-1}} \dots \frac{p_{t-(k-1)}}{p_{t-k}} - 1 = (1 + R_t) \cdot (1 + R_{t-1}) \dots (1 + R_{t-(k-1)}) - 1$

4. Compounded log-returns:
$r_t(k) := \log\left(\frac{p_t}{p_{t-k}}\right) = y_t - y_{t-k} = (y_t - y_{t-1}) + \ldots + (y_{t-(k-1)} - y_{t-k}) = r_t + \ldots + r_{t-(k-1)}$

With the purpose of displaying what has been defined, in figures 2.1 and 2.2 the log-returns and log-prices of the S&P500 have been plotted for the period that spans from 2015-01-01 to 2018-10-01.

Regarding the variation of returns, and hence the variation of prices, the term **volatility** comes into play. It will be denoted as $\sigma := \sqrt{\mathbb{E}[(r_t - \mathbb{E}[r_t])^2]}$, which is the standard deviation of log-returns.

Figure 2.1: Log-Returns of the S&P500

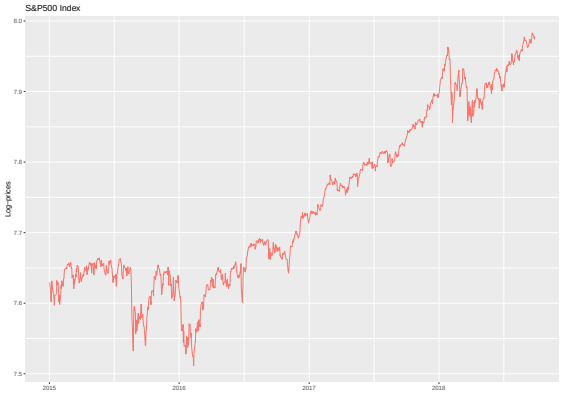


Figure 2.2: Log-Prices of the S&P500

2.2 Stylized facts

In an attempt to portray financial data without the need of a theoretical framework, Rama Cont presents financial data in [4]. To introduce empirical properties he uses stylized facts, which are broad generalizations that summarize data.

In other words, this section will serve as a brief introduction to properties of financial data which have been observed in various types of financial markets. However, bear in mind that this is a first approximation and that stylized facts are not by any means facts.

With that being said, the stylized facts relevant to this work presented in [4] are:

1. **Absence of autocorrelations:** linear autocorrelations of asset returns are insignificant, except for small intraday scales. See figure 2.3 for the autocorrelations of daily S&P500 log-returns.
2. **Heavy tails:** The pdf of returns is heavy-tailed. That is, its tails fall much slower than a Normal pdf, which falls exponentially to zero.
3. **Gain/loss asymmetry:** Extreme down movements are far more common than huge upswings.
4. **Aggregational Gaussianity:** The probability density function (pdf) of log-returns approaches the one of a Gaussian random variable as one increases the time scale over which returns are calculated (see figure 2.4).
5. **Intermittency:** Regardless of the time scale, returns display high variability.
6. **Volatility clustering:** As it can be seen in figure 2.1, high volatility (dispersion of log-returns) events tend to come in clusters.

2.3 Statistical properties

In figure 2.3 one can see the autocorrelations of daily log-returns. The definition, as in [12], of the log-returns autocorrelation of lag k is the following:

$$\rho_k = \frac{\text{Cov}(r_t, r_{t+k})}{\sqrt{\text{Var}[r_t]}\sqrt{\text{Var}[r_{t+k}]}}$$

Where $\text{Cov}(r_t, r_{t+k})$ is the covariance of r_t and r_{t+k} and $\text{Var}[r_t]$ the variance of r_t.

It can be seen that, by definition, $\rho_0 = 1$. Apart from that, the rest of lags are insignificant or barely surpass the significant threshold (dotted line). This confirms the first stylized fact presented in section 2.2.

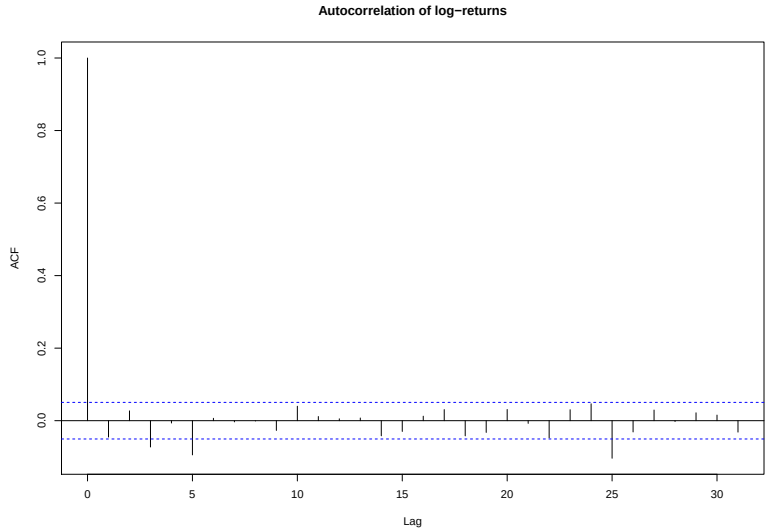


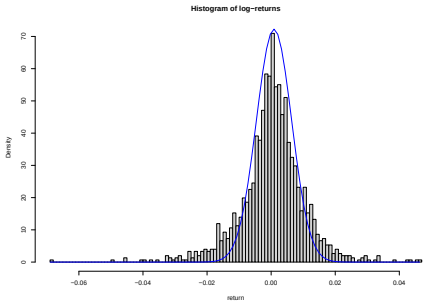
Figure 2.3: Autocorrelations of daily S&P500 log-returns

Figure 2.4 shows the histogram for daily, weekly and monthly log-returns of the S&P500. It has been introduced with the intention of illustrating stylized facts 2 and 4. It clearly shows the heavy tails and how it evolves towards a normal distribution as the returns time scale increases.

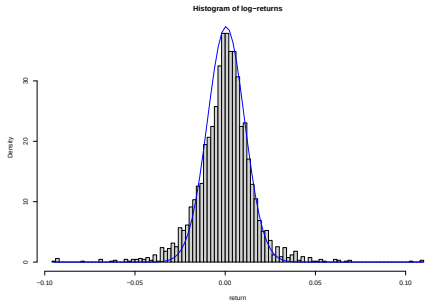
Lastly, figure 2.5 illustrates, via QQ (Quantile Quantile) plots, that log-returns do not follow a normal distribution. Beforehand, it is pertinent to introduce special notation for quantiles: given a r.v. X, a **quantile** q_α is a real number s.t. $P(X \leq q_\alpha) = \alpha$

That said, Quantile Quantile plots order the observations, and then map them to the graph such that the value of the x-axis is the equivalent quantile of $Z \sim N(0, 1)$, and the value of the y-axis is the quantile of the sample. That way, if one wants to test if $X \sim N(\mu, \sigma^2)$, then knowing that the quantiles would just be shifted and/or scaled, the corresponding QQ Plot should be a line.

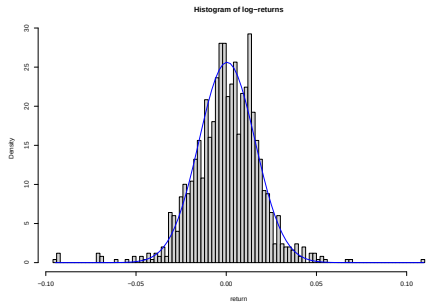
In the case of the S&P500 log-returns (see figure 2.5), the QQ Plots are not lines, implying that log-returns are not Normal. In addition, the curvature at the extremes is a clear sign of a heavy-tailed distribution.



(a) Daily log-returns

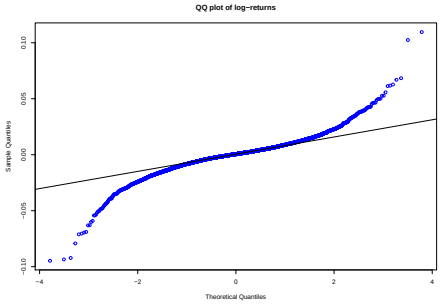


(b) Weekly log-returns

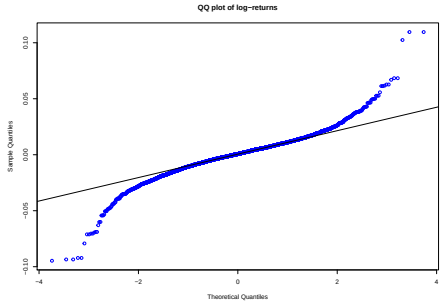


(c) Monthly log-returns

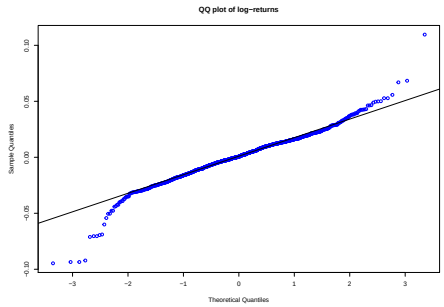
Figure 2.4: Pdf fitted to S&P500 log-returns



(a) Daily log-returns



(b) Weekly log-returns



(c) Monthly log-returns

Figure 2.5: QQ Plots of S&P500 log-returns

2.4 Portfolios

This section will introduce the concept of portfolios, which essentially means that one can allocate a certain budget to different financial products instead of buying a single asset. In this work, N will represent the number of securities and $\mathbf{w} \in \mathbb{R}^N$ the vector of weights assigned to each asset. Accordingly, the price of each stock considered in the portfolio will be noted as p_t^i, where $i \in \{1, \ \ldots, \ N\}$.

Regarding notation for portfolio returns it is useful to set the following terms:

- **Money available (budget):** B
- **Portfolio linear returns:** R_t^P
- **Portfolio log-returns:** r_t^P
- **Linear returns of asset** i: $R_t^i \longleftrightarrow \mathbf{R}_t = [R_t^1, \ \ldots, \ R_t^N]$
- **Log-returns of asset** i: $r_t^i \longleftrightarrow \mathbf{r}_t = [r_t^1, \ \ldots, \ r_t^N]$

Recognizing that the amount of money invested in asset i is simply Bw_i, R_t^P can be computed as:

$$R_t^P = \frac{\sum_{i=1}^N Bw_i \cdot (1 + R_t^i)}{B} - 1 = \sum_{i=1}^N (w_i + w_i \cdot R_t^i) - 1 = \mathbf{w} \cdot \mathbf{R}_t$$

$$r_t^P = \log(1 + R_t^P)$$

Where it has been used that:

- $\sum_{i=1}^N w_i = 1$ because in the cases that are going to be considered there is no leverage. This constraint just means that the sum of the money invested in each asset is B
- If Bw_i is the initial wealth of asset i at time $t-1$, then the final wealth is $Bw_i \cdot (1 + R_t^i) = Bw_i \cdot \frac{p_t^i}{p_{t-1}^i}$

2.5 Long and short positions

When opening a position it is really important to distinguish between long and short positions. A long position (or going long on some stock) is the most common way to invest since it just means that you buy an asset and you sell it at some point, expecting to earn a return.

On the contrary, if you short a stock, you first sell a stock that someone has lent you and then try to repurchase it at a lower price to return the stock to the lender. That way, if the stock goes down in price, you would earn a profit by selling high and buying low.

The following examples will attempt to clarify what was mentioned before:

Long position

Suppose the prices of stock **ABC** at times t_0 and $t_1 := t_0 + 1$ are $p_{t_0} = 100\$$ and $p_{t_1} = 115\$$ respectively. Then, the return is: $R_{t_1} = \frac{115}{100} - 1 = 0.15$. If at t_0 one buys 10 stocks of **ABC** at market value ($B = 10 \cdot p_{t_0} = 1000\$$), the end wealth at t_1 will be $B \cdot (1 + R_{t_1}) = 1150\$$. Thus, the final profit has been $150\$$.

Conversely, if the stock would have gone down in price at time t_1, money would have been lost. At this point, it is obvious that long positions will only be profitable if the stock price ends at a higher value than the one when the position was opened.

Short position

Following the example of the long position, suppose that at time t_0 one shorts 10 stocks of **ABC**. That means that 10 stocks will be lent to us to be sold immediately at market value, making one hold 1000$ in cash.

As the stock at time t_1 has gone up in value, one repurchases the shares at 1150$, realizing a 150$ loss. However, if at time t_1 the stock price would have been $p_{t_1} = 85$\$, then the stocks would have been repurchased at 850$, realizing a 150$ profit. From a practical standpoint, the end wealth of a short position can be calculated as $B \cdot (1 - R_{t_1})$ since:

$$R_{t_1}^{\text{short}} = \frac{\text{profits}}{p_{t_0}} = \frac{p_{t_0} - p_{t_1}}{p_{t_0}} = -R_{t_1}$$

where **profits** means the money you would earn if you shorted one stock (selling at p_{t_0} and buying at p_{t_1}) and $R_{t_1}^{\text{short}}$ the return realized at t_1 by shorting one stock at t_1.

As a final note, it should be pointed out that shorting a stock can lead to ever-growing losses. This is caused by the definition of shorting, which implies receiving stocks from someone who expects them back. So if a stock keeps going up in price, then the stocks will be repurchased at a price that will keep growing. On the other hand, if one goes long on a stock, the maximum capital that can be lost is the initial one, which would only happen if the stock price went to 0.

2.6 Financial Metrics

In this section, a series of financial metrics will be shown so as to compare results in the succeeding chapters:

- Sharpe Ratio
- Information Ratio
- Drawdown

Sharpe Ratio

The Sharpe ratio is extremely useful since it allows to compare assets independent of the returns and variability. This metric, as defined by William F. Sharpe in [14], was introduced with the concept of reward per unit of variability in mind:

$$\text{SR} = \frac{\mathbb{E}[R_t - r_f]}{\sqrt{\text{Var}[R_t - r_f]}} \tag{2.1}$$

The term r_f in equation 2.1 is the risk-free return, which is the rate of return you can achieve with zero risk. An example of this is the 1 month U.S. Treasury bills. As a side note, the term r_f was introduced because assets carrying risk must surpass r_f to encourage investors to buy them.

Information Ratio

Similarly to the Sharpe Ratio, the Information Ratio is useful to compare the returns achieved with a benchmark:

$$\text{IR} = \frac{\mathbb{E}[R_t - R_b]}{\sqrt{\text{Var}[R_t - R_b]}}$$

Where R_b are the returns of the benchmark, $\alpha := \mathbb{E}[R_t - R_b]$ is the excess return, and $\sqrt{\text{Var}[R_t - R_b]}$ is one of the many definitions of tracking error.

Drawdown

The drawdown of an investment at time t is noted as $D(t)$ and is a risk-related metric that measures the relative drop from a historical peak (High Water Mark):

$$D(t) = \frac{\text{HWM}(t) - p_t}{\text{HWM}(t)}$$

Where

$$\text{HWM}(t) = \max_{1 \leq \tau \leq t} p_t$$

2.7 High Frequency Data

The purest form of financial data is called tick data, which collects the information of sellers and buyers of a financial product. The way traders enter the market is by posting **orders**, that can be either **buy or sell orders**. A buy (sell) order represents the will of a trader to buy (sell) m units of an asset at a price p.

Before giving an overview of tick data, a few concepts should be defined:

- **Bid price** (b_t): maximum price out of all the buy orders at time t.
- **Ask price** (a_t): minimum price out of all the sell orders at time t.
- **Mid price:** $m_t = \frac{a_t + b_t}{2}$
- **Tick size:** smallest change in price a stock can move (e.g. 0.001$).

Note that orders are ordered in price and time. This means that if one posts an order to sell 10 units of an asset at time **10:00:21** at a price of 10.00$, then one would have less priority than a person that wanted to sell 4 units at time **10:00:18** at the same price.

Having said that, a few remarks follow:

1. If one were to buy or sell **one unit** of an asset, it could be done instantly at prices a_t and b_t respectively.
2. If the tick size did not exist then traders could jump ahead by increasing the bid or ask prices by a small amount ϵ.
3. The spread ($a_t - b_t > 0$) is a measure of the liquidity of the asset in question. That is, how buyers and sellers agree.
4. Data is **inhomogenous** in the sense that it does not arrive at fixed time intervals (see table 2.1). In fact, only when agreements are made between buyers and sellers will a new tick be recorded. Therefore, in a period of high activity a lot of **ticks** ($\{t, p_t, b_t, a_t, v_t\}$) will be recorded in a short time span, and vice versa.

Table 2.1: Example of tick data

t	p_t	b_t	a_t	v_t
2013-01-02 08:00:00	67.18	67.18	67.78	125
2013-01-02 08:12:56	67.70	67.19	67.70	125
2013-01-02 08:12:56	67.75	67.75	67.82	125
2013-01-02 08:47:15	67.91	67.21	67.93	150
2013-01-02 09:29:09	67.55	67.52	67.55	200
2013-01-02 09:29:09	67.56	67.51	67.57	200
2013-01-02 09:29:10	67.56	67.51	67.58	200
2013-01-02 09:29:10	67.58	67.51	67.57	100
2013-01-02 09:29:10	67.58	67.52	67.58	100
2013-01-02 09:29:11	67.57	67.52	67.58	200

Chapter 3

Meta-labeling

3.1 Introduction

Labeling is a well-known area in Machine Learning. It consists of gathering a matrix X, known as features, whose rows are the observations. Once a label y_i is assigned to every row $X_{[i,\cdot]}$ the main goal is to give a prediction $\widehat{y}_i$. Whenever $y_i \in I$ s.t. $|I| = 2$, the problem can be referred as a binary classification one, which is what this chapter will focus on.

When it comes to Finance, labeling is not as straightforward as a conventional case, i.e. predicting deterministic events; yes/no traffic light in the picture, yes/no happy face, etc. Since one is working with returns, one needs to determine whether a positive (negative) outcome will happen (or not) in a determined time horizon.

The investment literature has tried to label observations using what Marcos López de Prado (MLDP) [5] defines as *The Fixed-Time Horizon Method*:

$$y_i = \begin{cases} -1 & \text{if } r_{t_{i,0}+h}(h) < \tau \\ 0 & \text{if } |r_{t_{i,0}+h}(h)| \leq \tau \\ 1 & \text{if } r_{t_{i,0}+h}(h) > \tau \end{cases} \tag{3.1}$$

Where $r_{t_{i,0}+h}(h)$ is the log-return from time $t_{i,0}$ to $t_{i,0}+h$ (in this case, h is a time bar that can be days, months, etc.).

The problem with computing labels with a fixed threshold τ is that volatility, σ, changes overtime and should be updated regularly. Apart from that, restricting the model to a fixed h is not optimal at all since more flexible ways can be implemented.

3.2 Motivation

A portfolio of N securities is defined by the weights, $\mathbf{w} \in \mathbb{R}^N$ it gives to every instrument. In order to minimize the volatility, the Global Minimum Variance portfolio (**GMVP**) is defined as:

$$\begin{aligned} \min_{w} \quad & w^T \mathbf{\Sigma} w \\ \text{subject to} \quad & w \geq 0 \\ & \sum_{i=1}^{N} w_i = 1 \end{aligned} \tag{3.2}$$

Where $\mathbf{\Sigma}$ is the Covariance matrix of the N stock's log- returns.

At this point, remembering that the portfolio returns can be calculated as $\mathbf{w} \cdot \mathbf{R_t}$, it is appropriate to talk about a single time series. In other words, the portfolio indirectly transforms a multivariate time

series into a univariate one. Consequently, from now on, only the portfolio time series will be analyzed.

Regarding the motivation, as it can be seen in figure 3.1, the falls in late 2018 and early 2020 are a huge setback as far as the final wealth is concerned. Going back to section 3.1, the general idea is to label these periods accordingly so that these drawdowns can be predicted.

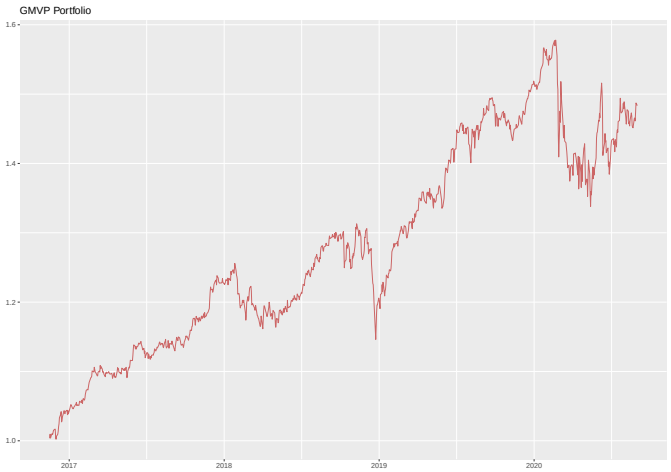


Figure 3.1: GMVP portfolio prices

3.3 What is meta-labeling?

In this chapter the concept of meta-labeling will be introduced using the diagram found in figure 3.2, adapted from [15].

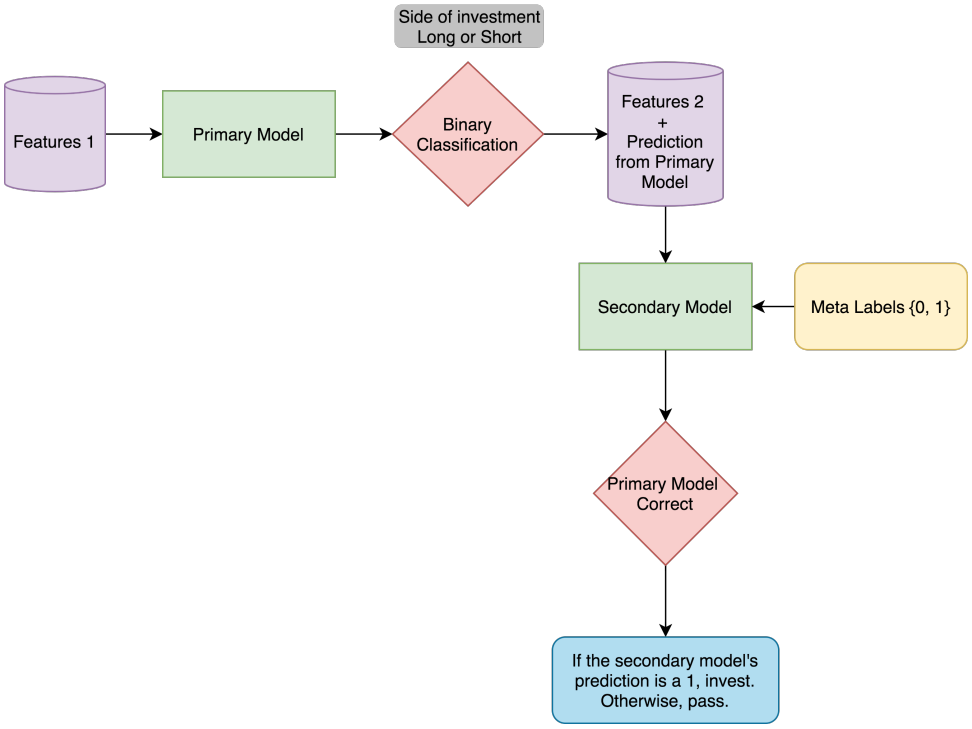


Figure 3.2: Meta-labeling model diagram

Bet-sizing is a common problem within the practitioner scene. That is, investors often know the side of the position they want to open (long/short or pass) but the size is unknown to them. Hence, an exogenous model that could base its predictions on some features, and the side prediction, comes into play.

The basic idea behind meta-labeling is having two models:

- **Primary Model:** in charge of determining whether one should open a long or short position on a given day.
- **Secondary Model:** exogenous model that will predict if the primary model was right.

Even though one could argue that the primary model is enough, it has a major flaw: lack of flexibility. That is, it always opens a position, and thus, it is not able to remain on the sidelines.

This is the point where the secondary model comes into play, since it will be trained to determine whether the primary model was correct or not. Consequently, if the secondary model predicts that the primary model was right, a position will be opened with the side predicted by the primary model. Otherwise, no position will be opened since it has been predicted that the primary model was wrong.

Labels

The labels of the primary model will be noted as $y_i^{\text{M1}} \in \{-1, 1\}$ and the predictions $\widehat{y}_i^{\text{M1}}$.

On the other hand, the predictions of the secondary model (or meta-labels) will be noted as $\widehat{y}_i^{\text{M2}}$, while the labels will be defined as:

$$y_i^{\text{M2}} = \begin{cases} 1 & \text{if } \widehat{y}_i^{\text{M1}} = y_i^{\text{M1}} \\ 0 & \text{otherwise} \end{cases}$$

That is, $y_i^{\text{M2}} \in \{0, 1\}$ and $y_i^{\text{M2}} = 1$ whenever the primary model made a correct prediction.

3.3.1 Binary classification

Before defining the global binary classification problem that will be dealt with, the concept of **meta-model** will be brought up. This model will be the combination of the primary and secondary models. Therefore, the meta-model will open a position (with the side of the primary model) when the secondary model predicts it. Also, an important remark is that the primary model by itself could be thought of as a meta-model where its secondary model always predicts that the primary model was right. That way, a position is always opened.

Having said that, it is time to define the binary classification problem by outlining the "positive" and "negative" outcomes:

- **1:** Open a position
- **0:** Do not open a position

This gives way to:

- **True Positive** (TP): $\widehat{y}_i^{\text{M2}} = 1 = y_i^{\text{M2}}$ - Opened a position that was profitable
- **False Positive** (FP): $\widehat{y}_i^{\text{M2}} = 1 \neq y_i^{\text{M2}}$ - Opened a losing position.
- **True Negative** (TN): $\widehat{y}_i^{\text{M2}} = 0 = y_i^{\text{M2}}$ - Did not open a position that was going to be unprofitable.
- **False Negative** (FN): $\widehat{y}_i^{\text{M2}} = 0 \neq y_i^{\text{M2}}$ - Took a pass at opening a position that was going to be profitable.

On the other hand, the metrics that will be used to evaluate the models will be the following:

- **Recall** = $\frac{\text{TP}}{\text{TP+FN}}$

- **Precision** $= \frac{\text{TP}}{\text{TP+FP}}$
- **F1 - Score** $= \frac{2}{\text{recall}^{-1}+\text{precision}^{-1}}$

3.4 Toy Project

Before applying meta-labeling to financial data, an experiment has been designed in order to explain how meta-labeling works with synthetic data. Labels of side (long/short) of investments will be created in a way that a designated set of features are responsible for it. That is, features that have predicting power. The desired output is to predict whether one should open a position or not, and the side of it. This will be done in a sequential manner, with a primary and secondary model.

The primary model will predict the side of the investment, and the secondary model will decide if the primary model was right. To be specific, the primary model will tell you to open a position (positive in the binary classification context) with a given side, and the secondary model will decide if it was a false or true positive.

As López de Prado states in [5], meta-labeling should be used when one wants to achieve higher F1-Scores (harmonic mean of precision and recall), because it lowers the high recall from the primary model while getting a much higher precision, thus boosting the F1-Score.

As for the data division, the primary model will use a set of features different than the secondary model ones. The latter will have some designated features plus the prediction from the primary model, which will be used to evaluate if the primary model made a correct decision.

3.4.1 Data

The data for this Toy Project will consist on 1000 observations of the following data points:

- **Features:** $\mathbf{X}_{k,i} \sim N(\mu_i,\ \sigma^2)$ for $i \in \{1, \ldots, 5\},\ k \in \{1, \ldots, 1000\}$
- $\omega_k = \textbf{sigmoid}\left(\alpha + \sum_{i=1}^{5} \mathbf{X}_{k,i} \cdot \beta_i + \epsilon_k\right)$ with $\epsilon_k \sim N(0,\ \sigma_\epsilon^2)$ and $\textbf{sigmoid}(z) = \frac{1}{1+e^{-z}}$

  Where:

  $\alpha = -0.5970,$
  $\boldsymbol{\beta} = [-0.7862,\ 0.7695,\ -1.3740,\ 0.6722,\ -0.4536],$
  $\boldsymbol{\mu} = [-0.3110,\ -0.1157,\ 0.0316,\ 0.3210,\ -0.5933],$
  $\sigma = 0.5$ and $\sigma_\epsilon \in \{0, 0.1, \ldots, 2.9, 3\}$

- **Labels:** The side will be designated using ω_k, which implies that all **5 features** have an effect on the correct side:

$$y_k^{\text{M1}} = \begin{cases} -1 & \text{if } \omega_k < 0.5 \\ 1 & \text{otherwise} \end{cases}$$

Note that, in order to avoid unbalanced labels, the following constraint has been applied: $0 = \alpha + \boldsymbol{\mu} \cdot \boldsymbol{\beta}$. That way, the expected value of $\alpha + \sum_{i=1}^{5} \mathbf{X}_{k,i} \cdot \beta_i + \epsilon_k$ is 0 ($\textbf{sigmoid}(0) = 0.5$).

By generating data this way, there are 5 explanatory variables that are responsible for the position opening. Lastly, the labels of the secondary model will be 1 whenever the primary model gave a correct prediction of the side and 0 otherwise:

$$y_k^{\text{M2}} = \begin{cases} 1 & \text{if } \widehat{y}_k^{\text{M1}} = y_k^{\text{M1}} \\ 0 & \text{otherwise} \end{cases}$$

3.4.2 Models

Since this section intends to give a general overview of the way meta-labeling works, the models will use different features in order to **simulate relative abundance or scarcity of data**. That is:

- M1: $\mathbf{X}_1$ $(N_{\text{M1}} = 1)$
  M2: $\hat{y}^{\text{M1}},\ \mathbf{X}_2,\ \ldots,\ \mathbf{X}_5$ $(N_{\text{M2}} = 5)$
- M1: $\mathbf{X}_1,\ \mathbf{X}_2$ $(N_{\text{M1}} = 2)$
  M2: $\hat{y}^{\text{M1}},\ \mathbf{X}_3,\ \mathbf{X}_4,\ \mathbf{X}_5$ $(N_{\text{M2}} = 4)$
- M1: $\mathbf{X}_1,\ \mathbf{X}_2,\ \mathbf{X}_3$ $(N_{\text{M1}} = 3)$
  M2: $\hat{y}^{\text{M1}},\ \mathbf{X}_4,\ \mathbf{X}_5$ $(N_{\text{M2}} = 3)$
- M1: $\mathbf{X}_1,\ \ldots,\ \mathbf{X}_4$ $(N_{\text{M1}} = 4)$
  M2: $\hat{y}^{\text{M1}},\ \mathbf{X}_5$ $(N_{\text{M2}} = 2)$

Where N_{M1} and N_{M2} are the number of features of the primary and secondary model respectively.

Primary Model (M1): It will use y^{M1} as labels. The underlying model used is a single layer neural network with a sigmoid activation function.

Secondary Model (M2): It will use y^{M2} as labels. The underlying model used is a neural network with a hidden layer (25 units - leaky ReLU) and an output unit with a sigmoid activation function.

Meta-model (M1 + M2): This model is the combination of the previous two models. It will decide to open a position with side ± 1 if the M1 predicts a side ± 1 and the M2 predicts a 1 (i.e. M1 is right). In contrast, if M2 predicts a 0 (M1 is wrong), then the meta-model will not open a position.

At this point, a relevant question regarding the meta-model is why cannot one train a single model instead of dividing the data between models. Although the single model will achieve higher precision scores, it will defeat the purpose of meta-labeling. In other words, one of the strengths of meta- labeling is being able to integrate ML into a fundamental/technical analysis approach or a model already up and running. That is, the secondary model will act as an exogenous model and not something that could have been designed from the start.

Lastly, the models will use 80% of data as in-sample and 20% of data as out-of-sample. The former will be further divided into training (80%) and validation (20%) so as to avoid over fitting.

3.4.3 Results

In the following subsections, the results of the toy project will be presented. In particular, in the subsection **Example** a σ_ϵ will be picked with the purpose of showing how meta-labeling works. In addition, the subsections **Precision**, **Recall** and **F1-Score** will analyze the metrics for every σ_ϵ considered.

Example

This subsection will exemplify what meta-labeling does in terms of relabeling false positives as true negatives, confusion matrices and metrics. The hyper parameters chosen are:

- $\sigma_\epsilon = 0.3$
- $N_{\text{M1}} = 2$
- $N_{\text{M2}} = 4$

		Prediction outcome		
		1	**0**	**Total**
Actual value	**1**	TP 143	FN 0	143
	0	FP 57	TN 0	57
	Total	200	0	

(a) Primary Model

		Prediction outcome		
		1	**0**	**Total**
Actual value	**1**	TP 136	FN 7	143
	0	FP 41	TN 16	57
	Total	177	23	

(b) Meta-model

Figure 3.3: Confusion Matrices in the Test data set

Table 3.1: Toy Project Metrics - Test

Model	F1-Score	Precision	Recall
Primary Model	0.8338	0.7150	1.0000
Meta Model	0.8500	0.7684	0.9510

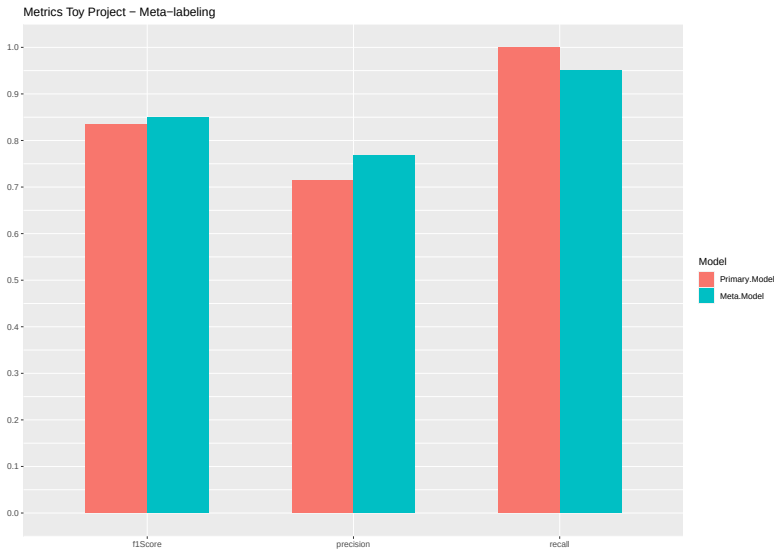


Figure 3.4: Toy Project - Metrics of example (Test)

As for the results, the confusion matrices of the primary model and meta-model are shown in figure 3.3. The primary model (figure 3.3a) only predicts opening positions, i.e., it does not have the ability to pass. Also, one should notice that the meta-model (figure 3.3b) re-labels more FP than TP.

The metrics obtained are shown in table 3.1 and figure 3.4. As it can be implied from the confusion matrices, the recall has gone down. However, the precision and F1-Score have gone up. In particular, the F1-Score has gone up by 2%, from 0.8338 to 0.85. This increase, even though on modest scale, is what meta-labeling was supposed to do.

Precision

In figures 3.5 to 3.8, the precision has been plotted for every σ_ϵ. Observe that whenever $N_{\text{M1}} \geq 3$, the meta-model fails to improve the precision. This could be attributed to the primary model being able to use the majority of the information, so the secondary model is not able to improve in the predictions given by the primary model.

On the other hand, if the primary model does not have a full picture of the information ($N_{\text{M1}} \leq 2$), this model is similar to the random model, which has a precision $= 0.5$. That way, the secondary model has more room to improve. To be more specific, if $\sigma_\epsilon \in [0,\ 0.7] \bigcup [2.6,\ 3]$, i.e., the observations have high/low signal-to-noise ratio (high/low σ_ϵ), the meta-model performs better.

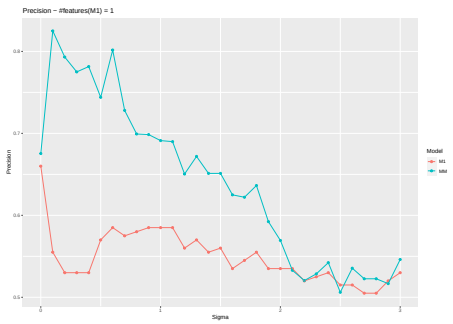


Figure 3.5: Precision (Test) - $N_{\text{M1}} = 1$

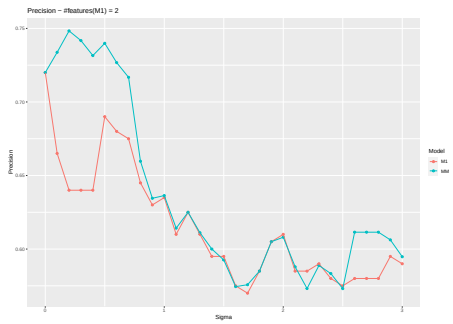


Figure 3.6: Precision (Test) - $N_{\text{M1}} = 2$

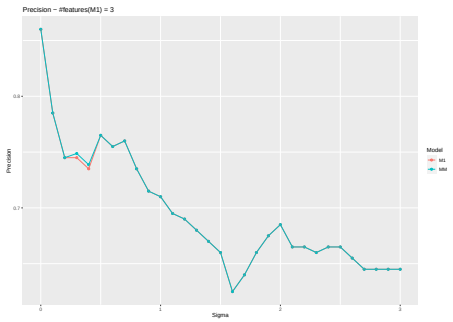


Figure 3.7: Precision (Test) - $N_{\text{M1}} = 3$

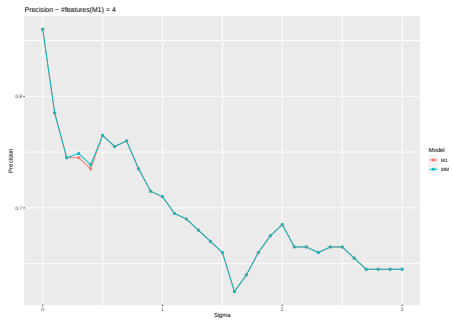


Figure 3.8: Precision (Test) - $N_{\text{M1}} = 4$

Recall

In figures 3.9 to 3.12, the recall has been plotted for every σ_ϵ. Before anything, observe that the primary model, independently of σ_ϵ and N_{M1}, always has a recall of 1. That is because the primary model does not have the ability to pass on a position. In other words, it always spits out a side, so in any event, a position is always opened (it only predicts 1's).

That being said, as in **Precision**, note that the meta-model is unable to change the recall whenever $N_{\text{M1}} \geq 3$. Additionally, if $\sigma_\epsilon \in [0,\ 0.7]$, then it is observed that for $N_{\text{M1}} \leq 2$ the recall falls considerably, so one can imply that if σ_ϵ is close to 0, then the recall falls due to the secondary model filtering trades.

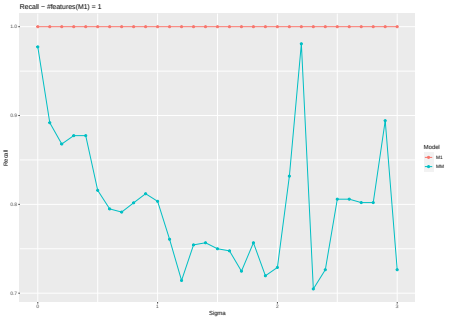


Figure 3.9: Recall (Test) - $N_{\text{M1}} = 1$

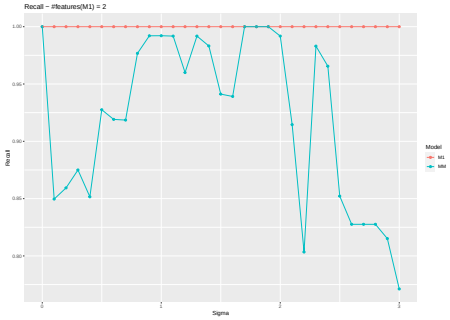


Figure 3.10: Recall (Test) - $N_{\text{M1}} = 2$

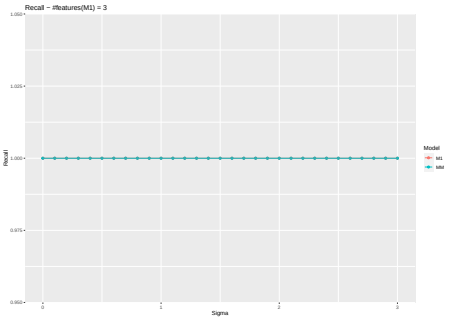


Figure 3.11: Recall (Test) - $N_{\text{M1}} = 3$

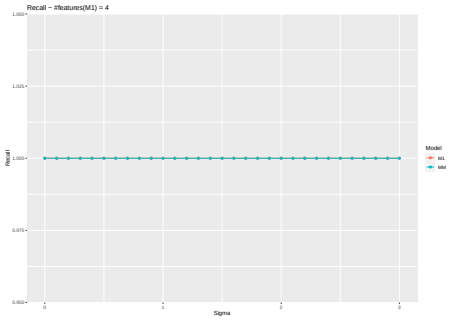


Figure 3.12: Recall (Test) - $N_{\text{M1}} = 4$

F1-Score

In figures 3.13 to 3.16, the F1-Score has been plotted for every σ_ϵ. As it has been seen in subsections **Precision** and **Recall**, if $N_{\text{M1}} \geq 3$, the performance of the primary and meta model is indistinguishable. Furthermore, if $N_{\text{M1}} \leq 2$, the F1-Score slightly improved when σ_ϵ was low ($\sigma_\epsilon \leq 0.6$).

Figure 3.13: F1-Score (Test) - $N_{\text{M1}} = 1$

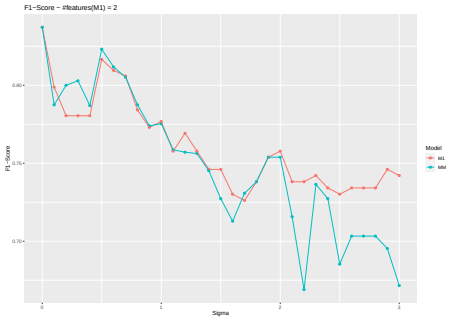


Figure 3.14: F1-Score (Test) - $N_{\text{M1}} = 2$

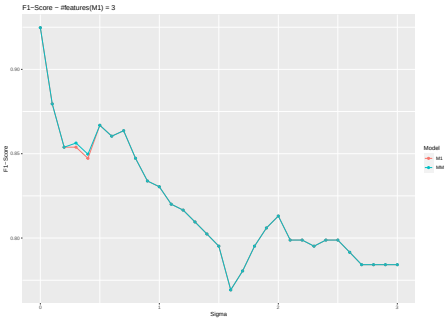


Figure 3.15: F1-Score (Test) - $N_{\text{M1}} = 3$

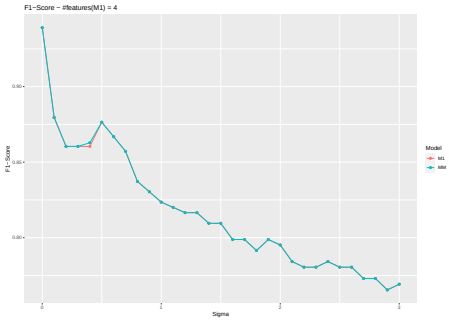


Figure 3.16: F1-Score (Test) - $N_{\text{M1}} = 4$

3.4.4 Conclusions

In conclusion, from the results observed in the toy project it can be concluded that meta-labeling needs a specific setting so as to deliver better results.

In addition, in an attempt to describe the behavior of meta-labeling, two typical situations have been identified:

1. M1 performs poorly (low precision) and/or has few features with predicting power (in this case, low N_{M1}).
2. M1 has the majority of the features with high predicting power (in this case, high N_{M1}), i.e., it is already a *good* model.

In situation 1, the meta-model will perform better, in an F1-Score sense, whenever σ_ϵ is low and N_{M2} is high. To put it another way, if the observations do not have a lot of noise, then the secondary model is able to correct the bad performance of the primary model. Note that since the 5 features are shared, whenever the primary model has few features with predicting power, the secondary model has a lot of predicting power.

In situation 2, M1 is already a decent model (recall = 1, and high precision when σ_ϵ is low). Consequently, it is very difficult to improve its performance. In fact, the F1-Score of the meta model is identical to the one of M1 because the secondary model fails to introduce new information (low N_{M2}).

The next step will be to try this methodology in financial data, which will be more challenging, since the synthetic data used was designed to be somewhat predictable, even though it had Gaussian white noise.

3.5 Financial Data

In order to build a GMVP a universe of stocks should be defined. In this case, the assets considered are the ones that have been part of the S&P500 in the period considered; 2000-01-01 to 2020-09-01. The

stocks that have missing values or have gone bankrupt have been dropped out of the dataset.

Bear in mind that this portfolio is not a proxy for the S&P500 GMVP since it presents look-ahead bias. That is, if one were to compute day-to-day the GMVP portfolio, one would not obtain the same data since the *losers* (companies that have gone bankrupt) have been removed, and the *winners* (future constituents of the index) have been added. Nonetheless, this chapter has not been designed in order to beat the index. In fact, this chapter has been designed with a focus on delivering better risk-adjusted results, i.e., higher Sharpe Ratios.

The tickers of the stocks considered are shown in tables 6.1 and 6.2 (pages 58 and 59 resp.).

3.5.1 Removal of outliers

In an attempt to reduce the influence of outliers on the models, the data has been cleaned using the R package imputeFin [10]. In particular, the function **impute_AR1_t** has been used with the following parameters:

- **outlier_prob_th** = 0.005 - Threshold of probability of observation to declare an outlier.
- **remove_outliers** = TRUE

In order to apply the function, data has been divided into 10 chunks. Then, log-prices will be fitted an autoregressive model of order 1 where the residuals follow a t-Student distribution.

$$\log(p_t) = \phi_0 + \phi_1 \cdot \log(p_{t-1}) + \epsilon_t$$

The AR(1) model together with **outlier_prob_th** identifies the outliers and imputes values so as to preserve statistical parameters in the time series. In figure 3.17 one can see the difference between the original log-price time series (GMVP Outliers) and the imputed one (GMVP Imputed).

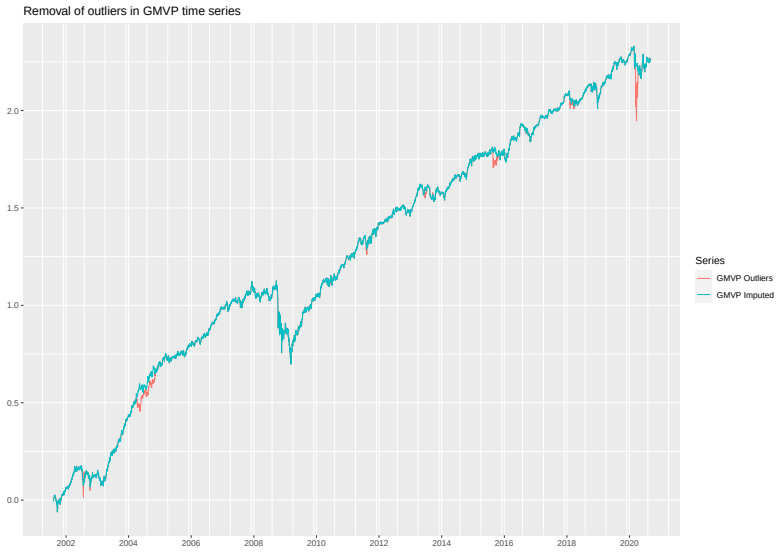


Figure 3.17: Imputed log-price time series of the GMVP

3.5.2 Division of Data

The data is divided as following (see figure 3.18):

In-Sample:

- Train (2001-08-06/2013-10-18): Data set that will be used to train the ML models.

- Validation (2013-10-21/2016-11-04): Data set to assess the performance of the ML models. It will be used to tune the hyper-parameters and/or to avoid overfitting.

Out-of-sample:

- Test (2016-11-07/2020-08-31): The models will use this data set to generate *out-of-sample* predictions and the performance observed will be close to real since data will be seen for the first time.

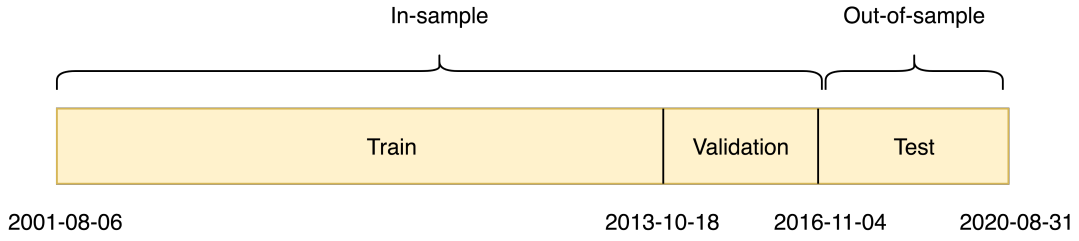


Figure 3.18: Data Division

3.6 Labeling (Triple barrier)

In section 3.1, *The Fixed-Time Horizon Method* was discussed. In an attempt to improve the previous method, MLDP defined the triple barrier method in [5]. It consists of:

- **Horizontal barriers:** Dynamic levels that depend on the 20 day rolling volatility.
- **Vertical barrier:** Set as a fixed time horizon. In this case, 10 days.

The variables/concepts that will be used are:

- The position is opened at the end of day $t_{i,0}$.
- The end of day $t_{i,0} + h$ is the vertical barrier.
- $p_{t_{i,0}}$ is the price at the end of day $t_{i,0}$.
- $p_{t_{i,0}}(1 + \text{pt} \cdot \sigma_{t_{i,0}})$ is the upper horizontal barrier.
- $p_{t_{i,0}}(1 - \text{sl} \cdot \sigma_{t_{i,0}})$ is the lower horizontal barrier.
- $\sigma_{t_{i,0}} := \sqrt{\frac{\sum_{k=0}^{19}(r_{t_{i,0}-k} - \bar{r})^2}{20-1}}$ the 20 day rolling volatility.
- $\bar{r} := \frac{\sum_{j=0}^{19} r_{t_{i,0}-j}}{20}$ the 20 day rolling mean of log-returns.
- tc: one-way transaction cost, which has been fixed at **5 bps** (0.05%).

It should be pointed out that hyper-parameters **pt** and **sl** both have been set to **2**, and that every variable has been carefully computed using information up to day $t_{i,0}$ to avoid look-ahead bias as spotted in [17].

With these variables in mind, it is time to introduce the triple barrier method. At day $t_{i,0}$, when the price is $p_{t_{i,0}}$, the horizontal and vertical barriers are set, creating a "cage" (see figure 3.19). Unlike the fixed-time horizon method, which automatically closes a position after h days (and computes the label), the triple barrier method closes a position (and computes the label) after the price hits one of the three barriers. Note that exiting is assured since at most the position will be open for 10 days (vertical barrier).

Therefore, if one opens a position on $t_{i,0}$, the triple barrier method will close the position the first time a barrier is hit ($t_{i,1}$). The corresponding return can be computed as:

$$R_i = (1 - \text{tc})^2 \cdot \left(\frac{p_{t_{i,1}}}{p_{t_{i,0}}} - 1 \right)$$

Combining everything, the observations are labeled as:

$$y_i = \begin{cases} 1 & \text{if } R_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

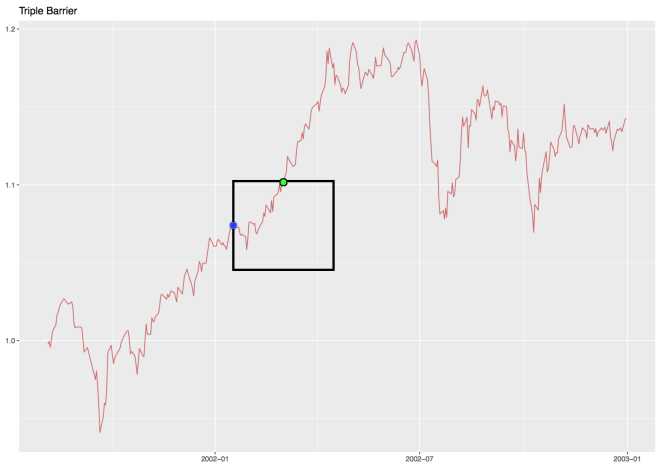


Figure 3.19: Triple Barrier Labeling (Symmetric barriers)

In figure 3.19, an observation **labeled** as **1** can be seen. In early January 2002 the barriers start and the price time series touched the upper horizontal barrier first, hence, obtaining a positive return.

3.6.1 Adaptation when the side is known

Let's suppose the primary model predicts the side of the trade starting on $t_{i,0}$ as $\widehat{y}_i^{\text{M1}} \in \{1, -1\}$. That is, either long or short. Accordingly, the method can be modified to include these changes:

$$R_i = (1 - \text{tc})^2 \cdot \left(\frac{p_{t_{i,1}}}{p_{t_{i,0}}} - 1 \right) \cdot \widehat{y}_i^{\text{M1}}$$

Combining everything, the observations are labeled as:

$$y_i^{\text{M2}} = \begin{cases} 1 & \text{if } R_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Apart from that, the horizontal barriers should be changed since it has been predicted that the stock will go up/down (long/short):

$$\delta_{+,t_{i,0}} = \begin{cases} \text{pt} \cdot \sigma_{t_{i,0}} & \text{if } \widehat{y}_i^{\text{M1}} = 1 \\ \min(0.5\%,\ \frac{1}{2} \cdot \text{pt} \cdot \sigma_{t_{i,0}}) & \text{otherwise} \end{cases}$$

$$\delta_{-,t_{i,0}} = \begin{cases} \text{sl} \cdot \sigma_{t_{i,0}} & \text{if } \widehat{y}_i^{\text{M1}} = -1 \\ \min(0.5\%,\ \frac{1}{2} \cdot \text{sl} \cdot \sigma_{t_{i,0}}) & \text{otherwise} \end{cases}$$

Therefore, the price will oscillate between $[p_{t_{i,0}}(1-\delta_{-,t_{i,0}}),\ p_{t_{i,0}}(1+\delta_{+,t_{i,0}})]$. Also, note that the position will be opened at the end of day $t_{i,0}$, so only information **up to that day** should be used.

In figure 3.20 an example can be seen which presents a side $= -1$ (short). As the prediction says the price should go down, the upper horizontal barrier has been lowered. In this case, since the price has hit the upper horizontal barrier and the side was -1, then the observation will be **labeled** as **0**.

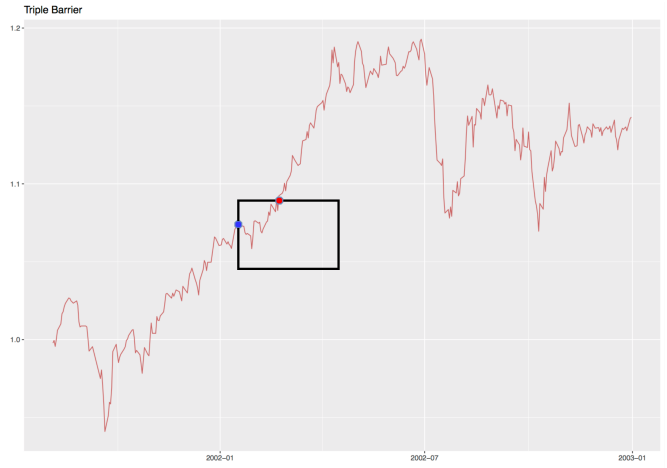


Figure 3.20: Triple Barrier Labeling - Side $= -1$ (Short)

3.7 Primary Model

The primary model can be based on fundamental analysis, Machine Learning, technical analysis, etc. With the purpose of showing how different models fare with meta-labeling, two types of primary models will be considered:

- **Moving average (MA) crossover strategy**: based on technical analysis (adapted from [8, 17])
- **Machine Learning (ML) strategy** with features as in [15]

3.7.1 MA based Primary Model

This primary model uses a Moving Average (MA) crossover strategy with a 20 day look-back period to establish the side of the investment. The entry points will be determined by structural breaks identified by the symmetric CUSUM (cumulative sum) filter, defined by:

$$\begin{aligned} \text{CUSUM}_{+,\,t} &= \max\left(0,\ \text{CUSUM}_{+,\,t-1} + r_t\right) \\ \text{CUSUM}_{-,\,t} &= \min\left(0,\ \text{CUSUM}_{-,\,t-1} + r_t\right) \end{aligned}$$

With the boundary conditions $\text{CUSUM}_{+,\,0} = \text{CUSUM}_{-,\,0} = 0$.

The filters **are reset to 0 whenever** $\text{CUSUM}_{+,\,t} > \sigma_t$ or $\text{CUSUM}_{-,\,t} < -\sigma_t$, where σ_t is the 20 day rolling volatility. Whenever this reset happens, it will be said that **the filters have been activated.**

Hence, at this point, a collection of days will have been gathered. To be specific, one can open a position at the end of days $t \in \{t_{1,0},\ \ldots,\ t_{i,0},\ \ldots,\ t_{N,0}\}$, where $t_{i,0}$ are the days where the filters have been activated.

Having filtered the days when one can open a position, the next step consists on computing the MA as in 3.3, and then the side (see equation 3.4).

$$\text{MA}_i = \frac{\sum_{k=0}^{19} p_{t_{i,0}-k}}{20} \tag{3.3}$$

$$\widehat{y}_i^{\text{M1}} = \begin{cases} 1 & \text{if } \text{MA}_i < p_{t_{i,0}} \\ -1 & \text{if } \text{MA}_i \geq p_{t_{i,0}} \end{cases} \tag{3.4}$$

On the other hand, the true labels are:

$$y_i^{\text{M1}} = \begin{cases} 1 & \text{if } R_i = (1-\text{tc})^2 \left(\frac{p_{t_{i,1}}}{p_{t_{i,0}}} - 1\right) > 0 \\ -1 & \text{otherwise} \end{cases}$$

Where $t_{i,1}$ is the exit day using the triple barrier method explained in subsection 3.6.1.

3.7.2 ML based Primary Model

This model is slightly different that the one based on MA. It aims to train a neural network with a hidden layer (20 fully connected units - Leaky ReLU) and an output layer (Sigmoid). It will be in charge of predicting the side (1 = long, -1 = short).

Firstly, let's present the features that the neural network will use:

- $\log\left(\frac{\text{MA}_t}{p_t}\right)$
- $\frac{\text{CUSUM}_{+,\ t}}{\sigma_t}$
- $\frac{\text{CUSUM}_{-,\ t}}{\sigma_t}$
- $\textbf{Reset}_{\text{CUSUM}_+,t}$: Binary variable that indicates if the positive CUSUM filter became activated at time t.
- $\textbf{Reset}_{\text{CUSUM}_-,t}$: Binary variable that indicates if the negative CUSUM filter became activated at time t.
- EWMSD_t: Exponentially weighted moving standard deviation of linear returns.

Where $\sigma_t^2 = \frac{\sum_{i=0}^{19}(r_{t-i} - \bar{r}_{t-19,t})^2}{20-1}$.

In this model one does not depend on the CUSUM filter to determine the date to open a position. The neural network, with the features given, will decide the side and the position will be opened. That being said, with the intention of matching features and labels, the latter have been defined as:

$$y_i^{\text{M1}} = \begin{cases} 1 & \text{if } R_i > 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.5}$$

Where $R_i = (1-\text{tc})^2 \left(\frac{p_{t_{i,1}}}{p_{t_{i,0}}} - 1\right)$ is the realized return using the triple barrier method, $t_{i,0}$ the start day and $t_{i,1}$ the day one of the barriers is hit. Also, remember that since side is unknown, the horizontal barriers will be defined with $\delta_{+,t_{i,0}} = \delta_{-,t_{i,0}} = \sigma_{t_{i,0}}$.

3.8 Secondary Model

The idea behind a secondary model is to train a Machine Learning model capable of learning how to use the primary model, deciding when one should/should not enter. Being more specific, it will filter the positives predicted from the primary model.

Similarly to what it was done in section 3.7, the models will be adapted to the 2 primary models defined.

3.8.1 MA based Secondary Model

This model will use a random forest trained with the following **labels:**

$$y_i^{\text{M2}} = \begin{cases} 1 & \text{if } y_i^{\text{M1}} = \widehat{y}_i^{\text{M1}} \\ 0 & \text{otherwise} \end{cases}$$

Note that this definition matches the one in subsection 3.6.1 since:

$$y_i^{\text{M1}} = \widehat{y}_i^{\text{M1}} \Rightarrow R_i = \left(\frac{p_{t_{i,1}}}{p_{t_{i,0}}} - 1\right) \cdot \widehat{y}_i^{\text{M1}} > 0$$

Additionally, the following features for day $\boldsymbol{t_{i,0}}$ will be used:

- $\log\left(\frac{\text{MA}_i}{p_{t_{i,0}}}\right)$
- $r_{t_{i,0}}$: return of the day $t_{i,0}$
- $\underline{r_{t_{i,0}}} := \left(\prod_{k=0}^{4}(1 + r_{t_{i,0}-k})\right) - 1$: Cumulative return using a 5 day window.
- $\widehat{y}_i^{\text{M1}}$
- $\text{RSI}_{t_{i,0},\ 9}$, $\text{RSI}_{t_{i,0},\ 14}$, $\text{RSI}_{t_{i,0},\ 25}$: Relative strength indicator (RSI) of linear returns using a 9, 14 and 25 days window respectively.
- $\tilde{\sigma}_{t_{i,0}}$: Exponentially weighted volatility using a 21 day ($\approx$ 1 trading month) window.
- $\sigma_{t_{i,0},\ 9}$, $\sigma_{t_{i,0},\ 14}$, $\sigma_{t_{i,0},\ 25}$: Volatility using a 9, 14 and 25 days window respectively.
- $\text{ACF}_{t_{i,0},\ 1}$, $\text{ACF}_{t_{i,0},\ 5}$: Auto-correlation function of log-returns with 1 and 5 days lag respectively.

The meta model (primary + secondary), together with the threshold thr_{M1}, will work using the following cases:

1. Check if a position is currently open
2. If no position is open and the CUSUM filters have been activated, then compute the side ($\widehat{y}_i^{\text{M1}}$) and predict the size ($\widehat{y}_i^{\text{M2}}$) using the primary and secondary model respectively.
3. Go long if:
   - $\widehat{y}_i^{\text{M1}} = 1$ and $\widehat{y}_i^{\text{M2}} > \text{thr}_{\text{M2}}$ - The primary model was right about going long.

   Go short if:
   - $\widehat{y}_i^{\text{M1}} = -1$ and $\widehat{y}_i^{\text{M2}} > \text{thr}_{\text{M2}}$ - The primary model was right about going short.

   Otherwise, the secondary model infers that there is not enough confidence to open a position and one should return to point 1 using data from the next possible entry point.
4. Open a position and exit using the triple barrier method.
5. Return to point 1 using data from the next possible entry point.

3.8.2 ML based Secondary Model

This model will use a neural network with a hidden layer (25 fully connected units - Leaky ReLU) and an output layer (Sigmoid). The labeling technique is the same as the one found in 3.8.1:

$$y_i^{\text{M2}} = \begin{cases} 1 & \text{if } y_i^{\text{M1}} = \widehat{y}_i^{\text{M1}} \\ 0 & \text{otherwise} \end{cases}$$

The features for day $\boldsymbol{t_{i,0}}$ are the same as the ones found in the MA based Secondary Model (3.8.1):

- $\widehat{y}_i^{\text{M1}}$
- $\log\left(\frac{\text{MA}_i}{p_{t_{i,0}}}\right)$
- $r_{t_{i,0}}$, $\underline{r_{t_{i,0}}}$

- $\text{RSI}_{t_{i,0},\ 9}$, $\text{RSI}_{t_{i,0},\ 14}$, $\text{RSI}_{t_{i,0},\ 25}$
- $\tilde{\sigma}_{t_{i,0}}$, $\sigma_{t_{i,0},\ 9}$, $\sigma_{t_{i,0},\ 14}$, $\sigma_{t_{i,0},\ 25}$
- $\text{ACF}_{t_{i,0},\ 1}$ and $\text{ACF}_{t_{i,0},\ 5}$

The meta model (primary + secondary) will follow the same procedure from subsection 3.8.1 to open/close a position. However, remember that it does not rely on the CUSUM filter to determine entry points.

3.9 Hyper-parameter tuning via cross-validation

3.9.1 Cross-Validation MA based Models

One of the ideas behind having a validation data set that has not been explicitly trained on, is to determine the value of hyper-parameters that make the model perform best in this data set. That way, one would expect this performance to transfer over to the test data set.

One should be aware that the hyper-parameters that are going to be tuned in the MA based models are thresholds. As an example of them, consider the raw predictions given by ML binary classifiers, which are real numbers in the interval $[0, 1]$. One way to determine $\widehat{y}_i \in \{0, 1\}$ is to assign it a value of 1 whenever the raw prediction surpasses a threshold, and 0 otherwise.

With that said, as thresholds play an important part at deciding whether to open a position or not, attention should be paid so as to achieve the best performance possible. This is where the Sharpe Ratio comes into play.

Suppose one has obtained a daily time series (R_t) of returns from a model. Then, the annualized Sharpe Ratio is defined as:

$$\text{SR} = \frac{\text{mean}(\widehat{R}_t)}{\text{SD}(\widehat{R}_t)} \cdot \sqrt{\frac{\#\text{oportunities}}{\#\text{years}}} \tag{3.6}$$

Where:

- $\widehat{R}_t$: time series without the zero returns (days where the model has decided not to enter)
- #opportunities: number of observations of the time series $\widehat{R}_t$
- #years: number of years elapsed from the first to the last observation of the time series $\widehat{R}_t$.

It is important to point out that a Buy & Hold strategy (buying an asset and holding it forever) presents the following properties:

1. $\widehat{R}_t = R_t$ because the strategy always keeps the position open. Hence, there are 0 days with zero returns.
2. $\frac{\#\text{opportunities}}{\#\text{years}} = 252 = \#\text{Trading days in a year}$: for the same reason as in point 1.

Having defined all these variables, one should be aware that the **MA based Models are defined by two thresholds: $\text{thr}_{\text{CUSUM}}$ and thr_{M2}.**

Since the primary model is based on MA crossovers, which are deterministic, there are not obvious hyper-parameters to tune. However, the CUSUM filter determined that a position should be opened when $\text{CUSUM}_{+,\ t} > \sigma_t$ or $\text{CUSUM}_{-,\ t} < -\sigma_t$. This can be adapted by scaling the volatility using $\text{thr}_{\text{CUSUM}}$, i.e., opening a position (and resetting) whenever:

- $\text{CUSUM}_{+,\ t} > \text{thr}_{\text{CUSUM}} \cdot \sigma_t$

- $\text{CUSUM}_{-,\,t} < -\text{thr}_{\text{CUSUM}} \cdot \sigma_t$

The cross-validation results of the primary model are shown in table 3.2. As it can be seen, the results are far from satisfactory. Apart from that, negative Sharpe Ratios are misleading since a small standard deviation can throw things off when returns are close to 0 but negative. Therefore, this method is useless to determine the "best" $\text{thr}_{\text{CUSUM}}$.

Table 3.2: Results cross-validation Primary Model (MA)

$\text{thr}_{\text{CUSUM}}$	Final wealth	SR Primary Model (Validation)
0.25	0.6533	-1.3585
0.60	0.6449	-1.4021
0.95	0.6566	-1.4137
1.30	0.6290	-1.6683
1.65	0.6758	-1.5189
2.00	0.7419	-1.1885

That is why it has been decided to determine $\text{thr}_{\text{CUSUM}}$ indirectly. This will be done by fixing a $\text{thr}_{\text{CUSUM}}$ and then computing the best Sharpe Ratio the meta-model can attain. Then, the $\text{thr}_{\text{CUSUM}}$ that gives the highest Sharpe Ratio will be chosen.

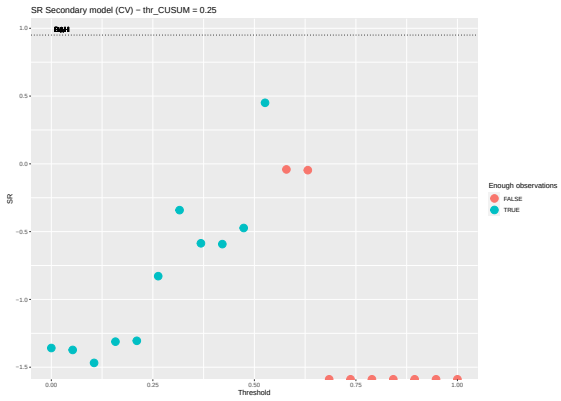
The obvious question is: having fixed a $\text{thr}_{\text{CUSUM}}$, how can one calculate the best Sharpe Ratio of the meta-model? The answer lies in thr_{M2}, which is the responsible of opening positions. In fact, it is so critical that if thr_{M2} is close to 1, most of the predictions $\widehat{y}_i^{\text{M2}}$ will be equal to 0. Therefore, the meta-model will have few observations since the secondary model will always predict that the primary model is wrong.

In an effort to discard the thr_{M2} that are too restrictive, a criteria has been set to determine if there are enough observations:

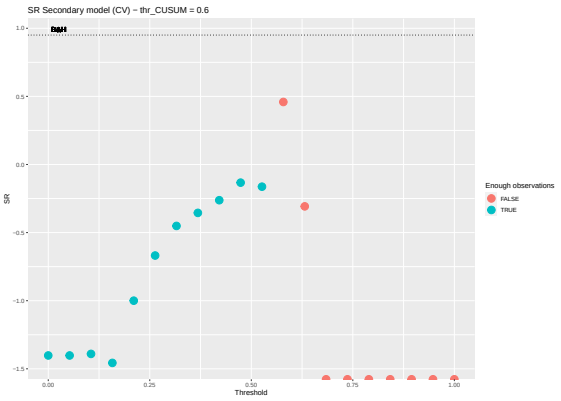
- For every thr_{M2} compute the number of times the meta-model decides to enter, which will be called number of discrete returns. Note that it is different than the number of days the model has kept a position open.
- Compute the median.
- If the number of discrete returns surpasses half the median, then it is labeled as having "Enough observations".

The idea behind this filter is to avoid over fitting. By choosing a thr_{M2} that has a low number of discrete returns, then it is likely that its out-of-sample performance will not be as good as in-sample because there are few observations.

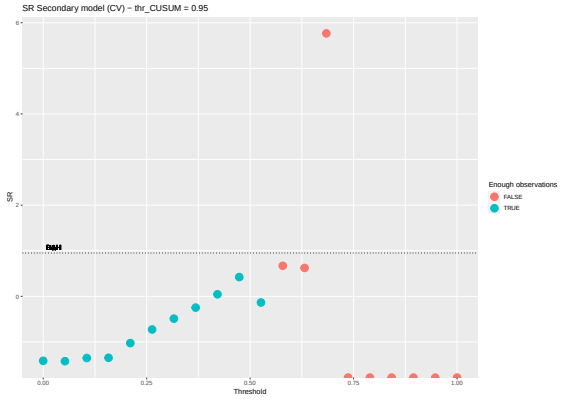
Given a $\text{thr}_{\text{CUSUM}}$, figure 3.21 shows a plot of the Sharpe Ratio of the meta-model as a function of thr_{M2}. Additionally, table 3.3 summarizes the previous figures mentioned. Also, from this table one can conclude that $\text{thr}_{\text{CUSUM}} = 0.25$ performs best, since the meta-model achieves a SR of 0.4971, using a $\text{thr}_{\text{M2}} = 0.53$. Hence, these two thresholds will be fixed from now on.



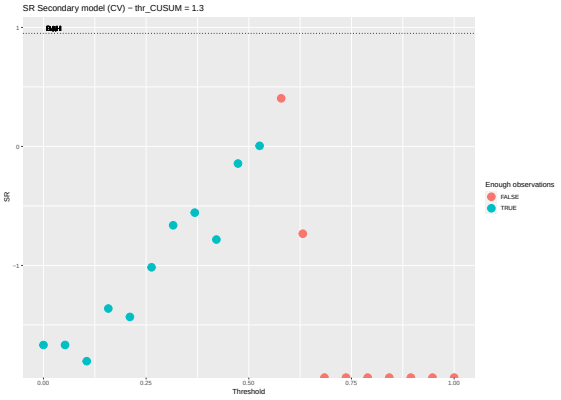
(a) $\text{thr}_{\text{CUSUM}} = 0.25$



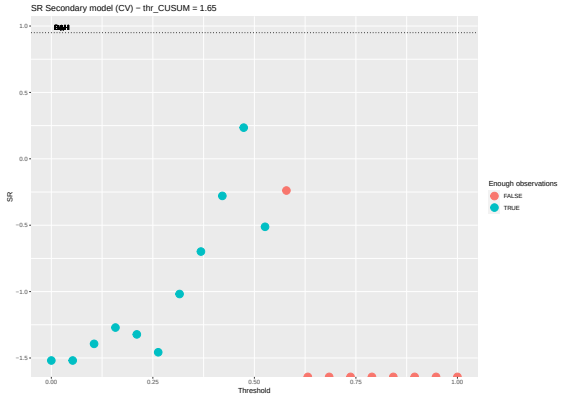
(b) $\text{thr}_{\text{CUSUM}} = 0.60$



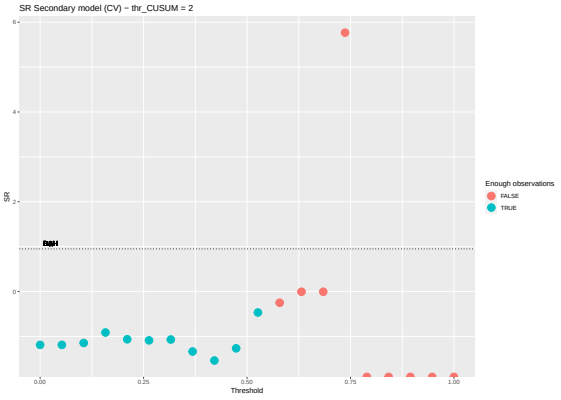
(c) $\text{thr}_{\text{CUSUM}} = 0.95$



(d) $\text{thr}_{\text{CUSUM}} = 1.30$



(e) $\text{thr}_{\text{CUSUM}} = 1.65$



(f) $\text{thr}_{\text{CUSUM}} = 2.00$

Figure 3.21: Sharpe Ratio of meta-model in the validation data set

Table 3.3: Results cross-validation Meta Model (MA)

$\text{thr}_{\text{CUSUM}}$	SR Meta Model (Validation)
0.25	0.4971
0.60	-0.1333
0.95	0.4230
1.30	0.0051
1.65	0.2348
2.00	-0.4674

3.9.2 Cross-Validation ML based Models

The purpose of cross-validation in the ML based models has been different than the ones based in MA. The validation data set has been used as an independent stopping indicator. That is, the neural networks have been trained on the train data set and whenever the loss function value in the validation data set increases, while the train one decreases, the algorithm stops. This is because the model is becoming too good a predictor in the train data set, without delivering the same results in unseen data sets, which is called overfitting.

That said, the thresholds for both the primary and secondary model have not been tuned. Hence, a $\widehat{y}_i$ will be given a positive value whenever its raw prediction surpasses 0.5, and negative otherwise.

3.10 Results

In subsections 3.10.1 and 3.10.2 a table and several graphs will be shown with the purpose of assessing the out-of-sample performance (Sharpe Ratio & Drawdown) of the models considered, which are the following:

- **Buy & Hold:** Buying the portfolio at the start and keeping the position until the end of the period.
- **Primary Model:** It is different from the previous model since it applies the corresponding side (long/short).
- **Meta Model:** Combination of the primary model and the secondary model. The latter is used to filter the trades from the former.

It should be pointed out that the **benchmark** is the **Buy & Hold** (B&H) model since it is the simplest one; it buys the portfolio and does nothing.

3.10.1 MA based Models

Since a huge emphasis has been put to select the threshold that performs best (SR-wise) in the validation set (in-sample), the results of the MA-based model in the validation data set are shown in table 3.4.

As it is seen in table 3.4, the meta-model does not even achieve satisfactory results (beating the SR of B&H) in the validation data set, so one should not expect to see positive results in the Test data set. Indeed, in table 3.5, the results in the Test data set show how the meta-model is unable to surpass the SR of the B&H.

Besides that, the precision of the primary model is so low (0.51 - validation, 0.58 - test) that the meta-model has needed to lower the recall considerably in order to achieve better precision results. That leads to a decrease in F1-Score in both data sets.

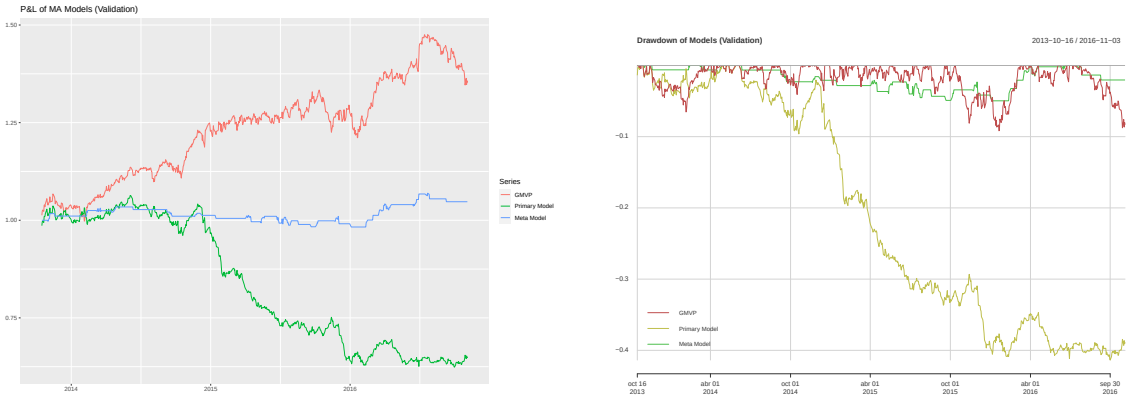
For more information, one can refer to figure 3.22 and 3.23 where the prices and drawdown of the models have been plotted.

Table 3.4: Results in the validation data set (MA)

Model	Max. Drawdown	SR	Precision	Recall	F1-Score
Buy & Hold	9.17%	0.9479	-	-	-
Primary Model	41.36%	-1.3585	0.5100	1	0.6757
Meta Model	4.97%	0.4504	0.6444	0.1657	0.2636

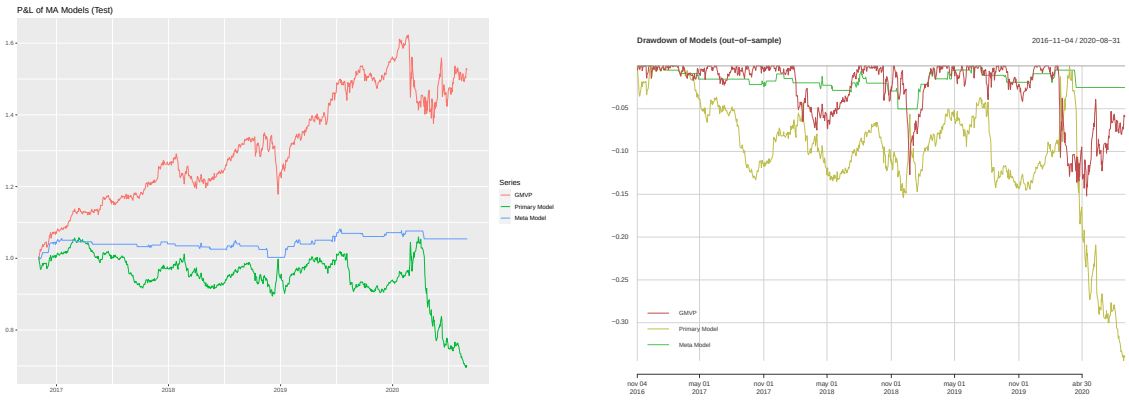
Table 3.5: Results in the Test data set (MA)

Model	Max. Drawdown	SR	Precision	Recall	F1-Score
Buy & Hold	15.20%	0.9682	-	-	-
Primary Model	34.41%	-0.7467	0.5758	1	0.7308
Meta Model	5.02%	0.4887	0.6769	0.1781	0.2821



(a) Prices (b) Drawdown

Figure 3.22: Results of MA Models in the validation data set



(a) Prices (b) Drawdown

Figure 3.23: Results of MA Models in the Test data set

3.10.2 ML based Models

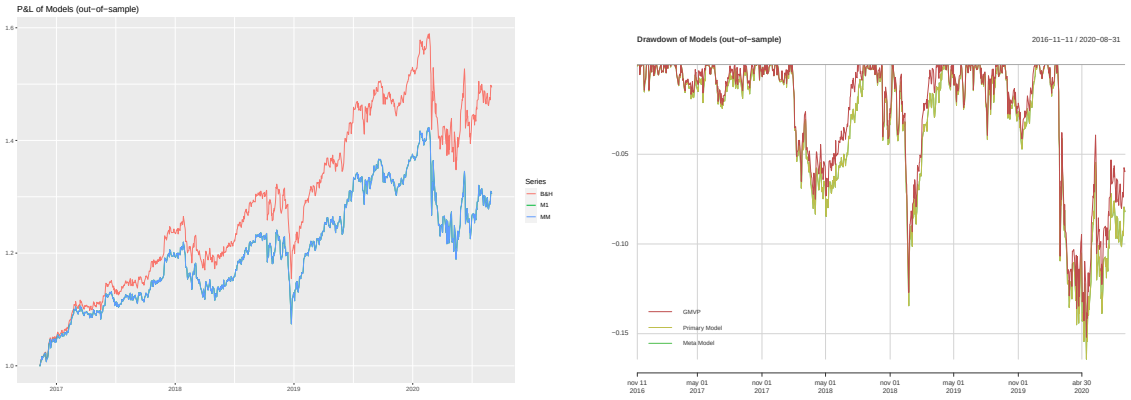
Table 3.6 shows the results of the different models based on ML in the Test data set. In addition, figure 3.24 shows the prices and drawdown for the different models in the Test data set.

The immediate thing to point out is that the performance between the primary model and the meta-model is indistinguishable. That means that the secondary model has predicted that the primary model was right all of the times. That is attributed to the lack of predictive power of the features of the secondary model. Consequently, the secondary model has decided to ignore the features and always predict a 1 since it is the majority class (it will result in the best naive metrics). Apart from that, in the primary model a similar thing happens since it always predicts to go long.

All of this is reflected in the metrics, which fail to change from model to model:

Table 3.6: Results in the Test Dataset (ML)

Model	Max. Drawdown	SR	Precision	Recall	F1-Score
Buy & Hold	15.20%	0.9682	-	-	-
Primary Model	16.42%	0.6630	0.6757	1	0.8065
Meta Model	16.42%	0.6630	0.6758	0.9969	0.8055



(a) Prices

(b) Drawdown

Figure 3.24: ML Models results in the Test data set

3.11 Coin flip correction

As it has been seen in the previous sections, the performance of the primary model has been suboptimal. In an effort to remove this condition from this work, a strategy has been devised towards achieving better (but artificial) primary models.

The idea lays in the concept of a coin flip, $S \sim Be(p)$ ($p = \Pr(S = 1)$). Suppose that every observation is linked to a random variable $S_i \sim Be(p)$. Then, one can define a new feature:

$$F_i = (1 - S_i) \cdot y_i^{\mathrm{M1}} - S_i \cdot y_i^{\mathrm{M1}} = (1 - 2 \cdot S_i) \cdot y_i^{\mathrm{M1}}$$

where $y_i^{\mathrm{M1}} \in \{-1, 1\}$ is the label of the observation in the primary model.

Interpreting p as the probability of swapping a feature, the model that defines $S_i \sim Be(0)$ will not swap features and the primary model will be the "perfect classifier". On the other hand, if $S_i \sim Be(0.5)$ then F_i is useless since the feature will alternate (in a random way) between $-y_i^{\mathrm{M1}}$ and y_i^{M1}.

These artificial models will use the framework from the **ML based models**. To be specific, the primary model will have the same features, besides the *coin flip* one, and the secondary model will be the same. Lastly, the probabilities that define the r.v. S_i are:

$$\mathbf{p} = [0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50]$$

3.11.1 Results

In figures 3.25, 3.26 and 3.27 the P&L plots of the artificial Primary model and meta-model for the Test data set are shown. Additionally, in figure 3.28, the performance of the Coin Flip Models is shown.

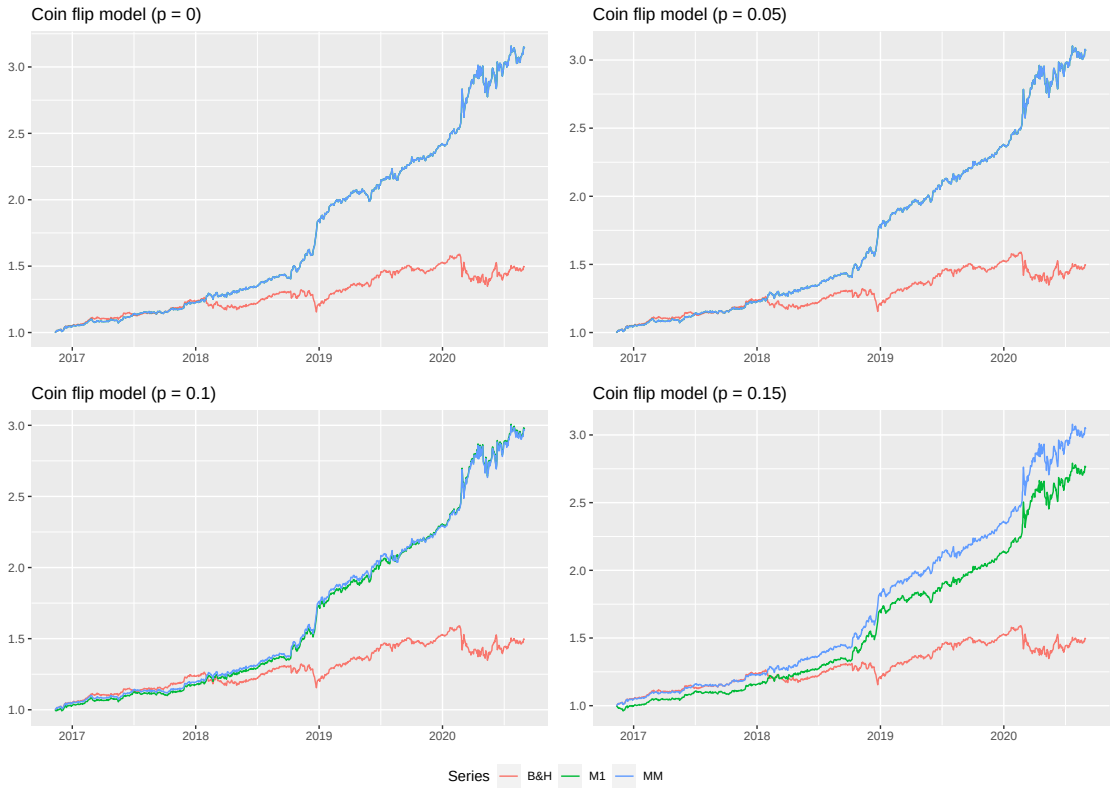


Figure 3.25: Out-of-sample P&L for $p \in \{0.00, 0.05, 0.10, 0.15\}$

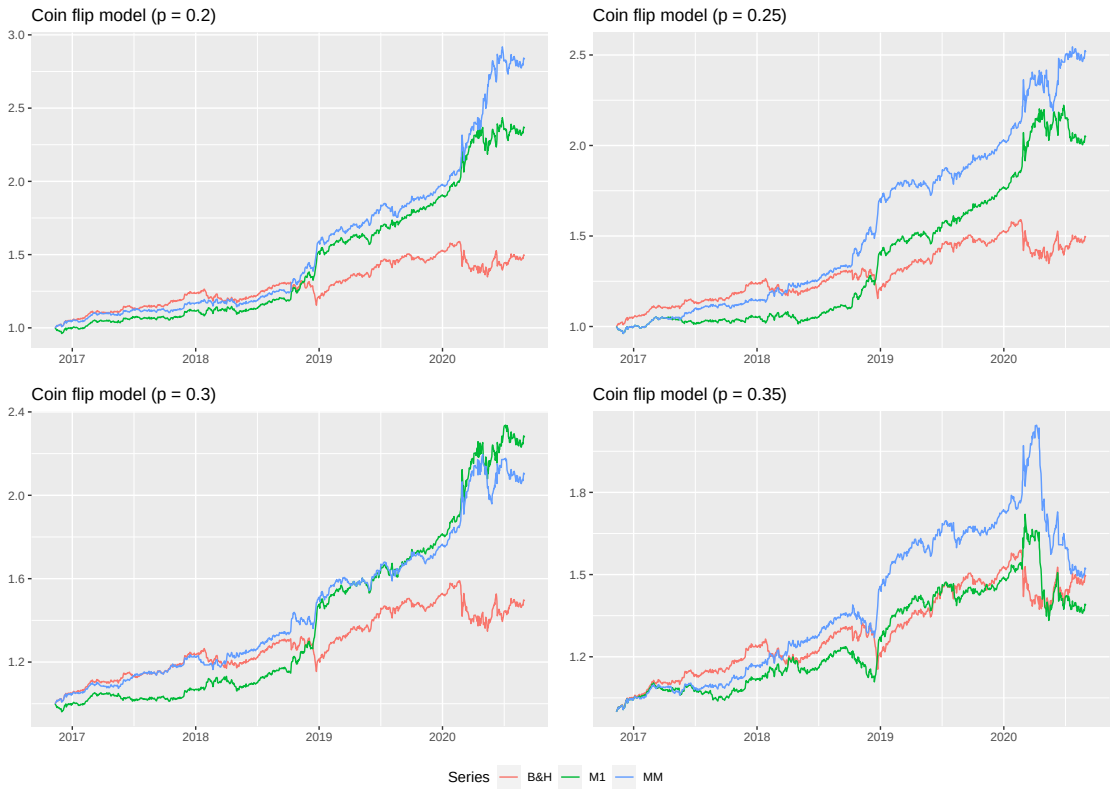
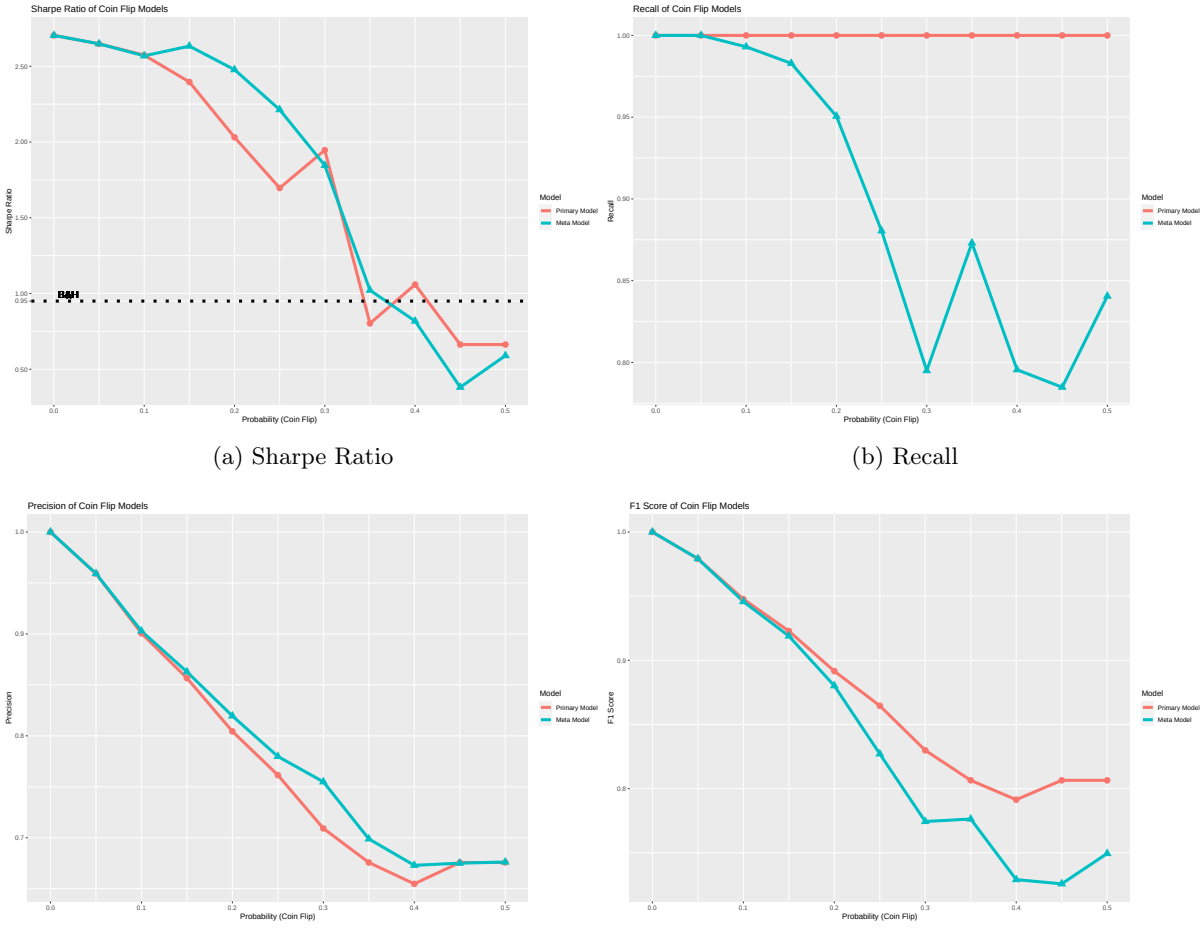


Figure 3.26: Out-of-sample P&L for $p \in \{0.20, 0.25, 0.30, 0.35\}$

Figure 3.27: Out-of-sample P&L for $p \in \{0.40, 0.45, 0.50\}$



(a) Sharpe Ratio

(b) Recall

(c) Precision

(d) F1 Score

Figure 3.28: Out-of-sample performance of Coin Flip Models

3.12 Conclusions

This chapter has explored the novel labeling technique meta-labeling. Beginning from the toy project, it was obvious that the main benefit of meta-labeling, improving the F1-Score, was hard to capture. In fact, the secondary model needed low levels of noise in order to work, besides having a primary model that was not near perfect (i.e. it made mistakes that could be corrected).

The conditions were obviously not met in the MA and ML based models. In the former, the primary model predictions were near random with precision of nearly 50%. That created a really bad model which was hard for the secondary model to correct. Even though it achieved some success in improving the SR, it was unable to beat a B&H strategy.

Apart from that, the ML based models did not have enough features with predictive power, so the results of meta-labeling are inconclusive. In fact, one would dare to say that the models did not "learn" anything which is common in systems where the models can not distinguish signal from noise. Note that this did not happen in the MA based model because even though it was almost random, decisions were made. In contrast, the ML based M1 did not have anything to base its predictions on, so it decided to always go long, since it would achieve the "best" performance for a naive classifier.

However, not everything is unfavorable. By creating better artificial primary models (coin-flip method), the hypothesis formulated in the toy project could be evaluated in a more controlled environment. In figure 3.28a, it can be seen that the SR of the meta model significantly surpasses the one of the primary model whenever the probability of the coin flip is between 0.1 and 0.3. That fits with the conditions in the toy project that made meta-labeling work: an M1 that is somewhat flawed and data with high signal-to-noise ratio.

All in all, meta-labeling applied to financial data is a tricky tool to use. First of all, a good primary model is required, which by itself is difficult to develop. Furthermore, the features of the secondary model need to have good predicting power so as to differentiate signal from noise, which in financial data is challenging as well.

Chapter 4

Fractional Differentiation

As it has been stated in previous chapters, Machine Learning models are trained over a specific data set. Then, its performance is assessed in what is called the Test data set, which is completely new to the ML model. If the metrics are similar, it can be concluded that the algorithm has "learned" and it can generalize over unseen data sets.

The phenomenon mentioned before is relevant when the data generation process is constant. In other words, if the Test data set does not follow the same distribution from the Train data set, the ML model should not be expected to behave as it did while training. That is because most ML models assume that data follows the same distribution.

This is where fractional differentiation comes into play. It will improve an ordinary technique, differentiating log-prices to compute log-returns, in order to obtain stationary data without compromising memory.

4.1 Stationary Time Series

A time series $\{x_t\}$ is said to be **strictly stationary** [1] if the joint distribution of $(x_{t_1},\ \ldots,\ x_{t_k})$ is the same as the one of $(x_{t_1+t},\ \ldots,\ x_{t_k+t})$ $\forall t$ and $\forall(t_1,\ \ldots,\ t_k)$ where $k \in \mathbb{N}$. That is, the joint distribution is time invariant.

However, as this definition of stationarity is complicated to verify empirically, a new definition will be introduced. That being said, a time series $\{x_t\}$ is said to be **weakly stationary** [1] if:

1. $\mathbb{E}[x_t^2] < \infty$
2. $\mathbb{E}[x_t] = \mu \in \mathbb{R}\ \forall t$
3. $\text{Cov}[x_t,\ x_{t-l}] = \gamma_l$, which only depends on the lag l.

With that said, let's present an Autoregressive model of order p, AR(p):

$$x_t = c + \sum_{i=1}^{p} \theta_i x_{t-i} + \epsilon_t$$

$$x_t = c + \sum_{i=1}^{p} \theta_i B^i\ x_t + \epsilon_t$$

$$\phi(B)\ x_t = c + \epsilon_t$$

Where $\epsilon_t \sim$ i.i.d. $N(0, \sigma^2)$, B is the backshift operator ($Bx_t = x_{t-1}$) and $\phi(z) = 1 - \theta_1 \cdot z^1 - \ldots - \theta_p \cdot z^p = 0$ is the characteristic equation.

The process will have a **unit root** (integrated of order 1 - $I(1)$) when $z = 1$ is a root of multiplicity 1 of the characteristic equation. When $z = 1$ is a root of multiplicity r, the process is integrated of order r - $I(r)$.

As a side note, proposition **3.1.4** (page 75) of [2] states that a unique stationary solution $\{x_t\}$ of an AR(p), with characteristic equation defined by $\phi(z)$, exists if and only if

$$\phi(z) = 1 - \theta_1 \cdot z^1 - \ldots - \theta_p \cdot z^p \neq 0 \quad \forall z \text{ s.t. } |z| = 1$$

Therefore, **if a process** following an AR(p) model **has a unit root** ($|z| = 1$), **it will not be stationary**.

4.1.1 Augmented Dickey-Fuller test

Let $\{y_t\}$ be a process that can be fitted a model defined by $\phi(B)\ y_t = \epsilon_t$, where $\phi(z)$ has one unit root:

$$\phi(z) = (1 - \theta_1 \cdot z^1 - \ldots - \theta_p \cdot z^p) \cdot (1 - z)$$

Then, as removing the unit root is a necessary (but not sufficient) condition to achieve stationarity, the first step towards it is differencing:

$$x_t := (1 - B) \cdot y_t$$

This transformation will leave one with a process $\{x_t\}$ that does not have a unit root. While this does not mean that $\{x_t\}$ is weakly stationary, in the financial industry it is often enough to have data without unit roots. Therefore, as the main goal of this thesis is to serve practitioners, from now on processes without unit roots will be considered stationary.

This is where the Augmented Dickey-Fuller (ADF) test comes into play. This test, which belongs to the family of unit root tests, will help one determine if a time series is stationary (from the financial standpoint).

Having said that, the first step towards introducing the ADF test is to consider the time series $\{x_t\}$ that comes from an AR(p). Then, rewriting $x_t - x_{t-1}$ in terms of the previous differences, as Nielsen did in [11] one obtains:

$$\phi(B)\ x_t = \left(1 - \sum_{i=1}^{p} \theta_i B^i\right) x_t = c + \epsilon_t$$

$$x_t = c + \sum_{i=1}^{p} \theta_i x_{t-i} + \epsilon_t$$

$$x_t - x_{t-1} = c + \left(\sum_{i=1}^{p} \theta_i - 1\right) x_{t-1} + \left(\sum_{i=2}^{p} \theta_i\right) \cdot (x_{t-2} - x_{t-1}) + \ \ldots + \ \theta_p \cdot (x_{t-p} - x_{t-(p-1)}) + \epsilon_t$$

$$\Delta x_t = c + \pi \cdot x_{t-1} + c_1 \cdot \Delta x_{t-1} + \ \ldots + \ c_{p-1} \cdot \Delta x_{t-(p-1)} + \epsilon_t$$

Where $\pi = \sum_{i=1}^{p} \theta_i - 1$ and $c_j = -\sum_{i=j+1}^{p} \theta_i$

Noting that $\phi(1) = -\pi$, then $z = 1$ is a root if $\pi = 0$. Consequently, the Augmented Dickey-Fuller test is defined:

$$H_0 : \pi = 0$$
$$H_a : \pi < 0$$

At this point, if a time series does not have a unit root, then H_0 will be rejected and it will be concluded that the time series is stationary.

Regarding the order p of the AR model, the R package **tseries** [16] will be used, which sets $p = \lfloor \sqrt[3]{\mathrm{T} - 1} \rfloor$ (T = length of the time series - $x_1, \ \ldots, \ x_{\mathrm{T}}$) as default. As they the authors of **tseries** state in [16], this value of p corresponds to "[...] the suggested upper bound on the rate at which the number of lags should be made to grow with the sample size for the general ARMA(p,q) setup".

4.2 Fractional differentiation

In financial data, it is often the case that the log-prices time series $\{y_t\}$ is not stationary. As it was mentioned in the previous section, a differentiation of order 1 is computed and the time series of log-returns $x_t = (1 - B) \cdot y_t = y_t - y_{t-1}$ is usually stationary. However, in the process, the "memory" of the time series is erased (see figure 4.1).

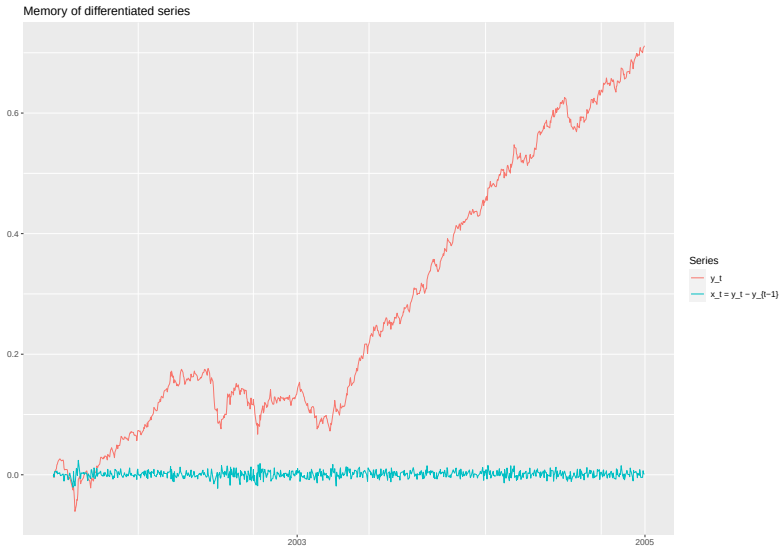


Figure 4.1: Memory of differentiated series

That is when fractional differentiation comes into play. Firstly introduced by Hosking in [9], it consists of defining the differentiated time series as $x_t^d = (1 - B)^d \; y_t$, $d \in (0, 1)$, where:

$$(1 - B)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k = \sum_{k=0}^{\infty} \frac{\prod_{i=0}^{k-1} (d - i)}{k!}$$
$$= \sum_{k=0}^{\infty} B^k (-1)^k \prod_{i=0}^{k-1} \frac{d - i}{k - i} = \sum_{k=0}^{\infty} w_k B^k$$

Note that:

- $w_k = -w_{k-1} \cdot \frac{d-(k-1)}{k}$ and $w_0 = 1$ (see figure 4.2).
- $d = 1 \Rightarrow x_t^1 = x_t$ - log-returns
- $d = 0 \Rightarrow x_t^0 = y_t$ - log-prices

Due to data limitations, fractional differentiation can not be computed using an infinite series of weights $\{w_k\}_{k=0}^{\infty}$. Hence, to mend that, a technique introduced in [5] called "Fixed-width window fractional differentiation" (FFD) will be defined. The purpose of this technique is to allow all observations to have the same amount of memory. That is, the discrepancy of terms in of the expression of x_T^d (uses the weights $w_0, \; \ldots, w_{T-1}$) and x_{T-l}^d (uses $w_0, \; \ldots, w_{T-l-1}$).

Consequently, if one did not solve this, some observations would have more "memory" than others, which is not desirable. Thus, by setting a threshold (e.g. $\tau = 10^{-4}$), the following set of weights can be defined:

$$\widetilde{w}_k = \begin{cases} w_k & \text{if } k \leq l^* \\ 0 & \text{if } k > l^* \end{cases}$$

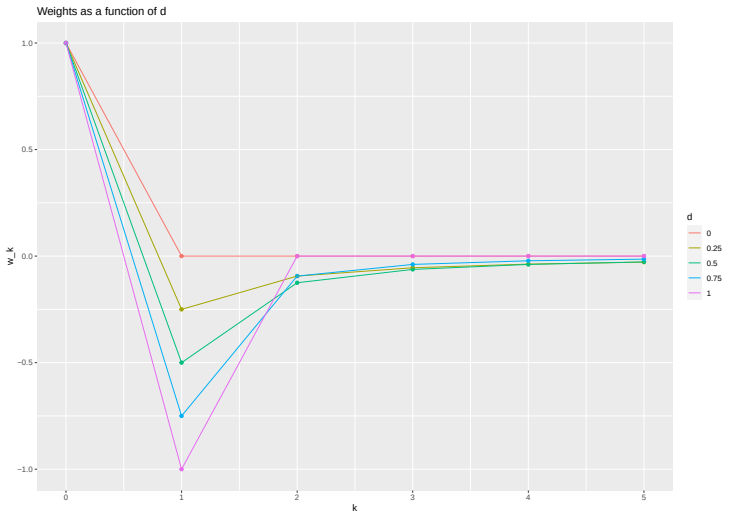


Figure 4.2: Weights as a function of d

Where l^* is such that $|w_{l^*}| > \tau$ and $|w_{l^*+1}| \leq \tau$. Note that the FFD differentiation will be valid only for $t \geq l^* + 1$ and the same set of weights, $\{w_k\}_{k=0}^{l^*}$, will be used for all estimates.

Finally, by setting a fixed τ, the FFD differentiation can be expressed as:

$$x_t^d = \sum_{k=0}^{\infty} \widetilde{w}_k B^k y_t = \sum_{k=0}^{l^*} \widetilde{w}_k B^k y_t =: \phi_d(B)\ y_t \quad (t \geq l^* + 1) \tag{4.1}$$

To clarify everything, figure 4.3 shows x_t^d for different values of d.

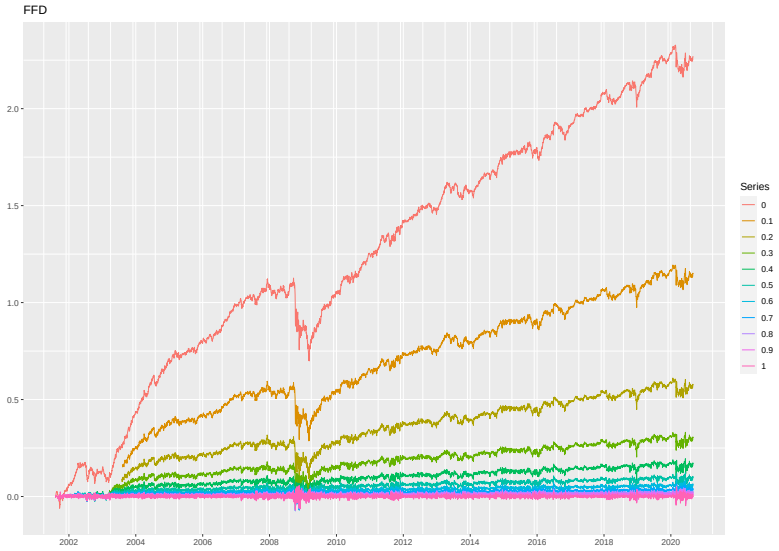


Figure 4.3: FFD time series - $\tau = 10^{-4}$

Now that the corresponding x_t^d time series have been computed, it is relevant to introduce again the ADF test. The objective is to find the minimum d, d^*, such that one can conclude that $x_t^{d^*}$ is stationary. In figure 4.4, the ADF test statistic is shown (using lag $p = \lfloor \sqrt[3]{\mathrm{T}_d - 1} \rfloor$).

Figure 4.4 also shows that $d = 0.2$ is sufficient to pass the ADF test and conclude that the time series is stationary.

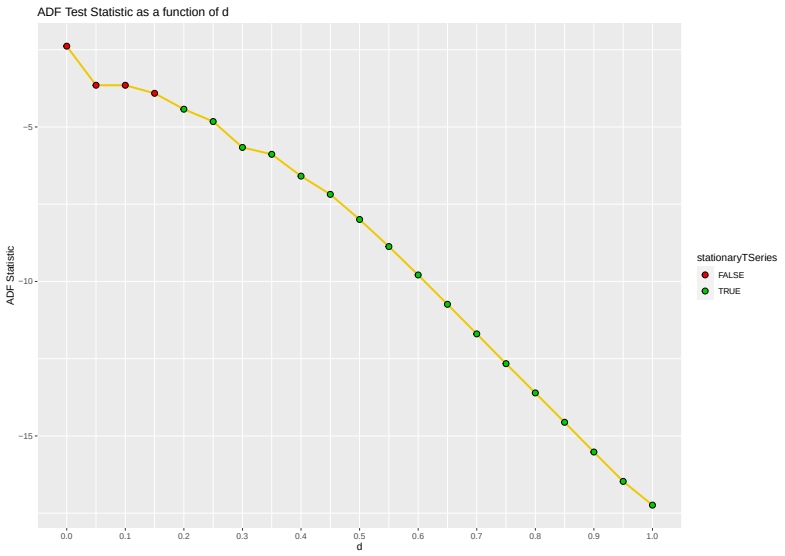


Figure 4.4: ADF statistic for FFD time series (p-value = 0.01)

4.2.1 Inverse of differentiation

In this section, the inverse of differentiation will be introduced since it is useful to go from one time series to the other. First of all, for convenience, let's define x_t^d for $1 \leq t \leq l^*$:

$$x_t^d = \begin{cases} \phi_d(B)\ y_t & \text{if } t \geq l^* + 1 \\ \sum_{k=0}^{t-1} \widetilde{w}_k \cdot B^k\ y_t & \text{if } 1 \leq t \leq l^* \end{cases} \tag{4.2}$$

Alternatively, if one defines $y_t = 0$ for $-l^* \leq t \leq 0$, equation 4.2 can be expressed as:

$$x_t^d = \phi_d(B)\ y_t \quad (t \geq 1)$$

Now that x_t^d has been defined $\forall t \geq 1$, the inverse of differentiation comes into play this way:

$$\begin{aligned} \frac{1}{\phi_d(B)}\ x_t^d &= y_t \\ \frac{1}{\sum_{k=0}^{l^*} \widetilde{w}_k \cdot B^k}\ x_t^d = \sum_{k=0}^{\infty} c_k \cdot B^k\ x_t^d &= \sum_{k=0}^{T-1} c_k \cdot B^k\ x_t^d = y_t \end{aligned} \tag{4.3}$$

Equation 4.3 shows that to compute y_t as a function of x_t^d, only a finite number of terms are needed. That, is inherent to the definition of x_t^d $(1 \leq t \leq T)$, which for $t < 1$ is 0. Therefore, it will suffice to compute the first T coefficients of the resulting polynomial. To do that, successive polynomial long divisions (see figure 4.5 for the first 2 terms) will be used until the first T terms are retrieved.

$$\begin{array}{llll} 1 & & & \widetilde{w}_0 + \widetilde{w}_1 B + \widetilde{w}_2 B^2 + \ldots + \widetilde{w}_{l^*} B^{l^*} \\ \hline 1 & \widetilde{w}_1/\widetilde{w}_0 B & \widetilde{w}_2/\widetilde{w}_0 B^2 \ldots & \widetilde{w}_{l^*}/\widetilde{w}_0 B^{l^*} \\ & -\widetilde{w}_1/\widetilde{w}_0 B & -\widetilde{w}_2/\widetilde{w}_0 B^2 \ldots & -\widetilde{w}_{l^*}/\widetilde{w}_0 B^{l^*} \\ & -\widetilde{w}_1/\widetilde{w}_0 B & -\widetilde{w}_1^2/\widetilde{w}_0^2 B^2 \ldots & -\widetilde{w}_1 \widetilde{w}_{l^*-1}/\widetilde{w}_0^2 B^{l^*} & -\widetilde{w}_1 \widetilde{w}_{l^*}/\widetilde{w}_0^2 B^{l^*+1} \\ & \ldots & \ldots & \ldots & \ldots \end{array} \qquad \begin{array}{l} \\ \\ \frac{1}{\widetilde{w}_0} - \frac{\widetilde{w}_1}{\widetilde{w}_0^2} B + \ldots \end{array}$$

Figure 4.5: First 2 terms of polynomial long division

4.3 Toy Project

In the previous sections, the *memory vs. stationarity* dilemma has been presented. On top of that, through fractional differentiation, stationarity has been achieved without totally giving up memory. Also, a method has been presented to differentiate and un-differentiate time series in a robust way.

That being said, the problem that this toy project attempts to solve will be presented. Given a time series $\{y_t\}$, and Λ_t representing all the values $\{y_t\}$ has taken up to time t, the aim of this project is to give a forecast $\widehat{y}_{t+1}$ given Λ_t. That way, by sequentially giving 1-day forecasts, a new time series will be generated.

To illustrate the effect fractional differentiation can have in the predicting power of a model, two models will be designed. One will have fully differentiated features and labels, and the other will use fractionally differentiated ones.

The last phase of the toy project will be computing prediction errors to compare the performance of the two models.

4.3.1 Synthetic data

Striving to create a log-prices time series with memory, the generation was the following:

$$y_t = \frac{1}{5} \cdot \sum_{i=1}^{5} y_{t-i} + \mu + \epsilon_t \quad (t \geq 6) \tag{4.4}$$

where $\mu = 0.005$, $\epsilon_t \sim N(0,\ \sigma = 0.01)$ and $y_{1:5} = \{0.000,\ 0.075,\ 0.150,\ 0.225,\ 0.300\}$.

A few remarks:

- $\left(1 - \frac{1}{5}\sum_{k=1}^{5} B^k\right) y_t = \phi(B)\ y_t = \mu + \epsilon_t$ has a unit root since $z = 1$ is a root of $\phi(z)$. Therefore, $\{y_t\}$ is not stationary and will have to undergo some degree of differentiation.
- Rewriting 4.4, one can obtain:

$$\begin{aligned} x_t := y_t - y_{t-1} = \frac{1}{5} \cdot \sum_{i=1}^{5} (y_{t-i} - y_{t-1}) + \mu + \epsilon_t = \frac{1}{5} \cdot \sum_{i=2}^{5} (y_{t-i} - y_{t-1}) + \mu + \epsilon_t = \\ = \frac{1}{5} \cdot \sum_{i=2}^{5} (y_{t-i} - y_{t-i+1} + \ldots + y_{t-2} - y_{t-1}) + \mu + \epsilon_t = \\ = -\frac{1}{5} \left(x_{t-4} + 2x_{t-3} + 3x_{t-2} + 4x_{t-1}\right) + \mu + \epsilon_t \end{aligned} \tag{4.5}$$

As it can be seen from equation 4.4, in order to determine the value of y_t, it is crucial to know the preceding 5 values of the time series, while x_t depends on the previous 4 values (see equation 4.5).

One big difference between $\{y_t\}$ and $\{x_t\}$ is that the latter is stationary (checking that the roots of the characteristic equation lie outside the unit circle is left to the reader). However, both processes have the same Gaussian white noise ($\{\epsilon_t\}$) which will be interesting to see how ML algorithms cope with it. To be specific, $\{y_t\}$ goes to infinity as $t \to \infty$, while $\{x_t\}$ is stationary, which means that as time passes $\{y_t\}$ becomes less noisy.

As for the length of the time series, $\{y_t\}$ will have 3000 observations, and 15% of them (450 obs.) will be saved for testing purposes.

4.3.2 Fractional differentiation of y_t

In figure 4.6 and 4.7 the differentiated time series are shown. Note that the method used is the one defined previously: fixed-width window fractional differentation method (FFD). Also, in order to determine the minimnum d^* that makes the time series stationary, figure 4.8 and table 4.1 are presented (only training data has been used).

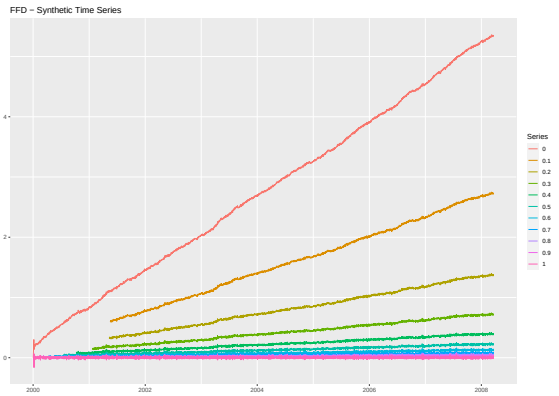


Figure 4.6: FFD of y_t with $\tau = 1e-4$

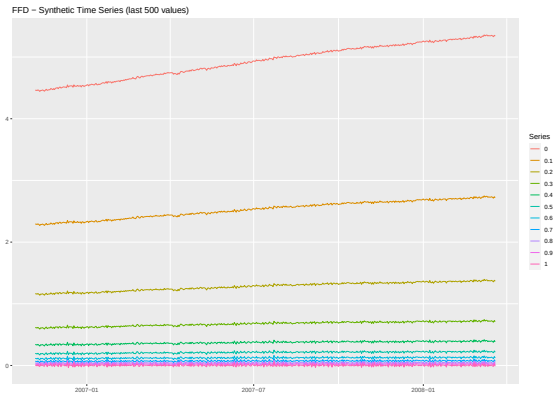


Figure 4.7: FFD of y_t (Zoomed in) with $\tau = 1e-4$

Table 4.1: ADF Test (with trend and drift) Statistics of x_t^d - Train

d	Lag order (p)	ADF Statistic
0.00	13	-3.130
0.05	12	-3.130
0.10	12	-3.430
0.15	12	-3.850
0.20	12	-4.254
0.25	12	-4.856
0.30	12	-5.352
0.35	13	-6.179
0.40	13	-6.939
0.45	13	-7.525
0.50	13	-8.141
0.55	13	-8.967
0.60	13	-9.819
0.65	13	-10.637
0.70	13	-11.515
0.75	13	-12.388
0.80	13	-13.265
0.85	13	-13.944
0.90	13	-14.684
0.95	13	-15.410
1.00	13	-16.077

Thus, now that it is known that $d^* = 0.2$, whenever the FFD time series is mentioned, the order of differentiation will be d^*.

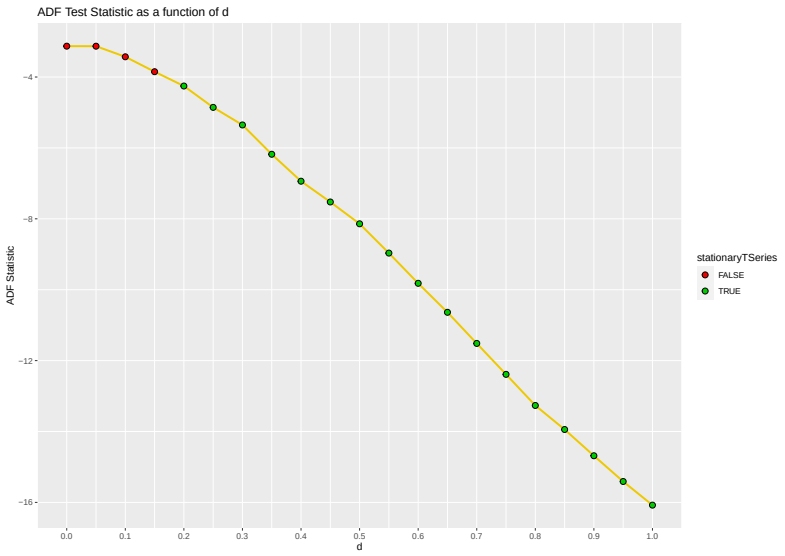


Figure 4.8: ADF Statistics of x_t^d - Train

4.3.3 Models

With the purpose of evaluating how FFD works in this toy project, three models will be used:

1. **Naive Model** (Benchmark): Simplest model than can be built. It will predict $\widehat{y}_{t+1} = y_t$
2. **FFD Model:** It will use $\{x_t^{d^*}\}$ to predict $\{y_t\}$.
3. **Returns Model:** It will use $\{x_t\}$ ($x_t \equiv x_t^1 = y_t - y_{t-1}$) to predict $\{y_t\}$.

The FFD and Returns models will be similar in the sense that they will use the same artificial neural network (ANN) structure and similar features. Both models will use the architecture seen in figure 4.10 but with 5 input units. Also, it is important to mention that they will use ELU (Exponential Linear Unit) as activation function.

Regarding the features and labels, due to the construction of the time series, the last 5 values will be used to predict the next one:

- **Features:** x_{t-5}^d, x_{t-4}^d, x_{t-3}^d, x_{t-2}^d and x_{t-1}^d
- **Labels:** x_t^d

Lastly, recall that 450 observations out of 3000 will be used for testing purposes. This does not mean that 2550 observations will be used for training, since using the FFD method implies that the series will not be defined up to a point. That is caused by the fact that all observations should have the same amount of memory.

4.3.4 Sequential forecasts

Considering that one is attempting to predict log-prices and differentiated features are being used, an emphasis should be put in the un-differentiation, which can be expressed (see equation 4.3) as:

$$y_t = \sum_{k=0}^{T-1} c_k \cdot B^k \; x_t^d \qquad (T = 3000)$$

Noting n_0 as the time stamp where predictions start, the predicted time series will be defined sequentially as:

$$\widehat{y}_t = c_0 \cdot \widehat{x}_t^d + \sum_{k=1}^{T-1} c_k \cdot B^k \; x_t^d = c_0 \cdot \widehat{x}_t^d + \sum_{k=1}^{T-1} c_k \cdot x_{t-k}^d \qquad (t \geq n_0)$$

Here one needs to remember that, by definition, $x_t^d = 0$ for $t \leq 0$.

4.3.5 Results

In the previous section, it has been shown how to obtain a new time series with the predictions. In order to compare the final time series in the test data set ($t \geq n_0$), three metrics will be used: Root-mean-square error (RMSE), Mean absolute percentage error (MAPE) and Median absolute deviation (MAD). The metrics mentioned will be defined after introducing the following parameters: $e_k = y_k - \widehat{y}_k$ and $\widetilde{e} = \text{median}(e_k)$.

$$\text{RMSE} = \sqrt{\frac{\sum_{k=n_0}^{T}(e_k)^2}{T - n_0 + 1}}$$

$$\text{MAPE} = \frac{1}{T - n_0 + 1} \cdot \sum_{k=n_0}^{T} \left| \frac{e_k}{y_k} \right|$$

$$\text{MAD} = \text{median}(|e_k - \widetilde{e}|)$$

With the metrics defined, the results in the test data set are the following (improvement over the benchmark is shown in parenthesis):

Table 4.2: Error metrics of the models considered - Test

	FFD model	Naive model	Returns model
RMSE	0.010184 (21.63%)	0.012995	0.013126 (-1.01%)
MAPE	0.001608 (23.90%)	0.002113	0.002148 (-1.66%)
MAD	0.006606 (24.48%)	0.008747	0.009192 (-5.09%)

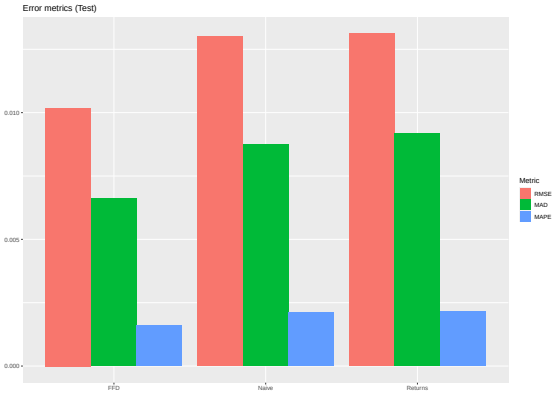


Figure 4.9: Error metrics of the models considered - Test

4.3.6 Conclusions

From the results shown in table 4.2 and figure 4.9, it can be implied that the FFD model has performed best across all error metrics. Also, the difference between the Naive and Returns Model is negligible, so it can be concluded that the Returns Model has not been able to detect the pattern behind the observations.

This goes to say that full differentiation is unnecessary. To be more specific, the FFD method has been able to retain some of the structure of $\{y_t\}$. Consequently, as mentioned in section 4.3.1, it has not suffered as much from the Gaussian white noise so predictions are bound to be better.

The next step is to apply the same method to a real financial time series. Although this controlled project has been successful in an FFD sense, it remains to be seen whether fractional differentiation of financial time series helps with forecasts.

4.4 Financial data

The data used will be the log-prices of the S&P500 from 2005-01-01 to 2019-09-01. It will include the Open (y_t^{Open}), High (y_t^{High}), Low (y_t^{Low}) and Close (y_t^{Close}). That is, the log-price it opens/closes and the highest/lowest values during the daily trading session.

As in the project of section 4.3, the goal of the models will be to make a 1-day forecast. In particular, with the OHLC (Open, High, Low, Close) data of day t, the models will predict the value the stock closes at time $t+1$. To do that, the features and labels will be the following:

- **Features:** y_t^{Open}, y_t^{High}, y_t^{Low} and y_t^{Close}
- **Labels:** y_{t+1}^{Close}

These features, as in [7], have been chosen because they provide information at different times of the day. That is, they represent the daily state of the time series.

4.4.1 Division Train/Test data sets

The data will be divided into a training and test data set. The distribution is the following:

Table 4.3: Division of Train/Test

	FFD model	Returns model
Start date (Train)	2006-10-06	2005-01-04
End date (Train)	2017-07-12	2017-07-12
Start date (Test)	2017-07-14	2017-07-14
End date (Test)	2020-08-28	2020-08-28
n_{Train}	2709	3152
n_{Test}	788	788

As it can be seen in table 4.3, both the FFD and the Returns models have 788 observations in the test data set. However, in the train data set, the FFD has 443 fewer observations. This can be explained via equation 4.1, since by **setting a threshold** $\tau = 10^{-4}$, an $l^*(\tau, d)$ exists such that the FFD is valid for $t \geq l^* + 1$. In particular, in the case of returns $d = 1$, and thus, $l^*(\tau, d = 1) = 1$ since the first observation does not have a preceding value.

Also note that n_{Test} has been kept equal because the error metrics will be evaluated in this data set. Consequently, the conditions should be the same to avoid misleading results.

4.5 Models

As in section 4.3, three models will be used: **Naive**, **FFD** and **Returns** model. The features will be the corresponding time series shown in section 4.4, but differentiated. To be specific, the FFD model will use the fractionally differentiated features with the minimum order ($d^* = 0.25$) that makes the 4 OHLC time series stationary (in training). Conversely, the **Returns** model will use fully differentiated features ($d = 1$).

The **FFD** and **Returns** models will use an ANN with the structure seen in figure 4.10. As in section 4.3, the activation function will be the Exponential Linear Unit (ELU).

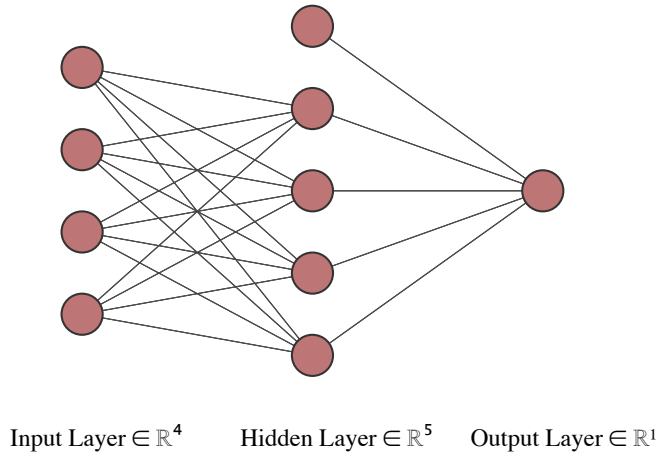


Figure 4.10: ANN used in the FFD and Returns Models

4.6 Data normalization

As a means to provide features to the neural network with the same range, all 4 features have been normalized:

$$z_t = \frac{y_t - y_{\min}}{y_{\max} - y_{\min}}$$

It should be pointed out that $y_{\max}$ and $y_{\min}$ have been computed in the **train** data set. It is extremely important to consider this, because if one were to use the maximum/minimum of the whole data set, look-ahead bias would have occurred. To be more specific, if the test observations started at time t_0 and the time stamp when the maximum is achieved is $t_0 + m > t_0$, then the forecast at time t_0 would have used information from $t_0 + m$, which is impossible.

That being said, in the train data set, $z_t \in [0, 1]$, while in the test data set, features are not guaranteed to be in $I = [0, 1]$. However, working under the assumption that the complete time series is stationary, values can not be far off.

It should also be brought up that since labels are not normalized, the predicted values will not have to undergo "un-normalization".

4.7 Results

As in section 4.3.4, the sequential forecasts have been calculated and the final result is 3 time series, which are forecasts of the close of the S&P500:

$$\begin{aligned} y_t^{\text{Naive}} \quad & (n_0 + 1 \leq t \leq T) \\ y_t^{\text{FFD}} \quad & (n_0 + 1 \leq t \leq T) \\ y_t^{\text{Returns}} \quad & (n_0 + 1 \leq t \leq T) \end{aligned}$$

Where $t = n_0$ is the time stamp that marks the start of the test data set. Therefore, the prediction will be of the log-price close of time $t + 1 = n_0 + 1$

Table 4.4: Error metrics of the models considered in the Test data set

	FFD model	Naive model	Returns model
RMSE	0.0187200 (-0.78%)	0.0185758	0.0181539 (2.27%)
MAPE	0.0017909 (-1.91%)	0.0017573	0.0017179 (2.24%)
MAD	0.0047790 (-6.85%)	0.0044728	0.0044403 (0.73%)

In table 4.4 the error metrics for the three models considered are shown, as well as the improvement over the Naive model (displayed in parentheses). For a graphical representation refer to figure 4.11. In addition, figures 4.12, 4.13 and 4.14 present the forecasted time series for different periods of time.

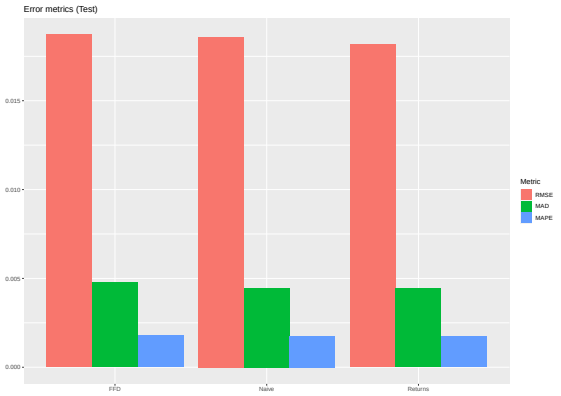


Figure 4.11: Error metrics of the models considered - Test

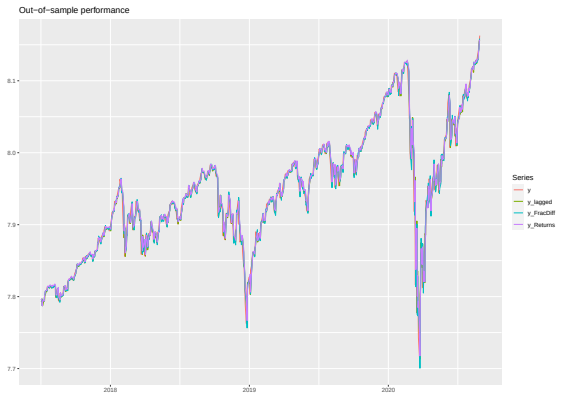


Figure 4.12: Forecasted time series - Test

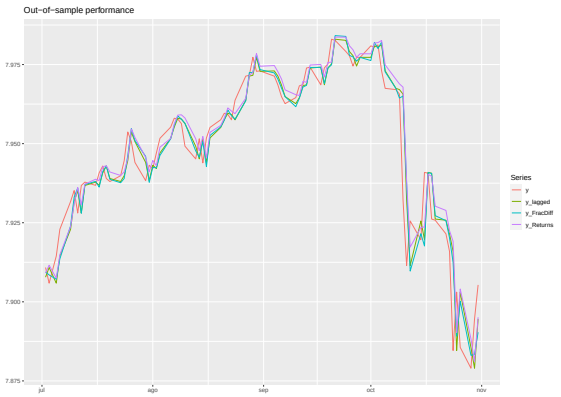


Figure 4.13: Results 2018-07/2018-10

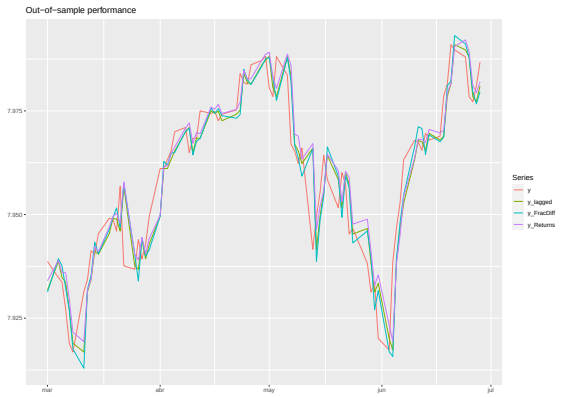


Figure 4.14: Results 2019-03/2019-06

4.8 Conclusions

This chapter has explored how fractional differentiation can be applied to financial data. Through a toy project it was observed that in a process that had memory, i.e. future values depended on the past, one could achieve better forecasts by using fractionally differentiated features instead of a full differentiation.

However, when it came to financial data the results were not as one would have expected. As shown in section 4.7, the FFD model was the worst performer among the three models. In particular, it should be pointed out that the metrics of the FFD model were slightly worse than the ones of the Naive model. In contrast, the performance of the Returns model was a bit better than the Naive.

On the grounds that the performance was so similar, the differences are attributed to randomness. That is, one should not expect the FFD or Returns model to outperform or underperform the Naive Model, indicating that the features used were not useful at making 1-day forecasts.

All in all, this chapter has encountered a similar situation than in meta-labeling. Fractional differentiation is a tool that should be used whenever one knows that the underlying time series has memory. That is, it is not a plug and play solution and should only be implemented if the current value of the time series relative to previous ones (memory) is helpful at giving forecasts.

Chapter 5

Data parsing as bars

The previous chapters used low frequency data. In particular, daily data was used which just means that after every trading day a single observation is gathered. On the other side of the spectrum rests high frequency data, which can be viewed as the purest form of financial data.

However, raw data does not mean that everything will work better. To feed this data to ML algorithms, it must be preprocessed in a way that superfluous information is discarded and useful parts are kept.

Before diving into the actual data manipulation, a brief explanation will be given to understand better where the term "data bars" come. As López de Prado states in [5], most ML algorithms assume a table representation of the extracted data and the tables' rows are often referred by practitioners as "bars".

5.1 Data bars

In section 2.7 of chapter 2, it was seen that raw financial data comes in ticks, and thus, inhomogenous in time. In order to structure data, samples of the price will be recorded whenever a certain event happens (time elapsed, price change, etc.). In consequence, different events will give rise to different type of bars.

A useful concept that will help understand how data bars arose is the concept of stochastic clock, proposed by Clark in [3]. This clock will only "show the time" when certain events happen. For instance, a volume clock will only show the time when M_{volume} units are exchanged. Hence, volume bars will be recorded whenever the volume clock "shows the time".

In the following subsections, the bars that Marcos López de Prado designated as standard in [5] will be presented.

Time bars

These bars consist of sampling the price every M_{time} minutes (e.g. every 1, 5, 30 minutes or longer). Note that daily data lays in this category since it records the price of an asset once a day.

One of the drawbacks of time bars is the fact that they ignore periods of high activity. That is, if one was sampling every 30 minutes and the bulk of trading occurred during the first and last 15 minutes of the trading session, then information would be lost.

Tick Bars

As the name implies, the sampling will occur every certain amount of ticks (M_{ticks}). Since ticks are directly related with transactions, during periods of high activity more samples will be gathered.

The problem with sampling data using ticks is the lack of control over the number of stocks traded. To put it another way, if one wanted to buy 5 stocks and there were 5 separate offers of size 1 that

matched, then the whole operation would count as 5 transactions. However, if there was an offer of size 5 that matched, it would be recorded as 1 transaction, even though the same number of stocks have been exchanged.

That is why López de Prado states in [5] that order fragmentation introduces arbitrariness in the number of ticks.

Volume bars

Volume bars will sample the price every time M_{volume} stocks are exchanged. This will certainly help with the issue outlined with tick bars. In this type of bars the amount of transactions will not matter since it only cares about the actual amount of securities traded.

Also, note that one of the great virtues of volume bars is the fact that they will sample the price correctly in moments of high activity. In particular, whenever a lot of stocks are traded, plenty of bars will be recorded, while in a low activity environment few bars will be gathered.

Dollar bars

Before introducing these bars one should know what dollar volume is: $d_t := v_t \cdot p_t$. At first glance, one could infer that sampling every time a certain dollar volume (M_{dollar}) is traded will lead to bars similar to volume ones.

That is true to a certain extent because if p_t stays fairly constant, then accounting for a constant, they will sample in a similar fashion. However, if the stock presents high volatility, volume bars will under sample periods of high prices. To be specific, dollar bars can distinguish 300 stocks being exchanged at 80\$ ($d_t = 24,000$\$) rather than at 70\$ ($d_t = 21,000$\$).

5.2 Data

The security that will be used for this chapter is called **IVE**, which trades in the NYSE (New York Stock Exchange) Arca market. This asset tracks the S&P500, which is an index that comprises the 500 most important US stocks.

Having said that, the data used will be the tick data of IVE from 2013-01-02 to 2013-12-31. It contains 550,697 rows that include information on: price, bid, ask, and volume. To get an idea of the general structure of the data used, refer to table 2.1 which shows a few ticks from day 2013-01-02.

5.3 Sampling

Having presented the tick data that will be used, the next step is determining the hyperparameters M_{ticks}, M_{volume}, M_{dollar} and M_{time}, which are responsible of the sample gathering. Once the samples are collected the daily average of bars ($\bar{n}^{\text{daily}}$) will be calculated. Hence, every M directly determines $\bar{n}^{\text{daily}}$.

Before showing the values of M used, one should bear in mind that processing the huge dataset and creating the bars takes a considerable amount of time ($\approx$ 3 hrs). Therefore, "brute force" is not feasible to choose the values of M. However, what has been done is the following:

- Choose a sensible M. For example, López de Prado in [6] uses $\frac{1}{50}$ of the average daily volume as M_{volume}.
- Shift M until the fitted pdf of log-returns resembles more of a normal pdf (see section 5.4 for further details).

Having said that, the final hyperparameters chosen are shown in table 5.1.

Table 5.1: Sampling hyperparameters

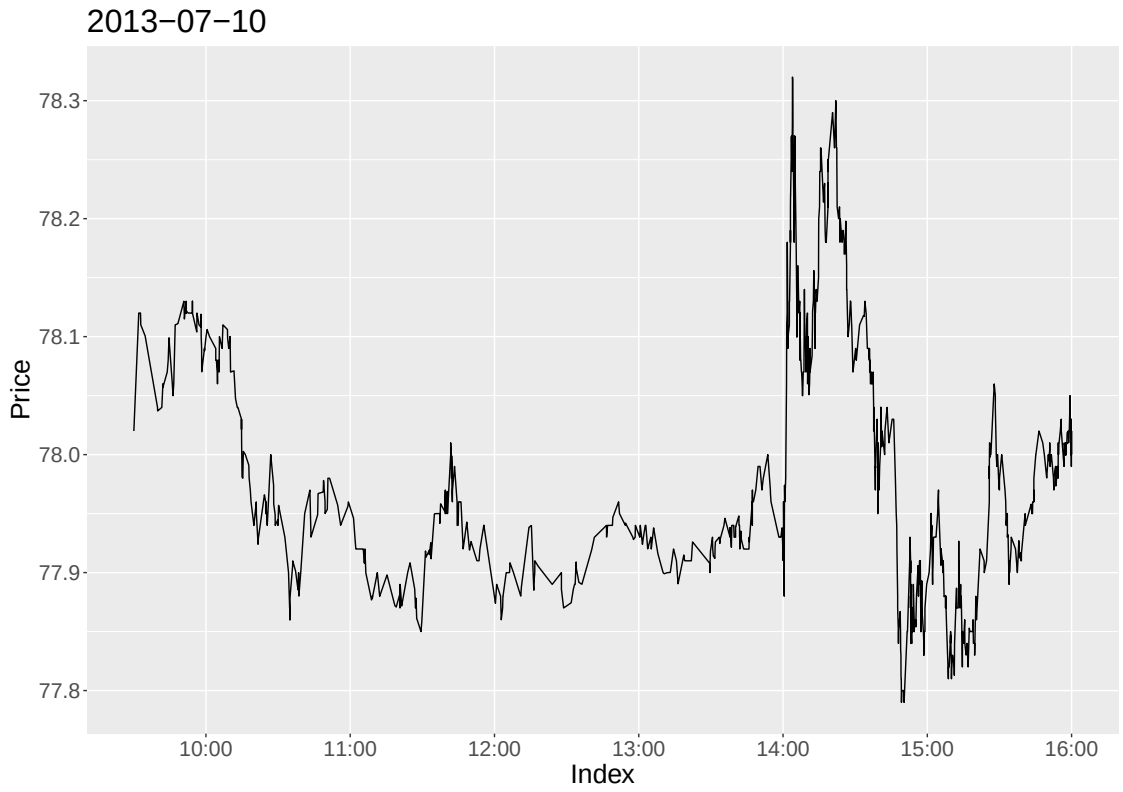
	Tick	Volume	Dollar	Time
M	60 ticks	8,000 stocks	800,000$	8 min.
$\bar{n}^{\text{daily}}$	36.42	73.77	49.20	49.93

Figure 5.1 shows three trading days and the samples that were extracted using tick, volume, dollar and time bars. Additionally, in figure 5.2 the samples of day 2013-07-10 are presented. This last image is really important because it symbolizes the data bars via their internal clock, i.e., one unit in the x-axis represents the occurrence of a certain event (stocks exchanged, ticks, etc.).

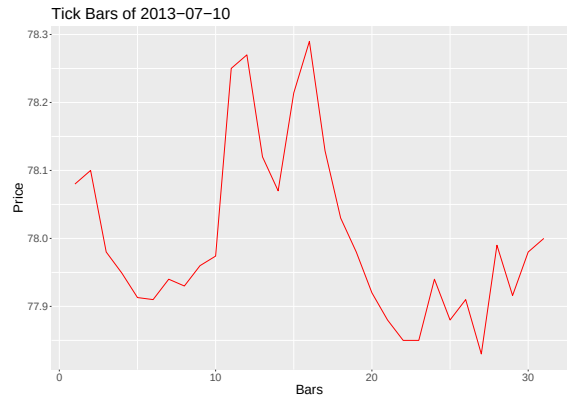
Furthermore, since it is of the utmost importance to understand how sampling occurs, a few remarks on figures 5.1 and 5.2 will be made:

- Tick bars, although not equally spaced as time bars, seem to be more distributed as time passes compared to volume and dollar bars.
- Volume and dollar samples are concentrated on periods of high volatility (downswings and upswings).
- In figure 5.2 it can be seen that tick, volume and dollar bars undersample periods where the price goes sideways, while they oversample the period from 14:00 to 15:00, where the bulk of activity was observed.

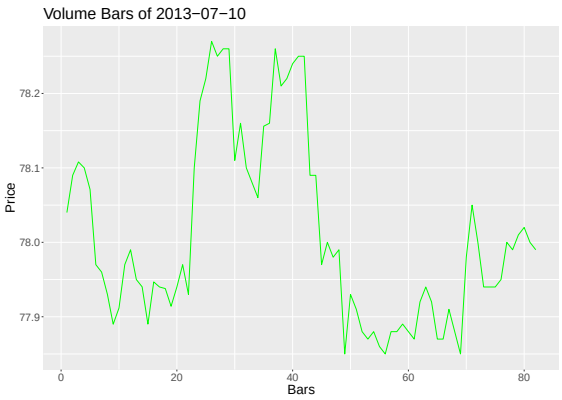
Figure 5.1: Tick, volume, dollar and time bars



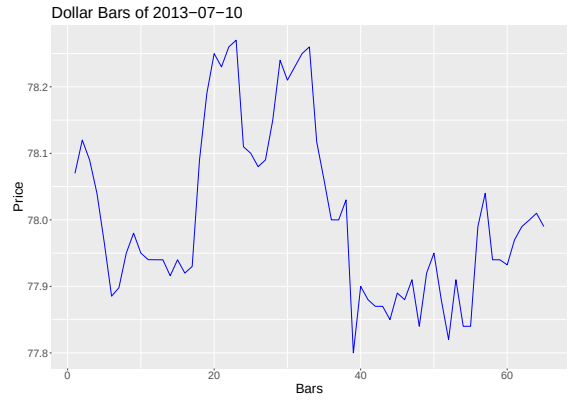
(a) Tick data



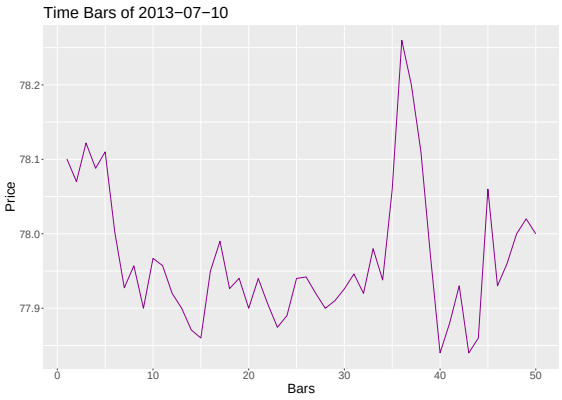
(b) Tick bars



(c) Volume bars



(d) Dollar bars



(e) Time bars

Figure 5.2: Sampling of 2013-07-10 tick data via tick, volume, dollar and time bars

5.4 Normality of log-returns

Once the samples have been gathered, it is time to compute log-returns and analyze their distribution. López de Prado argues in [6] that the time transformation induced by data bars allows a partial recovery of Normality.

However, after normalizing data, i.e. transform it such that it has a mean of 0 and a standard deviation of 1, the pdfs are far from normal (see figures 5.3 and 5.4). Even after plotting the pdf of a $N(\mu = 0, \sigma = 0.5)$, it can be seen that the pdfs of all of the bars considered are heavy tailed.

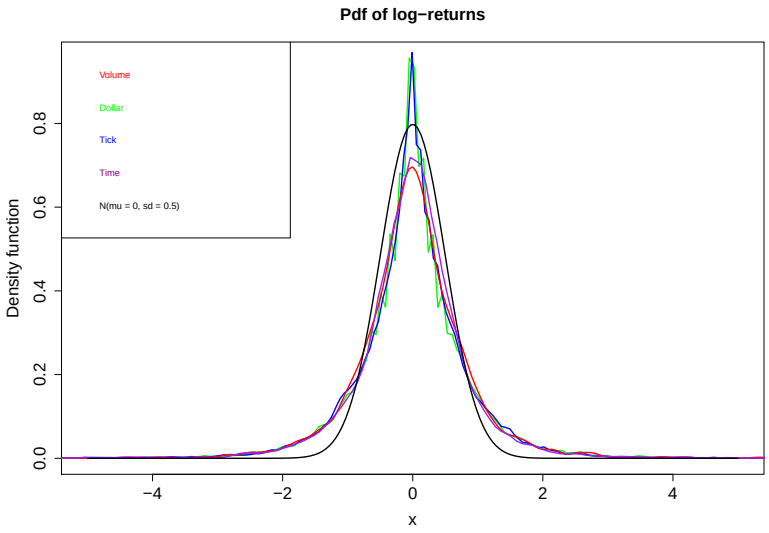
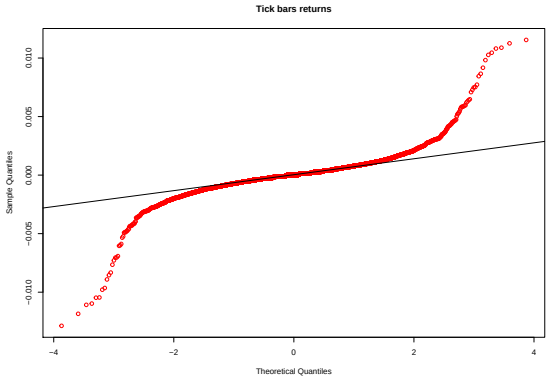
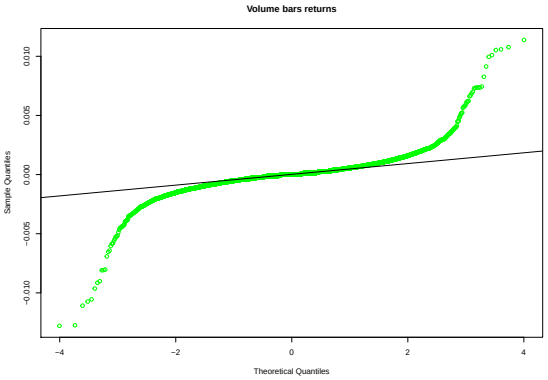


Figure 5.3: Fitted pdf of log-returns from data bars

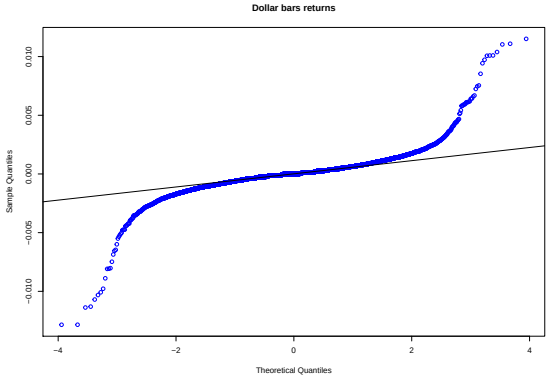
As a last comment on normality, note that in figure 5.4 the QQ Plots of tick, volume and dollar bars have more pronounced tails than time bars. This is attributed to oversampling periods of high volatility, which by definition imply drastic changes in prices.



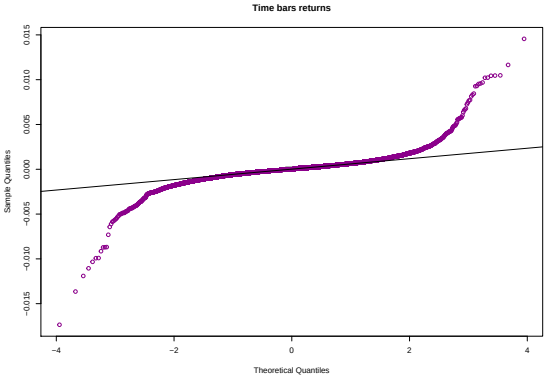
(a) Tick bars



(b) Volume bars



(c) Dollar bars



(d) Time bars

Figure 5.4: QQ Plots of data bars

5.5 Statistical analysis of bars

With the aim of understanding better the behavior of data bars some parameters have been calculated and presented in table 5.2, giving rise to the following remarks:

- Regarding ACF_1, the auto correlation function of lag 1, time bars are the only type of bars that have a statistically significant one.
- $\text{Var}[\widetilde{\sigma}^2]$ is the variance of $\widetilde{\sigma}^2$, where the latter is the square of monthly volatility. Therefore, a small variance will mean that volatility tends to stay fairly constant month over month. This is what happened with volume bars, whose variation of monthly volatility is the lowest.
- $\bar{n}^{\text{weekly}}$, the average number of bars in a week shows that tick bars have the fewest, dollar and time bars really similar numbers, and volume bars having the most.
- $\text{SD}[n^{\text{weekly}}]$, the standard deviation of the number of weekly bars, shows that time bars are the most stable ones. This should not be a surprise because being equally time spaced makes them deterministic, with the only variability introduced via weeks with holidays. Apart from that, tick and dollar bars follow along and volume bars are the ones that display the most variability.

Table 5.2: Statistical analysis of bars

	Tick	Volume	Dollar	Time
ACF_1	-0.01090561	-0.01225259	-0.01482256	-0.06833088
$\text{Var}[\widetilde{\sigma}^2]$	$2.936431 \cdot 10^{-13}$	$7.139122 \cdot 10^{-14}$	$1.054108 \cdot 10^{-13}$	$1.618769 \cdot 10^{-13}$
$\bar{n}^{\text{weekly}}$	173.1509	303.1698	233.9245	237.3962
$\text{SD}[n^{\text{weekly}}]$	63.95744	101.94866	73.20699	31.19069

To complement the information given in table 5.2, figure 5.5 shows the normalized (to a [0,1] interval) number of weekly bars.

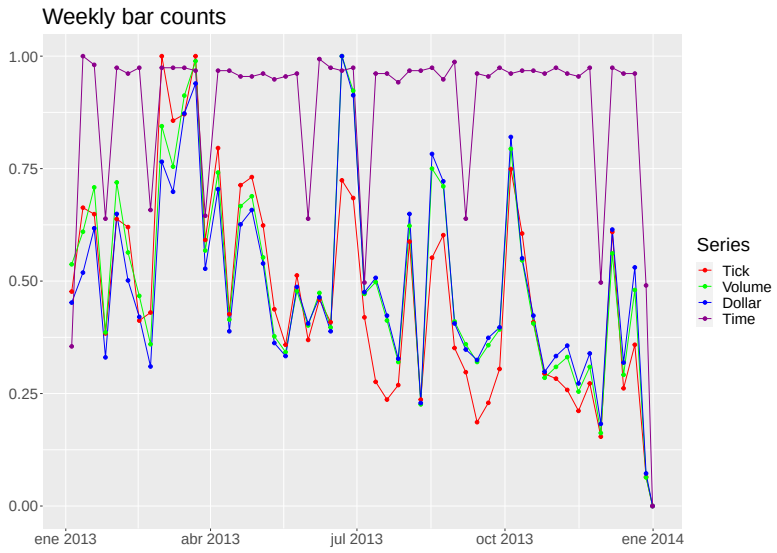


Figure 5.5: Normalized weekly bar counts

5.6 Results

Having presented the method to create data bars, the obvious question is: does sampling the price time series in a different way than time bars give a predictive edge?

In an attempt to shed some light on the previous question a simple autoregressive (AR) model will be used to fit the log-returns obtained via tick, volume, dollar and time bars:

$$r_k = c + \sum_{i=1}^{p} \theta_i r_{k-i} + \epsilon_k$$

Note that r_k is no longer indexed via t since it is no longer a time series, but a sequence. Additionally, the model order (p) will be determined with the R package stats [13], which uses the Akaike Information Criterion (AIC). This criterion is used to compare different models, in this case, different lags p.

As the datasets obtained with the bars are different, the training data will be the whole dataset except the **last 150 observations**. This number is fixed instead of a percentage because datasets have different sizes.

The forecasts will be given in a sequential manner. That is, if N is the number of bars in the dataset, the model will be trained on the first $N - 150$ bars and give a 1-bar forecast ($\widehat{r}_k$). Then, it will be trained on the first $N - 150 + 1$ and then give a 1-bar forecast and so on.

Lastly, considering that the datasets are different, an objective way to compare the errors $e_k = \widehat{r}_k - r_k$ is the Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{1}{150} \cdot \sum_{k=N-150+1}^{N} \left| \frac{e_k}{r_k} \right|$$

Note that the metrics used in previous chapters (RMSE, MAD) are inappropriate because they are not comparable over different datasets.

Table 5.3: AR results

	Tick	Volume	Dollar	Time
Lag (p)	0	1	10	2
MAPE	1.187730	1.129902	1.134345	2.041313
Improvement	41.82%	44.64%	44.43%	0%

As for the results, refer to table 5.3, which shows the chosen lag, the model's MAPE and the improvement on MAPE over the time bar model (**benchmark**). The most striking points are:

- The performance of the time model is the worst, with a MAPE of 2.04
- The tick model, even though the lag used is 0 which means that it predicts a constant, achieves a better MAPE than the time model.
- The volume and dollar models achieve similar results ($\approx 44\%$ improvement) but the complexity of the dollar model is much higher ($p = 10$). Thus, the volume model is preferred.

5.7 Conclusions

This chapter has introduced a useful technique to sample financial data. Although it was not possible to recover normality of log-returns through this new sampling method, the autoregressive models considered in section 5.6 delivered promising results. The next step would be to develop ML Models, that together with other features, would help predict the direction of the stock in a determined bar horizon.

Additionally, it also remains as future work to design a strategy that is able to incorporate data bars. High frequency trading is extremely difficult to replicate in an artificial environment since one has to account for information lag, order execution, etc. Apart from that, trading with volume bars is challenging since it introduces a "stochastic clock". That is, if one wants to open a position and the algorithm determines that it should be closed after m bars, ones does not know when that event is going to happen.

Chapter 6

Conclusions and future work

The goal of this thesis was to explore meta-labeling, fractional differentiation and data parsing as bars in the financial context with the purpose of determining if they were useful at delivering better forecasts and/or higher risk-adjusted returns. In order to fulfill this goal, three independent chapters have been presented, which have analyzed the techniques mentioned.

First of all, after studying meta-labeling thoroughly it can be concluded that this technique is very specific in the sense that it needs a certain environment to deliver better results. To be precise, as it was seen via the *coin flip* correction, to deliver higher Sharpe Ratios it needed a primary model that was good by itself, Sharpe Ratio wise. As the primary models developed did not achieve satisfactory Sharpe Ratios, the results mentioned before were not realized. Therefore, meta-labeling should only be used if one is confident that their primary model is good but needs minimal tweaking.

Secondly, fractional differentiation was useful at providing a framework to achieve stationarity without losing memory in time series. However, it did not translate into better forecasts. This is attributed to financial data, where memory was determined to not be highly relevant at giving predictions.

Lastly, the sampling technique known as data parsing as bars was helpful at giving better forecasts. It evidenced that in high frequency data, activity is more important than time.

Furthermore, it should be pointed out that there is still room for improvement. The data used in this thesis was centered around time series. However, in Finance there are alternative data sources such as financial news, macroeconomic data, FOREX, etc. It remains to be seen how these could have improved the performance of the Machine Learning models.

Regarding future work that might spin off of this thesis, it is important to mention the design of high frequency trading strategies. Having explored sampling of high frequency data, it is natural to continue working with this type of data, which is the perfect environment to further test meta-labeling, fractional differentiation ...

On the personal side, the author has learned that using Machine Learning in Finance is not a matter of what but when. One can have the most promising techniques but fail to deliver results due to not applying the model to the correct data set. In part, this is what happened in the meta-labeling and fractional differentiation chapters, which in financial data failed to deliver the results observed in the toy projects.

As a final note, it should be acknowledged that this thesis has been extremely useful to learn concepts that the author was oblivious of. That is, time series analysis, high frequency financial data and Machine Learning models. Also, this work has been an excellent introduction to academic research, since it has laid the foundations of the author's academic career, which he intends to continue in the coming years.

Bibliography

[1] Jan Beran. *Mathematical foundations of time series analysis: a concise introduction*. Springer, 2018.

[2] Peter J Brockwell and Richard A Davis. Introduction to time series and forecasting (3rd edn. ed.), 2016.

[3] Peter K Clark. A subordinated stochastic process model with finite variance for speculative prices. *Econometrica: journal of the Econometric Society*, pages 135–155, 1973.

[4] R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

[5] Marcos López de Prado. *Advances in Financial Machine Learning*. Wiley, 2018.

[6] David Easley, Marcos M López de Prado, and Maureen O'Hara. The volume clock: Insights into the high-frequency paradigm. *The Journal of Portfolio Management*, 39(1):19–29, 2012.

[7] Janusz Gajda, Rafał Walasek, et al. Fractional differentiation and its use in machine learning. Technical report, 2020.

[8] Paskalis Glabadanidis. Market timing with moving averages. *International Review of Finance*, 15(3):387–425, 2015.

[9] JRM Hosking. Fractional differencing. biometrika 68 165–176. *Mathematical Reviews (MathSciNet): MR614953 Zentralblatt MATH*, 464, 1981.

[10] J. Liu and D. P. Palomar. *imputeFin: Imputation of Financial Time Series with Missing Values*, 2019. R package version 0.1.1.

[11] Heino Bohn Nielsen. Non-stationary time series and unit root tests. *Unpublished Lecture Notes for Econometrics*, 2, 2005.

[12] Mohamed N. Nounou and Bhavik R. Bakshi. *Chapter 5 - Multiscale Methods for Denoising and Compression*, volume 22 of *Data Handling in Science and Technology*. Elsevier, 2000.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.

[14] William F Sharpe. The sharpe ratio, the journal of portfolio management. *Stanfold University, Fall*, 1994.

[15] Ashutosh Singh and Jacques Joubert. Does meta labeling add to signal efficacy?, 2019.

[16] Adrian Trapletti and Kurt Hornik. *tseries: Time Series Analysis and Computational Finance*, 2019. R package version 0.10-47.

[17] Valeriy Zakamulin. Revisiting the profitability of market timing with moving averages. *International Review of Finance*, 18(2):317–327, 2018.

Table 6.1: Tickers of the GMVP portfolio

MMM	ABT	ABBV	ABMD	ACN	ATVI
ADBE	AMD	AAP	AES	AFL	A
APD	AKAM	ALK	ALB	ARE	ALXN
ALGN	ALLE	LNT	ALL	GOOGL	GOOG
MO	AMZN	AMCR	AEE	AAL	AEP
AXP	AIG	AMT	AWK	AMP	ABC
AME	AMGN	APH	ADI	ANSS	ANTM
AON	AOS	APA	AIV	AAPL	AMAT
APTV	ADM	ANET	AJG	AIZ	T
ATO	ADSK	ADP	AZO	AVB	AVY
BKR	BLL	BAC	BK	BAX	BDX
BRK.B	BBY	BIO	BIIB	BLK	BA
BKNG	BWA	BXP	BSX	BMY	AVGO
BR	BF.B	CHRW	COG	CDNS	CPB
COF	CAH	KMX	CCL	CARR	CTLT
CAT	CBOE	CBRE	CDW	CE	CNC
CNP	CERN	CF	SCHW	CHTR	CVX
CMG	CB	CHD	CI	CINF	CTAS
CSCO	C	CFG	CTXS	CLX	CME
CMS	KO	CTSH	CL	CMCSA	CMA
CAG	CXO	COP	ED	STZ	COO
CPRT	GLW	CTVA	COST	CCI	CSX
CMI	CVS	DHI	DHR	DRI	DVA
DE	DAL	XRAY	DVN	DXCM	FANG
DLR	DFS	DISCA	DISCK	DISH	DG
DLTR	D	DPZ	DOV	DOW	DTE
DUK	DRE	DD	DXC	EMN	ETN
EBAY	ECL	EIX	EW	EA	EMR
ETR	EOG	EFX	EQIX	EQR	ESS
EL	ETSY	EVRG	ES	RE	EXC
EXPE	EXPD	EXR	XOM	FFIV	FB
FAST	FRT	FDX	FIS	FITB	FE
FRC	FISV	FLT	FLIR	FLS	FMC
F	FTNT	FTV	FBHS	FOXA	FOX
BEN	FCX	GPS	GRMN	IT	GD
GE	GIS	GM	GPC	GILD	GL
GPN	GS	GWW	HAL	HBI	HIG
HAS	HCA	PEAK	HSIC	HSY	HES
HPE	HLT	HFC	HOLX	HD	HON
HRL	HST	HPQ	HUM	HBAN	HII
IEX	IDXX	INFO	ITW	ILMN	INCY
IR	INTC	ICE	IBM	IP	IPG
IFF	INTU	ISRG	IVZ	IPGP	IQV
IRM	JKHY	J	JBHT	SJM	JNJ
JCI	JPM	JNPR	KSU	K	KEY
KEYS	KMB	KIM	KMI	KLAC	KHC
KR	LB	LHX	LH	LRCX	LW
LVS	LEG	LDOS	LEN	LLY	LNC
LIN	LYV	LKQ	LMT	L	LOW
LYB	MTB	MRO	MPC	MKTX	MAR
MMC	MLM	MAS	MA	MKC	MXIM
MCD	MCK	MDT	MRK	MET	MTD
MGM	MCHP	MU	MSFT	MAA	MHK

Table 6.2: Tickers (continued) of the GMVP portfolio

TAP	MDLZ	MNST	MCO	MS	MOS
MSI	MSCI	MYL	NDAQ	NOV	NTAP
NFLX	NWL	NEM	NWSA	NWS	NEE
NLSN	NKE	NI	NSC	NTRS	NOC
NLOK	NCLH	NRG	NUE	NVDA	NVR
ORLY	OXY	ODFL	OMC	OKE	ORCL
OTIS	PCAR	PKG	PH	PAYX	PAYC
PYPL	PNR	PBCT	PEP	PKI	PRGO
PFE	PM	PSX	PNW	PXD	PNC
POOL	PPG	PPL	PFG	PG	PGR
PLD	PRU	PEG	PSA	PHM	PVH
QRVO	PWR	QCOM	DGX	RL	RJF
RTX	O	REG	REGN	RF	RSG
RMD	RHI	ROK	ROL	ROP	ROST
RCL	SPGI	CRM	SBAC	SLB	STX
SEE	SRE	NOW	SHW	SPG	SWKS
SLG	SNA	SO	LUV	SWK	SBUX
STT	STE	SYK	SIVB	SYF	SNPS
SYY	TMUS	TROW	TTWO	TPR	TGT
TEL	FTI	TDY	TFX	TER	TXN
TXT	TMO	TIF	TJX	TSCO	TT
TDG	TRV	TFC	TWTR	TYL	TSN
UDR	ULTA	USB	UAA	UNP	UAL
UNH	UPS	URI	UHS	UNM	VFC
VLO	VAR	VTR	VRSN	VRSK	VZ
VRTX	V	VNO	VMC	WRB	WAB
WMT	WBA	DIS	WM	WAT	WEC
WFC	WELL	WST	WDC	WU	WRK
WY	WHR	WMB	WLTW	WYNN	XEL
XRX	XLNX	XYL	YUM	ZBRA	ZBH
ZION	ZTS				