Quant Workflow: A Scientific Method for Finance

Mathew Thiel

Faculty Mentors: Dr. Dominic Klyve, Dr. Donald Davendra, Dr. José Riera

Senior Capstone
Submitted in Partial Fulfillment of the Requirements for Graduation from
The William O. Douglas Honors College
Central Washington University

June, 2024

Accepted By:

Dominic Klyve	May 15th, 2024
Committee Chair (Dr. Dominic Klyve, Professor, Dept. of Mathematics)	Date
Davendra	May 24, 2024
Committee Member (Dr. Donald Davendra, Professor, Dept. of Computer Science)	Date
Christina Eberson	May 28, 2024
Director, William O. Douglas Honors College	Date

Contents

List of Figures

List of Tables

Abstract

Quantitative finance, or quant finance, is a field of investment management that leverages mathematical and statistical methods to trade and model a variety of asset classes. Practitioners in this field are called "quants", and may specialize in other areas of the field such as derivatives pricing, portfolio optimization, or equities. Quantitative finance encompasses three main fields: computer science, statistics, and finance. The goal of any quant is to generate *alpha*, which is "the measure of the excess return of an investment over a suitable benchmark, such as a market or an industry index" (Tulchinsky, 2019). Machine learning (ML) is becoming more prevalent in the field as practitioners develop models and methodologies to improve predictions and generate more alpha. Although ML has found more use in the field, it is generally misused, producing overfit models with inflated in-sample performance. Financial data is riddled with issues that challenge generic ML techniques, continuous discourse over efficacy of various methods creates a divide between practitioners and academics, and pseudo-discoveries from methods not conducted properly from a scientific perspective skew results. This project proposes "a scientific method for quantitative finance" – a complete, in-depth workflow that covers the entire strategy creation process and accommodates these finance-specific issues. This workflow aims to promote proper testing of strategy ideas and demystifies the strategy creation process to make the field more accessible for entry-level quants.

Chapter 1

Introduction

1.1 State of the Field of Finance

Two of the most controversial topics in the field of finance are the Efficient Market Hypothesis (EMH) and the Random Walk Hypothesis. The Efficient Market Hypothesis states that markets essentially fully reflect all information about an underlying security, and thus are efficient and not predictable (Fama, 1970). This is generally connected to the Random Walk Hypothesis, which states that changes in the price of a security are random, and therefore unpredictable (Samuelson, 2013). Over the years, these topics have seen heavy debate from both academics and practitioners, with no concrete conclusion from either side. The prevalence of this kind of discourse creates a situation where proponents of one school of thought chastise research from the other, providing an unnecessary schism that instills a deep-rooted distrust for opposing research. This sort of phenomenon was experienced by Andrew W. Lo and A. Craig MacKinlay when presenting their research rejecting the Random Walk Hypothesis, as "indeed, when we first presented our rejection of the Random Walk Hypothesis at an academic conference in 1986, our discussant – a distinguished economist and senior member of the profession – asserted with great confidence that we had made a programming error, for if our results were correct, this would imply tremendous profit opportunities in the stock market." (Lo & MacKinlay, 1999). This sort of issue is important context for the nature of the field of Finance as it displays the deep-seated distrust of research and unscientific nature of financial discourse. This issue is further personified through financial literature as finance books generally either contain "elegant mathematics that describe a world that does not exist" or explanations of actual observations that are "absent of rigorous academic theory" (Lopez de Prado, 2018b). These issues add to the divide between academics and practitioners, as "academic investigation and publication are divorced from practical application to financial markets, and many applications in the trading/investment world are not grounded in proper science" (Lopez de Prado, 2018b). Pressure to perform compounds the non-scientific nature of practical finance, as practitioners push out overfit models that lack the proper scientific rigor to validate their results. These practitioners "eventually settle for (1) a false positive that looks great in an overfit backtest or (2) standard factor investing, which is an overcrowded strategy with a low Sharpe ratio, but at least has academic support" (Lopez de Prado, 2018b). The false-positive issue is common not only in professional finance, but in academic literature as well.

1.2 Barriers to Entry in Finance

In the field of quantitative finance specifically, the barrier to entry for an individual practitioner, or retail trader, is substantially high. Even in the professional world, most established companies will not consider quants certified in a respective field with less than a Master's degree. Becoming a successful quant requires knowledge and experience from multiple distinguished fields, which further increases the challenge for retail traders. This challenge is exemplified by Lopez de Prado (2018b), as "it takes almost as much effort to produce one true investment strategy as to produce a hundred, and the complexities are overwhelming: data curation and processing, HPC (high performance computing) infrastructure, software development, feature analysis, execution simulators, backtesting, etc." Trading methodologies are secretive by nature, especially in large hedge funds and institutions. Many of these institutions also use proprietary, in-house software that is either inaccessible to individuals or locked behind massive paywalls. Even something as simple as basic data acquisition can cost hundreds to thousands of dollars depending on the resolution and extensiveness of the dataset. This is a clear disadvantage for the individual, as many ML models learn from examples in the data and require large sample size to be trained with any degree of accuracy.

1.3 Characteristics of Financial Data

Along with conceptual and practical issues, financial data is unique and exudes special characteristics that challenge general statistical practices. Although asset prices are extremely hard to accurately model, it is generally assumed that prices follow a Geometric Brownian Motion, as can be seen in the Black-Scholes model[1], which is a fundamental model for options pricing (Mota, 2012). A key assumption of Geometric Brownian motion is one of normality, and in this context, Black and Scholes (1973) use the assumption that stock prices follow a random walk, and that the log returns of a given stock are normally distributed. However, this assumption is one that lives in an "ideal world", as in most cases the log returns of an asset price are not normally distributed. This reality is present in multiple empirical works, as log-returns fail to exhibit normality at higher frequencies (Quigley, 2008), the assumption of normality fails for a high percentage of NASDAQ composite index companies (Mota, 2012), and "extreme tails of empirical distributions are higher (i.e., contain more of the total probability) than those of the normal distribution", which was discussed as early as 1963 (Fama, 1963). The lack of normality can also easily be seen through simple statistical tests, such as the Jarque-Bera [2] test. In short, many financial models like Black-Scholes make assumptions that are known to be false in a multitude of cases.

Data from a normal distribution has no skew and is symmetrical across the mean. Figure 1.1 displays a histogram of a random normal variable with 10,000 observations. On top of the plot is the normal probability density function (PDF), which defines the probability that a random variable will fall within a particular range of values.

The formula for log returns is the natural log of today's price, p_t, divided by yesterday's price, p_{t-1}, and is given below in Equation 1.1:

$$r_t = \ln(\frac{p_t}{p_{t-1}}). \tag{1.1}$$

[1] See Black and Scholes (1973) for more information.

[2] The Jarque-Bera test checks whether the skewness and kurtosis of a sample matches a normal distribution. See Jarque and Bera (1987), *A Test for Normality of Observations and Regression Residuals.*

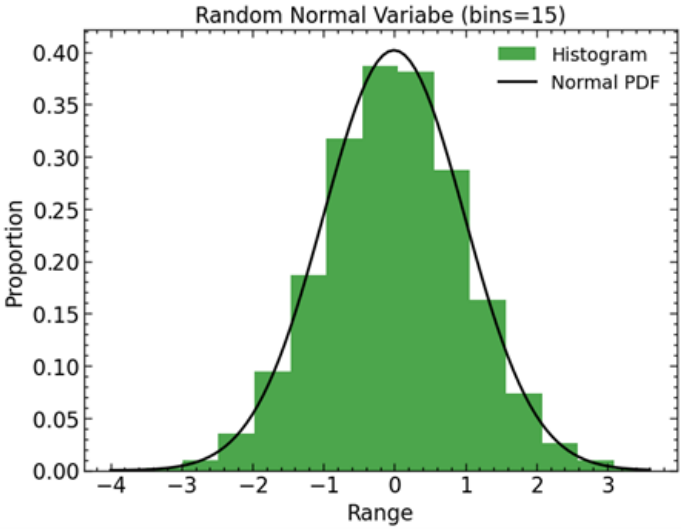

Figure 1.1: A normal distribution with 15 bins and 10000 observations

Figure 1.2 gives the log return of the S&P 500[3] adjusted closing price from January 1st, 2013, to January 1st, 2018, and it is evident that the characteristics of the S&P 500 log return differ from the random normal variable shown in Figure 1.1.

The wide shape of the data in Figure 1.2 exhibits a phenomenon called "fat tails", which refers to an increased probability of extreme outcomes on either side of the distribution. Conceptually, this makes sense as it is fairly likely that a stock price could drop 5% in any given day due to some extraneous event or factor like poor earnings. Conducting the Jarque-Bera test of normality on both distributions further confirms the different behavior of log returns. The results of the Jarque-Bera test are pictured in Table 1.1, where the null hypothesis that the given dataset is normal is rejected at $p < 0.05$ with 95% confidence.

Variable	Test Statistic	P-Value
Random Normal	0.1113	0.9458
Log Return of S&P 500	517.2939	4.69×10^{-113}

Table 1.1: Jarque-Bera Test of Normality for a normal distribution and the log return of the S&P 500 from 2013-2018.

From the results in Table 1.1, it is evident that log returns are clearly not normally distributed. Although the findings above cannot be generalized to every asset in any financial market, the conclusion that the assumption of normality in logarithmic asset returns can pose issues for accurate statistical modeling is supported. For example, if an asset exhibits low volatility with small, stable price movements, assumptions of normality in price modeling may lead to acceptable results.

[3]S&P 500 refers to Standard and Poor's 500, which is an index that tracks the largest 500 companies listed on United States stock exchanges.

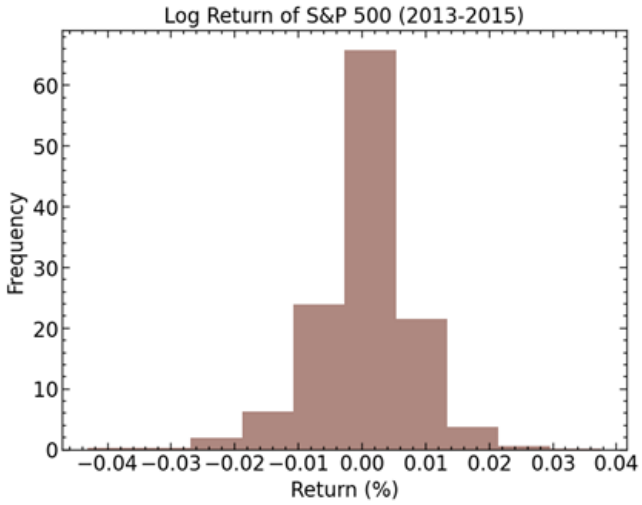

Figure 1.2: The log return of the S&P 500 from 2013-2015

However, if the asset is highly volatile and normality is assumed, large price movements (extreme outcomes on the tails of the distribution) would be drastically unaccounted for, leading to potentially catastrophic results.

Most ML practitioners can assume that observations are taken from an Independent and Identically Distributed (IID) process (Lopez de Prado, 2018b), which means that each term in the sequence of a random variable follows the same probability distribution and is mutually independent from one another. This is not the case in financial data, as prices exhibit serial correlation, which means that the price at a given time is related to a price at some time before that. This phenomenon is evident in empirical research as from a general study of five randomly selected stocks, it is concluded that "daily stock returns are not serially independent and that the market value of the corporations studied has a tendency to return to an interval around the trend value" (Ashley & Patterson, 1986). This adds complexity to financial ML, where "(1) labels are decided by outcomes; (2) outcomes are decided over multiple observations; (3) because labels overlap in time, we cannot be certain about what observed features caused an effect" (Lopez de Prado, 2018a). Financial data is also notoriously heteroskedastic, meaning that the variance of error terms is not constant. This is an unfortunate statistical property because many regression models are invalidated by heteroskedasticity.

Financial data is generally represented as a price at a fixed point in time, i.e. by price bars, and strategies and machine learning (ML) models are built upon these types of datasets. However, "markets do not process information at a constant time interval. The hour following the open is much more active than the hour around noon (or the hour around midnight in the case of futures)" (Lopez de Prado, 2018a). In a machine learning context, or any strategy for that matter, inconsistent sampling poses issues for predictions as datapoints at a more active period of time will reflect more information than a datapoint with less activity.

Prices are serially correlated and exhibit memory, meaning that information is preserved from one datapoint to the next. Since stationary, mean-reverting processes are a requirement for many statistical tests, one of the most common transformations of asset prices is differencing, or taking

the return over a period of time. The issue with performing a simple differencing like returns is that all memory before the current datapoint is eliminated (Lopez de Prado, 2018b). For any kind of ML model, preserving this memory is important as it could add a layer of predictability that is otherwise removed from simple differencing.

1.4 Backtesting and the Non-Scientific "Science" of Finance

Finance is laden with false reporting, misjudged performance, and out of sample degradation. The crisis of false reporting stems not only from the nature of financial data, but also from the approach to testing strategy ideas. Generally, strategy ideas are evaluated through a "backtest", which is a historical test of the strategy on past market data. The aim of a backtest is to give the researcher an idea of how the strategy may perform on new, future data, following the common belief that "history repeats itself." This mantra is not always true, as since markets are adaptive systems (Lo, 2019), they change over time and exhibit different characteristics. For this reason, it is possible that a given theory is false after the time of publishing, even if it may have originally been true (Lopez de Prado & Bailey, 2018). This phenomenon aides to both the reproducibility and degradation issues, as profitable strategies also lose efficacy over time.

We also never "truly" know whether a strategy is profitable, as we only have one historical path to test on, and experiments cannot be repeatedly conducted with constant conditions (Lopez de Prado & Bailey, 2018). Because of these reasons, it is especially challenging to determine whether a strategy is truly "significant", or whether the performance is just due to random chance or market factors at the time. Thus, in practice we generally estimate the out-of-sample performance of a strategy with the Sharpe Ratio[4] (SR), which is essentially just the ratio between expected return and risk of the strategy. The SR is the industry-standard of strategy evaluation metrics, yet it fails to properly estimate strategy performance due to multiple statistical issues that were previously discussed, like non-IID datapoints and non-Normal returns[5]. These problems can cause the standard SR to wildly overestimate strategy performance, which in some cases can exceed an upwards of 60% as found in Lo (2002).

Worse yet, the issue of multiple testing and p-hacking further skew proper estimations of strategy performance. P-hacking is defined as "a compound of strategies targeted at rendering non-significant hypothesis test results significant" (Stefan & Schönbrodt, 2023). In finance especially, p-hacking is mainly inspired by the pressure to perform and produce positive results. Finance is fairly unique in the sense that in both academia and the professional world, disproving a theory is often less important than proving one. Researchers and hedge fund managers care much more about well-performing, high SR strategies than terrible, low SR strategies. Compounded with tight deadlines, this pressure to perform can greatly influence the quality of financial research and strategy ideas. Multiple testing is a close relative of p-hacking, but is committed with much less nefarious intentions. Multiple testing occurs when multiple hypotheses are tested at the same time, which compounds the likelihood of a false discovery. For example, say we want to test 100 true hypotheses at once with a significance level of 0.95. For a single hypothesis test, the probability of the correct conclusion is $1 - \alpha = 0.95$, where $\alpha = 0.05$ is the error rate. However, when testing multiple hypothesis at once, the probability of a false conclusion increases to $1 - (1 - 0.05)^{100} = 0.994$, almost guaranteeing that a false discovery will be found (Bender & Lange, 2001). In finance, when a researcher searches

[4] See Chapter 12, Backtesting.
[5] See Chapter 12 for more information.

for the "optimal" strategy configuration through repeated backtesting, they are multiple testing. Each different strategy configuration can be considered a different hypothesis conducted under the same test, which in this case is the backtest. The researcher will then continue to select the best performing strategy from that multiple test, committing selection bias under multiple testing (Lopez de Prado & Bailey, 2018), where it is extremely likely that the strategy is a false discovery. Multiple testing is widely not accounted for in financial literature, as most academic financial journals assume that only a single test is conducted, and even some widely used textbooks ignore the issue entirely (Lopez de Prado & Bailey, 2018). Harvey et al. (2015) quantifies this issue, as in an empirical test of over 316 recent financial factors, the authors find that most are likely false due to multiple testing and data mining.

1.5 Overfitting: The Single Most Nefarious Problem in Finance

Overfitting, an issue not unique to the field of finance, is single-handedly one of the most problematic issues in the strategy creation process. Overfitting occurs when a model, or in this case a strategy, targets specific observations in a dataset instead of the overarching characteristics of the dataset (Bailey et al., 2014). Just about every phenomenon previously discussed, including label overlap, selection bias, and multiple testing, is a case of or contributes to overfitting. An investment strategy is generally a combination of multiple parts, including profit taking, stop loss, and risk levels, each of which can also be overfit. Overfitting is a common issue, as in addition to multiple testing, many academic studies that claim to be profitable are likely based on in-sample statistics (Bailey et al., 2014). An extra layer of complexity is added when using machine learning (ML) in finance, as not only can the strategy itself be overfit, but the ML model as well. It is exceptionally easy to overfit ML models, especially when increasing model parameters and complexity. Generic ML methods fail in finance, as label overlap causes common techniques like cross-validation[6] to leak data between the in-sample training sets and the out-of-sample test sets. As such, controlling for overfitting is incredibly challenging without the correct tools and checks along the strategy creation process to ensure that discoveries are actually significant.

1.6 The Need for the Quant Workflow

From the previous sections, it should be clear why a sound workflow is needed for the process of creating a profitable investment strategy. The Quant Workflow synthesizes some of the most advanced research currently available in the field to produce a start-to-finish method for strategy creation and evaluation. Marcos Lopez de Prado, among other authors like David H. Bailey and Andrew W. Lo, have been among the first to bring issues such as multiple testing and backtest overfitting to light in academic finance. In 2018, Marcos Lopez de Prado published *Advances in Financial Machine Learning* (Lopez de Prado, 2018b), which revolutionized the use of machine learning in the field and documented multiple steps of the strategy creation process. To date, Prado's research is still some of the most advanced in the field, and as such multiple citations to his work will be made throughout the rest of this work. This work attempts to demystify some of the advanced concepts present in current financial literature and introduce a practical

[6] See Chapter 10 for more information.

workflow that documents each part of the process so that the transition to practical application is more straightforward. The Quant Workflow leverages novel machine learning techniques specifically adapted for financial application to turn any simple rule or hypothesis about market movement into a full-fledged strategy. Chapters 4 through 12 document the entire strategy creation process, from data collection to strategy evaluation. After the necessary material has been introduced, Chapter 13 presents a practical example of the workflow when applied to a Simple Moving Average (SMA) Cross trading strategy, chosen due to its place as a "beginner-friendly" trading strategy. In the next chapter, we provide a visual overview of the Quant Workflow and compare it to a generalized machine learning workflow.

Chapter 2

The Quant Workflow: A Visual Overview

2.1 Generic ML Workflow

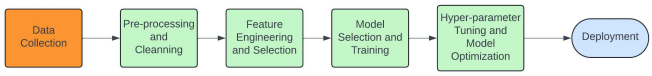

Figure 2.1: The generic machine learning workflow

As seen in Figure 2.1, the generic machine learning workflow begins with data collection and preprocessing[1]. In these steps, data is collected from relevant sources and prepared for the model by handling missing data, outliers, etc. In the next step, relevant features are created that describe the dataset. In cases where our features may exhibit high correlation with each other, we may drop similar features or use a dimension reduction technique to select features[2]. Next, the model is trained on the data and Cross-Validation[3] (CV) is conducted to compare various machine learning models. After a model has been selected, model hyperparameters are optimized. The generic workflow ends with the deployment of the model for use.

2.2 The Quant Workflow

The proposed Quant Workflow aims to adapt the generic machine learning workflow to solve finance-specific issues. The labeling step is perhaps the most drastic change, as instead of directly labeling the target variable (returns), we label using the Triple-Barrier Method[4]. The Quant Workflow is pictured in Figure 2.2.

[1] See Chapter 4
[2] See Chapter 11
[3] See Chapter 10
[4] See Chapter 7

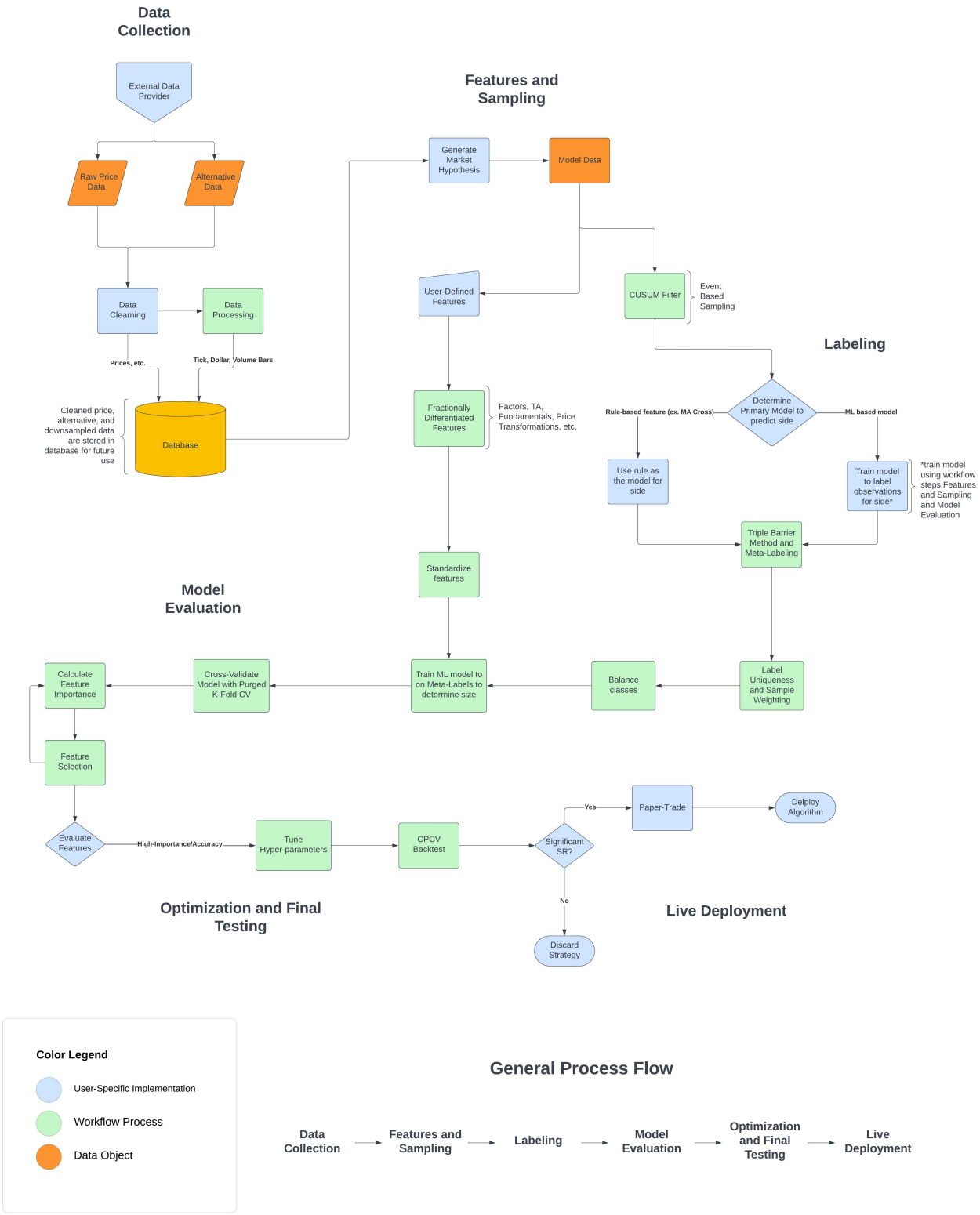

Figure 2.2: The proposed Quant Workflow

Chapter 3

Literature Review

3.1 A Brief History of Quant Finance

The first instances of computers formally used in financial markets was in 1971, as the National Association of Security Dealers Automated Quotations (NASDAQ) became the first electronic stock market (Markham & Harty, 2007/2008). Later in the mid 1970's, the first electronic order routing systems emerged, as the New York Stock Exchange (NYSE) released a system called "designated order turnaround" (DOT) (McGowan, 2010). In the 1980s, floor-based trading begin to cease as computerized strategies referred to as "program trading" began to see prevalence (McGowan, 2010). The late 1990s saw another financial revolution with the creation of electronic communications networks (ECN), which allowed for the trading of various financial products (equities, commodities, etc.) outside of traditional exchanges (Liebenberg, 2002). In 2001, decimal pricing was introduced, which changed the minimum tick size from $\frac{1}{16}$th of a dollar to \$0.01, dramatically benefiting algorithmic trading through decreased bid-ask spread and increased liquidity (Bessembinder, 2003). By 2005, algorithmic trading accounted for 25% of all market volume, increasing to a whopping 75% in 2009 (Kissell, 2013). Another turning point in the history of computerized finance was the flash crash of May 6th, where a large selling algorithm caused the Dow Jones Industrial Average (DJIA) to decline 998.5 points (equating to over \$1 trillion USD) in mere minutes (Easley et al., 2010). Some authors attribute the flash crash to high-frequency trading (HFT) (Kirilenko et al., 2017) which relies on rapid buy-and-sell orders to profit off of small price inefficiencies. HFT has grown in popularity in recent years, and HFT accounts for approximately 55% of current trading volume in US markets (Gerig, 2015).

3.2 ML and Its Use in Finance

Machine Learning (ML) began to take hold in the field of finance in the early 2000s. Early academic work was dominated by complex ML algorithms such as Neural Networks (NNs) and Support Vector Machines (SVMs) (Henrique et al., 2019). Many applications of financial ML attempt to predict market price and/or direction, such as Kim (2003) and Huang et al. (2005), and a large percentage of the most cited ML papers use NNs (Henrique et al., 2019). Ballings et al. (2015) compares the efficacy of multiple classifiers for stock price prediction, finding that Random

Forest (RF) models showed better performance than both NNs and SVMs. In terms of improving model performance, most sources agree in the importance of feature selection and processing. Tsai and Hsiao (2010) investigate the use of various selection methods, including Principle Component Analysis (PCA) and find that these methods improve predictions. Recent works have found success in the use of alternative data, such as social media and sentiment analysis, in predicting returns (Oliveira et al., 2017). Atsalakis and Valavanis (2009) conduct an empirical survey of more than 100 ML studies, finding that the average number of input variables is between four and ten, 30% of surveyed articles use price data or indicators depending solely on it as input variables, and 20% use technical analysis as input variables. In addition, Atsalakis and Valavanis (2009) finds that most of the finance-related performance metrics used are hit rate (percentage of correct predictions), annual rate of return, and average annual profit.

3.3 Practical Workflows: The Missing Link

While a multitude of studies that apply ML to financial markets exist, until recently, there has been a clear lack of studies dealing with proper application of ML methods to finance. The first hints of which began with the work of Marcos Lopez de Prado, David H. Bailey, etc. who will be heavily cited throughout the workflow. Before diving into machine learning workflows, it is important to cover more basic algorithmic trading workflows and literature.

While algorithmic trading and quant finance are similar, in some cases their definitions tend to be different. Quantitative finance deals strictly with the use of statistical and mathematical analysis applied to financial markets, while algorithmic trading deals with the computerized execution of trading strategies. In many cases, these two disciplines overlap, such as in the proposed workflow, but the literature surrounding the two is quite different. Introductory algorithmic trading textbooks[1] are generally written from a practitioners perspective, with a focus on the practical aspects of the strategy creation process. In quantitative finance textbooks[2] the focus is opposite; generally, the focus is on market microstructre and asset pricing theories. Introductory algorithmic trading textbooks tend to prescribe the walk-forward (WF) approach to backtesting, with is synonymous with the time series cross-validation approach used in ML[3]. Both the walk-forward approach and various statistical tests used in quantitative finance are subject to key issues, namely overfitting, multiple testing, and assumptions of normality.

The first realizations of overfitting and false reporting under multiple testing in financial literature was Lo (2002), as he found that reported Sharpe Ratios (SR) are subject to large estimation errors due to non-normal and non-independent and identically distributed (IID) returns. After correcting for estimation errors in mutual and hedge fund SRs, Lo finds that annual SRs can be overstated by upwards of 65% (Lo, 2002). The discussion continues as Lopez de Prado and Peijan (2005) show that hedge funds may also be substantially underestimating market risk. Subsequently, Christie (2005) is among the first to develop a framework for testing whether two portfolios can be accurately distinguished based on their SRs. In addition to the issues with SR estimation, multiple testing is especially prevalent in finance, as running multiple backtests is equivalent to running the same test with different hypotheses. With enough trials, we are bound to get a significant result regardless of the actual predictive power of the strategy (James et al., 2023). Bailey and Lopez de Prado (2012) build on Lo's work by introducing the Probabilistic Sharpe Ratio (PSR) which

[1] See Chan (2009), Carver (2015), among others.
[2] See Mazzoni (2018), among others.
[3] See Arlot and Celisse (2010)

accounts for non-Normality of returns and SR estimation errors. Two years later, Bailey and Lopez de Prado (2014) adapt the PSR to account for multiple trials, creating the Deflated Sharpe Ratio (DSR). The work of Bailey and Lopez de Prado was among the first to define a systematic performance metric that corrects for both multiple testing and non-Normality. PSR has set the standard for single-backtest performance reporting, as it has even become a standard performance metric in the popular backtestng platform Quant Connect[4].

3.3.1 Machine Learning in Finance: How it Falls Short

Aside from the testing process, other areas of the generic machine learning workflow are not entirely generalizable to finance without some adaptation. The general machine learning workflow usually proceeds as follows: data preprocessing, feature engineering and selection, model selection, hyperparameter tuning, and finally testing. (Quemy, 2020). Generally, machine learning is most beneficial in cases where we have large amounts of data to learn from, and models can protect against overfitting with cross-validation (Arnott et al., 2018). In the generic workflow, cross-validation occurs in the model selection and hyperparameter tuning steps. One of the biggest disadvantages of ML in finance is the limited amount of data. This exemplifies the issue of data mining, as not only will cross-validation falsely "validate" datamined patterns (Arnott et al., 2018), but the nature of non-IID returns and overlapping labels[5] further inflates cross-validation performance (Lopez de Prado, 2018b). While ML finds patterns in the data, it does not enforce sound economic rationale in its findings, so "working" models may not generalize to the future (Arnott et al., 2018). *Advances in Financial Machine Learning* (Lopez de Prado, 2018b) (AFML) by Marcos Lopez de Prado and his subsequent works not only revolutionized financial machine learning, but were the first to provide a properly scientific template for both strategy creation and ML application in finance. Aside from PSR and DSR, Lopez de Prado introduced multiple other breakthroughs, including Triple Barrier Labeling, Meta-Labeling, and Combinatorial Purged Cross Validation (CPCV) (Lopez de Prado, 2018b). Subsiquent to Lopez de Prado, other authors such as Arnott et al. (2018) began to publish studies aimed at backtest overfitting and machine learning applications in finance. In Arnott et al. (2018), the authors propose a "Seven-Point Protocol" for quant finance research, which will be further discussed in Chapter 12. Contemporary financial ML literature is still heavily dominated by Lopez de Prado's work, especially in terms of Triple Barrier Labeling, Meta-Labeling, and CPCV. For these reasons, Lopez de Prado and AFML will be heavily cited throughout the proposed workflow. While revolutionary, AFML is geared towards professionals with extensive backgrounds in machine learning. In this work, we attempt to demystify the methods described in AFML and synthesize the works of Lopez de Prado and other authors to define a concrete workflow that covers each part of the strategy creation process.

[4]See https://www.quantconnect.com/.
[5]See Chapter 7, Labeling.

Chapter 4

Data Processing

4.1 Data Cleaning

4.1.1 Stock Splits and Dividends

When working with historical stock price data, it is important to make sure that the data is adjusted for splits and dividends. Stock splits occur when the amount of shares offered for a specific company is increased by some factor in order to provide more liquidity to the market. For example, a 2:1 stock split would double the amount of currently available shares. Since stock splits only affect the number of shares traded, the price of the asset can change, as in a 2:1 split, each share would be worth half of its previous value. Dividends are payments made to shareholders in the form of additional shares or cash. Both splits and dividends can affect the historic value of an asset, which is why historical stock data needs to be adjusted for these events. To adjust for these events, the concept of forward adjustment is employed, where the adjustment factor A_t is initialized to a value of $A_t = 1$ at the first price bar and changes over time as we receive split and dividend data (Isichenko, 2021). Given stock split factor S_t, dividend D_t, and closing price C_t at time t, the adjustment factor is defined as (Isichenko, 2021):

$$A_t = \prod_{t=0}^{t} \frac{S_t}{1 - \frac{D_t}{C_t}}. \tag{4.1}$$

In some cases, historical data is already adjusted for these events, but higher-frequency data, like the trade data used in this workflow, is generally unadjusted. Figure 4.1 depicts the adjusted and unadjusted price of Walmart stock over the period 2018-2024, and it is clear that failing to adjust for splits and dividends can lead to pricing errors.

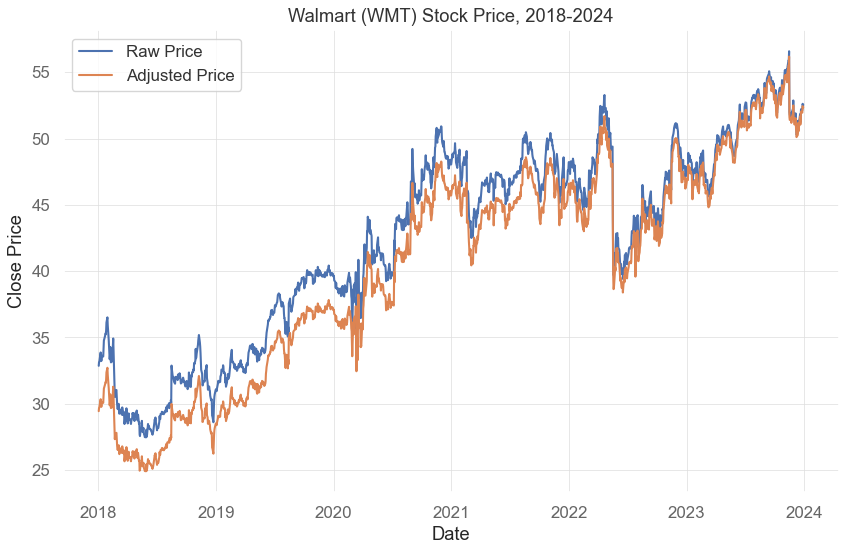

Figure 4.1: The raw vs. adjusted close price of Walmart stock from 2018 to 2024.

4.2 Bar Types

4.2.1 Time Bars

Financial price data is generally represented as a series of "bars" over time. A bar is a combination of generic price information, consisting of an open, high, low, and close price. Figure 4.2 shows a common representation of a time bar, where the color of the bar depicts whether the close price increased or decreased against the open price of that bar.

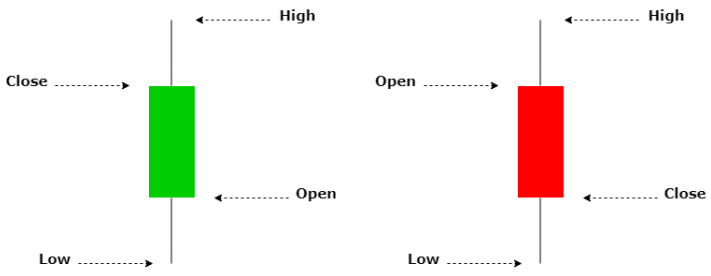

Figure 4.2: A diagram of the candlestick representation of a common time bar.

Time bars aggregate price data into bars sampled at fixed time steps, i.e. daily bars. Time bars are generally also used in tandem with volume aggregations sampled at the same time step. Time bars are likely the most popular financial data structure for academics and practitioners alike, yet they improperly sample data and have poor statistical properties (Lopez de Prado, 2018b). Since most markets have varying levels of activity at different times, such as higher activity at the open of the stock market versus at lunch time, time bars oversample at low-activity periods and undersample at high-activity periods (Lopez de Prado, 2018b). When comparing time-sampled and volume-sampled data, the volume-sampled data is "much closer to normal, exhibits less serial correlation, and is less heteroskedastic" (Easley et al., 2012). Thus, sampling as a function of volume instead of time improves data quality, resulting in a better model. There are three types of bars in which we can leverage these positive properties for a model: volume, tick, and dollar bars.

4.2.2 Volume Bars

Volume bars sample data after a certain number of volume is exchanged, more accurately modeling order fragmentation[1]. Sampling as a function of volume also has the advantage that price-volume analysis is valuable as "it is easy to demonstrate that given any price sequence, the distribution of future quotes and prices will differ depending on volume. So although price-based technical analysis is valuable, price and volume-based technical analysis is even more valuable." (Easley & O'Hara, 1992).

4.2.3 Tick Bars

Tick bars sample data after a certain number of transactions have occurred, i.e. 10 transactions. This is similar to a volume clock, but tick bars can encompass varying amounts of volume depending on the activity of the underlying asset between each sample threshold. Tick bars may have better return normality than volume-sampled bars, as "it is shown that, to recover normality in asset returns, the number of trades is a better time change than the traditionally used trading volume. The near-perfect normality in transaction time is exhibited through the reconstruction of the time-changed return process" (Ané & Geman, 2000). However, tick bars may contain outliers, as open and close auctions can cause over saturated trades and order fragmentation obscures trading activity (Lopez de Prado, 2018b).

4.2.4 Dollar Bars

Dollar bars sample after a certain number of currency has been exchanged. Dollar bars have the unique advantage that unlike tick and volume bars, they tend to be robust against actions like new share issues and buybacks[2] (Lopez de Prado, 2018b). Dollar bars also incorporate both price and volume into sampling, which generally leads to more robust bars in practice. Consider the case where a stock begins the year at \$10 and ends the year at \$20. Imagine we want to invest \$1000 at the start of the year, and another \$1000 at the end of the year. The amount of shares needed to fulfill an order of \$1000 at the end of the year is half the amount needed to fulfill the same order at the start of the year. If sampling by volume, the same order value will have half of the effect

[1]Order fragmentation refers to the process of splitting large orders across various exchanges or platforms.

[2]New share issues occur when a company offers additional shares to investors, while buybacks occur when a company repurchases outstanding shares.

on sample frequency as it did at the start of the year. For this reason, it may make sense to prefer dollar bars over the other sampling methods.

4.3 Return to Normality Through Sampling Methods

Figure 4.3 displays the PDFs of the log returns of each bar type against a normal distribution. Tick, volume, and dollar bars all tend closer to a normal distribution than time bars. This phenomenon is further confirmed in Mendelbrot and Taylor's seminal paper *On the distribution of Stock Price Differences* as "price changes over a fixed number of transactions may have a Gaussian distribution. Price changes over a fixed time period may follow a stable Paretian distribution, whose variance is infinite" (Mandelbrot & Taylor, 1967). Therefore, sampling with either of these methods will be more beneficial than sampling with a simple time clock.

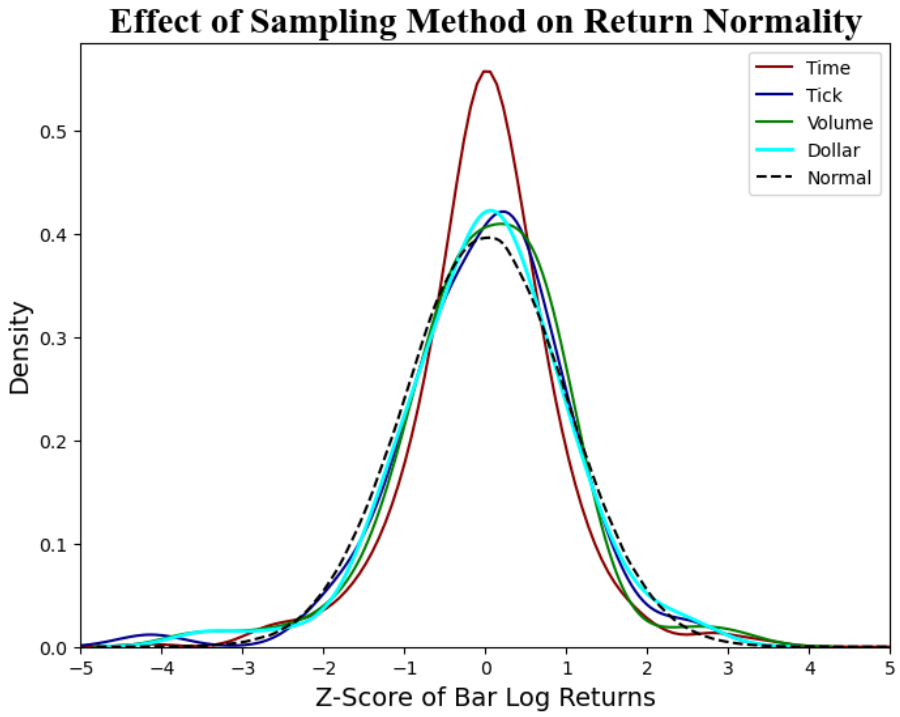

Figure 4.3: A graph of the distributions of each bar type against a Normal PDF, with time, tick, volume, and dollar bars sampled at 1 hour, 1.2×10^5, 3×10^7, and 1.1×10^{10} respectively.

4.4 Selecting Proper Thresholds

Selecting a threshold for any of the given bar types in this framework is left to the user, as there is no "tried-and-true" solution to selecting a sampling threshold. This is because the threshold to be used depends on multiple factors such as how frequently the user would like to trade and

generate signals. If a high threshold is chosen, then less bars would be sampled, but each bar would carry more information. If a low threshold is chosen, then more bars will be sampled, but each bar will carry less information and thus introduce more noise. The way to balance this trade-off is by aligning the threshold with the intended trade frequency to a certain extent. For example, if a trade frequency of hourly was chosen, i.e. make a trade roughly every hour, then it would not make sense for the sampling threshold to be so high that only one bar per day is sampled. In this case, bars sampled about every minute would make more sense, so a threshold that reflects that should be chosen. In practice, since we are not dealing with time bars, exact time sampling is fairly challenging to achieve without resampling the data post-sampling. A way to deal with this issue is by using either a fixed or dynamic threshold based on the market value of the asset. In the case of stocks, free floating market capitalization is a good candidate to use for a dynamic sampling threshold (Lopez de Prado, 2018b). Average volume, in the case of tick and volume bars, would also make a good candidate for a threshold as it directly relates to the sampling method. To set up a dynamic threshold in practice, the most intuitive way is to sample after some percentage of the given metric, market capitalization or average volume for example, has been exchanged, i.e, 5% of daily average volume. This percentage can be easily adjusted by the user to fit any trading frequency, or it can even be adjusted dynamically as a function of something like volatility. For example, say a dynamic threshold is implemented like the one provided in Equation 4.2:

$$\text{Threshold} = \text{Daily Average Volume} * (0.05 * \frac{1}{1 - |\text{Daily Volatility}|}). \tag{4.2}$$

In Equation 4.2, the fixed percentage 0.05 is scaled by the inverse of daily volatility, which will lower the threshold as daily volatility increases. This is but one example of the possible dynamic thresholds that can be implemented for any of the given sample methods, and the choice of which to use is left to the user. Proper data sampling is important for any ML model, and the sampling threshold must be sufficiently high to generate enough data for the ML model to learn from. In Chapter 5, we implement a method for capturing important events from the above methods, so capturing these events is not the purpose of bar sampling.

Chapter 5

Sampling Features

A "feature" is a predictor that describes an underlying dataset. Features can be categorical or continuous, where categorical features have values that relate to categories (i.e. "yes" or "no", 1 or 0) and continuous features take on any range of values (Müller & Guido, 2016). Machine learning models use features to generate predictions, and thus strong, well-crafted features with high predictive power are essential to a successful model. Features with low predictive power can be combined to create better features. Unfortunately, feature generation is mostly based on domain knowledge, and cannot be entirely automated. However, methods like feature importance can give insight into which features are relevant for the given machine learning algorithm.

In finance, it makes sense that we would want to pay attention to important events, as these are generally the basis of market inefficiency. There are multiple ways of incorporating important events into ML models, such as creating a feature based on fundamental events or setting minimum return thresholds for label generations. One method used in the proposed framework is called a cumulative sum (CUSUM) filter. The CUSUM filter is a filtering method that only considers datapoints outside a specified threshold. It is used to "detect a shift in the mean value of a measured quantity away from a target value" (Lopez de Prado, 2018b). In the seminal paper relative to finance by Kim Yam and H. C. Lam (1997), the CUSUM filter was used to generate trading signals for a trend-following approach, which relies on the assumption that prices moving in one direction will continue to move in that direction (Lam & Yam, 1997). Lopez de Prado (2018b) creates an adaptation of their work by introducing a symmetric CUSUM filter, which is mathematically defined in Equation 5.1 :

$$\begin{aligned} S_t^+ &= \max\{0, S_{t-1}^+ + y_t - E_{t-1}[y_t]\} \\ S_t^- &= \max\{0, S_{t-1}^- + y_t - E_{t-1}[y_t]\} \end{aligned} \tag{5.1}$$

where $S_0^+, S_0^- = 0$, and the filter would find an important event when $\mid S_t \mid$ is larger than some threshold h (Lopez de Prado, 2018b). This threshold can either be a fixed value specified by the user or a dynamic value based on some metric. The CUSUM filter is not limited only to asset prices, and it can be used with any given continuous feature. In this framework, the CUSUM filter is mainly used as an event sampling method for signal generation, but it can also be used as a standalone feature. In this framework, average daily volatility is used as the filter threshold, but other dynamic thresholds can be implemented, such as average high over t periods, for example. The daily volatility of S&P 500 dollar bars is given in Figure 5.1, with the red horizontal line representing the average

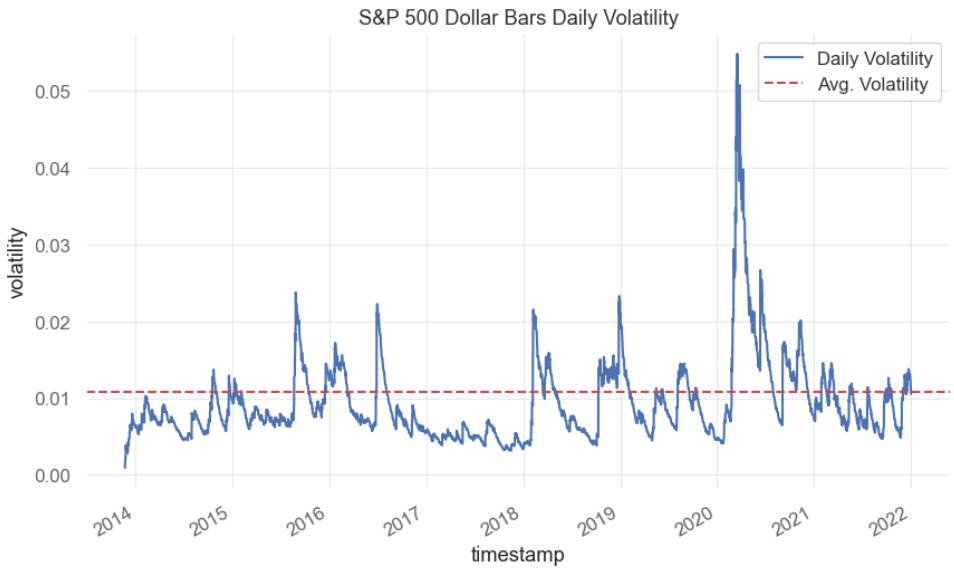

Figure 5.1: A graph of the daily volatility of S&P 500 dollar bars, with the red horizontal line representing the average daily volatility.

daily volatility used for the sampling threshold. Figure 5.2 represents the events sampled from the CUSUM filter, with values of 1.0 representing an event, and 0 values representing no event. Filtering events like this makes sense, especially in financial applications, as financial data generally contains a high amount of noise, and we want our model to learn from relevant examples in order to identify a robust pattern. Conceptually, volatility or return CUSUM filters are likely to capture an important pattern because high volatility or return events are generally due to some cataclysmic or significant event that can be learned from. By filtering out these events, we can allow our model to focus on just these events instead of smaller movements that are more likely attributed to random noise. Another important benefit of sampling events with something like the CUSUM filter is that it is robust to whipsaws and will not generate multiple events by hovering around a threshold level (Lopez de Prado, 2018b).

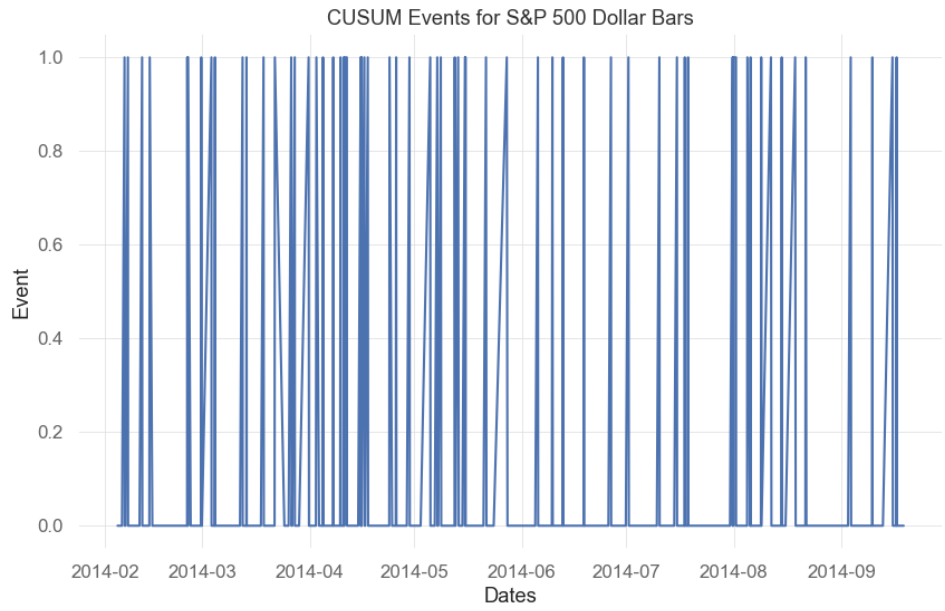

Figure 5.2: A graph of a subset of sampled events for S&P 500 dollar bars, where 1.0 values represent a sampled event and 0 values represent no event.

5.1 Finding Possible Features

5.1.1 Fundamental Analysis

Fundamental analysis refers to a form of investing that makes investment decisions based on the underlying value of the market. Fundamental analysis uses microeconomics and market factors, which generally include statistics from balance sheets, earning reports, etc. Fundamental analysis is generally used for longer-term investing, where the investment horizon is a month or more. Some classic examples of fundamental investment strategies are sector momentum (Faber, 2010) and asset class trend-following strategies (Faber, 2013). A popular example of factor investing is Fama and French's 5-factor model[1], which attempts to capture size, value, profitability, and investment pattern in average stock returns to model expected returns in the stock market (Fama & French, 2015). The 5-factor model builds on their previous 3-factor model and the Capital Asset Pricing Model[2] (CAPM) (Sharpe, 1964), and can be used not only for investing, but for estimation of expected returns for portfolio optimization or other asset pricing models. Specific to this workflow, the 5-factor model could be used to generate more complex features or serve as a baseline for another ML model to be trained for signal generation. More on this topic will be covered in Chapter 7, which discusses labeling observations.

One way that fundamental Analysis can be incorporated into higher frequency trading strategies

[1]See Fama and French (2015), *A Five-Factor Asset Pricing Model* for more information.

[2]See Sharpe (1964), *Capital Asset Prices, A Theory of Market Equilibrium Under Conditions of Risk* for more information.

is by using it to filter a large universe of assets. Say we have a daily feature that works well when asset prices exhibit a constant trend. We could then pick assets to use in the strategy by filtering a universe of stocks, such as S&P 500 constituents, by a momentum or trend following factor. The universe could be further filtered based on specific characteristics, but using fundamental analysis in this way is beneficial as it provides a computationally efficient method of filtering a large universe of assets.

5.1.2 Technical Analysis

Technical analysis refers to a study of price movement and market action as opposed to the underlying value, goods, or areas in which the market trades (Edwards et al., 2018). It is a polar opposite of fundamental analysis, which deals strictly in the value of the underlying market. Technical analysis essentially treats an asset price as a time series variable, where underlying characteristics of the asset are ignored and only things like price and volume are considered. Most technical analysis methods can be grouped into categories, such as trend-following/momentum, mean-reversion, chart patterns, etc. The efficacy of technical analysis has been a topic of heavy debate since its genesis, and discourse on the subject is still ongoing. There are countless books and academic papers in support of either side, such as a 2002 paper by authors Tian, Wan, and Guo who find that the 26 technical analysis trading rules they tested had no forecasting power in U.S. markets after 1975 (Tian et al., 2002). However, a 2016 paper by Smith et al. (2016) finds that technical analysis exhibits higher performance in high-sentiment periods and has little to no predictive power in low sentiment periods. This paper makes no argument for either side of the debate, but merely presents technical analysis as a possible method for feature generation. One possible benefit of technical analysis is that methods are easy to understand, generalize well to different assets, and easy to combine with other features. It is up to the user to discern which methods create actionable features with positive predictive power. Chapter 11 gives an in-depth explanation on how features can be evaluated and feature research can be conducted.

Chapter 6

Fractional Differentiating

6.1 Memory-Preservation for Predictions

In financial markets, prices have whats called memory, as the price today "remembers" the prices that come before it. Formally, this means that a price at time t is dependent on the prices $t-1, t-2, ..., t-n$ where n indicates the time since the creation of the asset. This phenomenon is called path-dependence, and in any sort of predictive model, this path-dependence is important to preserve as it introduces a layer of predictability. However, the path of prices is generally non-stationary, which causes issues when creating price-related features to be used in machine learning and statistical inferencing (Lopez de Prado, 2018b). We are thus presented with an interesting challenge; how can we convert these non-stationary features into stationary ones? Many practitioners and academics alike use price returns to achieve this stationarity. By definition, the return of price p at time t subtracts price p_t from price p_{t-n}, where n is the return period (i.e. daily), in turn removing all memory leading up to price p_{t-1}. This is called simple differencing, and methods such as this achieve stationarity at the cost of memory (Alexander, 2001). Instead, to preserve memory, we can take a fractional difference, which can be thought of as a sort of fractional derivative.

6.2 The Math Behind Fractional Differentiation

The mathematical definition of fractional differentiation[1] starts with a shifted identity matrix, called a backshift operator (Lopez de Prado, 2018b):

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.1}$$

The backshift operator B is an NxN matrix where N is the number of rows in the feature vector we are differencing. Consider the backshift operator B that is shifted, or lagged, by some value $k \geq 0$. It is evident that:

$$B^k X_t = X_{t-k} \tag{6.2}$$

[1]For a more in-depth mathematical description of fractional differentiation, see Gray and fan Zhang (1988).

where X_t is a feature vector over the span of time t. For example, if $k = 2$ we can see that:

$$(1 - B)^2 = 1 - 2B + B^2 \tag{6.3}$$

where

$$B^2 X_t = X_{t-2}. \tag{6.4}$$

By expanding Equation 6.4, we can see that:

$$(1 - B)^2 X_t = X_t - 2X_{t-1} + X_{t-2} \tag{6.5}$$

and this forms the basis for the backshift operator B used in the next section. Now, given some real number d, we can set up the following binomial series:

$$(1 + x)^d = \sum_{k=0}^{\infty} \binom{d}{k} x^k. \tag{6.6}$$

This is possible because given some positive integer n, we can write:

$$\sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k. \tag{6.7}$$

Substituting in the backshift operator B, we can rewrite Equation 6.6 as:

$$(1 - B)^d = \sum_{k=0}^{\infty} \binom{n}{k} (-B)^k = \sum_{k=0}^{\infty} \frac{\prod_{i=0}^{k-1} (d - i)}{k!} (-B)^k \tag{6.8}$$

which is equivalent to:

$$= \sum_{k=0}^{\infty} (-B)^k \prod_{i=0}^{k-1} \frac{d - i}{k - i}. \tag{6.9}$$

We can then expand the series in Equation 6.9 to:

$$= 1 - dB + \frac{d(d-1)}{2!} B^2 - \frac{d(d-1)(d-2)}{3!} B^3 + ... \tag{6.10}$$

Let Equation 6.10 be defined as a weights vector w so that:

$$w = \{1 - dB + \frac{d(d-1)}{2!} - \frac{d(d-1)(d-2)}{3!} + ..., (-1)^k \prod_{i=0}^{k-1} \frac{d - i}{k}, ...\} \tag{6.11}$$

The stationary series $\tilde{X}_t$ is given by (Lopez de Prado, 2018b):

$$\tilde{X}_t = \sum_{k=0}^{\infty} w_k X_{t-k} \tag{6.12}$$

where d is the fractional differentiation value and k is a positive integer shift value where $k = 0, ..., t - 1$ (Lopez de Prado, 2018b). An interesting point to note about Equation 4.10 is that it

looks suspiciously similar to the generalized power rule for derivatives. For some positive integer level of differentiation k, the generalized power rule is defined as:

$$\frac{d^k}{dx^k}(x^n) = \frac{n!}{(n-k)!}x^{n-k}. \tag{6.13}$$

Applying fractional differentiation in practice as-is results in a negative drift, pictured in Figure 6.1. Lopez de Prado (2018b) solves this issue by implementing a fixed-width window, which drops weights after the absolute value of the weight is less than a given threshold τ. Implementing this change, the resulting series is shown in Figure 6.2. Over-differentiating a series can also be a problem, and Prado finds that a differentiation value of about $d = 0.35$ is sufficient for most applications (Lopez de Prado, 2018b). To pragmatically determine the minimum level d, the Augmented Dickey-Fuller[2] (ADF) test for stationarity can be used (Shumway & Stoffer, 2006). The simplest method of finding the minimum d is to compute the fractionally differentiated series for different fractional steps until the ADF test returns a significant p-value ($p < 0.05$). However, fractional differentiation is computationally intensive for large series, so large fractional steps (a step size of 0.1 is usually sufficient) should be used.

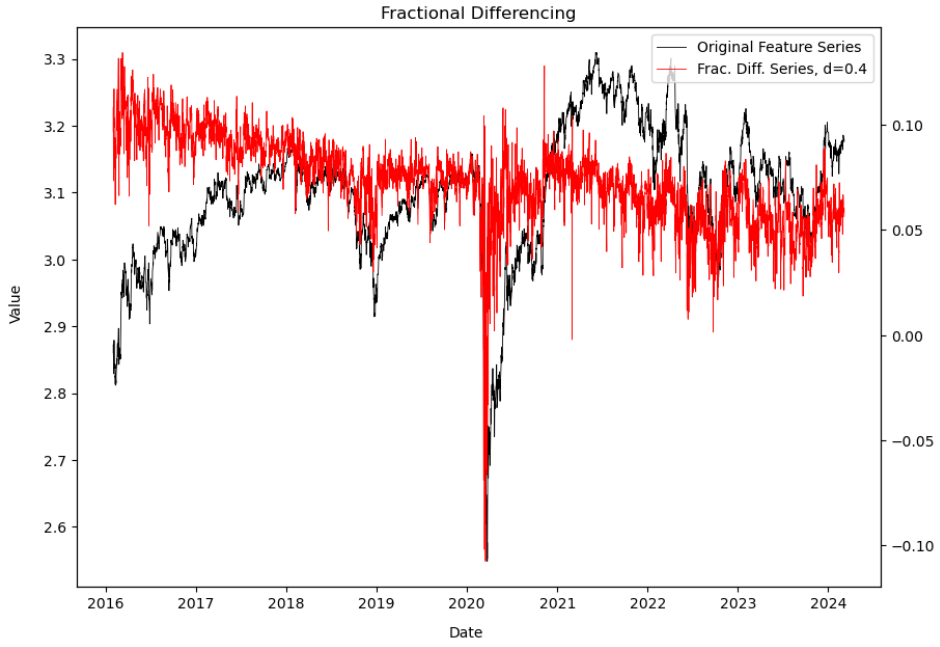

Figure 6.1: The fractionally differentiated series (red) over the close price of EWA stock. The fractionally differentiated series shows a clear negative drift.

[2]See Dickey and Fuller, 1979, *Distribution of the Estimators for Autoregressive Time Series With a Unit Root* for more information.

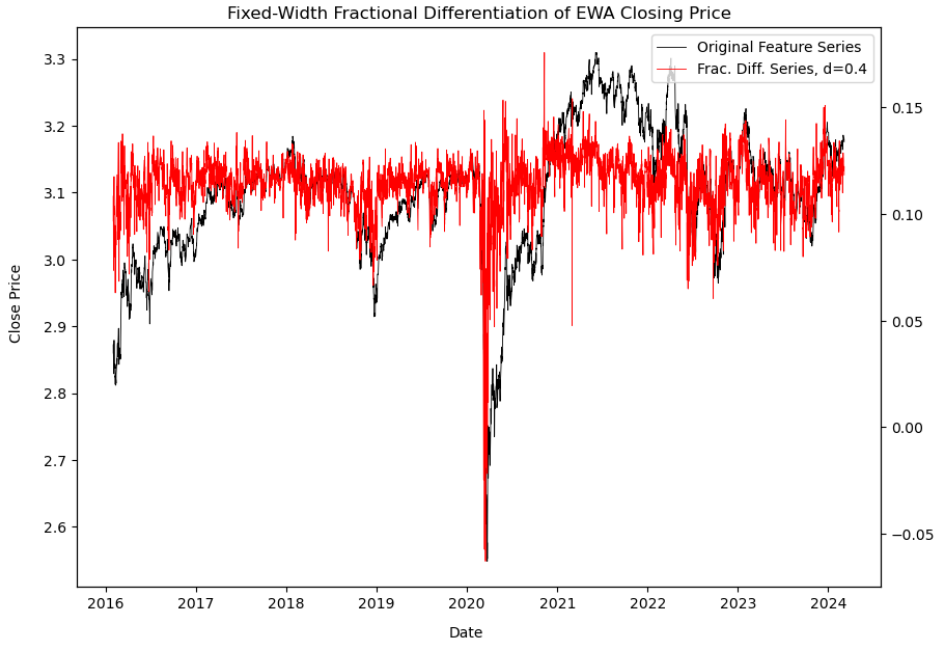

Figure 6.2: The fractionally differentiated series (red) over the close price of EWA stock. Drift is removed and the resulting series is stationary.

Chapter 7

Labeling Observations

In machine learning, an observation refers to a single point in time at which data is sampled. In financial ML, this includes both price data and feature data at any given time. A label is a value, generally 1, 0, or -1, that classifies an observation based on a target variable. In financial ML, labels can generally be thought of as a trading signal at a given time, however this definition can change when layering multiple models, which will be evident later in the workflow. One example of a common labeling scheme in finance is to label by predicted price direction. With this labeling scheme, if an observed feature indicated that the price might increase, that observation could be given a 1 label, which might indicate a positive price prediction.

7.1 Generic Labeling

Lopez de Prado (2018b) notes that most financial research papers label observations with a fixed-time horizon, similar to the previously mentioned labeling example. The fixed-time horizon method essentially attributes the sign of the return over a given period to a label, where 1 represents a positive return greater than some threshold, -1 represents a negative return under some threshold, and 0 represents a return within the signed threshold. Consider a feature matrix X with I number of rows, drawn from bars with index $t = 1, ..., T$. Formally, label y at feature bar i is defined as follows (Lopez de Prado, 2018b):

$$y_i = \begin{cases} -1 & \text{if } r_{i,0,t_{i,0}+h} < -\tau \\ 0 & \text{if } r_{i,0,t_{i,0}+h} \leq \tau \\ 1 & \text{if } r_{i,0,t_{i,0}+h} > \tau \end{cases} \tag{7.1}$$

where τ is a predefined return threshold, $t_{i,0}$ is the index of the bar after row i occurs, and $t_{i,0} + h$ is the index of the h-th bar after $t_{i,0}$ (Lopez de Prado, 2018b). The value $r_{i,0,t_{i,0}+h}$ is given by the return from bar $t_{i,0}$ to $t_{i,0} + h$, defined as (Lopez de Prado, 2018b):

$$r_{i,0,t_{i,0}+h} = \frac{p_{t_{i,0}+h}}{p_{t_{i,0}}} - 1. \tag{7.2}$$

Essentially, this method labels significant returns above a specified threshold by their sign. In the context of finance, this labeling method is sub-optimal for a few important reasons. First, the

aforementioned method does not allow for profit taking and stop loss levels, and second, since the threshold τ is fixed, volatility effects are not taken into account (Lopez de Prado, 2018b). These are big problems in practice as without profit taking and stop loss levels, we could be stuck in a position for an indefinite amount of time, and low-volatility periods are likely to be under-labeled.

7.2 Triple Barrier Method

The triple-barrier method, a novel approach introduced in Lopez de Prado (2018b), solves the issues found in the fixed-time horizon labeling method by implementing two horizontal and one vertical "barriers" that determine how to label the observation. Similar to the fixed-time horizon method, the triple-barrier method creates labels based on a period of returns beginning at the given observation and ending when a barrier is hit. The horizontal barriers are the profit taking and stop loss levels, while the vertical barrier is a time expiry that will end the labeling period if touched. The possible values for these labels are as follows; if the stop loss level is hit first, a -1 label is emitted. If the profit taking level is hit first, a 1 label is emitted. However, if the vertical barrier is hit first, either a 0 label or the sign of the return between the labeling period can be returned, depending on user configuration (Lopez de Prado, 2018b). The beauty of this method is that both stop loss and profit taking levels can be dynamic as a function of a user-defined metric. In most cases, it makes sense to use something like daily volatility, but other metrics like average true range (ATR), commonly used for stop loss levels, can easily be implemented. Figure 7.1 shows an example of the triple-barrier method with fixed profit taking and stop loss levels. In this example, the profit taking and stop loss levels are constant for the entire labeling period. The labeling period begins at the dotted black line and ends at the time of the first barrier touch, which happens to be at the vertical time expiry line.

With the triple-barrier method, it is also possible to learn the side of the bet when there is no underlying model for the side given by the user (Lopez de Prado, 2018b). In this case, horizontal barriers must be symmetric, and since we have no information about the side, it is arbitrarily set to long (buy) (Lopez de Prado, 2018b). Conceptually, this means that the top horizontal barrier is the profit taking level, and the bottom horizontal barrier is the stop loss level. When the price hits the profit taking barrier, the trade profits, and when the stop loss barrier is hit, the trade ends in a loss. In cases where we are unsure of the side we should take for an investment strategy, learning the side using this method may be beneficial, but generally, most investment strategies are based on an underlying model that inherently determines the side. For example, in a momentum-based trading strategy, a model for the side could be implied by the sign of the momentum. However, say a trend-following model is instead used that is able to determine when a trend occurs, but gives no information to the direction of the trend. In this case, learning the side may be beneficial, but even in this case it might be more beneficial to find a complimentary feature that models the side.

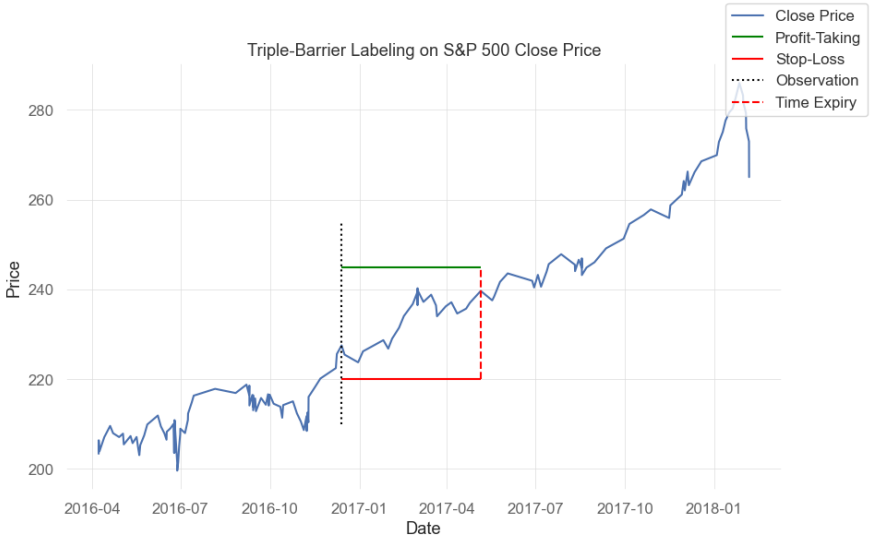

Figure 7.1: An implementation of the triple-barrier method on S&P 500 closing price. In this example, the horizontal barriers are static with the green line representing the profit taking level, red representing the stop loss level, the dashed red line representing the time expiry, and the dotted black line representing the observation date.

7.3 Meta-Labeling

Consider the case that a model for the side is defined, either by the user or an implementation of the triple-barrier labeling method. In practice, a financial algorithm generally should not consist solely of a prediction of the side we want to bet on. It is important to properly size each bet, as betting too much on losing positions can break an entirely valid, winning strategy. Meta-labeling, also introduced by Lopez de Prado (2018b), allows us to create a secondary model on top of our model for the side that solely determines the size of the bet. In the meta-labeling model, or meta model, labels are comprised of values in the set $\{0, 1\}$, where 0 labels indicate a size of zero, or no trade. To determine the actual size of the bet, we can multiply the probability of the label by the label itself to generate a vector of bet sizes. This is but one example of a bet sizing model, and more complex models will be discussed in Chapter 9. Splitting signal generation and bet sizing into two models have numerous advantages, the first of which is that training separate models allows for different features to be used in each model. This is a substantial benefit, as if a feature exhibits low predictive power in terms of determining the direction of price movement, but accurately predicts the risk at a given time, we can use this feature in our bet sizing model and exclude it from our side model. With only one model, features with these kinds of behaviors would likely be ignored and not used in the final model. Another large advantage of meta-labeling is that it can filter out false positives and increase F1-scores (Lopez de Prado, 2018b). In machine learning, the F1-score is the harmonic average between recall and precision, where recall, defined in Equation 7.3, is the ratio between true positives and false positives, and precision, defined in Equation 7.4, is the ratio

between true positives and false negatives (Müller & Guido, 2016).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7.3}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{7.4}$$

Generally, this increase in F1-score improves overall model accuracy, resulting in improved performance over the base side model without application of the meta model. Meta-labeling is also extremely versatile, as the model for side only needs to be a vector of labels. This means that any model for the side, no matter how simple or complex, can have a ML layer added on top by meta-labeling. This a key characteristic of the workflow, as it allows for simple models to reap the benefits of machine learning. One simple technical rule, such as a Simple Moving Average (SMA) crossover, could be implemented as-is and used with meta-labeling. One the other side of the spectrum, multiple models could be and used in conjunction with meta-labeling, which would be especially important if we have separate features for long and short trading. Models could even be assigned to different market regimes, volatility levels, asset classes, etc. The flexibility of meta-labeling allows for anyone to leverage the proposed workflow at any level of complexity, which makes the workflow more accessible to up-and-coming quants.

Chapter 8

Sample Weights and Class Balancing

8.1 Sample Weighting

As discussed in Chapter 1, in most cases financial data is not Independent and Identically Distributed (IID). In Chapter 7, we discussed a couple of labeling schemes for financial data. It was stated that with the triple-barrier method, an observation is labeled according to a return threshold $r_{i,0,t_{i,0}+h}$ (Equation 7.1). Since financial data is not IID, at any given time labels may "overlap" with one another, where data used to create one label, and the return $r_{t_{i,0}}$, overlaps with the data from another label (Lopez de Prado, 2018b). Figure 8.1 depicts a graphical example of label overlap, where the area between the dotted red lines is the overlapping data, and as such is the common return that both labels depend on.

This overlap can cause a lot of issues when creating machine learning models, the most notable of which is during k-fold cross validation (k-fold CV)[1]. In k-fold CV, we split the data into k train and test splits and evaluate a given model on each split. Consider the case where the training split occurs directly after the testing set. Since labels overlap, it is entirely possible to have a label y_t in the training set that overlaps with label y_{t-1} in the testing set. This "leakage" can artificially inflate model performance and contribute to overfitting, and is one of the most common issues with financial ML (Lopez de Prado, 2018b). For this reason, it is important to properly weight samples and control for possible overlap in our modeling process.

[1] see Chapter 10

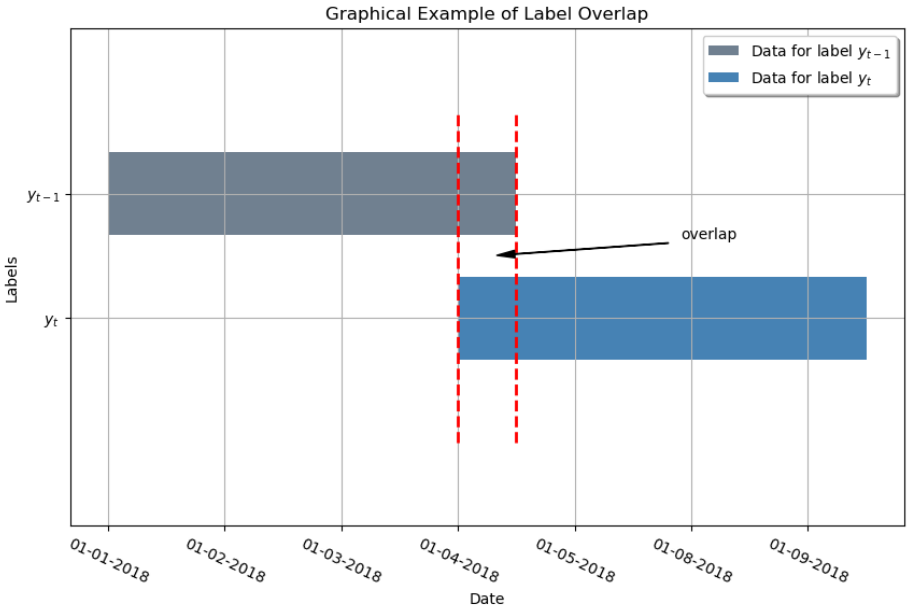

Figure 8.1: A graphical depiction of label overlap.

8.2 Sample Weights as a Function of Uniqueness

Two labels y_i and y_j are said to be concurrent at time t when both labels are a function of at least one common return (Lopez de Prado, 2018b):

$$r_{t-1,t} = \frac{p_t}{p_{t-1}} - 1 \tag{8.1}$$

where p is the price of the asset. The number of labels concurrent at time t is defined as (Lopez de Prado, 2018b):

$$c_t = \sum_{i=1}^{I} 1_{t,i} \tag{8.2}$$

where label $1_{t,i}$ is given by (Lopez de Prado, 2018b):

$$1_{t,i} = \begin{cases} 1 & \text{if } [t_{i,0}, t_{i,1}] \text{ overlaps with } [t-1, t] \\ 0 & \text{otherwise} \end{cases} \tag{8.3}$$

Using this definition for concurrency, we can then find the uniqueness of a given label, which is a measure of the lack of overlap present in the label, defined as (Lopez de Prado, 2018b):

$$u_{t,i} = 1_{t,i} c_t^{-1}. \tag{8.4}$$

The average uniqueness of a label is given by the product of sums of $u_{t,i}$ and $1_{t,i}$:

$$\bar{u}_i = \left(\sum t = 1^T u_{t,i} \right) \left(\sum t = 1^T 1_{t,i} \right)^{-1}. \tag{8.5}$$

Average uniqueness allows us to give proper weighting to unique samples and penalize overlapping labels. Proper sample weighting can simply improve model performance (Zhang & Pfister, 2021) and uniqueness weighting can give more importance to new information that enters the model. Figure 8.2 plots label concurrency values for the EWA ETF over the period 2014-2023. Figure 8.3 shows the percent average uniqueness of S&P 500 stock over the period 2014-2021. It is interesting to note that in this example, some labels are as low as 20% unique.

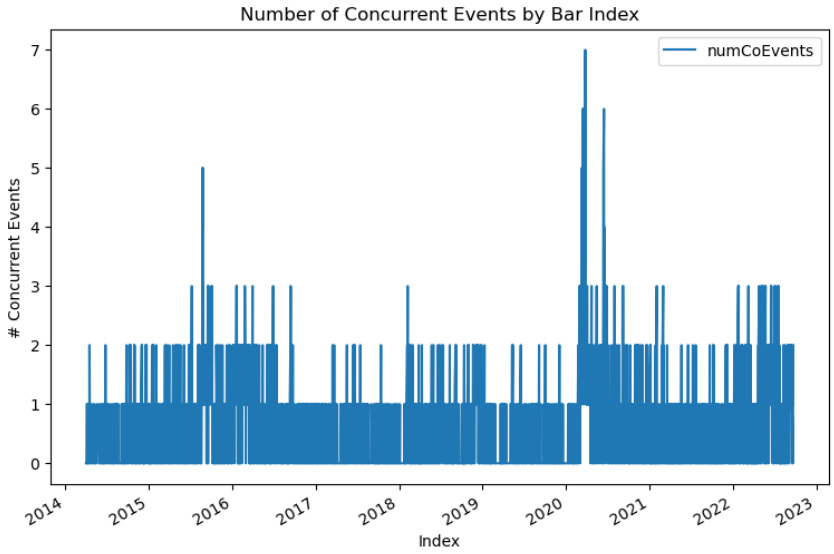

Figure 8.2: Concurrency of labels trained on EWA dollar bars between 2014 and 2023.

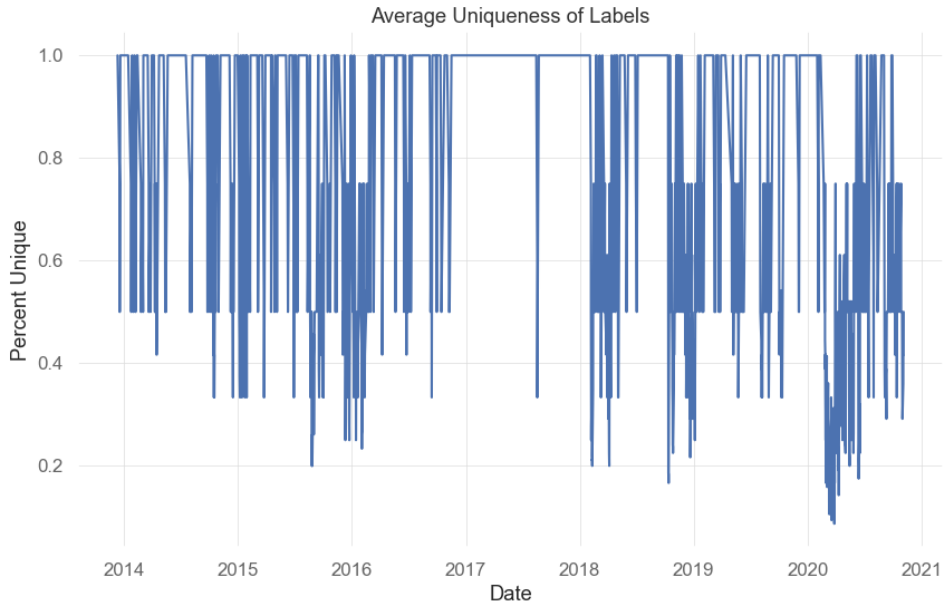

Figure 8.3: Percent average uniqueness of S&P 500 dollar bars between 2014-2021.

8.3 Return-Based Weighting

In finance, events characterized by large returns are generally considered important events, and in some cases they can be more predictable than low-return, noisy events. Return-based weighting allows for these large return events to be given more importance than those with less return. Lopez de Prado (2018b) implements this method, where weights are defined in terms of the sum of returns over the label's lifespan $[t_{i,0}, t_{i,1}]$:

$$\tilde{w}_i = \left| \sum_{t=t_{i,0}} t_{i,1} \frac{r_{t-1,t}}{c_t} \right| \tag{8.6}$$

$$w_i = \tilde{w}_i I \left(\sum_{j=1}^{I} \tilde{w}_j \right)^{-1} \tag{8.7}$$

where I is equal to the sum of all weights for label i, $\sum_{i=1}^{I} w_i = I$, c_t is number of labels concurrent (overlapping) at time t, and $r_{t-1,t}$ is the return over times $t, t-1$ (Lopez de Prado, 2018b). This method essentially weights labels as a function of unique returns over the labels lifespan, where unique returns refer to returns that don't overlap between labels. Lopez de Prado (2018b) also advises that neutral cases, where the return is below a given threshold, should be dropped altogether as they are unnecessary and likely attributed to noise. Figure 8.4 shows an example of return-based weights in practice.

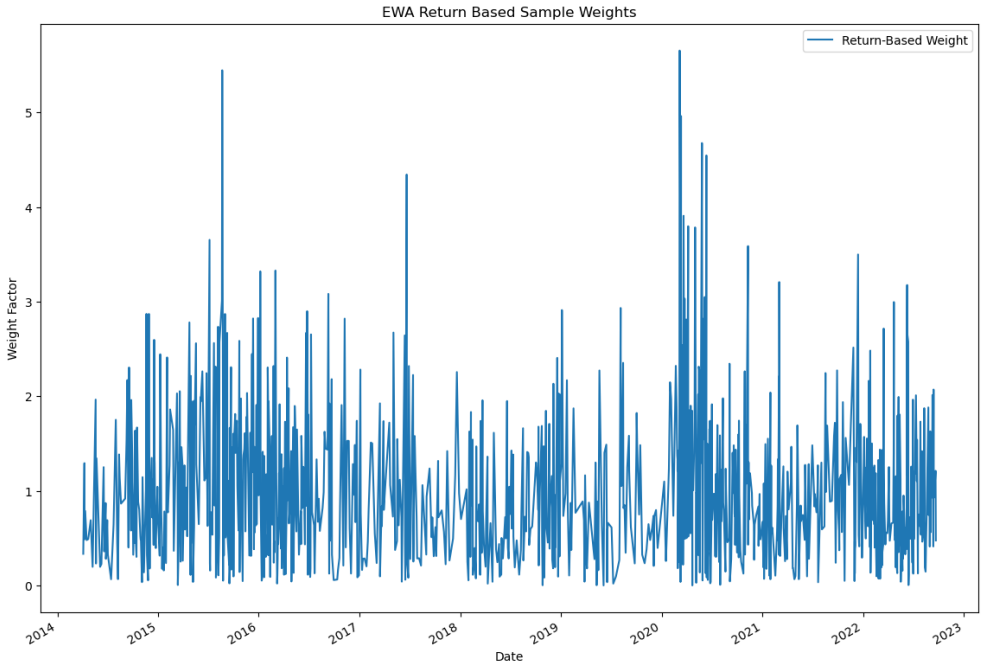

Figure 8.4: Return-based weights of EWA dollar bars between 2014 and 2023.

8.4 Time-Based Weighting

Financial markets adapt and change over time[2], and therefore a past event may not be as relevant as a recent event in a contemporary model. This makes sense conceptually as the market factors driving growth in the mid 2010's may be different than the market factors affecting the sell-offs around the 2020 COVID-19 pandemic[3]. For these reasons, it makes sense to include a time-decay factor in our weighting scheme, giving more importance to newer observations and less to older ones. Given sample weights defined in the previous section and the cumulative uniqueness $x \in \left[0, \sum_{i=1}^{I} \bar{u}_i\right]$, let $d[x]$ be the linear time decay factor that scales weights x (Lopez de Prado, 2018b). Let $c \in (-1, 1]$ be a user defined parameter that determines the decay function, where if $c \in [0, 1]$, then $d[1] = c$. However, if $c \in (-1, 0)$, then $d\left[-c\sum_{i=1}^{I} \bar{u}_i\right] = 0$ with linear decay between the two endpoints $\left[-c\sum_{i=1}^{I} \bar{u}_i, \sum_{i=1}^{I} \bar{u}_i\right]$ and $d[x] = 0 \; \forall x \leq -c\sum_{i=1}^{I} \bar{u}_i$ (Lopez de Prado, 2018b). The linear decay function $d = \max\{0, a + bx\}$ follows the conditions (Lopez de Prado, 2018b):

$$d = a + b\sum_{i=1}^{I} \bar{u}_i = 1 \Rightarrow a = 1 - b\sum_{i=1}^{I} \bar{u}_i = 1 \tag{8.8}$$

[2]See Lo (2019) (Lo, 2019) and Urquhart, McGroarty (2016) (Urquhart & McGroarty, 2016) for more information about changing market dynamics and predictability.

[3]See Fathmaningrum, Utami (2022) (Fathmaningrum & Utami, 2022) and Erdem (2020) (Erdem, 2020) for more information.

$$d = a + b0 = c \Rightarrow (1-c)\left(\sum_{i=1}^{I} \bar{u}_i\right)^{-1}, \ \forall c \in [0,1] \tag{8.9}$$

$$d = a - bc\sum_{i=1}^{I} \bar{u}_i = 0 \Rightarrow b = \left[(c+1)\sum_{i=1}^{I} \bar{u}_i\right]^{-1}, \ \forall c \in (-1,0). \tag{8.10}$$

In the given implementation, time is defined as the cumulative uniqueness $x \in \left[0, \sum_{i=1}^{I} \bar{u}_i\right]$, as a purely chronological decay would scale down the weights too fast due to overlap (Lopez de Prado, 2018b). When choosing a value for c, it is important to note how c changes the time decay factor (Lopez de Prado, 2018b); $c = 1$ means that there will be no decay, while $c = 0$ means that weights will linearly converge to zero over time. For values between $0 < c < 1$, weights will linearly converge to c, and for negative values $-1 < c < 0$, observations cT, where T is the number of observations, will receive zero weight (Lopez de Prado, 2018b). However, in most cases, values $c \geq 0$ are more practical. Figure 8.5 displays linear time decay functions for various values of c. Note that this method of weighting is intended to be used in conjunction with return-based weighting, however it is theoretically possible to use either in isolation.

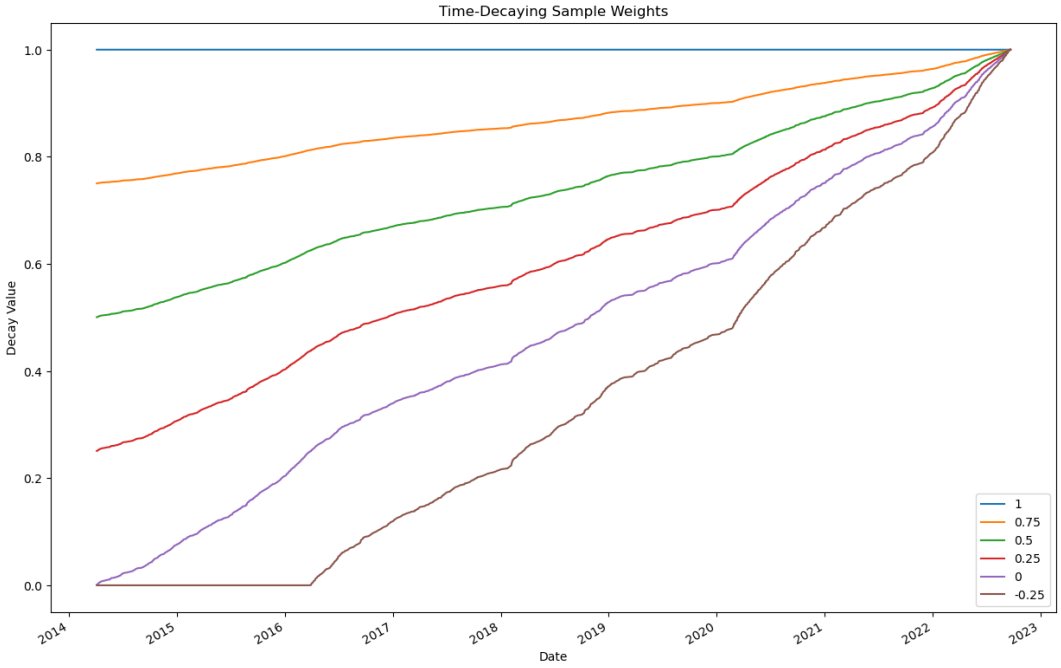

Figure 8.5: Linear time decay functions for various values of c.

8.5 Class Imbalance

Class imbalance occurs when one class of labels (i.e 0 labels) occur more often than another. In practice, class imbalance is extremely common and can sometimes even reach an unbalance factor of 90% or more (Denil & Trappenberg, 2011). In cases where the imbalance is large, standard classifiers can be overwhelmed by large classes and even ignore smaller classes altogether (Javaheri et al., 2014). Pertaining to this workflow, consider a model that is trained on a consistently upward-trending market. The model will likely have a high class imbalance towards a positive label, which may cause the model to predict upward movement more often than downward movement. If a market factor changes and it starts to trend downward, performance of the model on this new data could be unnaturally skewed due to overfitting on the other class. This interaction is characterized in Denil and Trappenberg (2011), as in combination with class overlap, class imbalance can introduce "covert overfitting" not detectable in CV and validation sets. In imbalanced datasets, classification accuracy can also give misleading information about classifier performance (Ali et al., 2015). In essence, classification imbalance is a problem that deserves attention, and it is especially important for financial machine learning, as classes tend to be imbalanced.

8.5.1 Solutions

In the case of finance, labels generally lie within the set $\{-1, 1\}$ where 0 labels can be implied by probabilities at or near 0.5 (Lopez de Prado, 2018b). Since both classes should be considered equal, we can simply use a balancing approach to weigh the classes. This is a basic approach and is present in most data science programming packages. Given n number of samples, c number of classes, and b bin counts for each class, the class weights vector w of size c classes is given by (King & Zeng, 2001):

$$w = \frac{n}{c * b} \tag{8.11}$$

This is just one approach to solving the class imbalance problem, however it is generally sufficient for financial ML. Other solutions to this problem exist such as over-sampling or under-sampling, ensemble learning, and use of specific imbalance-robust ML algorithms (Ali et al., 2015). In an effort to keep the proposed workflow generalizable, the balanced class weight approach is adopted, but the user is free to experiment with other methods.

Chapter 9

Sizing Bets

9.1 Probability-Based Sizing

In any trading system, it is important to properly size bets. In conjunction with the triple-barrier method and meta-labeling[1], we can define bet size as a function of the predicted label probability at a given time t. Let Z be the standard Normal distribution and $Z[.]$ be the cumulative density function (CDF) of Z. In the case of two outcomes (i.e. labels $x \in \{-1, 1\}$), the test statistic z is defined as (Lopez de Prado, 2018b):

$$z = \frac{p[x = 1] - \frac{1}{2}}{\sqrt{p[x = 1](1 - p[x = 1])}} = \frac{2p[x = 1] - 1}{2\sqrt{p[x = 1](1 - p[x = 1])}} \sim Z \tag{9.1}$$

where $z \in (-\infty, \infty)$. The bet size is then (Lopez de Prado, 2018b):

$$m = 2Z[z] - 1, \ m \in [-1, 1]. \tag{9.2}$$

In the case of more than two outcomes, let $X = -1, ..., 0, ...1$ be labels associated with bet sizes and the predicted label $x \in X$ (Lopez de Prado, 2018b). For each label $i = 1, ..., ||X||$, we want to estimate a probability p_i where the sum of all probabilities equals one. $\tilde{p} = \max_i\{p_i\}$ is the probability of x with null hypothesis H_0: $\tilde{p} = \frac{1}{||X||}^2$ (Lopez de Prado, 2018b). The test statistic z is then defined as (Lopez de Prado, 2018b):

$$z = \frac{\tilde{p} - \frac{1}{||X||}}{\sqrt{\tilde{p}(1 - \tilde{p})}} \sim Z \tag{9.3}$$

where $z \in [0, +\infty)$. The bet size is then (Lopez de Prado, 2018b):

$$m = x(2Z[z] - 1) \tag{9.4}$$

where $(2Z[z] - 1) \in [0, 1]$ and $m \in [-1, 1]$. Negative values for bet size m imply a negative, or short, position while positive bet sizes imply a positive, or long position. The cumulative density function (CDF) of bet sizes from predicted probability is shown in Figure 9.1. Since labels can

[1] see Chapter 7

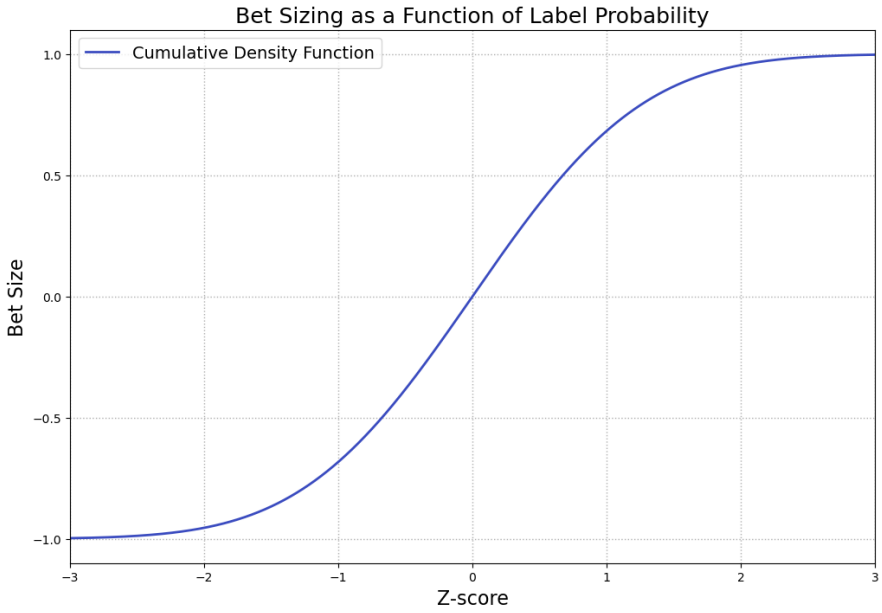

Figure 9.1: Bet sizes from probabilities of labels $x \in -1, 1$, where bet size is given by Equation 9.2, and Z-score is given by Equation 9.1.

overlap, concurrent bets should be averaged to avoid excessive trades (Lopez de Prado, 2018b). This can simply by done by taking the average bet size for each label overlapping at time $[t_{i,0}, t_{i,1}]$. Since it is possible that bet sizes can change in small amounts after each prediction, there may still be excessive trading. To remedy this, we can discretize the bet size per a degree of discretization $d \in (0, 1]$ where (Lopez de Prado, 2018b):

$$m^* = \text{round}\left[\frac{m}{d}\right] d \tag{9.5}$$

This results in a "stepped" bet size function that corrects for small changes in the bet size as seen in Figure 9.2. Figure 9.3 depicts probability based bet-sizes in practice with averaging and discretization applied. Notice that bet sizes differ in a stepwise manner, which will result in less turnover (trades) than continuous bet sizes.

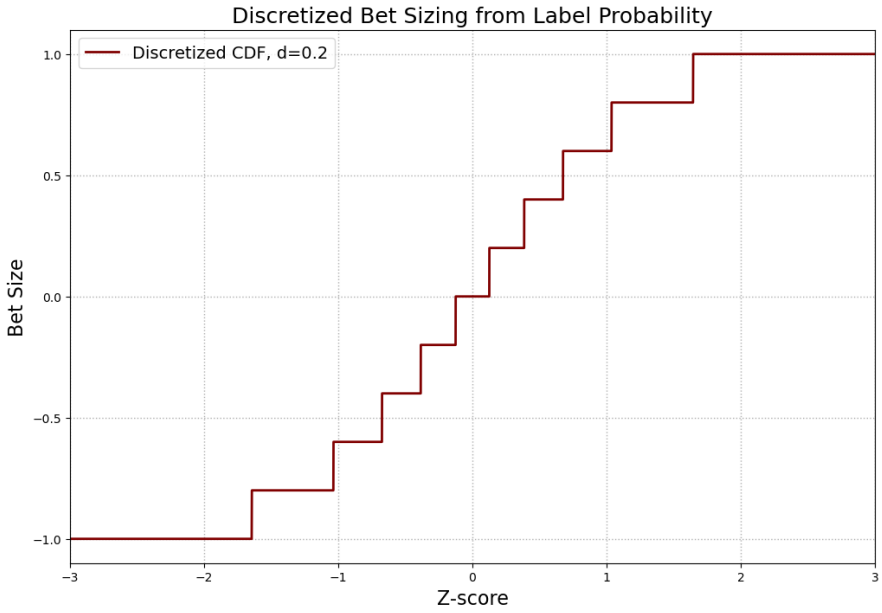

Figure 9.2: Discretized bet size cumulative density function (CDF) where $d = 0.2$.

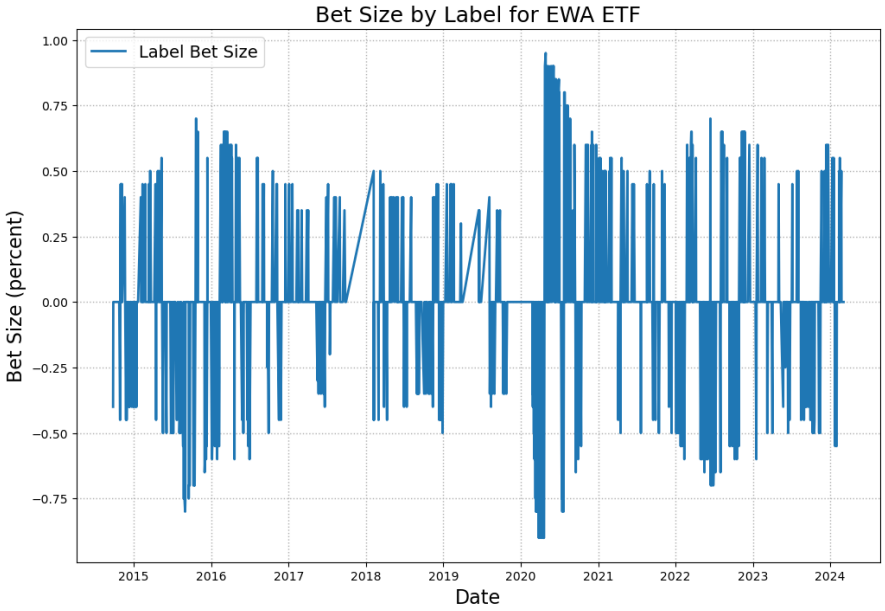

Figure 9.3: Bet sizes in practice for EWA ETF over the period 2015-2024 after averaging and discretization have been applied.

9.2 Dynamic Bet Sizing

With the triple-barrier labeling method, bar i is formed at time $t_{i,0}$ and a forecast is made for the first barrier that will be touched (Lopez de Prado, 2018b). The time it takes until the outcome takes place is defined as $t \in [t_{i,0}, t_{i,1}]$, and during this time price p_t fluctuates, possibly forming additional forecasts $E_{t_{j,0}}[p_{t_{i,1}}]$, where $j \in [i+1, I]$ and $t_{j,0} \leq t_{i,1}$ (Lopez de Prado, 2018b). Recall that I is the number of rows in the feature matrix X, and i is the i-th row. Let p_t equal the market price at time t and f_i the forecast price at row i. We want to find the current position q_t with absolute maximum position size Q. For forecast f_i, the target position size $\hat{q}_{i,t}$ is defined as (Lopez de Prado, 2018b):

$$\hat{q}_{i,t} = \text{int}[m[w, f_i - p_t] * Q] \tag{9.6}$$

$$m[w, x] = \frac{x}{\sqrt{w + x^2}} \tag{9.7}$$

where int[.] refers to the integer value of the given argument. $m[w, x]$ defines the bet size according to parameters w, which is a coefficient scaling the width of the sigmoid function, and x, which is the difference between forecast f_i and price p_t, $f_i - p_t$ (Lopez de Prado, 2018b). Target position size $\hat{q}_{i,t}$ is bounded by the maximum position size Q such that $-Q < \hat{q}_{i,t} < Q$ and bet size is within bounds $-1 < m[w, x] < 1$ (Lopez de Prado, 2018b). If bet size $m[w, x]$ was allowed to be outside bounds $-1 < m[w, x] < 1$, $|m[w, x] > 1|$ would imply bet sizes larger than the maximum position size Q and lead to over-allocation of capital. This is technically possible through leverage, which is essentially the act of borrowing money from a broker to fill a position, however this practice is extremely risky and should not be done without careful implementation. Target position size $\hat{q}_{i,t}$ can be dynamically adjusted as p_t changes, since $\hat{q}_{i,t} \to 0$ as $p_t \to f_i$ implies a break-even limit price $\bar{p}$ for order size $\hat{q}_{i,t} - q_t$ (Lopez de Prado, 2018b). Limit price $\bar{p}$ is defined as (Lopez de Prado, 2018b):

$$\bar{p} = \frac{1}{|\hat{q}_{i,t} - q_t|} \sum_{j=|q_t + \text{sgn}[\hat{q}_{i,t} - q_t]|}^{|\hat{q}_{i,t}|} L\left[f_i, w, \frac{j}{Q}\right] \tag{9.8}$$

where $L[f_i, w, m]$ is the inverse function of $m[w, f_i - p_t]$ defined as (Lopez de Prado, 2018b):

$$L[f_i, w, m] = f_i - m\sqrt{\frac{w}{1 - m^2}} \tag{9.9}$$

The sgn text string represents the sign operator:

$$sgn(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases}.$$

Since Equation 9.8 is monotonic, meaning it always increases over its domain, the case $m^2 = 1$ will not result in division by zero as $|\hat{q}_{i,t}| < 1$ and losses cannot be realized as $p_t \to f_i$ (Lopez de Prado, 2018b). To calibrate w, a user-defined parameter (x, m^*) is needed such that $x = f_i - p_t$ and $m^* = m[w, x]$ so that w can be defined as the inverse function of $m[w, x]$ (Lopez de Prado, 2018b):

$$w = x^2(m^{*-2} - 1) \tag{9.10}$$

This dynamic bet sizing framework is best for cases when price forecasts are accurate, as position sizes will be scaled according to the price difference between the forecasted and actual prices. However, since we are implementing meta-labeling and have prediction probabilities, bet size is derived from probability in this workflow.

Chapter 10

Cross Validation

Cross validation (CV) is a method of evaluating the performance of a model by training and testing the model on different sections of the data. CV generates multiple partitions, called train-test splits, and trains/tests the model on each split. A training set is a part of the dataset that is used to train the model, while a testing set is another part of the data that the model tests its predictions on. This allows us to evaluate the performance of the model at each of the splits.

10.1 k-Fold Cross Validation

In k-fold cross validation, (k-fold CV), we split the dataset into k train and test splits of approximately equal size, which are called folds (Müller & Guido, 2016). At each split, there are k folds, with one test fold and $k-1$ training folds. The fold number of the test set increases sequentially over k splits, where fold 1 is the test fold for the first split, fold 2 is the test fold for the second split, etc. A visual example of k-fold CV is given in Figure 10.1.

Figure 10.1: Train/test splits generated by k-fold cross validation.

A common use of k-fold CV is to generalize model accuracy over different splits. One implementation of this is to average the accuracy of the model at each split. The average accuracy provides an indication of how the model may perform when trained with various datasets. This is important because with only one train-test split, it is possible to get "lucky" in the sense that examples easy to classify end up in the training set, while harder examples stay in the testing set (Müller & Guido, 2016). This would result in an inflated model accuracy that likely would not generalize well out-of-sample. With CV, each label appears in the testing set exactly once, allowing for the model to be trained and tested on different conditions. For financial applications, CV conceptually makes sense as markets change over time (Lo, 2019) and it is important to evaluate the accuracy of the model across various market conditions.

10.2 Applications for Cross Validation

Cross validation is used in multiple settings across the modeling process. The two most common applications are for model selection and hyperparameter tuning. When selecting a ML model, some metric such as accuracy is evaluated between each model, and the model with the best metric is selected. It is important to use a generalized accuracy metric for this comparison, as models trained on a single train-test split are likely to have bias. Once a model is selected, we can apply CV again to tune the hyperparameters of the model. Hyperparameters refer to model parameters that control how the model learns from the data.

10.3 Purged k-Fold Cross Validation

Since financial data is not independent and identically distributed (IID), data from the testing set can "leak" into training sets[1], causing artificially inflated performance and overfit models. For this reason, Lopez de Prado (2018b) introduces a new CV method, purged k-fold cross validation, that controls for overlap. A label is purged from the testing set if it overlaps[2] with a label in the training set. This ensures that possible overlap between training and testing sets are purged from each split. Additionally, due to serial correlation[3] of features, observations that directly follow a testing set should be eliminated to control for overlap not caught by purging (Lopez de Prado, 2018b). Figure 10.2 depicts training and testing splits generated by purged k-fold CV, where red indicates purged or embargoed data points from the split.

[1] See Chapter 8 for more information on label overlap.

[2] Label y_i is said to overlap with label y_j if both labels contain a common return $r_{t-1,t}$, i.e. some of the data used to create y_i was also used to create y_j.

[3] Serial correlation is the correlation of a variable to its past values, i.e a trending stock exhibits positive serial correlation as it gains momentum (Tsay, 2010).

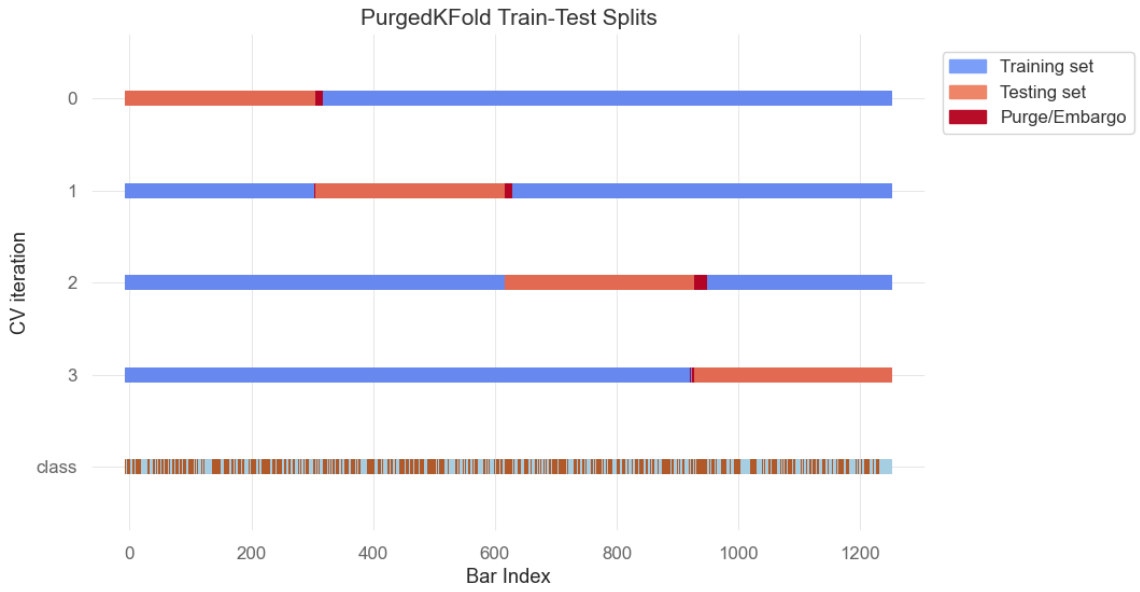

Figure 10.2: Train/test splits generated by purged k-fold cross validation. Red indicates purged and embargoed data points.

10.3.1 Purging Training Sets

Overlap occurs between observations i and j when their labels y_i and y_j depend on a common return. If label y_j is a function of observations in the range $[t_{j,0}, t_{j_1}]$ and label y_i is a function of observations in the range $[t_{i,0}, t_{i_1}]$, y_i overlaps with y_j if (Lopez de Prado, 2018b):

$$t_{j,0} \leq t_{i,0} \leq t_{j,1} \tag{10.1}$$

$$t_{j,0} \leq t_{i,1} \leq t_{j,1} \tag{10.2}$$

$$t_{i,0} \leq t_{j,0} \leq t_{j,1} \leq t_{i,1} \tag{10.3}$$

If a label meets any one of these conditions, it is purged from the training set. In Figure 10.2, purged data points are depicted in red at the start of each testing set.

10.3.2 Embargo

Since serial correlation is temporal, we only need to embargo observations after training sets. To achieve this, we can add a small threshold, generally $0.01T$, where T is the number of observations, to the end of the time period of label y_j, $[t_{j,0}, t_{j_1}]$, such that y_j spans observations $[t_{j,0}, t_{j_1} + 0.01T]$ (Lopez de Prado, 2018b). Figure 10.2 indicates embargoed observations after each training set in red.

Chapter 11

Feature Importance and Strategy Research

For the reasons discussed in Chapter 12, backtesting should not be used as a research tool due to multiple testing and overfitting. Instead, research should be conducted through feature importance methods as they give insight into the drivers of positive performance (Lopez de Prado, 2018b). In the following sections, methods for feature importance and selection will be introduced. It is important to be aware of the drawbacks of each method, as not properly accounting for issues like collinearity can lead to misleading results. In general, when using feature importance as a research tool, multiple methods should be implemented and used in conjunction. In the case of a random forest model, we may find that a single feature has high mean decrease impurity (MDI) importance. We can then evaluate that feature in isolation to see if the feature is important independently or through a combination effect systematically introduced in the random forest model.

11.1 Feature Selection

Feature selection is an important step in model creation, as some features may be redundant or lack importance in the prediction of the model. Additionally, as dimensionality of the data increases, data analysis and classification can become more challenging, and a large number of features can lead to lower classification accuracy (Janecek et al., 2008). Dimensionality reduction is the process of reducing the number of dimensions in a dataset, or in this case, reducing the number of features used to train the model. A common method for this is Principle Component Analysis (PCA), which reduces dimensionailty by creating new, uncorrelated variables such that variance is maximized (Jolliffe & Cadima, 2016). PCA also reduces collinearity (also called substitution effects) between features, which occurs when multiple features are highly correlated to each other. Collinearity is a problem for machine learning algorithms as it can reduce classification accuracy (Howley et al., 2006) and feature importance scores (Lopez de Prado, 2018b). In the next sections, common feature selection methods will be briefly introduced[1].

[1] Guyon and Elisseeff (2003) provide an introduction and check-list to solving the feature selection problem, while Kuhn and Johnson (2019) provide a comprehensive coverage of feature selection methods.

11.1.1 Filter Methods

Filter feature selection methods refer to filtering the feature set through rank metrics. This is one of the more basic feature selection methods, as we simply aim to filter out features with low effect on the overall classification accuracy of the model. Among other benefits, feature ranking is computationally light and avoids overfitting (Chandrashekar & Sahin, 2014). One simple ranking metric is Pearson Correlation, defined as (Chandrashekar & Sahin, 2014):

$$R(i) = \frac{\text{cov}(x_i, Y)}{\sqrt{\text{var}(x_i) * \text{var}(Y)}} \tag{11.1}$$

where x_i refers to the ith feature, Y is the class labels, and cov() is the covariance and var() is the variance. If the feature x_i is not significantly correlated with the class labels Y, it is likely that the feature has little predictive power. Filter methods suffer drawbacks in that combination effects can be discarded, and there is no ideal method for choosing the dimension of the feature subspace (Chandrashekar & Sahin, 2014).

11.1.2 Wrapper Methods

Wrapper methods essentially treat the classifier as a "black box" and use classifier performance to optimize a given subset of features (Chandrashekar & Sahin, 2014). Since evaluating feature subsets can become computationally intractable as the number of features increase, various optimization methods such as heuristic and sequential algorithms are used. Since performance is optimized, wrapper methods are prone to overfitting and computationally inefficient, as some optimizers may re-evaluate the same subsets of features (Chandrashekar & Sahin, 2014).

11.1.3 Embedded Methods

Embedded Methods attempt to solve some of the issues of wrapper and filter methods by incorporating feature selection as part of the model training process (Chandrashekar & Sahin, 2014). One possible implementation is to use class weights to rank features for possible removal from the feature set. For more information, see Guyon and Elisseeff (2003).

11.1.4 Combination Effects

In some cases, two weak features may be combined to create a stronger, more predictive feature. Kuhn and Johnson (2019) give an intuitive conceptual example of how combining predictors can improve predictions. Imagine we want to predict corn yield from the amount of water and fertilizer in the soil. If the soil has no water but enough fertilizer, there will be no yield since plants need water to grow. Conversely, if the soil has water but no fertilizer, some yield will occur, but not the optimal yield since there is no fertilizer. If the features are combined, however, the optimal yield can be achieved since it depends on correct amounts of both water and fertilizer. In a more general sense, a feature may have little predictive power on its own, but when combined with the right feature, better predictions can be achieved. However, combinations are generally unknown in practice and need to be discovered. In some cases, such as in García-Magariños et al. (2009), tree-based models such as Random Forest can discover interaction effects between predictors (Breiman et al., 1984). If the feature matrix is small enough, brute force can also be used to discover pair-wise

combination effects, however the probability that the combination effect is due to random chance is increased as more features are evaluated (Kuhn & Johnson, 2019). For an in-depth look at combination effects (also called interaction effects), see Kuhn and Johnson (2019).

11.2 Feature Importance Methods

11.2.1 Single Feature Importance

Single feature importance is a feature importance method that computes the out-of-sample performance score of each feature in isolation (Lopez de Prado, 2018b). Essentially, this method iterates over a given feature matrix X and evaluates the performance of each feature through cross validation. The importance of that feature is given by the average performance score across all CV splits. This method is helpful because it is robust to multicollinearity and can be computed on any classifier. Figure 11.1 displays single feature importance scores.

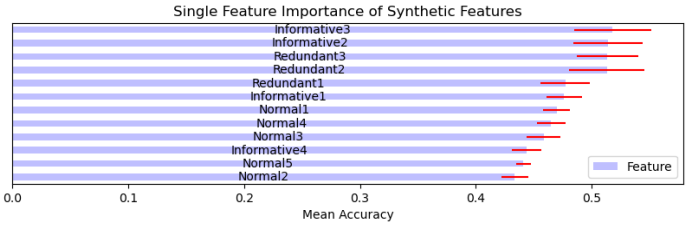

Figure 11.1: Single feature importance (SFI) scores for twelve synthetic features. Four features are informative and have predictive power, three are redundant, and five are normal features without bias. Red lines at the ends of the mean accuracy scores represent the standard deviation of CV scores.

Note that since the metric used in this example is accuracy, importance scores above 0.5 indicate that the feature produced a model that was better than random choice. The drawback to single feature importance is that combination effects between features are ignored, and a classifier trained on two features can perform better than combining two single-feature classifiers (Lopez de Prado, 2018b).

11.2.2 Mean Decrease Impurity

Mean decrease impurity (MDI) is a feature importance method that evaluates a feature by its decrease in impurity in tree-based classifiers. In the context of a descision tree, nodes split their received set into subsets, where impurity represents how "pure" (containing only one class) the subset is. Nodes classify observations by attempting to decrease impurity, and thus we can relate each feature to its average impurity decrease (Lopez de Prado, 2018b). The importance of feature X_i for predicting labels y is given by the weighted sum decrease impurity averaged across all trees $\varphi_m, m = 1, ..., M$ for all nodes t where feature X_i is used, formally defined as (Louppe, 2015):

$$\text{Imp}(X_i) = \frac{1}{M} \sum_{M}^{m=1} \sum_{t \in \varphi_m} 1(i_t = i) \left[p(t) \Delta i(s_t, t) \right] \tag{11.2}$$

where the weighted impurity decrease is $p(t)\Delta i(s_t, t)$, $p(t)$ is the proportion $\frac{N_t}{N}$ where N is the number of samples reaching t, and i_t is the variable (feature) used for splitting the subset passed to node t. Note that in the case of a single decision tree, $M = 1$ and thus $\text{Imp}(X_i)$ is the mean impurity decrease for that single tree. MDI has some importance caveats (Lopez de Prado, 2018b), the most notable of which is that tree-based classifiers can systematically ignore features in favor of others, and as such only one feature should be considered per tree level. MDI is not robust to collinearity, and the importance of substitute features will be diluted (Lopez de Prado, 2018b). Figure 11.2 plots MDI importance for twelve synthetically-created features. Notice that in some cases, a redundant and informative feature have approximately the same importance. This gives insight into the collinearity problem, as the true importance of the informative feature has likely been halved between the two features.

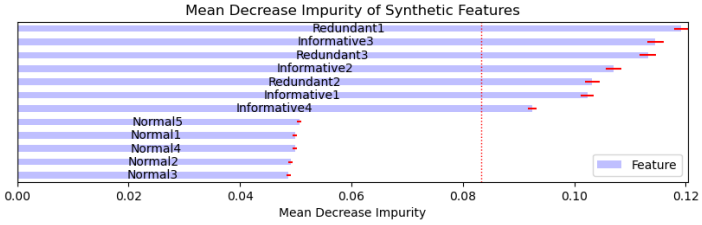

Figure 11.2: Mean decrease impurity (MDI) feature importance for twelve synthetic features. Four features are informative and have predictive power, three are redundant, and five are normal features without bias. The red bars at the end of each feature bar is the average MDI across all trees, while the dotted red line represents the standard deviation of that average.

11.2.3 Mean Decrease Accuracy

As oppose to MDI, mean decrease accuracy (MDA) evaluates the importance of a feature X_i by measuring the mean decrease in accuracy (synonomously mean increase error) that X_i contributes when the feature is randomly permuted and evaluated out-of-sample (Louppe, 2015). MDA derives the out-of-sample performance of the classifier, then permutes each feature in the given feature matrix, re-evaluating the out-of-sample performance after each permutation (Lopez de Prado, 2018b). For this reason, MDA is also called permutation importance, and it is able to be used with any classifier. The idea behind MDA is that if a feature is important, the mean decrease accuracy will be high because the relationship between the feature and the target variable is broken. The importance for feature X_i is given by the difference between base classifier performance s (without permutation) and the average performance after permutation, formally ("4.2. Permutation Feature Importance", n.d.):

$$X_i = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,i} \tag{11.3}$$

where $s_{k,i}$ represents the accuracy score after each permutation for a number of repetitions $k = 1, ..., K$. Similar to MDI, MDA is not robust to collinearity and can assign low importance scores to correlated features, even if they are actually important (Lopez de Prado, 2018b). MDA can also be used with other scoring metrics, which can be beneficial in cases like meta-labeling, where we may want to instead evaluate F1-score (Lopez de Prado, 2018b). Figure 11.3 plots MDA importance for

twelve synthetically-created features. In this example, the standard deviation of mean accuracies is quite large, however informative features still have MDA scores higher than zero.

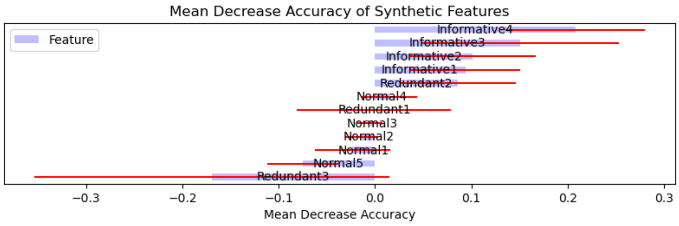

Figure 11.3: Mean decrease accuracy (MDA) feature importance for twelve synthetic features. Four features are informative and have predictive power, three are redundant, and five are normal features without bias. The red bars at the end of each feature bar represent the standard deviation of the mean accuracy.

Chapter 12

Backtesting

A backtest is a simulation of how an algorithmic strategy or model would have performed over some historical time period. Performance can be evaluated through various metrics such as profit/loss, cumulative return, etc., but the most commonly used metric for generalizing strategy performance is the Sharpe Ratio (SR) (see Equation 12.6), where values above 1.0 are generally considered "good". A common misuse of backtesting is as a research tool (Lopez de Prado, 2018b), where the user may draw some conclusion from an out-of-sample test period and re-tune the strategy to perform better over that period. This is extremely problematic as it leads to overfit models and false strategies that fail on live markets. Worse yet, with enough attempts, any researcher can find a backtest with the desired SR regardless of sample length (Bailey et al., 2014). For these reasons, research should instead be conducted through feature importance, and backtests should only be used after the strategy creation process is finished to discard bad models (Lopez de Prado, 2018b).

12.1 Common Forms of Bias

12.1.1 Survivorship Bias

Survivorship Bias is a common form of bias in which only assets that "survive" the test of time, i.e. are not delisted or removed from the investment universe, are used in a model. This is especially prevalent in the case of indices like the S&P 500, as the index constituents change over time. For example, say we wanted use the constituents in the S&P 500 index as a broad universe for a more specific selection algorithm, and we had price data available between the period 2018-2024. If the current (2024) constituent list was used to source our data, we would be introducing survivorship bias into our model unless only the period of data containing strictly those constituents was used for the model. Models affected by survivorship bias can exhibit artificial performance gains and reduced consequence of losses (Wang et al., 2014).

12.1.2 Look-Ahead Bias

Look-ahead bias refers to the act of using future information at a time when it is not available. This can present itself in many ways throughout a backtest, the most obvious being financial statement (fundamental) data (Wang et al., 2014). Although companies report fundamentals on a

regular fiscal period (i.e. quarterly), it takes an average of two months after the fiscal period ends to release their financial statements (Wang et al., 2014). This means that historical fundamental data is on average two months ahead of the time it was actually reported. Look-ahead bias also commonly presents itself through use of known data about an underlying market ahead of time, i.e. including a specific asset in a universe because it performed well in the past.

12.1.3 Data-Snooping Bias

Data-snooping refers to the process of extensively searching for rules or patterns that maximize model performance in-sample (Wang et al., 2014). Since the amount of data in finance is limited, data-snooping bias is especially prevalent. One example of data-snooping is excessive in-sample parameter optimization, but bias can also occur through model decisions like trading on close vs. open prices, asset universes to trade on, etc. (Chan, 2009). Data-snooping bias usually follows hand-in-hand with look-ahead bias, as we may see that our out-of-sample performance is not desirable and be tempted to refit the model based on that performance, an example of look-ahead bias.

12.1.4 Selection Bias

Selection bias occurs under multiple testing, where we test multiple hypotheses and select the ones that pass a given test statistic, ignoring those that don't (Bailey & Lopez de Prado, 2014). This can manifest itself in multiple ways in the field of finance, such as partial reporting of back-test results, where negative results are ignored, or selecting specific assets based on out-of-sample performance (Bailey & Lopez de Prado, 2014).

12.1.5 Trading Costs and Reality Modeling

In historical simulation, it may not be possible to capture all of the nuances of real-time trading such as bid ask spreads, order times, and shorting costs. Most backtests are generated on the premise that the model will execute trades at an exact price, such as open or close price. In reality, bid-ask spreads and liquidity can affect the order execution price and time. Transaction and shorting costs can also be increasingly hard to properly model as they requires an extensive amount of data (such as the entire order book) that may not be available or is hard to obtain (Wang et al., 2014). In these cases, generalizations are usually made, such as setting transaction costs equal to 20bps[1]. For these reasons, it is important to first "paper trade" a strategy before it is used on live markets. Paper trading refers to the process of deploying a strategy on live market data using fake or "paper" money. This allows us to gain insight on how the strategy performs with liqidity and transaction cost constraints, which are challenging to model in a backtesting environment.

12.1.6 Confirmation Bias

Especially when dealing with "black-box" type systems like machine learning algorithms, it is easy to falsely rationalize unknown behavior as significant. Confirmation bias most frequently appears in finance after backtesting, where the user may rationalize performance by some false

[1] "bps" refers to basis points, where one basis point is one one-hundredth of a percent. 20bps is equal to 0.002%.

economic claim. Say we randomly try different features in a model and stumble upon a good-looking backtest. Confirmation bias would occur if we tried to rationalize the positive performance by some explanation related to these random features (Wang et al., 2014).

12.2 Single-Split Backtesting

Single-split backtesting is a case of walk-forward backtesting, where given a historical period, a strategy is created on an in-sample period (training set) and tested on an out-of-sample period (testing set). This form of backtesting only considers one historical path and thus is easy to overfit and draw false conclusions (Lopez de Prado, 2018a). For example, if we had stock data available for the time period 2014-2018, we may use the first three years (2014-2017) as a training period and the last year (2017-2018) as a testing period. If some market factor caused the universe of stocks used in our strategy to exhibit substantial positive performance from 2017-2018, our backtest may also show substantial positive performance due to that rally and not necessarily because of our underlying strategy. With a single backtest path, it is challenging to get a general idea of strategy performance across different market conditions, especially since data is limited. Figure 12.1 depicts a visual look at single-split backtesting, where we only create one training and one testing set over our historical data period.

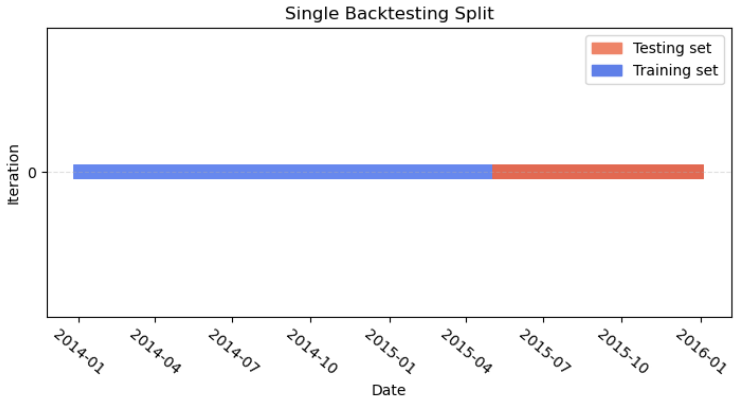

Figure 12.1: Single-split backtesting example over the period 2014-2016.

12.3 Walk-Forward Backtesting

The most common form of backtesting is walk-forward (WF) backtesting, where a strategy is trained/created on a portion of the historical dataset and tested against a period after the training set. Similar to single-split backtesting, walk-forward separates the data into various train test splits, but instead of considering just one historical path, multiple train-test splits are created sequentially in a step-wise fashion. This allows for the model to be evaluated on multiple "out-of-sample" test splits over the period of the backtest. Figure 12.2 illustrates walk-forward backtesting, where each train test split is "walked-forward" in a manner than prevents test split overlap.

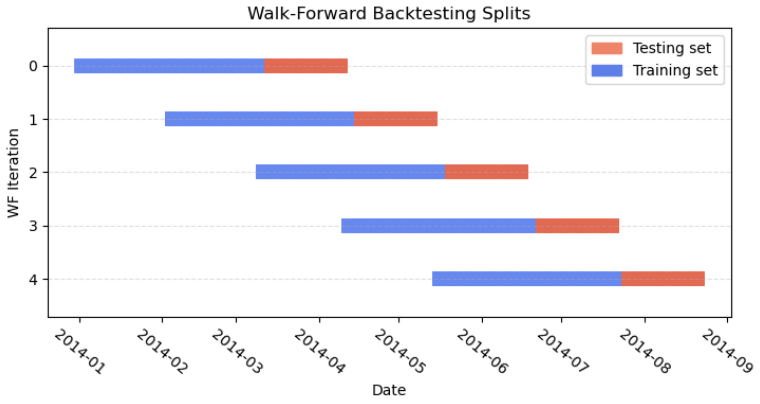

Figure 12.2: Walk-forward backtesting example over the period 2014-2016.

One possible drawback is that training sets can overlap, placing some events in multiple splits. It is possible to "walk-forward" so that training splits don't overlap, however this would further limit the maximum number of splits allowed for a given data size. Walk-forward backtesting is susceptible to leakage, and therefore training sets need to be purged, which is generally not accounted for in practice (Lopez de Prado, 2018b). As with single-split backtesting, WF only considers a single historical path (in this case, separated into n-splits), and can be influenced by the underlying performance of the investment universe (sequence of datapoints) (Bailey et al., 2014). The main argument for walk-forward backtesting is that predicting the past would produce optimistic performance estimates, however Lopez de Prado (2018b) notes that "it is as easy to overfit a walk-forward backtest as to overfit a walk-backward backtest, and the fact that changing the sequence of observations yields inconsistent outcomes is evidence of that overfitting". Splitting the dataset for evaluation through walk-forward backtesting also limits the amount of data used to train a model, which can have detrimental effects especially in cases when the underlying model is a ML model.

12.4 Proper Backtesting Practice

12.4.1 CPCV

Walk-forward backtesting only tests one historical path, and is thus prone to overfitting. Lopez de Prado (2018b) introduces a new backtesting method called Combinatorial Purged Cross Validation (CPCV), which generates a user-defined number of backtest paths φ through a combination of train-test splits needed to achieve φ paths. For T observations and N groups, groups $n = 1, ..., N-1$ are of equal size $\frac{T}{N}$ and each group has the same number of training and testing sets (Lopez de Prado, 2018b). For a testing set of size k groups (i.e. $k = 2$ means the testing set size is $2 * \frac{T}{N}$), the number of possible train-test splits is defined as (Lopez de Prado, 2018b):

$$\binom{N}{N-k} = \frac{\prod_{i=0}^{k-1}(N-i)}{k!}. \tag{12.1}$$

This results in a total number of backtest paths $\varphi[N, k]$, where (Lopez de Prado, 2018b):

$$\varphi[N, k] = \frac{k}{m}\binom{N}{N-k} = \frac{\prod_{i=1}^{k-1}(N-i)}{(k-1)!}. \tag{12.2}$$

A clear advantage of CPCV is that backtest paths φ, groups N, and testing set size k groups are all user-defined, allowing for customization of the backtesting process. Lopez de Prado (2018b) recommends $k \leq \frac{N}{2}$ as the portion of data used for training sets $\theta = 1 - \frac{k}{N}$ should be at least $\theta \geq \frac{1}{2}$. Backtest paths are generated by combining testing set predictions from various train-test splits. Figure 12.3 illustrates the distribution of train-test splits for CPCV with paths $\varphi = 5$, groups $N = 6$, and test set size $k = 2$ groups.

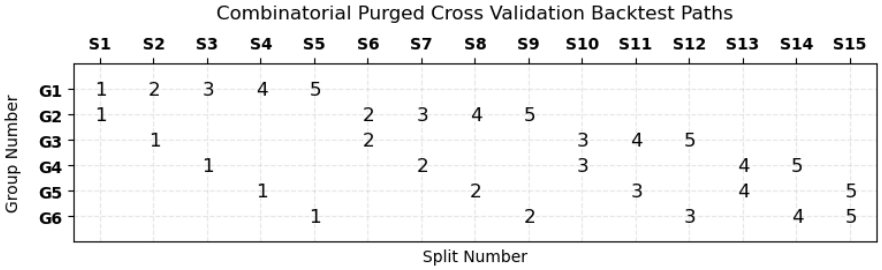

Figure 12.3: A representation of Combinatorial Purged Cross Validation train-test splits and backtest paths for groups $N = 6$ and test set size $k = 2$, generating $\varphi = 5$ backtest paths.

In Figure 12.3, numbers 1-5 on the chart indicate testing sets, where the number itself indicates which path that test set belongs to. Since there are $N = 6$ groups, each backtest path contains six test sets from various splits. Within each split, overlapping observations between train and test sets are purged, and an embargo period is added after testing splits followed by a training set. Figure 12.4 represents a CPCV implementation again with $\varphi = 5$, groups $N = 6$, and test set size $k = 2$ groups. The red areas in each split represent the purged and embargoed datapoints. It is clear in Figure 12.4 that test sets do not overlap, and any possible overlap is removed due to the purge and embargo process.

12.4.2 Scientific Backtesting: Thinking Like a Scientist

The creation of an investment strategy usually begins with a prior belief that some rule or pattern can predict and profit from some market event (Bailey et al., 2014). In the Quant Workflow, we are informally testing a given hypothesis about its efficacy in a market. For example, say we believe that markets follow trends, and trending markets are likely to continue moving in the direction of the trend (trend-following). We then create features, such as moving averages, and research the hypothesis through feature importance. When we finish our research (i.e. we have found important features), we then evaluate the hypothesis through a backtest. There are an extremely large number of possible model configurations (trails) and we would usually like to select the configuration with the best performance (Bailey et al., 2014). The number of possible strategy configurations (trails) is extremely large due to the number of possible securities and model parameters (Bailey et al., 2015). For most statistical tests, we would normally evaluate a given statistic at a 95% confidence

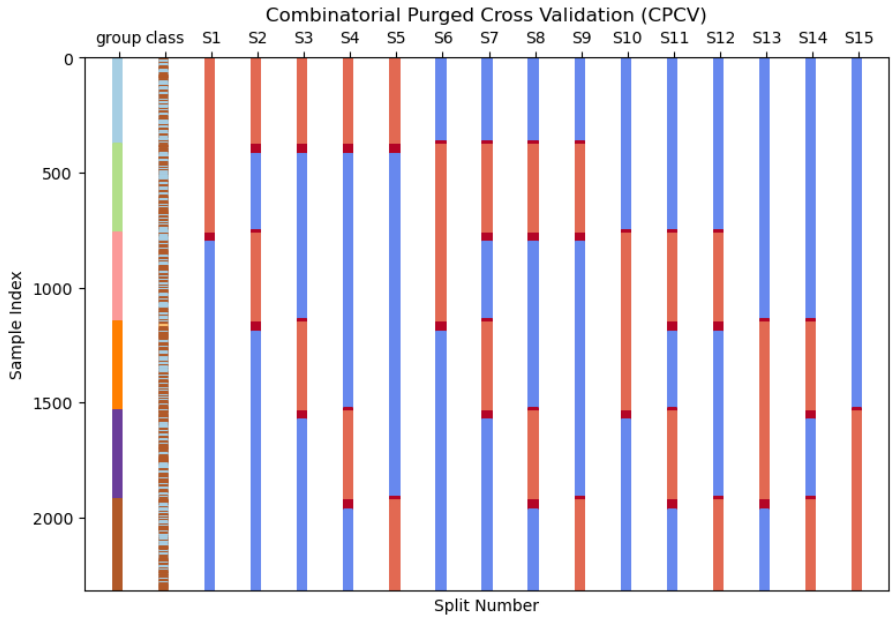

Figure 12.4: A representation of Combinatorial Purged Cross Validation train-test splits and backtest paths for groups $N = 6$ and test set size $k = 2$, generating $\varphi = 5$ backtest paths.

rate with a false-positive rate of 5%. However, since we are conducting the same test more than once (one time for each strategy configuration), the 5% probability of a false positive does not hold and as such reaching a false positive is almost certain (Bailey et al., 2015).

A backtest is said to be "realistic" if the in-sample performance is consistent with the out-of-sample performance (Bailey et al., 2014).

12.4.3 Hypothesis Testing

Hypothesis testing is the process in which we use statistical methods to answer conjectures related to a problem space. The hypothesis testing process usually proceeds as follows (James et al., 2023):

1. Define the null and alternative hypothesis.
2. Construct a test statistic that summarizes the strength of evidence against the null hypothesis.
3. Compute a p-value that quantifies the probability of obtaining that test statistic under the null hypothesis.
4. Decide whether to reject or accept the null hypothesis based on the p-value.

A given hypothesis is divided into two possibilities: the null hypothesis and the alternative hypothesis. The null hypothesis, usually denoted H_0, is the "default" or opposing outcome to our hypothesis. The alternative hypothesis, usually denoted H_a is the unexpected outcome, which is

usually our underlying hypothesis. For H_a to be true, H_0 must be false, and thus we usually attempt to disprove H_0 in order to support H_a. The test statistic is the metric that summarizes how consistent the data is with H_0, where the p-value is the probability of observing a test statistic equal to or more extreme than the observed statistic assuming the null hypothesis H_0 is true (James et al., 2023). The smaller the p-value, the more evidence that is provided against H_0, and we define a threshold to reject H_0 in favor of H_a. Normally, we define this threshold as $p \leq 0.05$, thus meaning under H_0, we would expect to see a p-value this small no more than 5% of the time, and we reject H_0 (James et al., 2023).

Since we are dealing with probabilities and not certainties, the possibility for error exists, called Type I and Type II errors (James et al., 2023):

1. If we reject H_0 when it is in fact true, we commit a Type I error.
2. If we accept H_0 when it is in fact false, we commit a Type II error.

Multiple testing occurs when we do not just want to test one hypothesis, but multiple concurrently. In this case, we would have m null hypotheses, $H_{01}, ..., H_{0m}$ (James et al., 2023). The biggest issue with multiple testing is that when testing a large number of hypotheses at once, small p-values are bound to occur purely by chance, thus increasing the chance for Type I errors (James et al., 2023). Consider a case where we want to test $m = 100$ null hypothesis at a confidence level of 99%, and we know beforehand that all are true. Purely due to chance, one of those one-hundred hypotheses will have a significant p-value, regardless of the fact that the null hypothesis is true. Family-wise error rate (FWER) is a metric that controls for the issue of multiple hypothesis by generalizing the probability of Type I errors to m hypotheses, formally defined as (James et al., 2023):

$$FWER(\alpha) = 1 - \prod_{j-1}^{m}(1-\alpha) = 1 - (1-\alpha)^m \tag{12.3}$$

Not properly controlling for multiple testing is a common issue in finance, and in the following sections we will introduce some best-practices for dealing with this problem.

12.4.4 Evaluating Strategies

Sharpe Ratio (SR) is one of the most common metrics used to evaluate the performance of a given strategy. For a given portfolio, let R_t denote the period t return between t and $t-1$. The mean and variance of that portfolio is defined as (Lo, 2002):

$$\mu = \mathbb{E}(R_t) \tag{12.4}$$

$$\sigma^2 = Var(R_t). \tag{12.5}$$

SR is the ratio of excess expected return to standard deviation, formally defined as (Lo, 2002):

$$SR = \frac{\mu - R_f}{\sigma} \tag{12.6}$$

where R_f denotes risk-free return. Since we almost never know the true mean and variance of an asset, we must estimate SR using historical estimates for μ and σ, giving us the estimated Sharpe Ratio (Lo, 2002):

$$\widehat{SR} = \frac{\hat{\mu} - R_f}{\hat{\sigma}}. \tag{12.7}$$

Expected Sharpe Ratio is subject to estimation error, which can have catastrophic effects if not properly accounted for. In practice, it is common to annualize monthly SR by multiplying it by a factor of $\sqrt{12}$. However, Lo (2002) finds that in most cases, this method is subject to large estimation errors since returns are not IID. In the following sections, multiple methods for producing more accurate Sharpe Ratios are introduced.

12.4.5 Probabilistic Sharpe Ratio

The Probabilistic Sharpe Ratio (PSR) attempts to evaluate the probability that an estimated Sharpe Ratio exceeds a given threshold when returns are non-Normal (Bailey & Lopez de Prado, 2012). Given a pre-defined benchmark Sharpe Ratio SR^*, PSR is formally defined as (Bailey & Lopez de Prado, 2012):

$$\widehat{PSR}(SR^*) = Z\left[\frac{(\widehat{SR} - SR^*)\sqrt{n-1}}{\sqrt{1 - \hat{\gamma_3}\widehat{SR} + \frac{\hat{\gamma_4}-1}{4}\widehat{SR^2}}}\right] \tag{12.8}$$

with returns skewness:

$$\hat{\gamma_3} = \frac{\mathbb{E}[r-\mu]^3}{\sigma^3} \tag{12.9}$$

and kurtosis:

$$\hat{\gamma_4} = \frac{\mathbb{E}[r-\mu]^4}{\sigma^4} \tag{12.10}$$

where n is the track record (period of time $\widehat{SR}$ is observed), r is the excess return, Z is the cumulative denstiy function (CDF) of the standard Normal distribution, and $\widehat{SR}$ is the observed SR. It is important to note that $\widehat{SR}$ and SR^* should follow the same frequency as the returns time series (historical return) (Bailey & Lopez de Prado, 2012). PSR has mutiple advantages (Bailey & Lopez de Prado, 2012):

- PSR takes into account fat tails and non-Normal returns, delivering a corrected measure of performance in terms of probability of skill.
- PSR is robust to irregular trading frequencies, and does not do any annualization as it uses the same frequency as the data.
- PSR gives us a simple metric to validate hypotheses, as PSR is a probability, and we can rule out strategies with $\widehat{PSR} < 0.95$ (or any confidence level).

However, a few caveats exist for PSR, the first of which is that values for skew and kurtosis are an estimate, so they can be subject to error. In these cases, Bailey and Lopez de Prado (2012) recommends setting skew $\hat{\gamma_3}$ to a lower bound and kurtosis $\hat{\gamma_3}$ to an upper bounds per a given confidence level. PSR also assumes that while returns may not be normal, they are still IID since theoretically the skill of the portfolio should not change during the observation period (Bailey & Lopez de Prado, 2012). However, Opdyke (2006) shows that the non-Normality equation also used in PSR still holds under the more general assumption of stationarity. Track record also has an impact on PSR, where longer track records may compensate for uncertainty introduced by non-Normal returns (Bailey & Lopez de Prado, 2012). A longer track record is required for smaller

$\widehat{SR}$, and as such the minimum track length ($MinTRL$) for a given Sharpe Ratio $\widehat{SR}$ is defined as (Bailey & Lopez de Prado, 2012):

$$MinTRL = n^* = 1 + \left[1 - \hat{\gamma}_3\widehat{SR} + \frac{\hat{\gamma}_4 - 1}{4}\widehat{SR}^2\right]\left(\frac{Z_a}{\widehat{SR} - SR^*}\right)^2. \tag{12.11}$$

Track records less than $MinTRL$ indicate that there is not enough observations to conclude with confidence that the observed Sharpe Ratio $\widehat{SR}$ is above the benchmark Sharpe Ratio SR^*. Conceptually, the strategy is punished due to non-Normal returns, but confidence can be regained over a longer time period (Bailey & Lopez de Prado, 2012). An important caveat for $MinTRL$ is that it may recommend a track record less than the required amount of observations for the Central Limit Theorem[2] (CLT) to hold, which must be satisfied for the calculations of skewness $\hat{\gamma}_3$ and kurtosis $\hat{\gamma}_4$ (Bailey & Lopez de Prado, 2012). However, this should not be much of an issue in practice as $MinTRL$ values that would invalidate the CLT are generally too small for a practical backtest (Bailey & Lopez de Prado, 2012).

12.4.6 Deflated Sharpe Ratio

The Deflated Sharpe Ratio (DSR) builds upon the PSR by additionally accounting for selection bias and multiple testing. DSR is formally defined as (Bailey & Lopez de Prado, 2014):

$$\widehat{DSR} \equiv \widehat{PSR}(\widehat{SR_0}) = Z\left[\frac{(\widehat{SR} - \widehat{SR_0})\sqrt{T-1}}{\sqrt{1 - \hat{\gamma}_3\widehat{SR} + \frac{\hat{\gamma}_4 - 1}{4}\widehat{SR^2}}}\right]. \tag{12.12}$$

DSR requires definition of additional parameters, the first of which is $\widehat{SR_0}$ defined as (Bailey & Lopez de Prado, 2014):

$$\widehat{SR_0} = \sqrt{V\left[\{\widehat{SR_n}\}\right]}\left((1-\gamma)Z^{-1}\left[1 - \frac{1}{N}\right] + \gamma Z^{-1}\left[1 - \frac{1}{N}e^{-1}\right]\right) \tag{12.13}$$

where $V\left[\{\widehat{SR_n}\}\right]$ represents the variance across each trial's estimated SR with N number of independent trials (Bailey & Lopez de Prado, 2014). Since DSR is meant for use in the case of multiple trials (strategy configurations), parameters relate to a given configuration, where for a given strategy, $\widehat{SR}$ is the estimated SR for that given strategy, T is the sample length, and parameters $\hat{\gamma}_3$ and $\hat{\gamma}_4$ represent the skew and kurtosis for that strategy's return distribution (Bailey & Lopez de Prado, 2014). When conducting multiple trials, the notion of when to stop testing is important, and as such it is recommended to follow the $\frac{1}{e}$ law[3], in which we sample $\frac{1}{e}$ configurations from the set of theoretically justifiable strategies and continue sampling one by one until an optimal is found against the previous samples (Bailey & Lopez de Prado, 2014). As DSR addresses both multiple testing errors and non-Normality, it should be used over Sharpe Ratio to accept or reject a given hypothesis. In practice, single trials can be evaluated through PSR and backtest length can be validated against $MinTRL$. When conducting multiple trials (i.e. attempting to pick an optimal strategy configuration), DSR should be used and $MinTRL$ can still be found with a given trial's $\widehat{SR}$.

[2]The Central Limit Theorem (CLT) states that as sample size increases, the mean of the sample data will tend closer to the mean of the population. A general rule is that the CLT holds for sample sizes ≥ 30. See Kwak and Kim (2017) for more information.

[3]See Bruss (1984) for more details.

12.4.7 Putting it All Together: How CPCV Controls for Backtest Overfitting

Walk-forward backtests have high variance between SRs simply because a large portion of decisions are based on a small portion of the dataset (Lopez de Prado, 2018b). Since CPCV derives SRs from a large number of paths $j = 1, ..., \varphi$ with mean $E[\{y_{i,j}\}_{j=1,...,\varphi}] = \mu_i$ and variance $\sigma^2[\{y_{i,j}\}_{j=1,...,\varphi}] = \sigma_i^2$ the variance of the sample mean of CPCV paths is defined as (Lopez de Prado, 2018b):

$$\sigma^2[\mu_i] = \varphi^{-2}\left(\varphi\sigma_i^2 + \varphi(\varphi-1)\sigma_i^2\bar{\rho}_i\right) = \varphi^{-1}\sigma_i^2\left(1 + (\varphi-1)\bar{\rho}_i\right) \tag{12.14}$$

where $\gamma = 0.5772156649$ represents the Euler-Mascheroni constant, σ_i^2 is the variance of SRs across all paths for strategy i, and $\hat{\rho}_i$ is the average correlation (not including the diagonal) among $\{y_{i,j}\}_{j=1,...,\varphi}$ (Lopez de Prado, 2018b). Because $\hat{\rho}_i < 1$ implies that the variance of the sample mean is lower than the variance of the sample, CPCV selects SRs closer to the true SR and results in less false discoveries than WF backtesting (Lopez de Prado, 2018b).

Generally, machine learning models are created on a training set and evaluated on an out-of-sample testing set. This is essentially the same process as a walk-forward backtest, and thus the results are likely to be reliant on that single path. To better estimate true results, we can evaluate the performance of the model on the testing set and compare it with CPCV performance. Consider a case where a strategy generates a SR of 1.5 on the test set. We may think our backtest is overfit, so we conduct a CPCV backtest. If the CPCV backtest generates an average SR close to the observed SR of 1.5, then we can be confident in our results. However, if there is a discrepancy between SRs, we are given an indication that the backtest on the test set may have been overfit or subject to the underlying path of the prices.

12.5 Backtesting Recommendations

Lopez de Prado (2018b) and Arnott et al. (2018) both highlight a list of backtesting recommendations and best practices. Below, a brief summary of the most notable recommendations from both works is provided:

1. Establish a valid economic hypothesis before a strategy/model is created.
2. Develop models for a large investment universe.
3. Define the test sample before creating a model.
4. Clean data before use.
5. Do not arbitrarily exclude outliers.
6. Keep track of trials and strategies tried.
7. The more simple a model, the better. Increased dimensionality works against ML algorithms due to data limitations.
8. If out-of-sample results are not satisfactory, do not change the model and test again.
9. Do not justify results after the fact.
10. Start from scratch if a profitable strategy is not found.

Chapter 13

Results: The Quant Workflow in Practice

To test the Quant Workflow and evaluate the improvements it proposes, a Simple Moving Average (SMA) Cross trading strategy was implemented. As a disclaimer, these results are not meant to introduce a profitable strategy or give investment advice. Instead, these results are meant to show how the Quant Workflow can improve even basic investment strategy ideas. The SMA cross strategy was chosen as it is generally the starting point for beginners in algorithmic trading. A simple moving average is quite simply just the mean of the given price series over a specified window. The strategy begins with the definition of two or more SMAs, each with different periods. The strategy then proceeds as follows; if the "fast" (moving average with the lowest period) crosses above the "slow" (moving average with the highest period), then a buy signal is emitted. If the opposite happens, and the fast SMA crosses under the slow SMA, then a sell signal is emitted. In the following example, this will be our model for side. We will then apply meta-labeling to determine bet size, train the model, and finally compare the meta model with the basic SMA cross strategy.

13.1 Primary Model for Side

The SMA cross strategy is a simple rule that attempts to capture trends in a given market. It is generally composed of two moving averages, one with a fast period, and one with a slow period. In this case, we use the 20-period SMA as the fast period moving average and the 50-period SMA as the slow period moving average. The moving averages are calculated on the closing price of the asset as defined in Equation 13.2. A "buy" signal is generated when the fast moving average crosses the slow moving average from below, and a "sell" signal occurs when the fast moving average crosses the slow moving average from above. The signal generation process is defined formally as follows:

$$\text{signal}_t = \begin{cases} 1 & \text{SMA}_{20,t} > \text{SMA}_{50,t} \\ -1 & \text{SMA}_{20,t} < \text{SMA}_{50,t} \end{cases} \tag{13.1}$$

and thus our model for side is generated. We now proceed to the Quant Workflow, where our aim is to generate a meta model to predict bet size.

13.2 Data Collection

The first step in the Quant Workflow is data collection. In this example, trade data for the S&P 500 over the period 2015-01-14 to 2024-05-06 is used. Dollar bars are then created on the price series, resulting in 22,648 observations. It is important to note that in this case, our data is sampled with a fairly high frequency. Data was also only sampled during open market hours as to provide a more realistic representation of live-market performance. However, important considerations like transaction costs and liquidity are not taken into account in this example, which is one of the reasons why this demonstration is not meant to be used as an actual trading strategy.

13.3 Sampling Features

For the purpose of demonstration, we use similar features as Patel et al. (2015) with slight modifications. After calculation, we convert each feature into a categorical feature based on the information it provides about the trend of the underlying asset (Patel et al., 2015). The rationale behind this method is as follows; without machine learning to find patterns in the continuous data, technical analysis features would generally be represented this way, and encoding continuous features into categorical features may remove some obscurity for the model. Additionally, we sampled each feature for four different periods: 5, 10, 25, and 50 bar. The next sections detail the construction of each feature and the transformation used to convert them into categorical features.

13.3.1 Simple Moving Average (SMA)

The Simple Moving Average (SMA) is a weighted average of a price series over a given period. In this case, close price C_t is used, where the SMA at time t is given by:

$$\text{SMA}_{p,t} = \frac{\sum_{i=0}^{p-1} C_{t-i}}{p} \tag{13.2}$$

where p is the period of the SMA. To convert to a categorical feature, we emit a "buy" signal when the closing price is above the SMA, and a "sell" signal when the closing price is below the SMA. This labeling scheme is formally defined as:

$$\text{SMA\_Trend}_{p,t} = \begin{cases} 1 & \text{SMA}_{p,t} < C_t \\ -1 & \text{SMA}_{p,t} > C_t \end{cases} \tag{13.3}$$

13.3.2 Weighted Moving Average (WMA)

The Weighted Moving Average (WMA) is similar to the SMA, but samples closer to the current time are given more weight. The WMA is defined as (Kara et al., 2011):

$$\text{WMA}_{p,t} = \frac{\sum_{i=0}^{p-1} C_{t-i} * (p-i)}{p} \tag{13.4}$$

where p is the period of the WMA. Similar to the SMA, we emit a "buy" signal when the closing price is above the WMA and a "sell" signal when the opposite occurs, formally defined as:

$$\text{WMA\_Trend}_{p,t} = \begin{cases} 1 & \text{WMA}_{p,t} < C_t \\ -1 & \text{WMA}_{p,t} > C_t \end{cases} \tag{13.5}$$

13.3.3 Momentum (MOM)

Momentum (MOM) is quite simply the difference between the current close price and the close price p periods ago, defined as (Kara et al., 2011):

$$\text{MOM}_{p,t} = C_t - C_{t-p} \tag{13.6}$$

In this case, since momentum is a simple difference, a "buy" signal is emitted when the momentum value is greater than zero and a "sell" signal is emitted otherwise. This is formally defined as:

$$\text{MOM\_Trend}_{p,t} = \begin{cases} 1 & \text{MOM}_{p,t} > 0 \\ -1 & \text{MOM}_{p,t} < 0 \end{cases} \tag{13.7}$$

13.3.4 Williams %R

Williams %R (WillR) is a momentum indicator that relates the closing price to the high price over a given period. For closing price C_t, high price H_t, and low price L_t at time t, the p period Williams %R is defined as (Kara et al., 2011):

$$\text{WillR}_{p,t} = \frac{H_{t-p} - C_t}{H_{t-p} - L_{t-p}} * 100 \tag{13.8}$$

To convert the Williams %R to a categorical feature, a "buy" signal is emitted when the current value of the indicator is greater than the previous value, and a "sell" signal is emitted when the current value is less than the previous value. This is formally defined as:

$$\text{WillR\_Trend}_{p,t} = \begin{cases} 1 & \text{WillR}_{p,t} > \text{WillR}_{p,t-1} \\ -1 & \text{WillR}_{p,t} < \text{WillR}_{p,t-1} \end{cases} \tag{13.9}$$

13.3.5 Moving Average Convergence Divergence (MACD)

Moving Average Convergence Divergence (MACD) is a momentum oscillator that is commonly used to trade trends similar to a moving average. MACD is normally calculated with two Exponential Moving Average (EMA) indicators with periods 12 and 26, however in order to apply the indicator to various periods, a different approach was used. The EMA indicator is calculated as follows (Kara et al., 2011):

$$\text{EMA}_{p,t} = C_t * \alpha + \text{EMA}_{p,t-1} * (1 - \alpha) \tag{13.10}$$

where α is a smoothing factor $\alpha = \frac{2}{1+p}$. Instead of fixed period values, we use a ratio of the MACD period for each EMA as follows:

$$\text{MACD}_{p,t} = \text{MACD}_{p-1,t} + \frac{2}{n+1} * \left(\text{EMA}_{p,t} - \text{EMA}_{floor(p/2),t} - \text{MACD}_{p-1,t}\right) \tag{13.11}$$

where $floor()$ is the floor operator that rounds a given value down to the nearest integer. As this is a momentum indicator, we emit a "buy" signal when the current MACD value is greater than the previous MACD value, and a "sell" signal when the previous MACD value is greater than the current MACD value. This is formally defined as:

$$\text{MACD\_Trend}_{p,t} = \begin{cases} 1 & \text{MACD}_{p,t} > \text{MACD}_{p,t-1} \\ -1 & \text{MACD}_{p,t} < \text{MACD}_{p,t-1} \end{cases} \tag{13.12}$$

13.3.6 Stochastic Oscillator

The Stochastic Oscillator is a momentum indicator that is comprised of two signal lines, the %K and %D lines. For closing price C_t, highest price over p periods $HH_{p,t}$, and lowest price $LL_{p,t}$ over p periods, the %K line (StochK) is defined as (Kara et al., 2011):

$$\text{StochK}_{p,t} = \frac{C_t - LL_{p,t}}{HH_{p,t} - LL_{p,t}} * 100 \tag{13.13}$$

The %D line (StochD) is just the simple moving average of the %K line, defined as (Kara et al., 2011):

$$\text{StochD}_{p,t} = \frac{\sum_{i=0}^{p-1} \text{StochK}_{t-i}}{p} \tag{13.14}$$

Since the Stochastic Oscillator is a momentum indicator, "buy" singals are emitted when each respective line is greater than its previous value, and a "sell" signal is emitted when each respective line is less than its previous value. This results in two feature series, defined as follows:

$$\text{StochK\_Trend}_{p,t} = \begin{cases} 1 & \text{StochK}_{p,t} > \text{StochK}_{p,t-1} \\ -1 & \text{StochK}_{p,t} < \text{StochK}_{p,t-1} \end{cases} \tag{13.15}$$

$$\text{StochD\_Trend}_{p,t} = \begin{cases} 1 & \text{StochD}_{p,t} > \text{StochD}_{p,t-1} \\ -1 & \text{StochD}_{p,t} < \text{StochD}_{p,t-1} \end{cases} \tag{13.16}$$

13.3.7 Relative Strength Index (RSI)

The Relative Strength Index (RSI) is a popular momentum indicator commonly used to identify oversold and overbought conditions. RSI is the ratio of upward movements to downward movements, defined as (Wilder, 1978):

$$\text{RSI}_{p,t} = 100 - \frac{100}{1 + RS_{p,t}} \tag{13.17}$$

where $\text{RS}_{p,t}$ is the relative strength defined as:

$$\text{RS}_{p,t} = \frac{\text{Average Gain}_{p_t}}{\text{Average Loss}_{p_t}} \tag{13.18}$$

with Average Gain over p periods:

$$\text{Average Gain}_{p,t} = \frac{\sum_{i=0}^{p-1} U_{t,i}}{p} \tag{13.19}$$

$$U_{t,i} = \begin{cases} C_{t-i} - C_{t-i-1} & C_{t-i} > C_{t-i-1} \\ 0 & \text{otherwise} \end{cases} \tag{13.20}$$

and Average Loss over p periods:

$$\text{Average Loss}_{p,t} = \frac{\sum_{i=0}^{p-1} D_{t,i}}{p} \tag{13.21}$$

$$D_{t,i} = \begin{cases} C_{t-i-1} - C_{t-i} & C_{t-i} < C_{t-i-1} \\ 0 & \text{otherwise} \end{cases} \tag{13.22}$$

The RSI oscillates between the range 0 and 100, and an asset is said to be "overbought" when the RSI is greater than 70 and "oversold" when the RSI is less than 30. Taking this into account, we emit "buy" signals when the RSI is below the oversold level and additionally when the RSI is greater than its previous value if between the oversold and overbought levels. Conversely, a "sell" signal is emitted when the RSI is above the overbought level and additionally when the RSI is less than its previous value when between the oversold and overbought levels. This is formally defined as:

$$\text{RSI\_Trend}_{p,t} = \begin{cases} 1 & \text{RSI}_{p,t} < 30 \\ 1 & \text{RSI}_{p,t} > \text{RSI}_{p,t-1} \wedge 30 < \text{RSI}_{p,t} < 70 \\ -1 & \text{RSI}_{p,t} > 70 \\ -1 & \text{RSI}_{p,t} < \text{RSI}_{p,t-1} \wedge 30 < \text{RSI}_{p,t} < 70 \end{cases} \tag{13.23}$$

13.3.8 Average Price Difference Oscillator (AD)

This indicator is an adaptation of the Accumulation Distribution Oscillator, which simply compares the difference between the high price and the close price with the difference between the high price and low price (Kara et al., 2011). In this case, the formula was modified to compare the high with the close price $t - p$ periods ago, defined as:

$$\text{AD}_{p,t} = \frac{H_t - C_{t-p}}{H_t - L_t} \tag{13.24}$$

A "buy" signal is generated when the current AD value is greater than the previous AD value, and a "sell" signal is generated when the current AD value is less than the previous AD value, formally:

$$\text{AD\_Trend}_{p,t} = \begin{cases} 1 & \text{AD}_{p,t} > \text{AD}_{p,t-1} \\ -1 & \text{AD}_{p,t} < \text{AD}_{p,t-1} \end{cases} \tag{13.25}$$

13.4 Algorithm Selection

To determine which ML algorithm should be used for the meta model, six models were tested through cross validation on the training set: Ada Boost (ADA), Random Forest (RF), Support Vector Classifier (SVC), Extreme Gradient Boost (XGB), Bagged Random Forest (bgRF), and Bagged Decision Tree (bgDT). Table 13.1 depicts the initial hyperparameters used for each model. Figure 13.1 depicts a box plot of accuracy scores for each of the models. It is clear that accuracy scores are extremely low, and in some cases not much better than random guessing. This is an initial sign that our features may not contain much predictive power. Figure 13.2 depicts the negative log-loss measures for the same algorithms. Negative log-loss depicts how accurate the model's confidence in a prediction is, which is important in this application because these probabilities directly relate to the bet sizes of our model. For this example, we choose to use the Random Forest model, however different features and parameters can greatly affect these initial accuracy scores.

Table 13.1: Hyperparameters for Various Models.

Model	**Hyperparameter**	**Description**
Random Forest	`n_estimators`	400
	`max_depth`	None
	`min_samples_split`	2
	`min_samples_leaf`	1
	`max_features`	1
	`min_weight_fraction_leaf`	0.05
	`class_weight`	'balanced_subsample'
	`criterion`	'gini'
Ada Boost Classifier	`max_depth`	DescisionTreeClassifier(max_depth=1)
	`n_estimators`	400
	`learning_rate`	1
Extreme Gradient Boost Classifier	`learning_rate`	0.02
	`max_depth`	3
	`min_child_weight`	1
Bagging Classifier (Random Forest)	`base_estimator`	RandomForest(n_estimators=1, criterion='entropy', class_weight='balanced_subsample', bootstrap='False')
	`n_estimators`	400
	`max_features`	1
Bagging Classifier (Descison Tree)	`max_depth`	DescisionTreeClassifier(criterion='entropy', max_features='auto', class_weight='balanced')
	`n_estimators`	400
	`max_features`	1
Support Vector Classifier (SVC)	`C`	1.0
	`kernel`	'rbf'
	`class_weight`	'balanced'

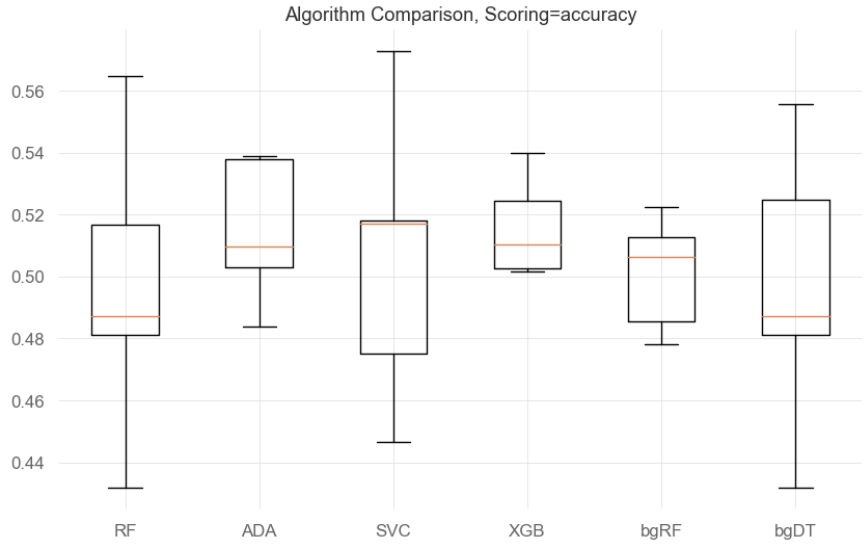

Figure 13.1: Box plot of accuracy scores generated through CV on the training set for six different models.

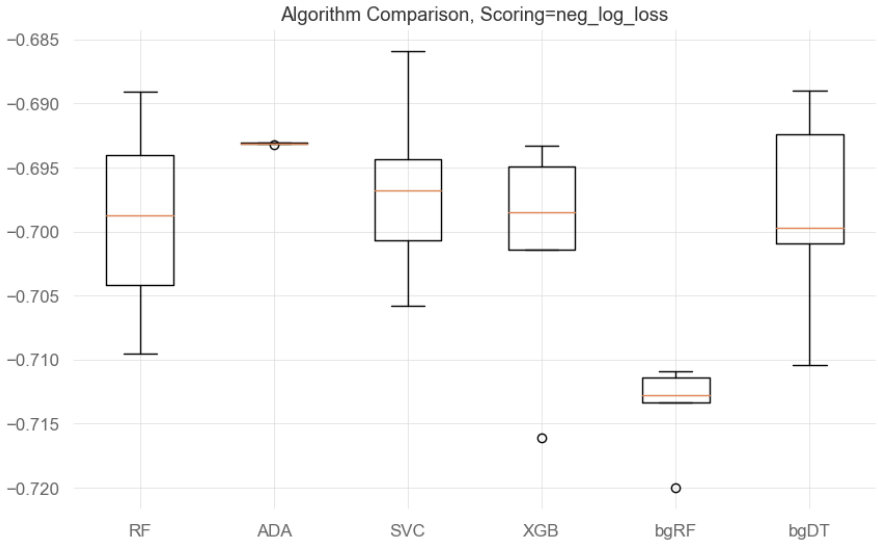

Figure 13.2: Box plot of negative log-loss scores generated through CV on the training set for six different models

13.5 Feature Importance

After a model was selected, feature importance scores were calculated as per Chapter 11. Each feature importance method was calculated on the training set, as using the testing set would be a case of look-ahead bias. Figure 13.3 depicts the mean decrease impurity (MDI) scores for each feature. From the figure, it is evident that quite a few features have impurity scores below the mean. Figure 13.4 depicts the mean decrease accuracy (MDA) feature importance scores for the model, and we see that for some features like the 50-period MACD, both MDI and MDA are in agreement that the feature may have a detrimental impact on the model. The last feature importance method, single feature importance (SFI), is shown in Figure 13.5. The SFI scores further support our original hypothesis that our features may have little predictive power. In almost every case, each feature was about as accurate as random chance, with some features actually decreasing the model accuracy from the 50% level. The low SFI scores do not mean that all features have no predictive power, however, as it fails to encompass combination effects between the features. Still, from these results, we can see that our features have low signal-to-noise ratios, which is to be expected for such simple features. If the goal is to create a robust strategy, the next step from here would be to revisit the feature creation step and possibly remove or adjust the current features to add more predictive power. However, for the purpose of demonstration, we continue with the workflow.

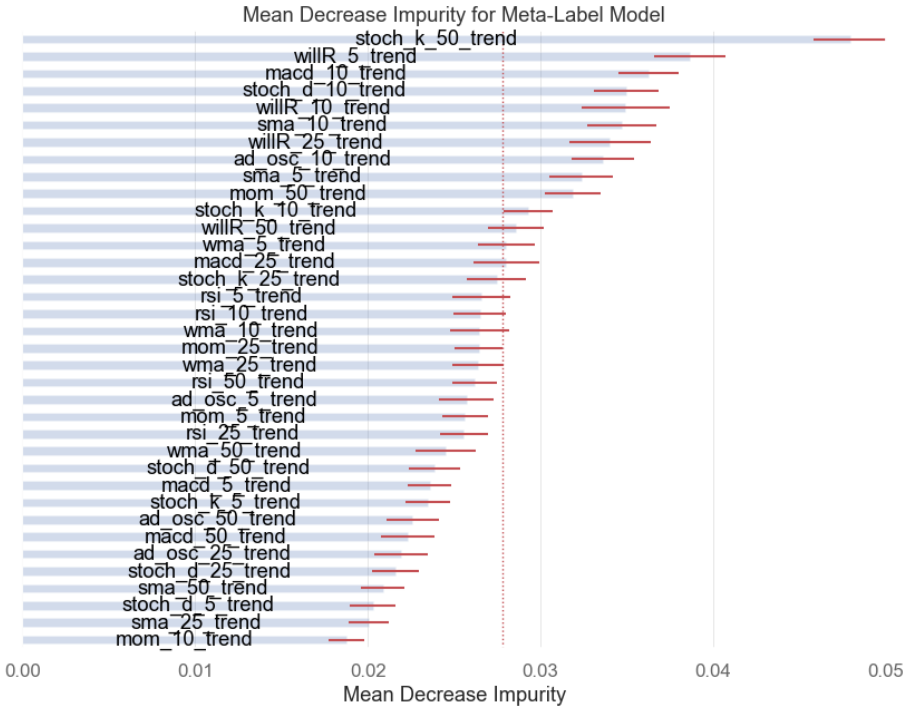

Figure 13.3: Mean decrease impurity (MDI) feature importance scores for the Random Forest model on the training set.

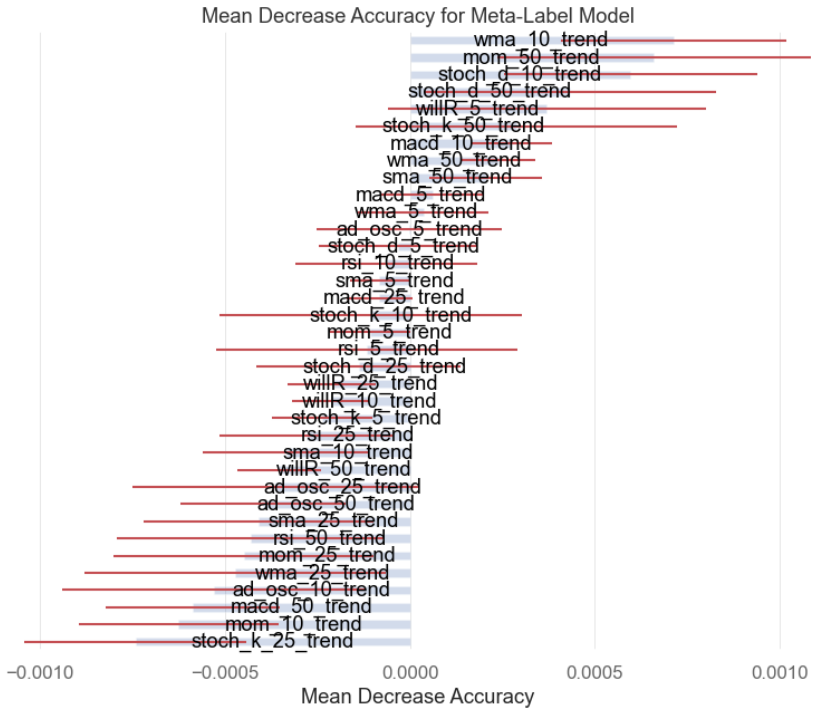

Figure 13.4: Mean decrease accuracy (MDA) feature importance scores for the Random Forest model on the training set.

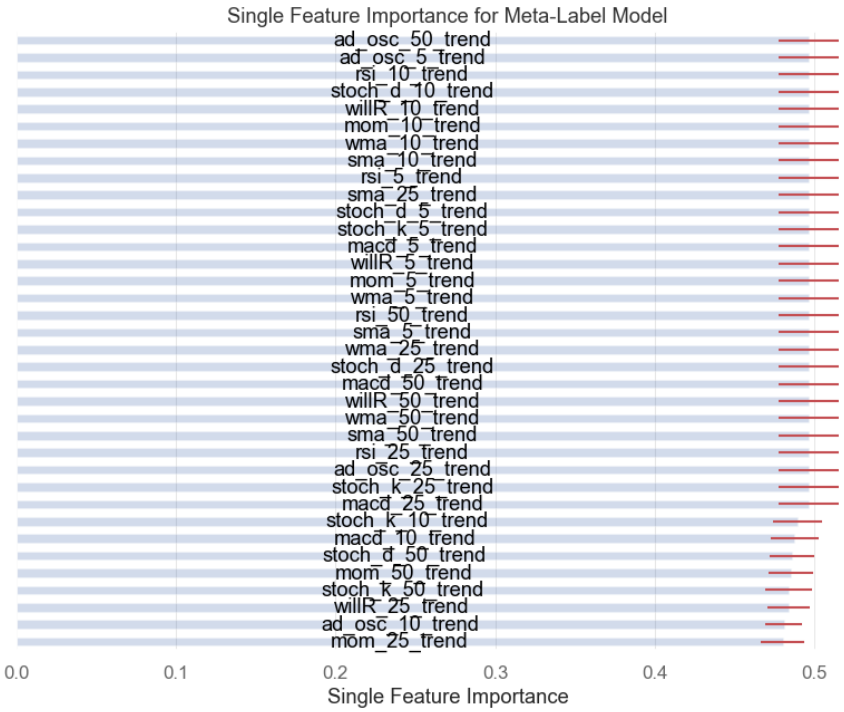

Figure 13.5: Single feature importance (SFI) feature importance scores for the Random Forest model on the training set.

13.6 Training Set Statistics

Table 13.2 depicts the training set statistics of the RF model after it was fitted on the training set. Table 13.3 shows the confusion matrix for the same model on the training set. Further supporting our previous observations, the training set accuracy is low, meaning that our model is likely lacking in predictive power. However, a positive observation is that the model does not seem to be overfit on the training set. The Receiver Operating Characteristic (ROC) curve is pictured in Figure 13.6. This plot further confirms the lack of model accuracy.

Table 13.2: Classification Report RF Model on the Training Set.

Class	Precision	Recall	F1-Score	Support
0	0.55	0.58	0.56	802
1	0.63	0.59	0.61	950

Accuracy: 0.587

Table 13.3: Confusion Matrix of RF Model on the Training Set.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	466	336
	Negative	387	563

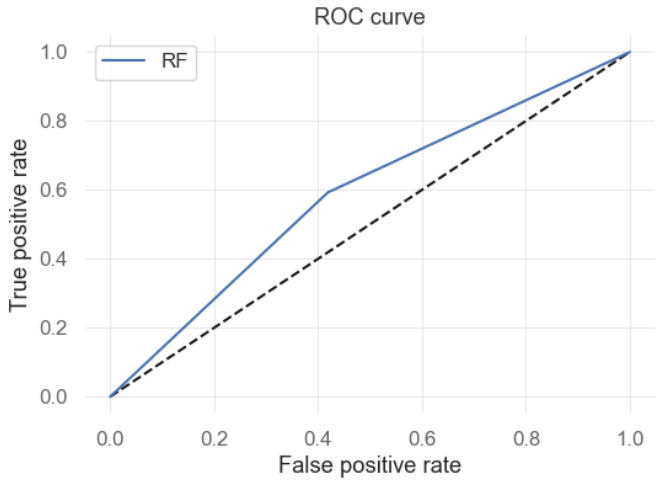

Figure 13.6: ROC curve of the RF model on the training set.

13.7 Hyperparameter Tuning

A simple Grid Search CV model was used on the training set to optimize the hyperparameters of the model. The same CV method as mentioned in Chapter 10 was implemented in order to control for label overlap. Table 13.4 highlights the new parameters chosen for the RF model. It is interesting to note that both the number of estimators and the max_features parameters were decreased. This likely means that many of our features may be redundant, as the model is only benefiting from a select few.

Table 13.4: New Random Forest Hyperparameters after Grid Search CV.

Hyperparameter	**Description**
`n_estimators`	20
`max_features`	0.9

13.8 Combinatorial Purged Cross Validation Backtesting

After hyperparameter tuning, CPCV backtesting is run on the entire dataset with $n = 6$ groups and $k = 2$ test set size. This results in 15 splits and 5 backtest paths, where in each split the model is retrained on the training set and tested on the testing set. Table 13.5 depicts the statistics for each of the five backtest paths.

Table 13.5: Combinatorial Purged k-Fold Backtest Statistics, $n = 6$ and $k = 2$.

	Path 1	**Path 2**	**Path 3**	**Path 4**	**Path 5**	**Average**
Sharpe Ratio	1.07	0.90	0.93	1.18	1.01	**1.018**
Annual Return (%)	13.8	12.5	13.1	16.7	14.3	**14.08**
Cumulative Returns (%)	248.3	211.6	226.5	343.7	260.8	**258.18**
Annual Volatility (%)	12.8	14.2	14.4	13.9	14.2	**13.9**
Maximum Drawdown (%)	-19.6	-24.8	-33.1	-24.9	-30.4	**-26.56**
Calmar Ratio	0.70	0.50	0.40	0.67	0.47	**0.548**
Sortino Ratio	1.78	1.52	1.44	1.88	1.51	**1.626**
Skew	2.61	4.39	2.17	1.03	-0.20	**2.0**
Kurtosis	68.37	115.15	91.79	45.60	72.47	**78.676**
Tail Ratio	1.17	1.12	1.33	1.29	1.38	**1.258**
Daily Value at Risk (%)	-1.6	-1.7	-1.8	-1.7	-1.7	**-1.68**

13.9 Final Performance on the Test Set

The final step in the workflow is to compare backtest results between CPCV and walk-forward. Table 13.6 depicts the classification statistics for the RF model on the test set, and Table 13.7 shows the resulting confusion matrix. The ROC curve is also pictured in Figure 13.7, and it is clear that the predictive power of the model is not much better than random chance.

Table 13.6: Classification Report for the RF Model on the Testing Set.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.45	0.42	0.43	432
1	0.57	0.61	0.59	558
Accuracy: 0.524				

Table 13.7: Confusion Matrix of RF Model on the Testing Set.

		Predicted	
		Positive	**Negative**
Actual	**Positive**	181	251
	Negative	220	338

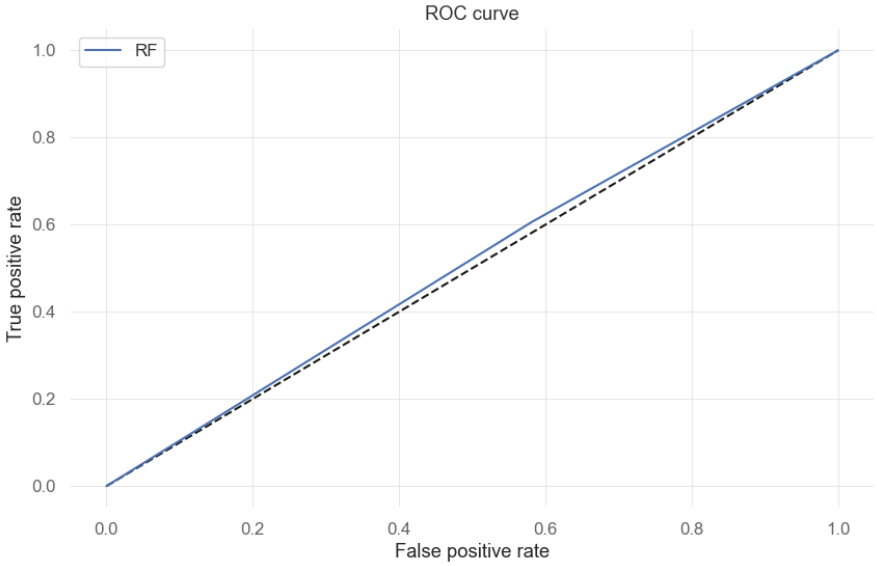

Figure 13.7: ROC curve for the RF model on the test set.

For a final comparison, we evaluated the performance of the model at three stages of the Quant Workflow, including the base side model, the triple-barrier labels without the meta model (TB), and finally the triple-barrier labels with the meta model (TB-Meta). Table 13.8 depicts the backtest statistics for each model. The cumulative return plots for the base side model, TB labels, and TB with meta labels are pictured in Figures 13.8, 13.9, 13.10 respectively. It is clear that although the cumulative return went down for the meta model, the Sharpe Ratio increased, which indicates better risk/return characteristics.

Table 13.8: Final WF Backtest Statistics.

	Base Side Model	TB	TB-Meta
Sharpe Ratio	0.09	1.38	**1.48**
Probabilistic Sharpe Ratio (%)	10.8	99.65	99.79
CAGR (%)	0.74	27.88	13.05
Cumulative Returns (%)	3.16	183.31	68.13
Annual Volatility (%)	1.32	20.84	9.22
Maximum Drawdown (%)	-5.11	26.35	-16.41
Calmar Ratio	0.14	1.06	0.8
Sortino Ratio	0.13	2.0	2.12
Skew	-0.03	-0.24	-0.42
Kurtosis	15.65	0.63	4.29
Tail Ratio	0.98	0.97	0.99
Daily Value at Risk (%)	-0.14	-2.05	-0.9

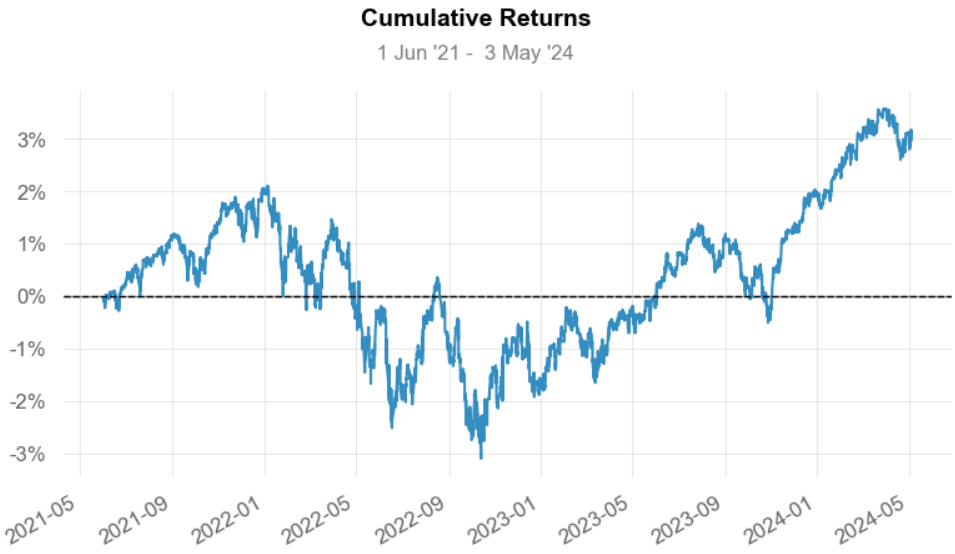

Figure 13.8: Cumulative return plot of the base side model on the test set.

Figure 13.9: Cumulative return plot of the triple-barrier labels (TB) on the test set.

Figure 13.10: Cumulative return plot of the triple-barrier labels with the meta model (TB-Meta) on the test set.

13.10 Findings and Significance

From the aforementioned results, it is clear that the Quant Workflow can improve even simple side models. However, the most significant improvements are manifested throughout the strategy creation process, as different "checks" help to prevent overfitting and better estimate true performance of the model. The first "check" that was seen in our demonstration was the model selection step. The low overall accuracy of each model shows that our features may not have enough predictive power. The feature importance plots in Figures 13.3 to 13.5 further support this observation, as the MDA importance plot in Figure 13.4 shows that accuracy decrease/increase is marginal for each feature. The SFI plot in Figure 13.5 also shows that when trained in isolation, each feature is rarely more predictive than random chance. From these conclusions, we could then go back to the feature creation step and add more predictive features. The reason why the ability to make these conclusions is so important is because they happened in the training set – we are not multiple testing on the test set.

After we have finished conducting feature research, we can then get insight into how overfit the model is through the classification report on the training set. We expect the accuracy of the training set predictions to be inflated since the model was trained on that dataset, but the accuracy score in Table 13.2 is within reasonable bounds for the previous results, and thus the meta model is likely not overfit to the training set. After tuning the hyperparameters of the model, we begin the final backtesting of the model. At this step, we are now evaluating the model from an objective perspective; if the SR of the model is not significant, then it is discarded entirely. If conducting multiple backtests with different parameters, then trials must be accounted for and methods described in Chapter 12 must be used. After CPCV backtesting with five paths, an average Sharpe Ratio of 1.018 was generated. The WF backtest produced a SR of 1.48, and since we know that the CPCV average SR was 1.018, we can conclude that the testing set performance is likely inflated due to the path of the prices during that time. It is also evident that the meta model did not have much improvement over just the triple-barrier labels, which makes sense due to the low model accuracy. However, the triple-barrier method alone greatly improved the performance of the underlying side model over the testing period, which is a clear indicator that the Quant Workflow has the potential to improve a given model for side. With more predictive features, it is clear that the meta model could further increase strategy performance while subsequently quantifying possible model overfitting throughout the strategy creation process.

13.11 Hindsight Considerations and Best Practices

From the results in the previous sections, we made multiple observations that the given features for the model may be lacking in predictive power. To further quantify these observations, we can employ some simple data exploration methods. Especially when dealing with technical analysis indicators of multiple periods, it is important to understand what signals are being generated by each indicator. One way that this is possible is through a bar plot of each feature as shown in Figure 13.11. Figure 13.11 should be used in conjunction with statistics from the training set or SFI plots, which may be able to provide insight into how each feature is behaving when we have many features in a model. Although Figure 13.11 says nothing about the accuracy of the features, it provides extra context as to why the model accuracy could be so low; most features emit approximately the same amount of signals for each class, meaning that many of the features may be redundant and not providing much information to the model. As covered in Chapter 11

on Feature Importance, both MDA and MDI are not robust to collinearity. As such, it may be valuable to first view a correlation matrix between each of the features before looking at feature importance scores. Figure 13.12 depicts the correlation matrix for the training set, and it is clear that some features, namely SMA and WMA, exhibit high correlation with other features. This observation could be useful when we are deciding which features to remove, as some features may have decreased importance due to collinearity in MDA and MDI. While putting the Quant Workflow into practice, multiple observations were made that could help improve model accuracy in the future. The first of which is that in the case of continuous features, PCA may boost accuracy scores. However, application of PCA to our model presents a challenge in discerning feature importance scores, as the transformation of the feature matrix complicates the translation of scores to their original features. One way to remedy this is to calculate feature importance scores on the PCA components, then relate the feature importance of each PCA component to the weights of each feature in that component. For example, consider a case where PCA component 1 is found to be significant through MDA feature importance. We can then compare the weight of each feature in that component and make an inference that features with large weights may also be important. On the topic of model creation, an important implementation detail is to make sure that each part of the model is re-trained in each cross validation split. This is especially important when applying data transformations like PCA. If the transformation is not reapplied in each train-test split, then data leakage will occur in each split. When using the Scikit-Learn[1] package in Python, instead of specifying the model and conducting transformations separately, a Pipeline[2] object should be used, which allows for the transformation and model fitting process to happen sequentially in each split whenever the model is fit. This way, data will not overlap between splits and CV will more accurately estimate general model performance.

[1] See https://scikit-learn.org/stable/ for more details.

[2] See https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html for more details.

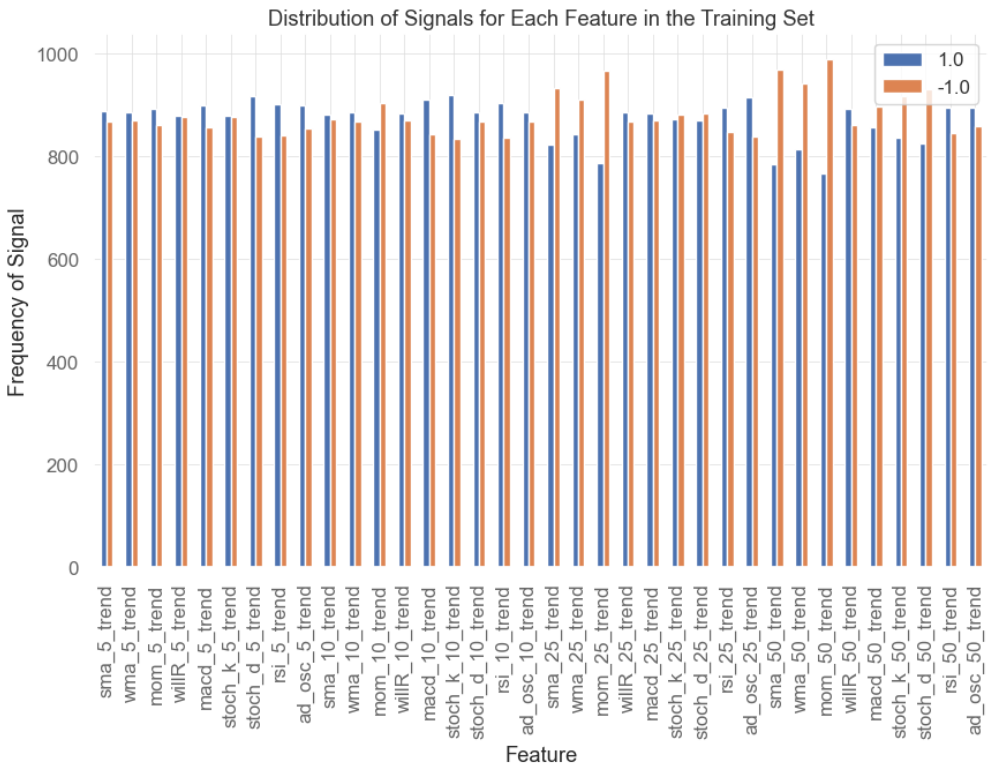

Figure 13.11: Distribution of training set signals for each feature.

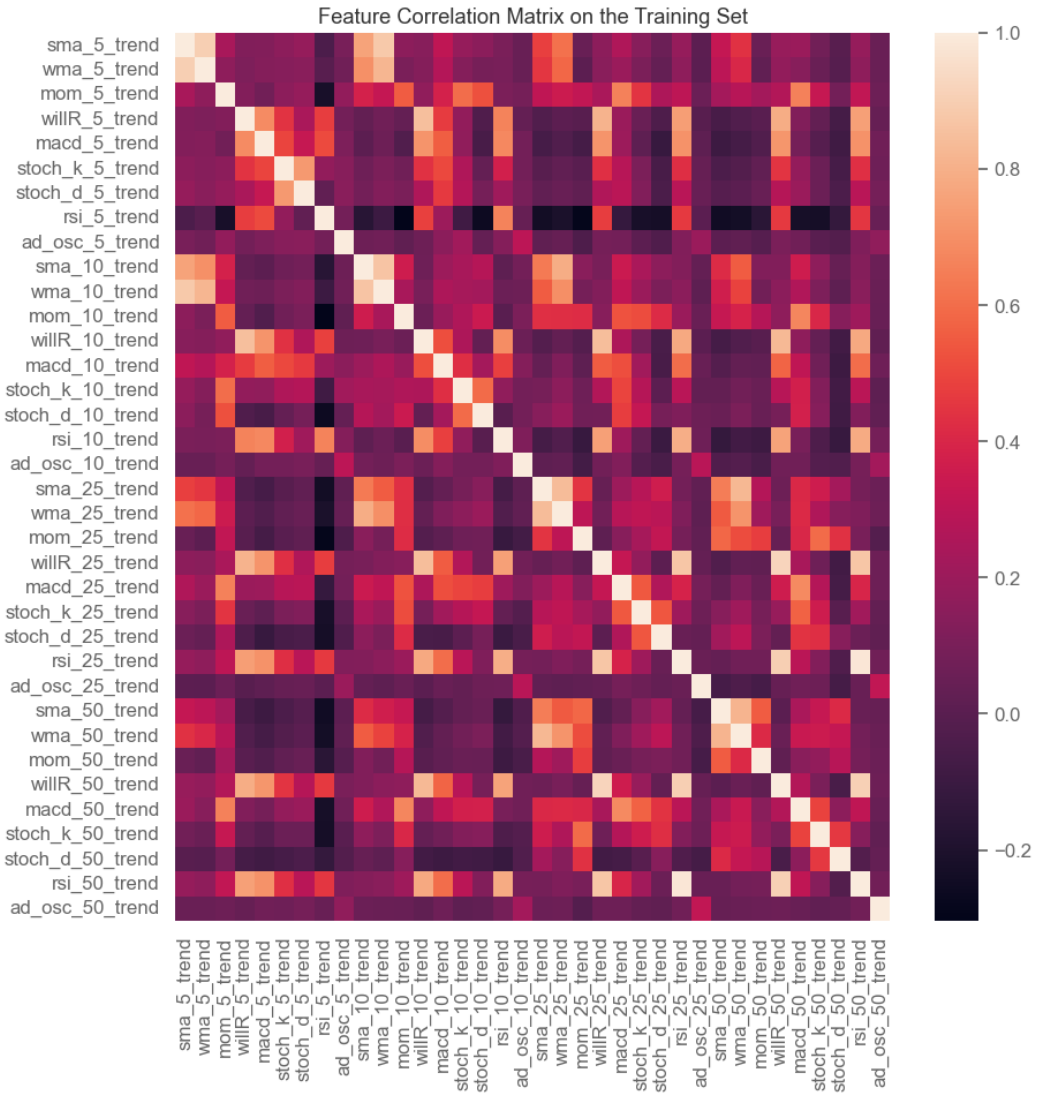

Figure 13.12: Correlation matrix of features in the training set.

Chapter 14

Conclusion

In the proposed workflow, we have synthesized cutting-edge literature in financial machine learning and backtest practice to demystify the strategy creation process. Previous introductory works in the field either fail to describe a complete workflow, fail to account for necessary issues like non-Normality of returns, or succumb to multiple-testing and other biases. Not only have we covered every part of the process, but an implementation showcasing the power of the proposed workflow was presented. From our results, it is clear that the proposed workflow has the potential to improve the performance of a given rule or model. Along with performance improvements, the Quant Workflow provides sound methodologies for controlling overfitting throughout multiple steps in the strategy creation process, instilling various "checks" that quantify the predictive power of the model and its subsequent features. Through these "checks," the Quant Workflow advocates for a research process not done through backtesting on the testing set, but instead through feature importance methods on the training set. This important distinction helps to avoid many of the common biases that cause published trading strategies to be false. The proposed workflow also provides rigorous methodologies for determining the significance of results, adjusting for the effects of multiple testing and potentially filtering false positives. This work hopes to address the lack of comprehensive backtesting workflows available to individual quants and act as a gateway into the proper scientific practice of finance.

The future of the Quant Workflow is extremely promising. Multiple methods not currently present in the workflow could further improve performance and flexibility, the most notable of which is simulated data and synthetic backtesting. In finance, the amount of available data is a clear limitation, and the generation of synthetic data allows for a strategy to be tested on a large number of unseen testing sets (Lopez de Prado, 2018b). Along with portfolio optimization frameworks and high performance computing (HPC) methods, large statistical tests and parameter optimizations could be efficiently conducted, where metrics such as DSR[1] can be used to effectively discern expected SRs. The modeling process could also be expanded, as various market behaviors and regimes[2] could be delegated to their own specific models. An obvious next step for the Quant Workflow is to implement a multi-asset trading framework that allows for a single strategy to be deployed across a wide variety of assets. Notably, a multi-asset framework would allow for assets

[1] DSR is a metric that controls for the likelihood of false SRs under multiple testing. See Chapter 12 for more information.

[2] A regime is period of consistent market behavior that influences subsequent market variables. See Hamilton (2010) for more information.

with varied risk characteristics to be included in the same strategy portfolio, introducing a level of diversification. Developing strategies for multiple assets instead of a single asset may also decrease the likelihood of a false positive (Lopez de Prado, 2018b). Aside from these future improvements, the current state of the Quant Workflow is more than sufficient for effective strategy creation and allows for a truly scientific approach to finance.

References

4.2. Permutation feature importance. (n.d.). Retrieved May 7, 2024, from https://scikit-learn/stable/modules/permutation%5C_importance.html

Alexander, C. (2001, December). *Market Models: A Guide to Financial Data Analysis*. Wiley.

Ali, A., Shamsuddin, S. M., & Ralescu, A. (2015). Classification with class imbalance problem: A review. *7*, 176–204.

Ané, T., & Geman, H. (2000). Order Flow, Transaction Clock, and Normality of Asset Returns. *The Journal of Finance*, *55*(5), 2259–2284.

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*(none). https://doi.org/10.1214/09-SS054

Arnott, R. D., Harvey, C. R., & Markowitz, H. (2018, November). A Backtesting Protocol in the Era of Machine Learning. https://doi.org/10.2139/ssrn.3275654

Ashley, R. A., & Patterson, D. M. (1986). A Nonparametric, Distribution-Free Test for Serial Independence in Stock Returns. *The Journal of Financial and Quantitative Analysis*, *21*(2), 221–227. https://doi.org/10.2307/2330739

Atsalakis, G. S., & Valavanis, K. P. (2009). Surveying stock market forecasting techniques – Part II: Soft computing methods. *Expert Systems with Applications*, *36*(3, Part 2), 5932–5941. https://doi.org/10.1016/j.eswa.2008.07.006

Bailey, D. H., Borwein, J., Lopez de Prado, M., & Zhu, Q. J. (2014, April). Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance. https://doi.org/10.2139/ssrn.2308659

Bailey, D. H., Borwein, J., Lopez de Prado, M., & Zhu, Q. J. (2015, February). The Probability of Backtest Overfitting. https://doi.org/10.2139/ssrn.2326253

Bailey, D. H., & Lopez de Prado, M. (2012, April). The Sharpe Ratio Efficient Frontier. https://doi.org/10.2139/ssrn.1821643

Bailey, D. H., & Lopez de Prado, M. (2014, July). The Deflated Sharpe Ratio: Correcting for Selection Bias, Backtest Overfitting and Non-Normality. https://doi.org/10.2139/ssrn.2460551

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, *42*(20), 7046–7056. https://doi.org/10.1016/j.eswa.2015.05.013

Bender, R., & Lange, S. (2001). Adjusting for multiple testing—when and how? *Journal of Clinical Epidemiology*, *54*(4), 343–349. https://doi.org/10.1016/S0895-4356(00)00314-0

Bessembinder, H. (2003). Trade Execution Costs and Market Quality after Decimalization. *The Journal of Financial and Quantitative Analysis*, *38*(4), 747–777. https://doi.org/10.2307/4126742

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC. https://doi.org/10.1201/9781315139470

Bruss, F. T. (1984). A Unified Approach to a Class of Best Choice Problems with an Unknown Number of Options. *The Annals of Probability*, *12*(3), 882–889. https://doi.org/10.1214/aop/1176993237

Carver, R. (2015, September). *Systematic Trading: A unique new method for designing trading and investing systems*. Harriman House Limited.

Chan, E. P. (2009). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*. John Wiley & Sons.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

Christie, S. (2005, May). Is the Sharpe Ratio Useful in Asset Allocation? https://doi.org/10.2139/ssrn.720801

Denil, M., & Trappenberg, T. (2011, September). A Characterization of the Combined Effects of Overlap and Imbalance on the SVM Classifier. https://doi.org/10.48550/arXiv.1109.3532

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, *74*(366), 427–431. https://doi.org/10.2307/2286348

Easley, D., Lopez de Prado, M., & O'Hara, M. (2010, November). The Microstructure of the 'Flash Crash': Flow Toxicity, Liquidity Crashes and the Probability of Informed Trading. https://doi.org/10.2139/ssrn.1695041

Easley, D., Lopez de Prado, M., & O'Hara, M. (2012, February). Flow Toxicity and Liquidity in a High Frequency World. https://doi.org/10.2139/ssrn.1695596

Easley, D., & O'Hara, M. (1992). Time and the Process of Security Price Adjustment. *The Journal of Finance*, *47*(2), 577–605. https://doi.org/10.2307/2329116

Edwards, R. D., Magee, J., & Bassetti, W. H. C. (2018, July). *Technical Analysis of Stock Trends*. CRC Press.

Erdem, O. (2020). Freedom and stock market performance during Covid-19 outbreak. *Finance Research Letters*, *36*, 101671. https://doi.org/10.1016/j.frl.2020.101671

Faber, M. (2010, April). Relative Strength Strategies for Investing. https://doi.org/10.2139/ssrn.1585517

Faber, M. (2013, February). A Quantitative Approach to Tactical Asset Allocation.

Fama, E. F. (1963). Mandelbrot and the Stable Paretian Hypothesis. *The Journal of Business*, *36*(4), 420–429.

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383–417. https://doi.org/10.2307/2325486

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, *116*(1), 1–22. https://doi.org/10.1016/j.jfineco.2014.10.010

Fathmaningrum, E. S., & Utami, T. P. (2022). Determinants of Investment Decisions in the Capital Market During the COVID-19 Pandemic. *Journal of Accounting and Investment*, *23*(1), 147–169. https://doi.org/10.18196/jai.v23i1.13408

García-Magariños, M., López-de-Ullibarri, I., Cao, R., & Salas, A. (2009). Evaluating the ability of tree-based methods and logistic regression for the detection of SNP-SNP interaction. *Annals of Human Genetics*, *73*(Pt 3), 360–369. https://doi.org/10.1111/j.1469-1809.2009.00511.x

Gerig, A. (2015, January). High-Frequency Trading Synchronizes Prices in Financial Markets. https://doi.org/10.2139/ssrn.2173247

Gray, H. L., & fan Zhang, N. (1988). On a New Definition of the Fractional Difference. *Mathematics of Computation*, *50*(182), 513–529. https://doi.org/10.2307/2008620

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, *3*(null), 1157–1182.

Hamilton, J. D. (2010). Regime switching models. In S. N. Durlauf & L. E. Blume (Eds.), *Macroeconometrics and Time Series Analysis* (pp. 202–209). Palgrave Macmillan UK. https://doi.org/10.1057/9780230280830_23

Harvey, C. R., Liu, Y., & Zhu, C. (2015, February). . . . and the Cross-Section of Expected Returns. https://doi.org/10.2139/ssrn.2249314

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*, *124*, 226–251. https://doi.org/10.1016/j.eswa.2019.01.012

Howley, T., Madden, M. G., O'Connell, M.-L., & Ryder, A. G. (2006). The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data. *Knowledge-Based Systems*, *19*(5), 363–370. https://doi.org/10.1016/j.knosys.2005.11.014

Huang, W., Nakamori, Y., & Wang, S.-Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, *32*(10), 2513–2522. https://doi.org/10.1016/j.cor.2004.03.016

Isichenko, M. (2021, August). *Quantitative Portfolio Management: The Art and Science of Statistical Arbitrage*. John Wiley & Sons.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023, June). *An Introduction to Statistical Learning: With Applications in Python*. Springer Nature.

Janecek, A., Gansterer, W., Demel, M., & Ecker, G. (2008). On the Relationship Between Feature Selection and Classification Accuracy. *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, 90–105.

Jarque, C. M., & Bera, A. K. (1987). A Test for Normality of Observations and Regression Residuals. *International Statistical Review / Revue Internationale de Statistique*, *55*(2), 163–172. https://doi.org/10.2307/1403192

Javaheri, S. H., Sepehri, M. M., & Teimourpour, B. (2014, January). Chapter 6 - Response Modeling in Direct Marketing: A Data Mining-Based Approach for Target Selection. In Y. Zhao & Y. Cen (Eds.), *Data Mining Applications with R* (pp. 153–180). Academic Press. https://doi.org/10.1016/B978-0-12-411511-8.00006-2

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Kara, Y., Acar Boyacioglu, M., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert Systems with Applications*, *38*(5), 5311–5319. https://doi.org/10.1016/j.eswa.2010.10.027

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*(1), 307–319. https://doi.org/10.1016/S0925-2312(03)00372-2

King, G., & Zeng, L. (2001). Logistic Regression in Rare Events Data. *Political Analysis*, *9*, 137–163.

Kirilenko, A., Kyle, A. S., Samadi, M., & Tuzun, T. (2017). The Flash Crash: High-Frequency Trading in an Electronic Market. *The Journal of Finance*, *72*(3), 967–998.

Kissell, R. (2013, October). *The Science of Algorithmic Trading and Portfolio Management*. Academic Press.

Kuhn, M., & Johnson, K. (2019, August). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman and Hall/CRC. https://doi.org/10.1201/9781315108230

Kwak, S. G., & Kim, J. H. (2017). Central limit theorem: The cornerstone of modern statistics. *Korean Journal of Anesthesiology*, *70*(2), 144–156. https://doi.org/10.4097/kjae.2017.70.2.144

Lam, K., & Yam, H. C. (1997). Cusum Techniques for Technical Trading in Financial Markets. *Financial Engineering and the Japanese Markets*, *4*(3), 257–274. https://doi.org/10.1023/A:1009604804110

Liebenberg, L. (2002, July). *The Electronic Financial Markets of the Future: Survival Strategies of the Broker-Dealers*. Springer.

Lo, A. W. (2002). The Statistics of Sharpe Ratios. *Financial Analysts Journal*, *58*(4), 36–52.

Lo, A. W. (2019, May). *Adaptive Markets: Financial Evolution at the Speed of Thought*. Princeton University Press.

Lo, A. W., & MacKinlay, A. C. (1999). *A Non-Random Walk down Wall Street*. Princeton University Press.

Lopez de Prado, M. (2018a, January). The 10 Reasons Most Machine Learning Funds Fail. https://doi.org/10.2139/ssrn.3104816

Lopez de Prado, M. (2018b, January). *Advances in Financial Machine Learning*. John Wiley & Sons.

Lopez de Prado, M., & Bailey, D. H. (2018, July). The False Strategy Theorem: A Financial Application of Experimental Mathematics.

Lopez de Prado, M., & Peijan, A. (2005, January). Measuring Loss Potential of Hedge Fund Strategies.

Louppe, G. (2015, June). Understanding Random Forests: From Theory to Practice. https://doi.org/10.48550/arXiv.1407.7502

Mandelbrot, B., & Taylor, H. M. (1967). On the Distribution of Stock Price Differences. *Operations Research*, *15*(6), 1057–1062. https://doi.org/10.1287/opre.15.6.1057

Markham, J. W., & Harty, D. J. (2007/2008). For Whom the Bell Tolls: The Demise of Exchange Trading Floors and the Growth of ECNs. *Journal of Corporation Law*, *33*, 865.

Mazzoni, T. (2018, March). *A First Course in Quantitative Finance*. Cambridge University Press.

McGowan, M. (2010). The Rise of Computerized High Frequency Trading: Use and Controversy. *Duke Law & Technology Review*, *9*(1), 1–25.

Mota, P. P. (2012). Normality assumption for the log-return of the stock prices. *Discussiones Mathematicae Probability and Statistics*, *32*(1-2), 47. https://doi.org/10.7151/dmps.1143

Müller, A. C., & Guido, S. (2016, September). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. "O'Reilly Media, Inc.".

Oliveira, N., Cortez, P., & Areal, N. (2017). The impact of microblogging data for stock market prediction: Using Twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert Systems with Applications*, *73*, 125–144. https://doi.org/10.1016/j.eswa.2016.12.036

Opdyke, J. D. (2006, March). Comparing Sharpe Ratios: So Where are the P-Values?

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

Quemy, A. (2020). Two-stage optimization for machine learning workflow. *Information Systems*, *92*, 101483. https://doi.org/10.1016/j.is.2019.101483

Quigley, L. (2008). Statistical Analysis of the Log Returns of Financial Assets.

Samuelson, P. A. (2013, September). Proof that Properly Anticipated Prices Fluctuate Randomly. In *The World Scientific Handbook of Futures Markets* (pp. 25–38, Vol. Volume 5). WORLD SCIENTIFIC. https://doi.org/10.1142/9789814566926_0002

Sharpe, W. F. (1964). Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk*. *The Journal of Finance*, *19*(3), 425–442. https://doi.org/10.1111/j.1540-6261.1964.tb02865.x

Shumway, R. H., & Stoffer, D. S. (2006). *Time Series Analysis and Its Applications: With R Examples*. Springer.

Smith, D. M., Wang, N., Wang, Y., & Zychowicz, E. J. (2016). Sentiment and the Effectiveness of Technical Analysis: Evidence from the Hedge Fund Industry. *Journal of Financial and Quantitative Analysis*, *51*(6), 1991–2013. https://doi.org/10.1017/S0022109016000843

Stefan, A. M., & Schönbrodt, F. D. (2023). Big little lies: A compendium and simulation of p-hacking strategies. *Royal Society Open Science*, *10*(2), 220346. https://doi.org/10.1098/rsos.220346

Tian, G. G., Wan, G. H., & Guo, M. (2002). Market Efficiency and the Returns to Simple Technical Trading Rules: New Evidence from U.S. Equity Market and Chinese Equity Markets. *Asia-Pacific Financial Markets*, *9*(3), 241–258. https://doi.org/10.1023/A:1024181515265

Tsai, C.-F., & Hsiao, Y.-C. (2010). Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, *50*(1), 258–269. https://doi.org/10.1016/j.dss.2010.08.028

Tsay, R. S. (2010, October). *Analysis of Financial Time Series*. John Wiley & Sons.

Tulchinsky, I. (2019). *Finding Alphas: A Quantitative Approach to Building Trading Strategies*. John Wiley & Sons, Incorporated.

Urquhart, A., & McGroarty, F. (2016). Are stock markets really efficient? Evidence of the adaptive market hypothesis. *International Review of Financial Analysis*, *47*, 39–49. https://doi.org/10.1016/j.irfa.2016.06.011

Wang, S., Luo, Y., & Alvarez, M.-A. (2014). Seven Sins of Quantitative Investing. *Signal Processing*.

Wilder, J. W. (1978). *New Concepts in Technical Trading Systems*. Trend Research.

Zhang, Z., & Pfister, T. (2021, September). Learning Fast Sample Re-weighting Without Reward Data. https://doi.org/10.48550/arXiv.2109.03216