Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master Thesis

Revisiting Equity Strategies with Financial Machine Learning

Luca Schneider

Department of Management, Technology and Economics
Chair of Entrepreunarial Risks

Supervisor:
Prof. Dr. Didier Sornette

September 2019

Acknowledgements

First of all, I would like to express my deep gratitude to my thesis supervisor, Professor Sornette, for this research opportunity, his patience, availability and his suggestions that contributed to my reflection.

I also thank Samuel Fux and Loher Gerson from the IT-Services of ETH Zurich for providing the computational resources for this thesis.

I would also like to extend my thanks to my previous employer*, Banque Paris Bertrand SA*, and its *Systemic Asset Management* team for providing the raw data and the expertise in quantitative portfolio management.

I wish to acknowledge the patience and continuing support from my partner, Jasmin Rose Lewis, and my relatives who have made the achievement of this thesis and the Master's degree possible.

Finally, a big thank you to my student colleagues from ETH, Timothée Barattin, Martin Butterscotch, Emmanuel Profumo and Cédric Travelletti, for their extensive advice in code development and statistics, and their support throughout the writing of this thesis.

Abstract

With the recent trend around Artificial Intelligence and Machine Learning, these techniques have been widely used by academics and practitioners to forecast financial markets. With the most modern concepts of financial machine learning and data science, this paper attempt to derive a sustainable and interpretable Equity strategy with common Equity strategies signals as input. This paper also reviews numerous challenges and issues encountered in financial machine learning application as early modelling have been compromised by data leakage despite cautious procedures. After remedying it, the strategies proposed remain profitable and still beats the Russell 1000 Index, including transaction costs.

Contents

1 Introduction **1**

2 Thesis Outline **3**

3 Essential Concepts in Financial Machine Learning **8**

4 Feature Engineering **11**

4.1 Context **11**

4.2 Feature Cleaning **15**

4.3 Exploratory Data Analysis **16**

4.4 Initial Feature Selection **21**

5 Predictive Modelling **25**

5.1 Context **25**

5.2 Primary Model **34**

5.3 Feature Selection **41**

5.4 Data Leakage **44**

5.5 Pipeline Performances **47**

6 Portfolio Allocation and Portfolio Analysis **48**

6.1 Context **48**

6.2 Results **50**

7 Conclusion and Further Research **55**

References **58**

Appendix A: Python and Dataset Files 70

Appendix B: Downloading Bloomberg data with Python 71

Appendix C: List of Bloomberg Fields downloaded 73

Appendix D: Dataset Cleaning: a Detailled Description 76

Appendix E: List of Features 79

Appendix F: Equity Strategies Computation 87

Appendix G: Portfolio Statistics Computation 112

Appendix H: AutoML and Deep Learning 115

List of Figures

Figure 2.1 Graphical illustration of the trading framework. 4
Figure 2.2 Graphical illustration of the machine learning pipeline. 5
Figure 4.1 Binary representation of missing values projected in the first two dimensions. ...13
Figure 4.2 Visualization of missing data patterns over the years. 14
Figure 4.3 Visualizations of the absolute and relative differences of SIDs over the years after feature cleaning. 16
Figure 4.4 Variability summarized across the first 20 components of the FAMD. 17
Figure 4.5 Summary analysis of the first and second components of the FAMD. 19
Figure 4.6 Summary analysis of the third and fourth components of the FAMD. 20
Figure 4.7 Visualization of the quantitative predictors' correlation matrix. 22
Figure 4.8 Visualization of the uncertainty coefficient matrix. 24
Figure 5.1: Graphical illustration of the meta-labelling approach. 25
Figure 5.2: Graphical illustration of the data splitting and the training methodology used. ..30
Figure 5.3: Graphical illustration of the Repeated Holdout validation. 31
Figure 5.4: Comparison of training methodologies: Means and standard deviations of metrics computed monthly 36
Figure 5.5: Comparison of training methodologies. 37
Figure 5.6: Comparison of training with missing values: Means and standard deviations of metrics computed monthly. 38
Figure 5.7: Comparison of training with missing values 38
Figure 5.8: Comparison of training with rolling pre-processed data: Means and standard deviations of metrics computed monthly. 39
Figure 5.9: Comparison of training with rolling pre-processed data: 40
Figure 5.10: Predictor selection percentage for selected feature subsets. 41
Figure 5.11: The cross-validated results for RFE and SFS using GBT. 42
Figure 5.12: Comparison of training with different feature subsets: Means and standard deviations of metrics computed monthly. 43
Figure 5.13: Graphical illustration of data leakage. 45
Figure 5.14: Comparison of pipeline performances on the validation set: Means and standard deviations of metrics computed monthly. 47
Figure 6.1 Empirical Sharpe Ratio distribution for strategies 50
Figure 6.2 Top panel: Out-Sample cumulative returns for strategies and benchmarks. 52
Figure 6.3 Top panel: Out-Sample ratio of the strategies cumulative returns over the Russell 1000 Index cumulative returns 53

List of Tables

Table 4.1 Association between qualitative variables. 23
Table 6.1 Strategies performances for out-of-sample data. 54

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AUC	Area Under the receiver operating characteristic Curve
AutoML	Automated Machine Learning
B/P	Book-to-Price
BDH	Bloomberg Data History
BDP	Bloomberg Data Point
BDS	Bloomberg Data Set
BICS	Bloomberg Industry Classification System
BLPAPI	Bloomberg API
BQL	Bloomberg Query Language
CA	Correspondence analysis
CAGR	Compound Annual Growth Rate
CPCV	Combinatorial Purged Cross-Validation
CUSIP	Committee on Uniform Security Identification Procedures
CV	Cross-Validation
DART	Dropouts meet Multiple Additive Regression Trees
DL	Deep Learning
DT	Decision Tree
EDA	Exploratory Data Analysis
EMH	Efficient-Market Hypothesis
EPS	Earnings Per Share
FAMD	Factor Analysis of Mixed Data
FPR	False Positive Rate
GA	Genetic Algorithm
GBT	Gradient Tree Boosting
GPGPU	General-Purpose GPU programming
GPU	Graphical Processor Unit
HDF5	Hierarchical Data Format

IID	Independent and identically distributed random variables
IPO	Initial Public Offering
ISIN	International Securities Identification Number
JSON	JavaScript Object Notation
***k*-NN**	*k*-Nearest Neighbors
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Logistic Regression
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAR	Missing At Random
MCA	Multiple Correspondence Analysis
MCAR	Missing Completely At Random
MCFD	Mean Correct Forecast Directions
MedAE	Median Absolute Error
MFTR	Mean Forecast Trading Returns
MI	Mutual Information
ML	Machine Learning
M&A	Mergers and Acquisitions
NA	Non-Available
NB	Naive Bayes
NLP	Natural Language Processing
NMAR	Not Missing At Random
NYSE	New York Stock Exchange
OLS	Ordinary Least Squares
PCA	Principal Component Analysis
PIT	Point-In-Time
Rep-Holdout	Repeated Holdout validation
RFE	Recursive Feature Elimination
RF	Random Forests
RL	Reinforcement Learning
RNN	Recurrent Neural Networks

ROC	Receiver Operating characteristic Curve
SA	Simulated Annealing
SBS	Sequential Backward Selection
SDK	Software Development Kit
SHAP	SHapley Additive exPlanation
SID	Security Identifier
SUE	Standardized Unexpected Earnings
SVM	Support-Vector Machine
TPE	Tree-structured Parzen Estimator
TPR	True Positive Rate

Notation

This section provides a concise reference describing the notation used throughout this thesis.

Number & Arrays

a	A scalar
A	A scalar constant
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
a	A scalar random variable
a	A vector-valued random variable
A	A matrix-valued random variable

Indexing

a_i	Element i of vector a, with indexing starting at 1
a_{-i}	All elements of vector a except for element i
$A_{i,j}$	Element (i, j) of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,j}$	Column i of matrix $\boldsymbol{A}$

Sets

$\{0,1\}$	The set containing 0 and 1
$\{0,1,...,n\}$	The set of all integers between 0 and n
$[a,b]$	The real interval including a and b
$(a,b]$	The real interval excluding a but including b

Calculus

$f'(a)$	Derivative of f at input point a
$\int_{\mathbb{S}} f(x)dx$	Definite integral with respect to x over the set $\mathbb{S}$

Probability & Information Theory

$\mathrm{P}(a \mid b)$	Short hand for the probability $\mathrm{P}(\mathrm{a} = a \mid \mathrm{b} = b)$
$P(\mathrm{a})$	A probability distribution over a discrete variable a
$p(\mathrm{a})$	A probability distribution over the continuous variable a
H(x)	Shannon entropy of the random variable x

Datasets and Machine Learning

$\mathbb{X}$	A set of training examples
$\boldsymbol{x}^{(i)}$	The i-th example from dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning
$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $\boldsymbol{X}_{i,:}$
$P(\mathrm{x}, \mathrm{y})$	A data generating distribution
$\mathbb{F}$	Hypothesis space of function to be learnt, i.e., a model
(x', y')	A testing pair
$\hat{y}$	Label predicted by a function f, i.e., $\hat{y} = f(x')$

Functions

$f(\boldsymbol{x}, \theta)$	A function of $\boldsymbol{x}$ paramatrized by θ
$\ln x$	Natural logarithm of x
$\mathbf{1}_{\text{condition}}$	is 1 if the condition is true, 0 otherwise

1 Introduction

Financial markets have always been leading to a scientific and technological arms race among its participants. After the eras of trend-following, of statistical arbitrage, and high frequency-trading, it seems that quantitative trading has been transitioning into a new era recently, with the democratization of Artificial Intelligence and Machine learning brought by digital companies (Sirotyuk, 2018). However, nor machine learning or trying to forecast financial markets with it are new. The academic literature abounds of application of different models and techniques. (Cavalcante, Brasileiro, Souza, Nobrega, & Oliveira, 2016) provides a comprehensive review of techniques used by publication ranging from 2009 to 2015. More recently, (Ryll & Seidens, 2019) perform a meta-analysis on more than 150 articles related financial machine learning published from 1995 to 2018. In addition to rank deep learning models higher than other models, the authors observed the lack of standards shared in financial machine learning as compared to the rest of the machine learning space.

Besides, the exercise of financial machine learning differs sensibly from the other machine learning areas due to the nature of financial markets. Main cited reasons are :

- **Size of datasets in finance**: In computer vision, a single 3-channels picture of 224 pixels format provides already more than 150'528 bits of information, as much as 150 years of OLHCV data for a single stock.
- **Non-Stationarity**: Machine learning theories have been formulated with the assumption of constant data pattern, while financial markets are known for their adaptivity and regime changes (W. Lo, 2017).
- **Noise to signal-ratio**: Machine learning algorithms will always identify a pattern even it is a false signal (Black, 1986).
- **Decision space**: In comparison to the main applications of Reinforcement Learning (RL) (Sato, 2019) in decision space with defined rules, finance involves the choice between numerous assets classes, markets,… without forgetting external uncertainties to be considered such as political or regulatory events.

As such, defining a trading strategy with machine learning is far from trivial due to the challenging nature of the finance area. Recently published, (López de Prado, 2018) laid the ground for the rigorous application of machine learning in real investment scenario cases. As such, this thesis relies on such advanced methods to attempt to derive a sustainable trading strategy from well-known equity anomalies.

The rest of this thesis is organized as follows. Section 2 outlines the thesis workflow and describes computational tools and data used. Section 3 reviews common bias encountered in strategy backtesting. Section 4 describes data exploration and early data pre-processing. Section 5 sets the context of the machine learning models employed for classifying stock returns and deals with issue of data leaking. Section 6 translates predictions into tradable signals before backtesting the final strategies. Finally, concluding remarks are presented in Section 7.

2 Thesis Outline

Context

This thesis focuses on the study of the predictability of stock returns within an interpretable machine learning framework. In addition, it is aimed to be as realistic as possible for an asset management environment. As such, the methodology provided here was designed to avoid common pitfalls encountered in financial machine learning application and strategy backtesting, and ideally result in a sustainable trading strategy. As this thesis deals co-jointly with the fields of quantitative trading, machine learning and statistical learning, the models and methodologies employed here are not sufficiently detailed in order to keep the paper consistent. However, their choices and consequences remain contested, and references are provided for interested readers. Therefore, it is expected that the reader is familiar with machine learning concepts in order to be able to grasp the content of this thesis. Finally, concepts are described progressively along the thesis to match with the paper's workflow, instead of compiling them in an early chapter, as may usually be encountered in other theses.

Predictors

The assumed set of predictors is as given in the equity section of (Kakushadze & Serur, 2018).

Investment Universe

The investment universe is defined as the stock components of the Russell 1000 Index. This choice is motivated by data availability, market efficiency and liquidity. The index measures the performance of the large-cap segment of the U.S. equity universe. It is a subset of the Russell 3000 Index and includes approximately 1000 of the largest securities, representing approximately 92% of the U.S. market capitalisation[1]. The main index is updated annually, usually on the last Friday of June. Furthermore, the eligible quarterly Initial Public Offering (IPO) may be included along the year. For more information about the full methodology of the index construction, please refer to the methodology available online[2]. Finally, being heavily scrutinised by financial professionals and academic studies, it is expected that an American mid-/large-cap stock subset is an accurate measure to test the sustainability of any trading scheme.

[1] FRED: https://fred.stlouisfed.org/series/RU1000TR
[2] FTSE Russell: https://research.ftserussell.com/products/downloads/Russell-US-indexes.pdf?704&_ga=2.184114918.1931757306.1563636511-2056835715.1563636511

Trading Frequency

As equity strategies from (Kakushadze & Serur, 2018) range from intra-day to quarterly, a monthly set up is chosen as a fair trade-off to encompass technical analysis-based strategies, more short-term oriented trading, and fundamental analysis-based ones which are more long-term oriented. This choice is beneficial due to the inclusion of less noisy data, less turnover at the expense of less available data to train the algorithm, and lowered Sharpe ratio in comparison to a daily strategy (E. P. Chan, 2017). As such, monthly predictions are computed after the close of the last trading of month and trades are entered on the close of the following trading day until and kept for one month. This gap prevents look-ahead bias, ensures all data is available and allows for a realistic timeframe needed to run computations. This set up can be visualised in Figure 2.1:

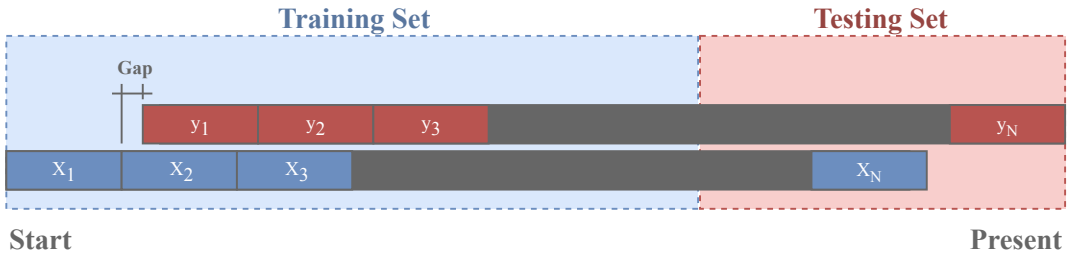

Figure 2.1 Graphical illustration of the trading framework.

Pipeline Outline

This thesis is guided around three core steps:

1. **Feature generation**: The dataset is generated, cleaned, and predictors computed.
2. **Predictive modelling**: A learning algorithm is trained and optimised for defined machine learning metrics to generate predictions.
3. **Portfolio allocation**: predictions are processed to allocate a strategy optimised for defined backtesting metrics.

A chart with all the details of the pipeline is presented in Figure 2.2.

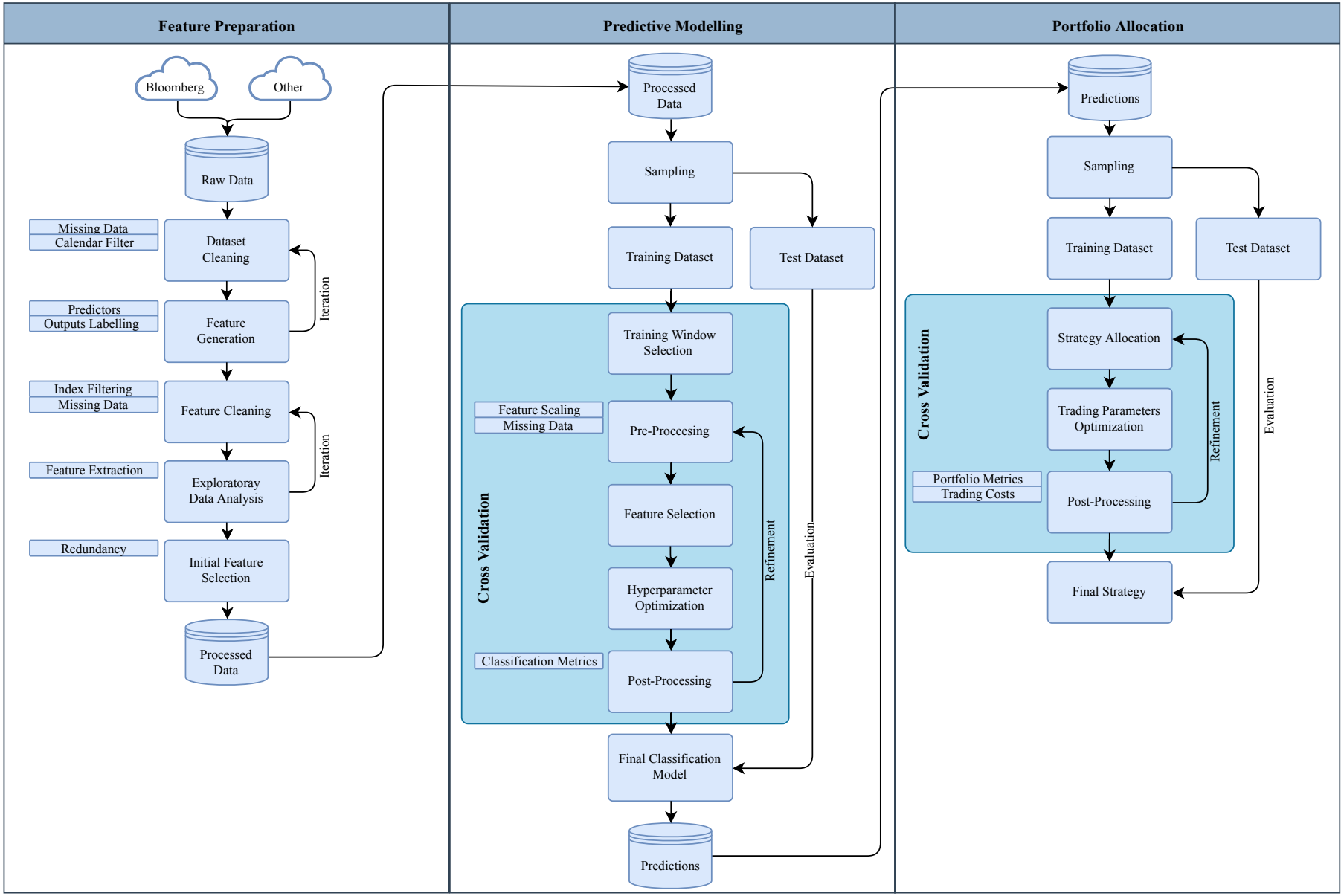

Figure 2.2 Graphical illustration of the machine learning pipeline.

Software

Most of the computations are conducted in Python; a free and open-source interpreted, interactive, object-oriented programming language (Van Rossum & De Boer, 1991). Due to its active community around providing a vast selection of libraries and support, Python is currently among one of the fastest-growing programming languages, especially in the space of data science and artificial intelligence. This thesis makes intensive use of its packages for scientific computing amongst others:

- IPython (Pérez & Granger, 2007), Jupyter (Kluyver et al., 2016), Dask (Dask Development Team, 2016) and Joblib (Joblib developers, 2011) for development;
- NumPy (Van Der Walt, Colbert, & Varoquaux, 2011) and Pandas (McKinney, 2010) for scientific computing and data processing;
- Matplotlib (Hunter, 2007), Seaborn (Waskom et al., 2018) and Yellowbrick (Bengfort et al., 2018) for data visualisation;
- SciPy (Jones, Oliphant, Peterson, & Others, 2001), Statsmodels (Seabold & Perktold, 2010), Scikit-Rebate (Urbanowicz, Olson, Schmitt, Meeker, & Moore, 2018), Dython (Zychlinski, 2018) and Scikit-posthocs (Terpilowski, 2019) for statistical models and statistical tests;
- Scikit-learn (Pedregosa et al., 2011), MLxtend (Raschka, 2018), Pomegranate (Schreiber, 2018) and LightGBM (Ke et al., 2017) for machine learning;
- Keras (Chollet & Others, 2015) and Tensorflow (Abadi et al., 2015) for deep learning;
- Hyperopt (Bergstra, Yamins, & Cox, 2013), Hyperas (Pumperla, 2016) and Optuna (Akiba, Sano, Yanase, Ohta, & Koyama, 2019) for hyperparameter optimisation;
- Lime (Ribeiro, Singh, & Guestrin, 2016), ELI5 (TeamHG-Memex, 2016) and SHAP (Lundberg et al., 2019; Lundberg, Erion, & Lee, 2018; Lundberg & Lee, 2017) for machine learning explainability;
- Auto-sklearn (Feurer et al., 2015) and TPOT (Olson, Bartley, Urbanowicz, & Moore, 2016) for Automated Machine Learning (AutoML);
- Pyfolio (Quantopian Inc., 2015) and Empyrical (Quantopian Inc., 2017) for portfolio analysis.

For readability and reproducibility purposes, Jupyter notebooks are designed according to (Rule et al., 2019). Other undirectly relevant utility packages used here are mentioned throughout this paper with references in footnotes.

As no equivalent Python packages could be found, R (R Development Core Team (R Foundation for Statistical Computing), 2008) and the following packages have been used:

- FactoMineR (Lê, Josse, & Husson, 2008) for multivariate analysis.

Links to the scripts and notebooks used can be found in Appendix A.

Hardware

Some of the calculations reported here were performed using the EULER cluster at ETH Zurich and on the Google Cloud Platform, while deep learning models were realised with a Graphical Processor Unit (GPU): an NVDIA Quadro P5000 with 2560 CUDA cores and 16GB GDDR5. To enable large scale distributed parameters optimizations, a PostgreSQL database (Stonebraker & Rowe, 1986) from ETH Zurich has been used.

Data

In this paper, a Bloomberg terminal has been used to retrieve most financial data. While the data provider dominates the market of financial data with 33.2% of market share for institutional investors in 2017[3], most encountered financial commercial data is not the most suitable for backtesting purposes. Such data might need additional efforts in order to be used appropriately within a strategy development. Bloomberg data can easily be downloaded through their add-in provided for Excel. In order to automate the processing of larger quantities of data, the Bloomberg API[4] (BLPAPI) for Python has been mostly used. More information about BLPAPI can be found in Appendix B. A list of all fields downloaded on Bloomberg can be found in Appendix C. To avoid survivorship-bias (see Chapter 3), security identifiers (SID) for the end-of-month constituents of the Russell 100 Index are obtained from January 1995 (earliest date available for the constituents on Bloomberg) to March 2019 and compiled into a binary matrix, indicating whether the stock is a constituent of the index for the subsequent month or not. Then, after a cleaning of the SIDs list referred in Appendix D, the list is used to download individual security data where stock prices have been adjusted for dividends, corporate actions and stock splits. Finally, after cleaning the equity data, referred to in Appendix D, the dataset is ready for feature generation.

[3] Financial Times: https://www.ft.com/content/622855dc-2d31-11e8-9b4b-bc4b9f08f381
[4] Bloomberg API: https://www.bloomberg.com/professional/support/api-library/

3 Essential Concepts in Financial Machine Learning

We review here essential concepts that should be taken into account before any data collecting. High-data quality is critical in a robust strategy development process and could be one of its most time-consuming steps. A common way to validate a trading strategy is to simulate it on past data, namely backtesting the strategy. It is as essential as not trivial to perform it. Naive approaches would lead to bias and flawed result metrics due to bias known among academics and financial professional. Adapted data and limiting bias is a necessary condition for a robust trading strategy. The following bias and errors are compiled from (López de Prado, 2018; Luo et al., 2014). The bias shares different terminologies between the fields of quantitative trading, statistical learning and machine learning but refers to the same underlying issues:

1. **Survivorship bias:** It is usually introduced by considering only the current investment universe across the backtesting. Excluding securities that previously left it (i.e. bankruptcy, acquisition or delisting), would artificially inflate portfolio metrics. (Daniel, Sornette, & Woehrmann, 2009) analyses deeply this issue.
2. **Look-ahead bias:** Also referred as data leakage, (Kaufman, Rosset, Perlich, & Stitelman, 2012), this bias consists of acting upon data that would have not yet been available at the simulated moment. It is ubiquitous that macroeconomic or companies' fundamental data are lagged and can even be corrected posteriorly. For the same reason data random shuffling or classic cross-validation should be avoided when dealing with non-stationary time series data. Data timestamps of data availability should be monitored carefully in order to limit the likelihood of peeking into the future. It also possible to find Point-In-Time (PIT) data from data providers. In this way, CRSP and Compustat have been the most widely used database in the empirical finance literature[5].
3. **Storytelling:** Overinterpreting and trying to justify past random patterns.
4. **Data snooping:** Also known as cherry-picking, p-hacking or selection bias, it occurs when a given set of data is used more than once for purposes of inference or model selection. When such data reuse occurs, there is always the possibility that any satisfactory results obtained may be due to chance rather than to any merit inherent in the method yielding the results. It is particularly relevant in finance due to its non-experimental nature. Splitting data into in-sample dataset (training set) and out-of-

[5] CRSP/COMPUSTAT: https://wrds-www.wharton.upenn.edu/pages/support/applications/linking-databases/linking-crsp-and-compustat

sample dataset (validation/testing set) and statistical tests, as the Bonferroni correction should be considered to prevent false discovery.

5. **Transaction costs:** Transaction costs could only be known if the actual trade would have been made and are thus complicate to be simulated precisely.
6. **Outliers:** Outliers are observations that deviates so much from others data points, raising the suspicion of being the result of a different generating process. Such extreme outcomes should be considered carefully They might never happen again and have non-desired effects on a model. Investigating the nature and the severity of outliers and various robust statistical tools can help to mitigate it.
7. **Shorting:** Short selling involves the availability of a lender, a cost associated with it and depends on different other market conditions and restrictions. Some regulators banned it temporarily during the financial crisis of 2007-2008 and other periods.

In addition to these common errors, it is crucial to continually remain critical and look for other flaws that could affect the validity of backtesting results. For instance, the following could also be considered:

- Using algorithms, especially machine learning, or exploiting financial anomalies on data corresponding to before their discovery.
- Daily and lower frequency strategies usually rely on open and closing prices. As these prices are formed through an auction process every day, data providers often use the *consolidated prices* among market centres and might not reflect the trade execution price that one could obtain[6].
- Original stock prices might display variations despite not reflecting a change in real value for investors. When a dividend is issued, the stock price immediately drops to take into account that the company no longer owns this money. Another case is stock split where a company decides to increase/decrease the number of its outstanding shares. Prices are thus expected to follow proportionally. To prevent these price variations, it is common to find adjusted prices from data providers. The adjustment formulas used should be checked as it is not consistent across data providers. In the case of Bloomberg, backtesting with their adjusted close would not simulate receiving

[6] Beware of Low Frequency Data: http://epchan.blogspot.com/2015/04/beware-of-low-frequency-data.html

dividend nor returns from re-investing it, underestimating strategy's long term expected return[7].

[7] Backtesting 101: Dividends and Adjustments: https://backtest-rookies.com/2018/10/12/backtesting-101-dividends-and-adjustments/

4 Feature Engineering

4.1 Context

Feature generation

After data collection and cleaning, 92 monthly predictors (57 quantitative and 35 qualitative) have been derived for each SID based on the Equity section of (Kakushadze & Serur, 2018). Depending on the data format required from the libraries used, qualitative data may need to be one-hot encoded, leading to the possibility of different numbers of predictors displayed across figures. The predictors are listed in Appendix E. Backgrounds and computations are presented in Appendix F. As opposed to training algorithms parallelly for each security i, all individual stock datasets $\{(\boldsymbol{X}_i, \boldsymbol{y}_i)\}_{i=1,\dots,I}$ are stacked into a single matrix $(\boldsymbol{X}, \boldsymbol{y})$ in order to fit one classifier on the investment universe simultaneously. This approach leads to more data for training and consecutively more general conclusion, reduced outliers influence and less overfitting (López de Prado, 2018). The decrease in computational complexity for optimising only one machine learning pipeline can also be added.

Target generation

As explained earlier, the returns to be predicted are computed between the close between each first trading day of months. This one-day gap in comparison to features prevents look-ahead bias in computation. Also, some SIDs might get delisted along the month, and should be taken into account to avoid survivorship bias. Delisted shares were assumed to equal a null share value, as the reason why the stock became delisted was not possible to automate for the quantity of SIDs on Bloomberg. For labelling, the methodology from (Fu et al., 2018) is followed here: stocks risk-adjusted returns are ranked for each timestamp. While the top and bottom quantiles are labelled positively and negatively, the candidates in the middle are discarded to filter out noise. It is proposed here to keep the middle candidates and to assess their values later with the meta-labelling approach in the predictive modelling chapter. Accordingly, target returns are defined as follows.

For each month, the targeted returns R_i of the n stocks constituting the index are firstly ranked with the empirical distribution function $\hat{F}_n(x)$:

$$\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(x_i \leq x)}$$

The associated cumulative probabilities are then labelled according to the quantile q:

$$y_i(q) \doteq \begin{cases} 1 \text{ if } \hat{F}_n(R_i) \geq 1 - q \\ -1 \text{ if } \hat{F}_n(R_i) \leq q \\ 0 \text{ otherwise} \end{cases}$$

While (Fu et al., 2018) labels only risk-adjusted returns, simple returns have also been labelled here, and the association between them tested statistically. For illustration purposes, $q = 1/3$ is set in the following section. Across the entire dataset, approximately 14% of the returns would be labelled differently than the risk-adjusted returns. The Spearman's rank correlation r_s (Spearman, 1904), Kendall's rank coefficient τ (Kendall, 1938), Chi-squared test χ^2 (Pearson, 1900), Cramer's value V (Harald, 1946) and their respective p-values[8] have been computed as:

$$r_s = 0.88, p \approx 0$$
$$\tau = 0.85, p \approx 0$$
$$\chi^2 = 582272, p \approx 0$$
$$V = 0.78, p \approx 0$$

It can be concluded that both types of returns are statistically too associated to be considered separately. Only simple returns are therefore kept.

Missing data assumptions

After filtering SIDs according to their index membership for each month, the predictors are first inspected for missing data. The heatmap of Figure 4.2 displays the percentage of missing data for each predictor along time. For a more condensed visualisation of missing patterns across predictors, a Principal Component Analysis (PCA) (Pearson, 1901) can be used. In this setting, the predictors are first converted to a binary matrix where missing data is labelled one, and zero otherwise (Kuhn & Johnson, 2019). The matrix is then transposed for predictors to be in rows before PCA is finally applied. Figure 4.1 illustrates the projections of the predictors on the two first components. A significant part of the missing data comes first from implied volatility-related predictors, for which no data is available before beginning 2005 and secondly from the predictors `Standardized Unexpected Earnings` and `Residual Momentum`.

[8] p-values are smaller than the system smallest representable positive number and the values 0 have been then returned. See https://stackoverflow.com/questions/20530138/scipy-p-value-returns-0-0

These two have the longest lookback window requirement for computation, namely 24 and 36 months and may most likely be the reason for missingness increase, in addition to the lower availability of fundamental data for early timestamps.

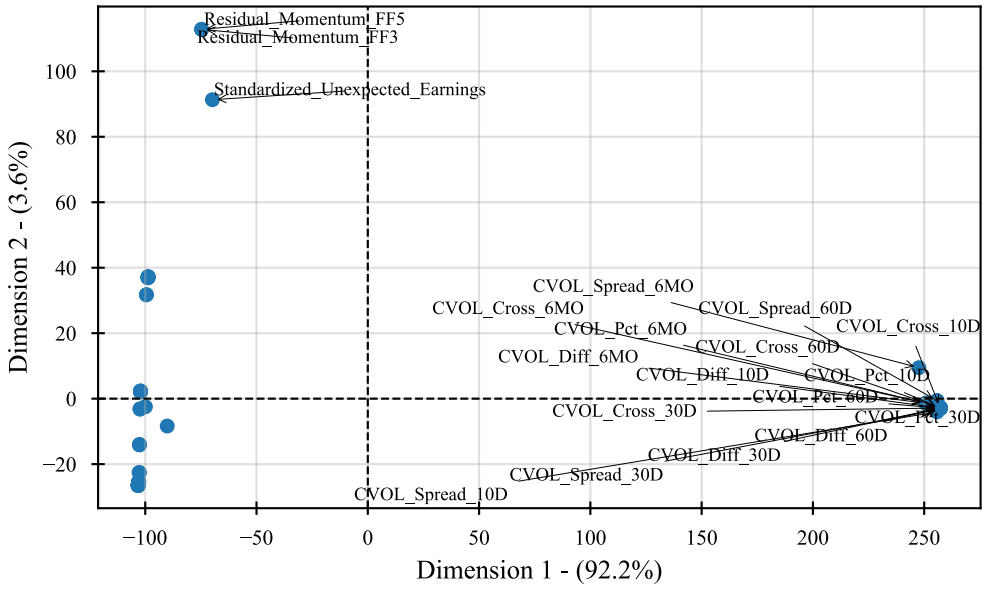

Figure 4.1 Binary representation of missing values projected in the first two dimensions**.**

Due to the previous explanation, the present missing data can be categorized as *Not Missing At Random* (NMAR) as opposed to *Missing Completely At Random* (MCAR) or *Missing At Random* (MAR) (Roderick & Donald, 2019). While missing information for categorical variables can be easily encoded as *Non-Available* (NA) (Kuhn & Johnson, 2019), the situation is more challenging for continuous variables. NMAR represents the most complex case of the three, and strategies to handle it are to explore the causes of missing data and perform sensitivity analyses. Due to the temporal aspect of the problem, it is assumed here that observations are not statistically independent leading to a cautious approach for not introducing any bias. Bayesian methods are proposed by (Kofman & Sharpe, 2000; Roderick & Donald, 2019). However, as they are outside of the scope of this thesis, such imputation is not attempted. Instead problematic predictors are momently dropped, and value of remaining missing data is assessed later with resistant models such as Decision-Tree (DT) models or Naïve Bayes (NB) (Hand & Yu, 2001).

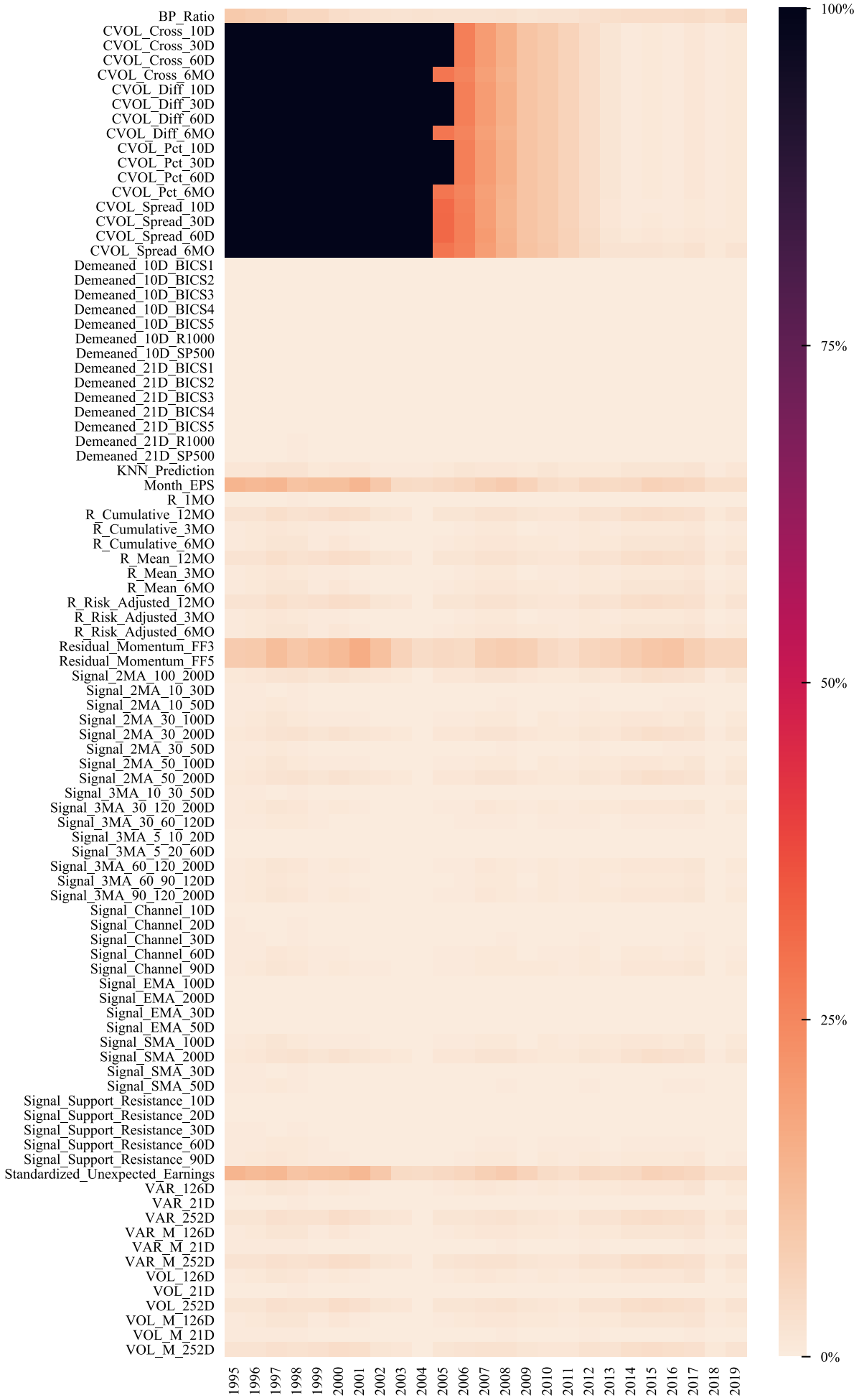

Figure 4.2 Visualization of missing data patterns over the years.

Outliers handling assumptions

Outliers have been presenting a constant challenge in empirical finance research, mainly when relying primarily on Ordinary Least Squares (OLS). A 30-year review of empirical finance articles was performed by Adams, Hayunga, Mansi, Reeb, & Verardi, (2019), using mainly OLS and erroneous practice of outliers' mitigation if any: winsorizing, trimming, dropping or univariate outliers' identification in place of multivariate analysis. Instead, the authors introduce a multivariate identification strategy and an estimator for both cross-sectional and panel regressions. This issue is approached here differently as classification methods have been chosen. Harris (2017) revisits statistical distributions used in the fields of economics and finance, questioning the most use of Frequentist approaches and advocating for a balance with Bayesian methods instead. Thus, this thesis approaches data with distribution-free and robust methods. It is assumed instead that outliers' origins are not the result of error and might contain valuable information. They should not be corrected nor dropped, assuming Bloomberg has entered its data correctly. In that sense robust algorithms, such as DT or data transformation techniques, such as spatial sign (Serneels, De Nolf, & Van Espen, 2006), should be applied.

4.2 Feature Cleaning

Based on the formulated assumptions, the predictors are cleaned as following:

1. Filtering individuals according to their index membership for each month
2. Encoding missingness for qualitative predictors.
3. Dropping implied volatility-related predictors momently. They will be re-assessed in the modelling process.
4. Dropping individuals with remaining missing data. They will be re-assessed in the modelling process.

Figure 4.3 illustrates the effect of the cleaning process. For each month, 12% SIDs are dropped on average, leaving a monthly average of 865 SIDs lefts for training. The main reason for such a drop is related to the window length requirement to compute certain features as mentioned before.

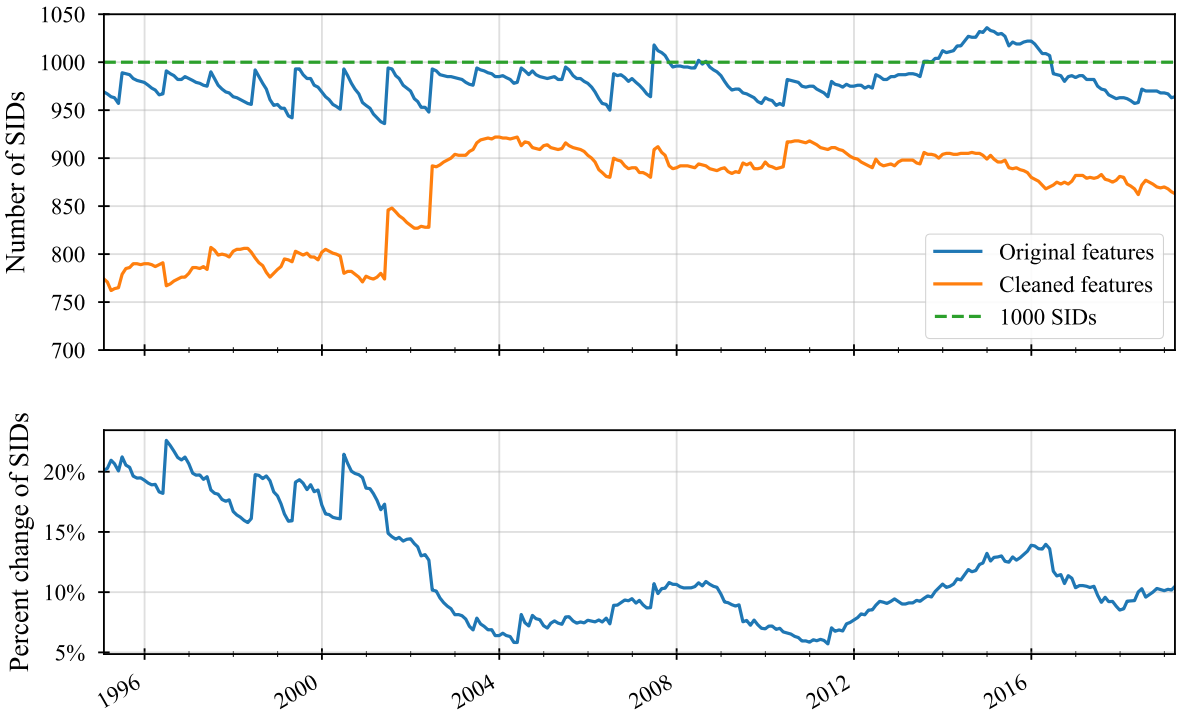

Figure 4.3 Visualizations of the absolute and relative differences of SIDs over the years after feature cleaning.

4.3 Exploratory Data Analysis

After laying out the dataset and main assumptions, the dataset can be analyzed, and its main characteristics summarised with the help of visual methods, a step namely called *Exploratory Data Analysis* (EDA). As each of the 92 predictors are evaluated here, the most insightful figures and analyses are presented in this thesis to remain consistent. The interested reader is invited to refer to the Jupyter notebook (see appendix A) displaying the full EDA process where univariate distributions, bivariate distributions and other visual representations are presented.

Feature extraction

To directly visualise 92 dimensions, dimension reduction techniques are applied. As PCA can only be performed on continuous data, a Factor Analysis of Mixed Data (FAMD) (Pages, 2004) is chosen here to simultaneously analyses both qualitative and qualitative data. As an analogy, FAMD can be said to behave as PCA on quantitative variables and as Multiple Correspondence Analysis (MCA) for qualitative variables. MCA designs the extension to more than two categorical variables of a Correspondence Analysis (CA) (Gibrat, 1978). Before performing FAMD with R package FactoMineR, quantitative data needs to be normalised. As many predictors display skewness, a Yeo–Johnson transformation (Yeo & Johnson, 2000) is applied first before standardisation. The data transformation revealed to be non-essential as FAMD

with raw values displayed the same results. Figure 4.4 displays the explained variance across the first 20 components and its cumulative amount. The first four components are here examined in detail, capturing around 45% of the overall variability.

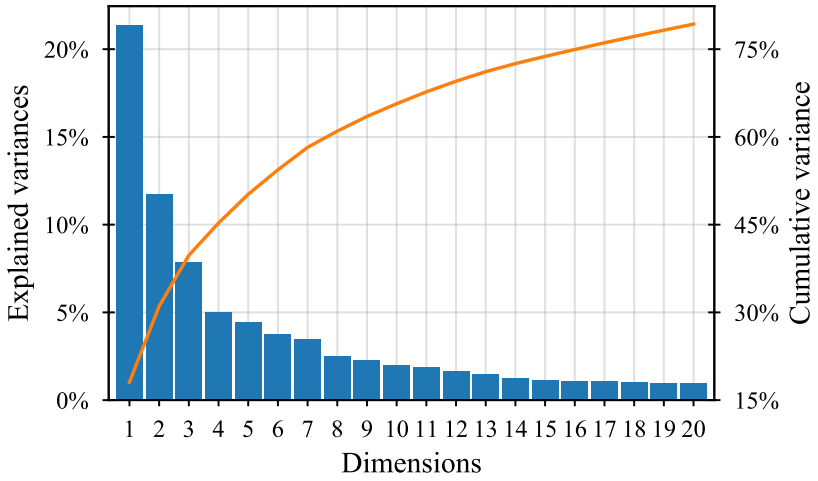

Figure 4.4 Variability summarized across the first 20 components of the FAMD.

Figure 4.5 provides a summary of the analysis for the first and second dimensions, while Figure 4.6 for the third and fourth dimensions. For both figures, part (a) and (b) displays the top 22 variables contributing to the respective dimensions. The red dashed line represents the expected average value if contributions were uniform. Both variables can be represented on plot (c), called the *relationship square*, where the correlation between variables and principal dimensions can be assessed. Quantitative variables can be visualized on the *correlation circle* (d) where the signs of correlations are specified. The categories of qualitative variables are represented in (e). Variable categories with a similar profile are grouped together, while negatively correlated variable categories are positioned on opposite sides of the plot origin. Finally (f) displays individuals projected on the selected plane. In addition, more information is displayed on the plots as follows: variable names have been shortened to preserve space. Variables are coloured according to their contribution to each projection (lowest are in blue and highest are in red) on plots (c), (d) and (e) and only top lowest, and highest contributors have been labelled. Lastly, individuals are coloured according to the return label associated in plot (f). From both figures, it can be observed that neither of the first four components separates the return categories.

In Figure 4.5, the first component seems to have captured a latent variable related to trend-following/momentum (momentum, moving averages, residual momentum, standardized unexpected earnings), while the second one focuses on mean-reversion predictors and related indicators (channels, support and resistance). Volatility and variance have an ambiguous role as

they are negatively correlated to dimension 1, but positively correlated to dimension 2. On the plot of categories, neutral signals are all grouped at the origin. The position of categories seems coherent with the previous interpretation of axis as sell-signals of long channels and of support-resistance are close to buying signals of moving averages and vice-versa.

In Figure 4.6, dimension three have captured all predictors related to volatility and variances. Axis 4 interpretation is more conflictual as no variables seem to be well represented. It can be noticed that short term moving averages contribute the most to it. Again, for categories, buy and sell signals from previously mentioned moving averages are opposed on the 3rd dimension. In general, the primary information to be collected from theses plots is the strong inter- and intra-association of certain continuous and qualitative variables. Redundancy of certainty can easily be suspected, as some variables have been computed with close parameters.

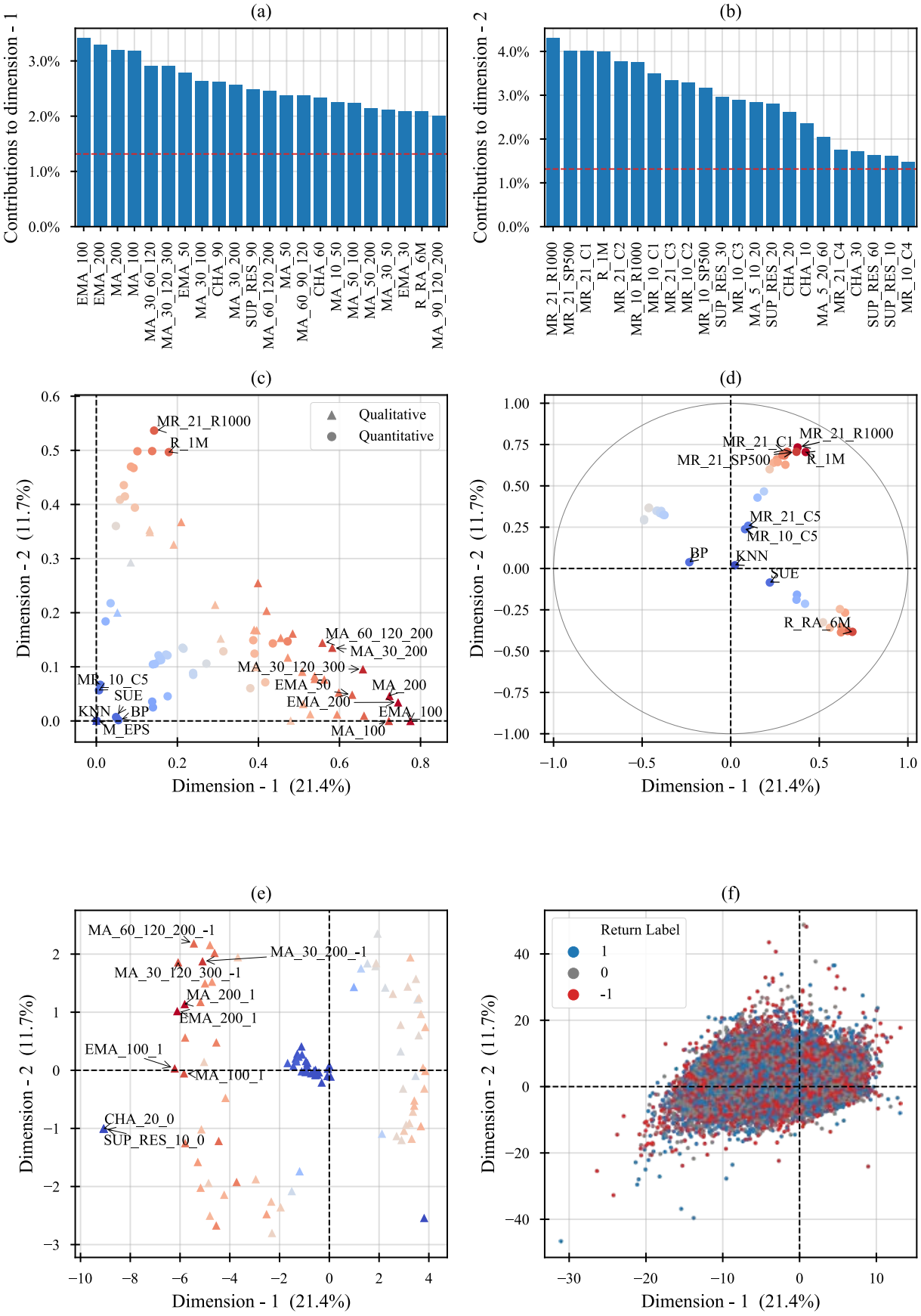

Figure 4.5 Summary analysis of the first and second components of the FAMD.

(a) The 22 predictors contributing the most to the first dimension. (b) The 22 predictors contributing the most to the second dimension. (c) Relationship between quantitative and qualitative predictors. (d) Relationship between quantitative variables and components. (e) Relationship between qualitative variables and components. (f) Projection of individuals on the components.

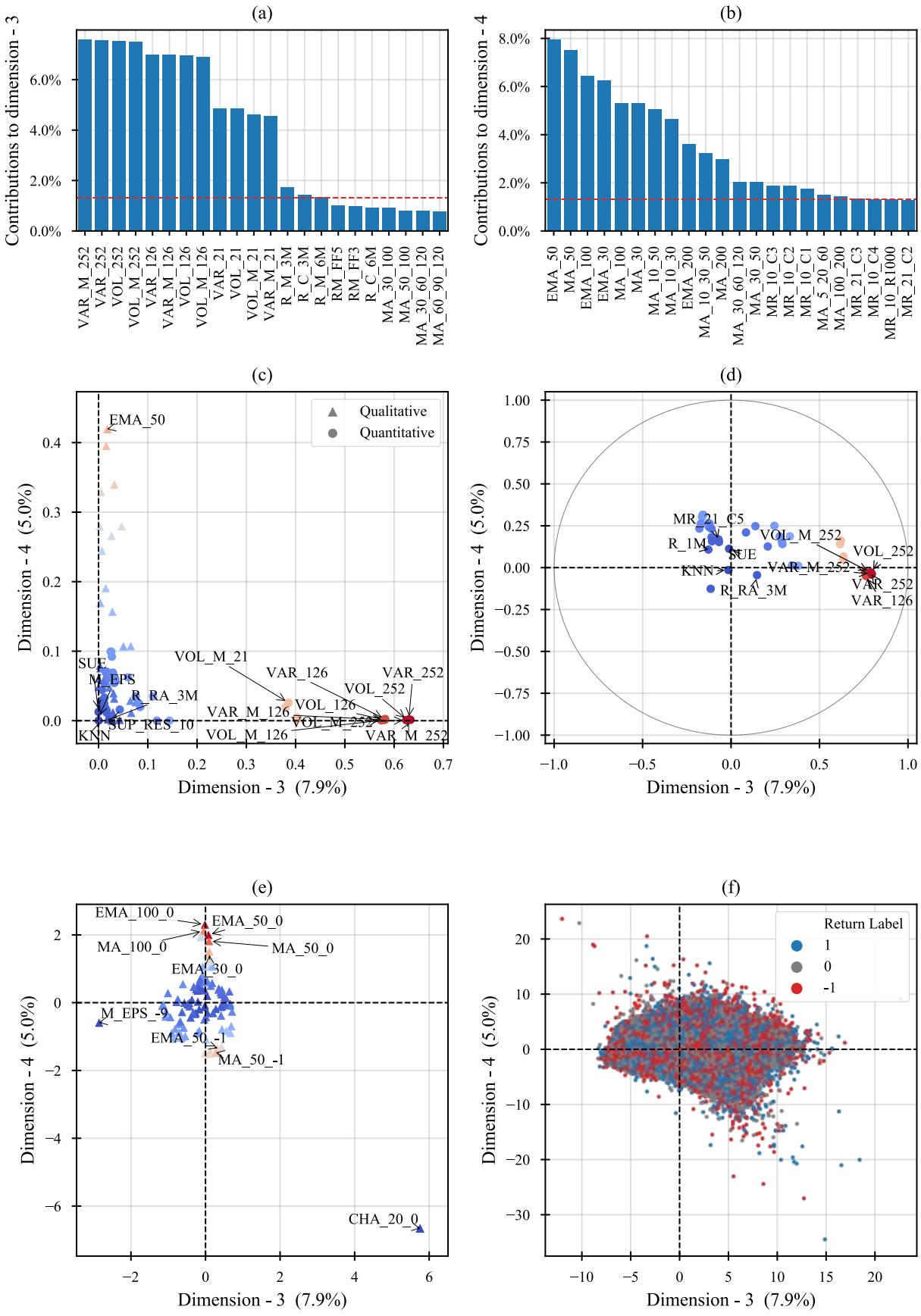

Figure 4.6 Summary analysis of the third and fourth components of the FAMD.

(a) The 22 predictors contributing the most to the third dimension. (b) The 22 predictors contributing the most to the fourth dimension. (c) Relationship between quantitative and qualitative predictors. (d) Relationship between quantitative variables and components. (e) Relationship between qualitative variables and components. (f) Projection of individuals on the components.

4.4 Initial Feature Selection

Context

In general, models face the trade-off between predictive performance and model interpretability: Extensively adding predictors might increase predictive ability, but it would be at the expense of model and computational complexity. In addition, some models may be sensitive to irrelevant predictors, multicollinearity or noise, and it always makes scientific sense to minimise the feature set that provides acceptable results. As such, the goal of feature selection is to reduce the number of predictors as much as possible without compromising predictive performance. Methods of feature selection are usually distinguished by three groups:

1. **Implicit methods**: Some models already incorporate a feature selection in their modelling approaches such as LASSO regression (Tibshirani, 1996) or DT models:
2. **Filter methods**: It designates an initial analysis of predictors before proceeding to modelling. Some algorithms or statistical tests can be used to judge the importance of predictors, such as RelieF (Robnik-Sikonja & Kononenko, 2003) or filtering out correlated predictors.
3. **Wrapper methods**: They are iterative procedures continuously supplying predictor subsets to a learning model. The resulting model performance then guides the selection of the next subset to evaluate. Examples might include Recursive Feature Elimination (RFE) (Guyon, Weston, Barnhill, & Vapnik, 2002) with a greedy approach or non-greedy ones such as Genetic Algorithms (GA) or Simulated Annealing (SA)

Each of these groups presents its advantages and drawbacks. The reader is invited to consult (Kuhn & Johnson, 2019) for an exhaustive description. It is common in early feature engineering steps to remove multicollinear features or irrelevant ones with filter methods. However, for classification tasks, (Guyon & Elisseeff, 2003; Guyon, Gunn, Nikravesh, & Zadeh, 2006) provide small but revealing illustrated anti-examples with continuous data. As such, correlation does not always imply redundancy. Two highly correlated features taken together are shown to achieve a perfect class separation while providing poor separation taken independently. Moreover, a variable seeming useless by itself can provide to improve classification when taken with others.

All variables

For exploration purposes, all predictors have been evaluated concerning the target labels with first dependency measurement with the Mutual Information (MI) (Ross, 2014) from Scikit-learn and secondly with the RelieF algorithm from Scikit-Rebate. While MI ranks each predictor separately, RelieF takes into account interaction effects. For raw and scaled data and two or three classes of returns, these two procedures scored `KNN_Prediction` highly discriminative and negligible for the rest. This situation can also be observed with univariate and bivariate representations of the predictor (see Jupyter Notebook from Appendix A)

Quantitative variables

Despite high multicollinearity among quantitative predictors (see Figure 4.7), no filtering is attempted here as discussed above. An RFE will be applied during the predictive modelling process instead.

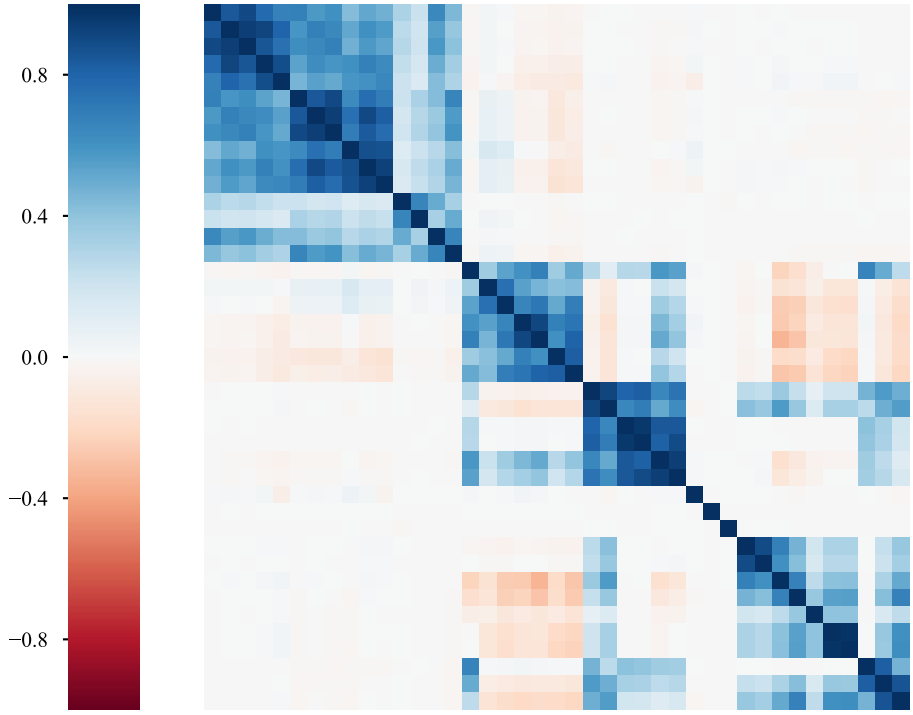

Figure 4.7 Visualization of the quantitative predictors' correlation matrix.

Qualitative variables

Redundancy among certain categorical variables can be assumed from the FAMD results. Instead of Cramer's V, the uncertainty coefficient (Theil, 1970) is used here as a measure of association between categorical variables. The uncertainty coefficient relies on information entropy and provides to be asymmetric in comparison to Cramer's V, with possible loss of

information. For two discrete random variables x and y, the uncertainty coefficient $U(x \mid y)$ is defined as[9]:

$$U(x \mid y) = \frac{H(x) - H(x \mid y)}{H(x)} = \frac{I(x,y)}{H(x)}$$

Where $H(x)$ is the Shannon entropy of a single distribution, $H(x|y)$ is the conditional entropy and $I(x,y)$ is the MI[10]:

$$H(x) = -\sum_{x} P(x) \ln P(x)$$

$$H(x \mid y) = -\sum_{x,y} P(x,y) \ln P(x \mid y)$$

Where $P(x,y)$ and $P(x \mid y)$ are the joint and conditional distributions. The first expression shows that the uncertainty coefficient is an MI normalised. Uncertainty coefficients between categorical variables are computed with the package Dython and can be visualised in Figure 4.8. The filter threshold is set to 0.8 and variables are removed according to Table 4.1. Exponential moving averages seems not to be statistically different from their simple moving averages counterparts.

Table 4.1 Association between qualitative variables.

Feature removed	High association with
Support_Resistance_20D	Support_Resistance_10D
	Channel_10D
	Channel_20D
Support_Resistance_10D	Support_Resistance_20D
	Channel_10D
EMA_30D	SMA_30D
EMA_50D	SMA_50D
EMA_100D	SMA_100D
EMA_200D	SMA_200D
3MA_60_120_200D	3MA_30_120_200D
	3MA_90_120_200D

[9] Wikipedia: https://en.wikipedia.org/wiki/Uncertainty_coefficient
[10] Wikipedia: https://en.wikipedia.org/wiki/Mutual_information

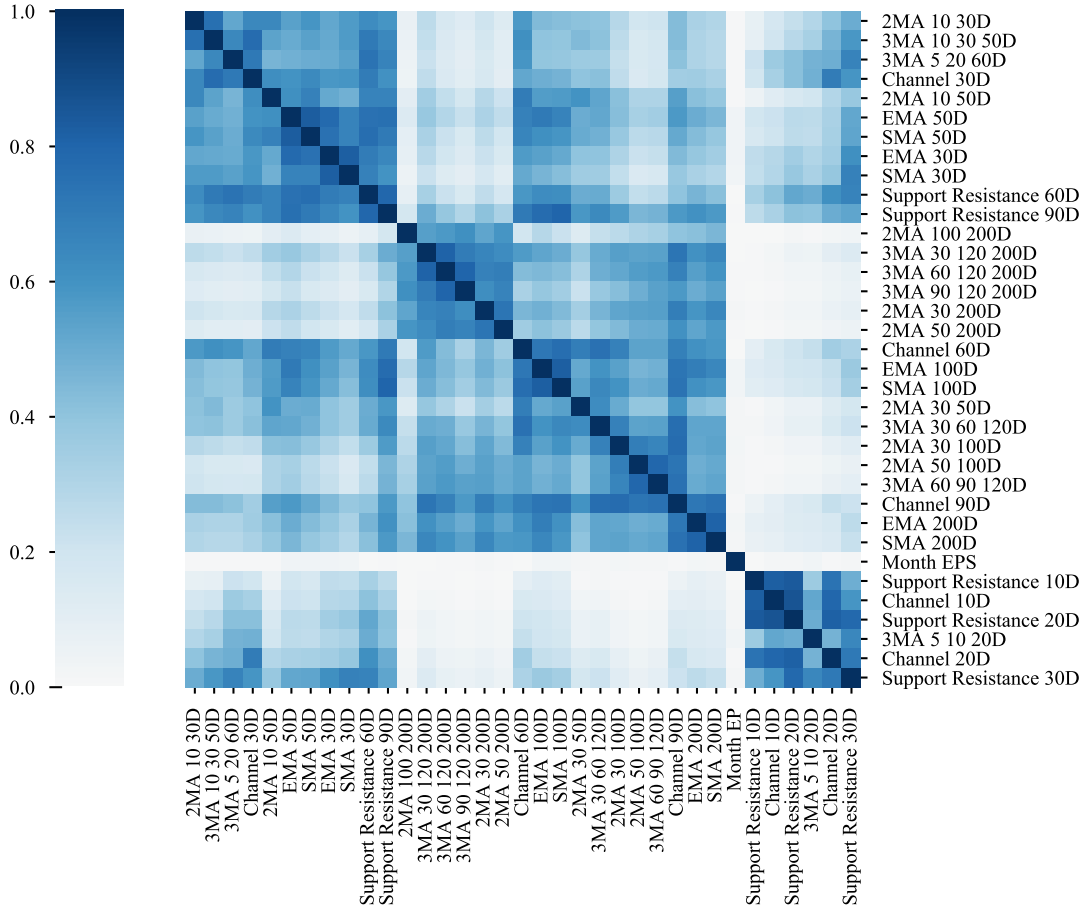

Figure 4.8 Visualization of the uncertainty coefficient matrix.

5 Predictive Modelling

5.1 Context

Forecasting stock returns can be translated into a classification problem where side returns define different classes. A third class for neutral cases can also be considered. Instead, the modelling is divided into two problems as presented in the *Meta-Labelling* approach of (López de Prado, 2018). The first modelling process focuses on predicting the side of the forecasted returns with targets {-1,1}. Neutral cases are dropped in this first step as it is assumed they would be assigned with low probabilities to both classes. Once sides are predicted, a second model re-assess the feature with side predictions to determine the size with targets {1,0}, that should be attributed to the bet. As such, the secondary model is expected to filter wrongly classified targets from the first model. This approach also reduces the likelihood of overfitting as models are trained for different targets. Figure 5.1 displays the meta-labelling approach.

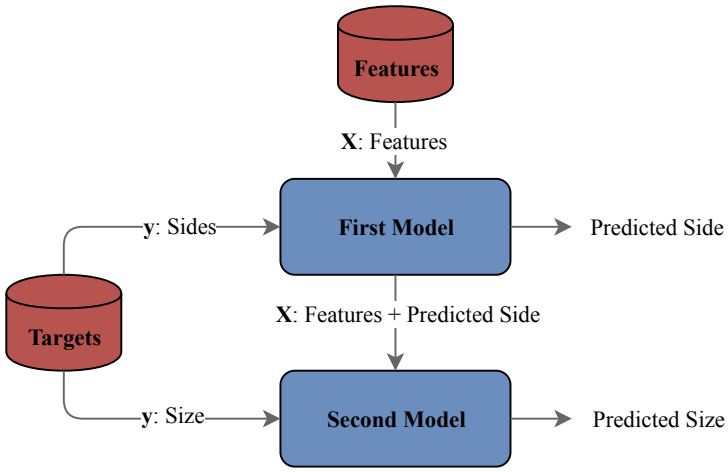

Figure 5.1: Graphical illustration of the meta-labelling approach.

Learning algorithms

As missing values and multicollinearity are still present in the dataset, early modellings are performed with DT algorithms for their non-parametric, distribution-free assumptions, and robustness to the presence of outliers and irrelevant attributes (Gama, Medas, & Rodrigues, 2004). DT models can be combined into Ensemble model (Yaliang Li, Gao, Li, & Fan, 2015) in order to obtain a better model than any individuals one. Three ensembling approaches can be distinguished:

- **Bagging**: Learners are created independently by resampling with replacement the dataset. The final forecast is the average of the learners' forecasts.
- **Boosting**: Learners are fit sequentially where subsequent learners address previous models errors. The final forecast is a weighted average of the learners' forecasts.
- **Stacking**: Learners forecasts are combined to train a meta-model to output a final forecast.

Bagging addresses overfitting while boosting addresses underfitting, at the cost of a greater risk of overfitting. López de Prado (2018) suggests the bagging approach for financial machine learning, due to greater concerns of easy overfitting because of the low signal-to-noise ratio. The two types of approaches are tested here. As such, Random Forest (RF) (Ho, 1995), Gradient Tree Boosting (GBT) (J. H. Friedman, 2001) and Dropouts meet Multiple Additive Regression Trees (DART) (Rashmi & Gilad-Bachrach, 2015) have been selected. GBT popularity is due to its immense success in online data science competitions, and a detailed analysis is provided by (Nielsen, 2016). A less formal description can be found here[11,12] and an intuitive description of RF can be found here[13]. Finally, DART is an extension of GBT (also called Multiple Additive Regression Trees (MART) according to authors) with an implementation of the concept of Dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012), designed to regularise and prevent overfitting for neural networks. Instead of the commonly used XGBoost (Chen & Guestrin, 2016), LightGBM is adopted here for its higher convergence speed in comparison to XGBoost and handling missing values as opposed to Scikit-learn.

Classification metrics

Major classification metrics, such as accuracy, sensitivity, specificity and others rely on the implicit assumption of a valid probability cutoff (in general 0.5). As the class separation is expected to be filled with noisy samples, cutoff-free metrics are used initially. As such, the Area Under the receiver operating characteristic curve (AUC) (Green & Swets, 1966; Japkowicz & Shah, 2011) is chosen to indicate the separation capacity of the classifier. For a

[11] A Kaggle Master Explains Gradient Boosting: http://blog.kaggle.com/2017/01/23/a-kaggle-master-explains-gradient-boosting/
[12] How to explain gradient boosting?: https://explained.ai/gradient-boosting/index.html
[13] Random Forests for Complete Beginners: https://victorzhou.com/blog/intro-to-random-forests/

random continuous variable x in a binary classification, AUC is defined as the integral of the Receiver Operating characteristic Curve (ROC)[14]:

$$AUC = \int_{x=0}^{1} TPR(FPR^{-1}(x))\,dx$$

Where $TPR(T)$ and $FPR(T)$ are the True Positive Rate and the False Positive Rate for a given threshold T:

$$TPR(T) = \int_{T}^{\infty} p_1(x)\,dx$$

$$FPR(T) = \int_{T}^{\infty} p_0(x)\,dx$$

Where $p_1(x)$ and $p_0(x)$ are the probability distribution for the positive and negative classes.

Additionally, the log loss or cross-entropy is employed for probabilistic confidence across classes. For N samples and C classes as, the log loss L is defined as:

$$L(Y,P) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j} \ln p_{i,j}$$

While AUC aims to maximise classes discrimination, the log loss is meant to penalise the divergence between actual and estimated probabilities.

The secondary model is optimized for F_1 for two reasons:

1. Relabelling has no impact on log loss.
2. There is more interest in focusing on prediction accuracy for the positive class, as compared in predicting both sides equally before.

F_1 score is defined as the harmonic mean of precision and recall

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

Where recall and precision are:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

[14] Wikipedia: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

As these metrics are not directly interpretable, benchmark values are computed for performances comparison. They represent the case of a classifier with random guesses for a balanced dataset:

$$AUC_{random} = \int_0^1 x\,dx = 0.5$$

$$L_{\text{random}_{2\text{ classes}}} = -\ln(1/2) \approx 0.69$$

$$L_{\text{random}_{3\text{ classes}}} = -\ln(1/3) \approx 1.10$$

$$F_{1random} = 0.5$$

The case for AUC represents the area under a diagonal curve. The full mathematical derivation can be found here[15].

Sample weight

In addition, Scikit-learn and LightGBM allow users to pass samples weights to learners fitting or to metrics. As higher absolute returns are defined as more important than negligible absolute returns, the sampling weight methodology of (López de Prado, 2018) is followed here, and sample weights are defined as the weighted absolute log returns at each timestamp. Logarithmic returns are chosen here due to their symmetric property.

Dataset shift

Machine learning algorithms have been formalised under the presumption of common data generating process between the training and the testing environment. In practice, this assumption often fails to hold, and when the data distribution is subject to change over time, it is subject of *dataset shift*. The academic literature has been using different terminologies to express the same concept (Moreno-Torres, Raeder, Alaiz-Rodríguez, Chawla, & Herrera, 2012), but such a shift may fall in three categories as presented initially by (Kelly, Hand, & Adams, 1999) for a set of predictors X and the class variable y:

1. **Covariate shift**: the input distribution $p(\mathbf{X})$ differs while the posterior distribution of the class memberships $p\left(\mathrm{y} \mid \mathbf{X}\right)$ remains unchanged.
2. **Prior probability shift**: the class prior distribution $p(\mathrm{y})$ differs while the posterior distribution $p\left(\mathbf{X} \mid \mathrm{y}\right)$ remains unchanged.

[15] StackExchange: https://datascience.stackexchange.com/questions/31872/auc-roc-of-a-random-classifier

3. **Concept shift**: the relationship between input and class variables, $p(\mathrm{y}\,|\,\mathbf{X})$ and $p(\mathbf{X}\,|\,\mathrm{y})$, differ while input and class distribution, $p(\mathbf{X})$ and $p(\mathrm{y})$, remains unchanged.

This paper's problem is naturally exposed to such shifts due to its temporal component. Markets are also known to be adaptive systems (W. Lo, 2017), and signal values are thus expected to keep changing. Developing algorithms to learn under dataset shift is an active area in machine learning research (Gama et al., 2004; Žliobaitė, 2010) and exploring the implementation of such learners is outside of the scope of this thesis. However, the following two adjustments are proposed and tested statistically to make our machine learning pipeline more adaptable:

1. **Sample weighting**: (López de Prado, 2018) time linear decay factors are used to multiply sample weights and to decay older ones. The convention of the author is reused here, and for a decay parameter $c \in (-1,1]$, the following cases can be observed:
    - $c = 1$ means no time decay.
    - $0 < c < 1$ means that weights decay linearly, but are still strictly positive.
    - $c = 0$ means that weights converge linearly to zero as they get older.
    - $-1 < c < 0$ means that the sample weights of the oldest c portion are set to zero and are consecutively erased from memory.
2. **Sliding window training**: Predictions are based on a learner periodically re-trained on a defined lookback window. Such methodology is standard in the financial machine learning literature. An example can be found in (Bao, Yue, & Rao, 2017)

These implementations can be seen in Figure 5.2, in the context of the data splitting, described below.

Data splitting

The dataset is broken in three sections, as suggested by (Hastie, Robert, & Jerome, 2009):

1. **Testing set**: defined as the last four years of the dataset, namely 16.5% of the entire dataset
2. **Training set**: defined as the first 70% of the remaining data, namely 58.5% of the entire dataset
3. **Validation set**: defined as the last 30% of the remaining data, namely 25% of the entire dataset

The testing set is locked until the entire model calibration process is finished. This prevents underestimation of the true model error and should provide a clean backtest for the last four years at least. Finally, the model is built on the training set, and prediction errors during the modelling process are calculated using the validation set. Figure 5.2 illustrates the splits used.

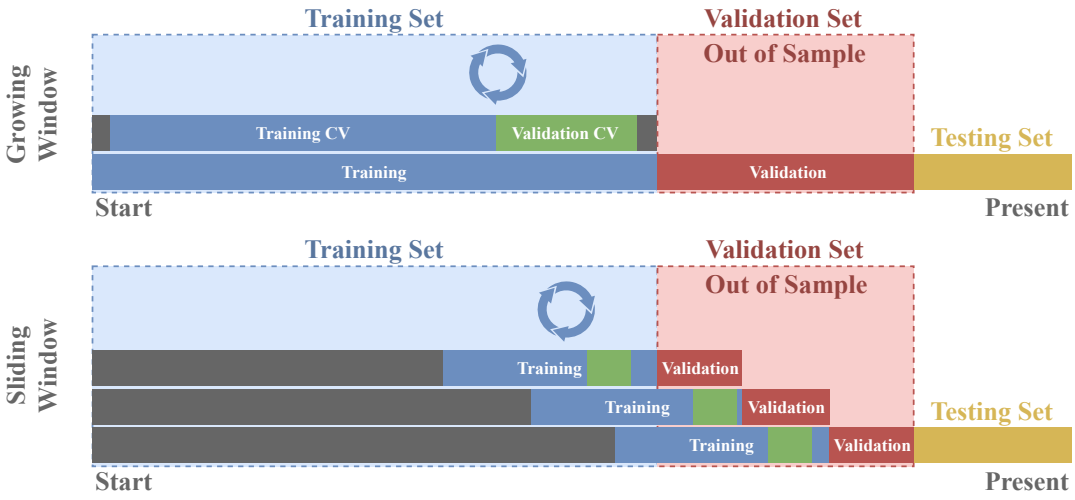

Figure 5.2: Graphical illustration of the data splitting and the training methodology used.

Top pannel: Growing window illustrates the case where all previous data is considered. Bottom pannel: The model is only trained on a specific lookback period.

Resampling

The commonly used k-fold Cross-Validation (CV) is known not to be suitable for time-dependent data. The in-sample performances are then estimated with Repeated Holdout validation (Rep-Holdout) motivated by the exhaustive comparison of resampling methods for time series from (Cerqueira, Torgo, & Mozetič, 2019). Due to serial correlation and data derived from overlapping points, a monthly purge is applied between the inner-training set and the inner-testing set (López de Prado, 2018). Rep-Holdout and the purging procedures are illustrated in Figure 5.3. If not mentioned otherwise, 30 resamplings are performed to estimate performances.

As observed by (López de Prado, 2018), Scikit-learn cross-validation implementations do not pass sample weights to the scoring function, but only to the estimator class. This issue has still not been solved as of the time of this paper[16] and is common among other major Python machine learning libraries due to their dependency to Scikit-learn. The author's solution is

[16] Github: https://github.com/scikit-learn/scikit-learn/pull/13432

followed, and a function estimating weighted score by repeated holdout validation has also been recorded here.

Figure 5.3: Graphical illustration of the Repeated Holdout validation.

For defined percentages of data assigned to the inner-training and inner-validation set, the remaining data available defines a window where sets can be randomly splitted. The gap between sets represents the training set being purged.

Tuning parameters

Because of the high complexity of LightGBM's parameters tuning[17], a grid-/random search (Bergstra & Bengio, 2012) would be computationally too demanding. Instead, Bayesian optimisation frameworks are chosen. Initially, with Hyperopt, hyperparameters search has been finally performed with Optuna. They both rely on a Tree-structured Parzen Estimator approach (TPE) (Bergstra, Bardenet, Bengio, & Kégl, 2011), but Optuna proved to be more performant due to its ability for parallel computing and pruning of unpromising trials. To enable large scale distributed optimisations with Optuna, a PostgreSQL database has been used. Unless mentioned otherwise, 200 trials are performed for each model.

Statistical tests for model comparison

The effective practice of statistical tests to compare classifiers has been controversial in the machine learning research space, mainly due to the use of multiple testing without significance tests correction or the use of parametric tests which assumptions cannot be proven to hold (Benavoli, Corani, Demšar, & Zaffalon, 2017; Japkowicz & Shah, 2011). Tests results presented in this thesis follows methodologies from (Demšar, 2006; Japkowicz & Shah, 2011; Salzberg, 1997). As such, monthly performance metrics are drawn from the validation set, meaning the learning algorithms has not been trained nor tuned with them. Then, the non-parametric Wilcoxon signed-ranks test, (Wilcoxon, 1945) and Friedman test (M. Friedman, 1937) are performed to compare two and more classifiers respectively. Finally, whenever the

[17] Description of all parameters in LightGBM: https://lightgbm.readthedocs.io/en/latest/Parameters.html

null hypothesis of the Friedman test is rejected (H_0: *no significant difference between models performances*), post hoc tests can be proceeded to identify which classifiers actually differ. For its complementarity with the Friedman test, the Nemenyi test (Nemenyi, 1962) is chosen here as a post hoc test.

Feature selection

Despite not affecting the predictive performances of DT algorithms, redundant predictors tend to dilute the feature importance scores (Kuhn & Johnson, 2019). In addition to the previous reasons mentioned, feature selection should also be applied in order to correctly interpret such scores. For this process, RFE is chosen for its complementarity with DT algorithms and its efficient screening of correlated variables (Gregorutti, Michel, & Saint-Pierre, 2017). RFE is a greedy Sequential Backward Selection (SBS) (Ferri, Pudil, Hatef, & Kittler, 1994) of predictors for learners with weight coefficients or importance scores. For tree-based models, RFE primarily works by eliminating features recursively using feature importance, whereas SBS eliminates features based on a user-defined classifier/regression performance metric. These processes need to be paired with external resampling as they could easily overfit.

RFE is performed with `RFECV`[18] from Scikit-learn and SFS with `SequentialFeatureSelector`[19] from Mlxtend. As before, these objects cannot pass sample weights to the scoring function. While sample weights can be passed to the learner for `SequentialFeatureSelector`, `RFECV` does not accept any sample weights. Due to their computational costs and the fact these implementations are already optimised for parallel computing, recoding them including a weighted scorer has been considered out of this thesis scope. Instead, both implementations are used and tested on the separate validation set with weighted scorer.

Furthermore, these computational complexities rise another issue concerning selection bias. CV methods are unbiased if all the aspect of classifier training takes place inside the CV (Hastie et al., 2009; Kuhn & Johnson, 2013). Ideally, it means that feature pre-processing, feature selection, classifier type selection and classifier parameter tuning should be performed co-jointly in each CV loop. Due to the size of the dataset and the complexity of this paper machine learning pipeline, such procedures cannot be considered. Instead, we rely on the more pragmatic

[18] RFECV: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html
[19] SFS: http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

approach from (Kuhn & Johnson, 2019) and here[20]. The following assumptions are formulated to mitigate selection bias:

- Precedent feature selections, i.e. filtering methods, have been done in an unsupervised way, meaning not taking into account targets, and should introduce any selection bias (Hastie et al., 2009).
- Learners less sensitive to the choice of tuning parameters might not need to be re-tuned in a nested CV. (Kuhn & Johnson, 2019) used a Random Forest (RF) model for their application of RFE and thus LightGBM's GBT is considered similarly less sensitive as opposed to Logistic Regression (LR) or Suppor-Vector Machine (SVM) (Corinna Cortes & Vapnik, 1995).

A selection of promising features subsets and hyperparameters would be tested on the validation set. Finally, to study the impacts of correlated predictors and of the scorer in the selection process, the exercise of (Kuhn & Johnson, 2019) is repeated here. RFE and SBS are performed for AUC and log loss for three initial feature sets defined as:

1. All predictors.
2. Predictors filtered for correlation (threshold set at 0.8).
3. Predictors filtered for multicollinearity with dependency estimation with RF[21].

In order to identify dependency among features, each feature can be regressed on the remaining ones with RF. Consecutively for each regression, the corresponding out-of-bag R^2 indicates the independent variable dependency, or how it can be easily predicted. Regarding the dependent variables, their importance scores can be interpreted as their predictive powers. Finally, dependent variables with importance scores close to one across several regressions can be removed. The author's package, `Rfpimp`[22], is used to illustrate this filtering.

[20] StackExchange: https://stats.stackexchange.com/questions/264533/how-should-feature-selection-and-hyperparameter-optimization-be-ordered-in-the-m
[21] Dealing with collinear features: https://explained.ai/rf-importance/index.html#corr_collinear
[22] Rfimp: https://github.com/parrt/random-forest-importances

5.2 Primary Model

As in the previous chapter, neutral cases, i.e. label 0, are momentarily dropped, leading to a binary classification problem with labels {-1,1}. Neutral cases are expected to be later predicted to both classes with low probability, allowing the second model to filter them out. In the next pages, different training configurations are compared. Classifiers are judged based on their capacity to distinguish return sides. For this purpose, performances are measured with non-/weighted log loss and AUC on predictions made on the validation set. In addition, each configuration hyperparameters have been retuned for weighted log loss.

Window training

As a first comparison, performances on the validation set are compared for models trained with growing windows and sliding windows. In order to describe training methodologies, the terminology of (Cerqueira et al., 2019) is referred to this study. The term growing window describe including all previous data for training while sliding window includes a fixed size look-back window for training. The following models are considered:

- Growing window training for time decay factor $c \in \{1, 0.75, 0.5, 0, -0.25, -0.5\}$
- Sliding window training for windows of 1, 2, 4 and 8 years.

Each model has been optimized for weighted log loss with internal Rep-Holdout validation. Figure 5.4 displays the statistics metrics computed over months on the validation set. All models seem to have the same level of class discrimination. However, growing windows models seem to have a slight advantage from a probabilistic view. Friedman tests have been then performed for the four metrics, and the null hypothesis is rejected. Nemenyi tests then proceed and respective p-values are displayed in Figure 5.5. Across tests, a significant difference can generally be found between growing window and sliding window models, especially for weighted log loss, the optimised metric. For the rest of the study, sliding window models are then not considered anymore. In addition, to slightly lower performances, they are computationally expensive to optimise as parameters search have to be redone at each training slide.

Missing values

So far, missing values have been dropped. Their value on performances metrics is re-assessed here as LightGBM handles them. In a first time, samples with missing data are added back to the dataset, while the implied volatility features are added in a second step. New models are

compared to the previous growing training models. Figure 5.6 displays the statistics metrics computed over months on the validation set and related p-values from Nemenyi tests are displayed in Figure 5.7. While non-weighted performances slightly decreased, weighted log loss show to be lower and have less variance. P-values seem also to indicates statistical differences between models with and without missing values. As we are more interested in better probabilities for returns with higher weights, missing values are therefore kept.

Rolling preprocessing

When stacking features, (López de Prado, 2018) recommends to standardize them on a rolling training window to ensure some distributional homogeneity in the dataset and numerous equity strategies of (Kakushadze & Serur, 2018) rely on ranking stocks. Ranking is also a way to mitigate outliers. Finally, as we perform cross-sectional analysis, rolling scaling could be suitable. Thus, models trained on monthly pre-processed feature are compared for:

- Ranking.
- `RobustScaler()`, as outliers are expected.
- `PowerTransformer()`, as skewness is expected.

Figure 5.8 displays the statistics metrics computed over months on the validation set and related p-values from Nemenyi tests are displayed in Figure 5.9. Monthly means and variances have both improved for weighted and non-weighted metrics, confirmed by the Nemenyi tests. `RobustScaler()` is kept as it is computationally more efficient than `PowerTransformer()` and to preserve the data distribution if it had to be discriminating for the secondary model.

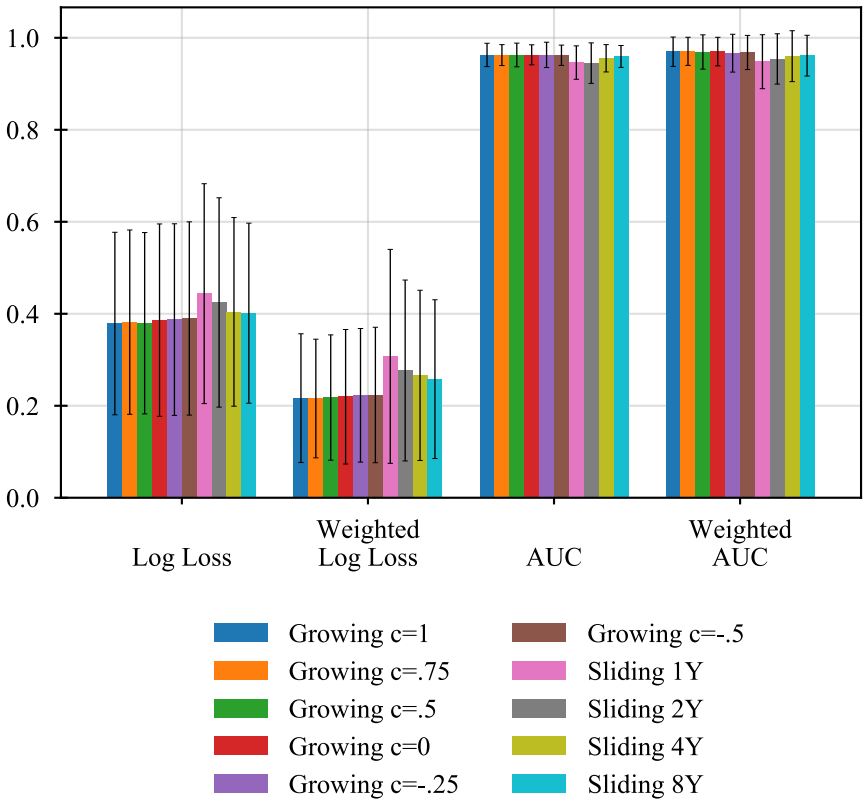

Figure 5.4: Comparison of training methodologies: Means and standard deviations of metrics computed monthly

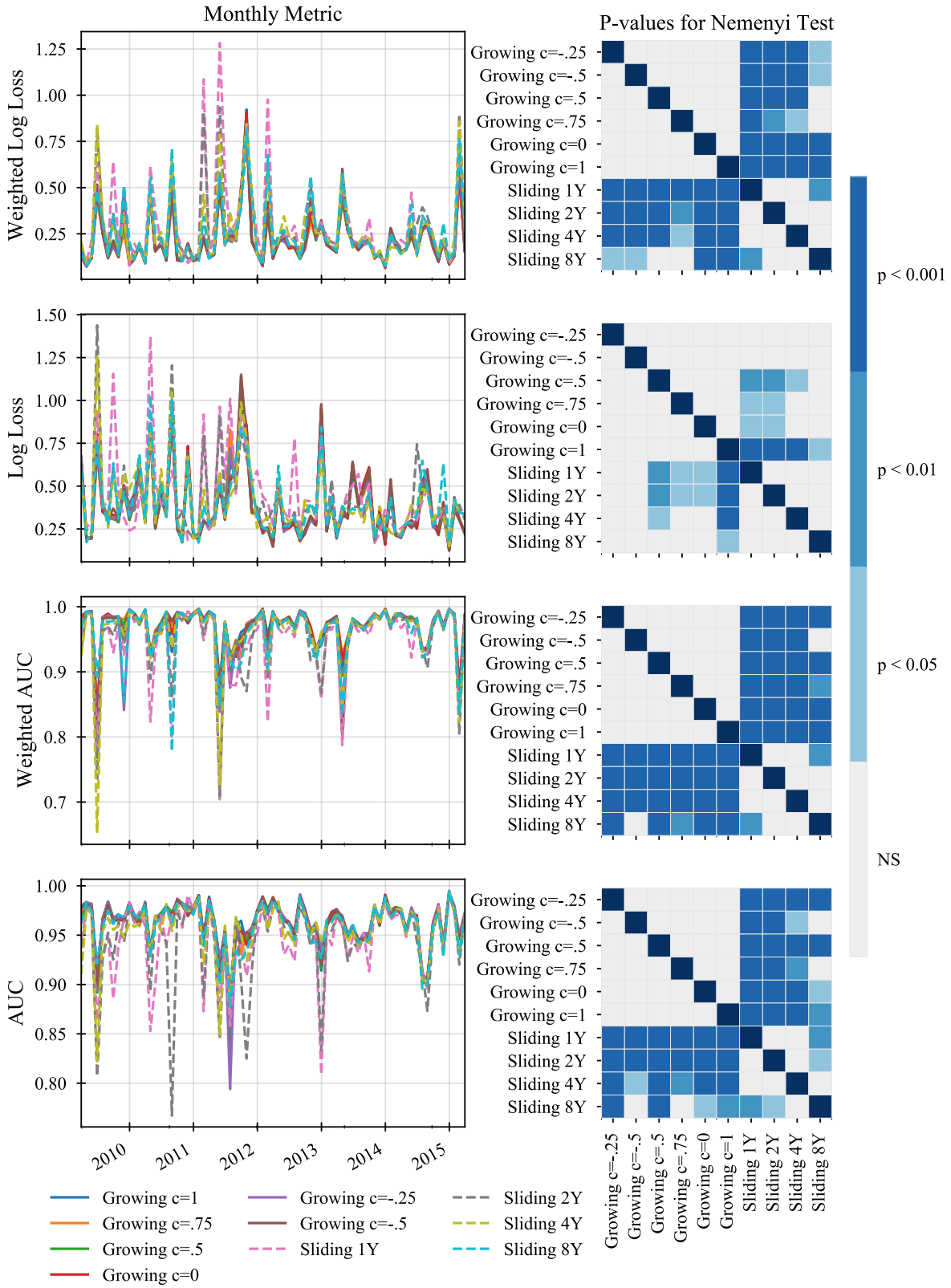

Figure 5.5: Comparison of training methodologies.

Left pannel: Monthly metrics time series. Right pannel: Visualization of p-values from Nemenyi tests.

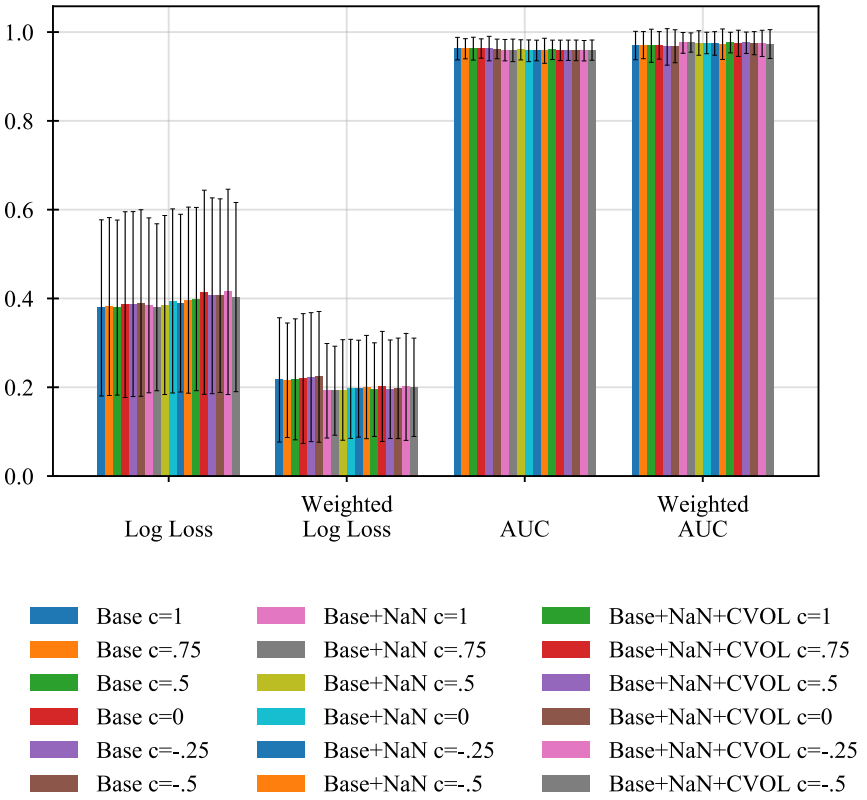

Figure 5.6: Comparison of training with missing values: Means and standard deviations of metrics computed monthly.

Base models refers to previous growing window models.

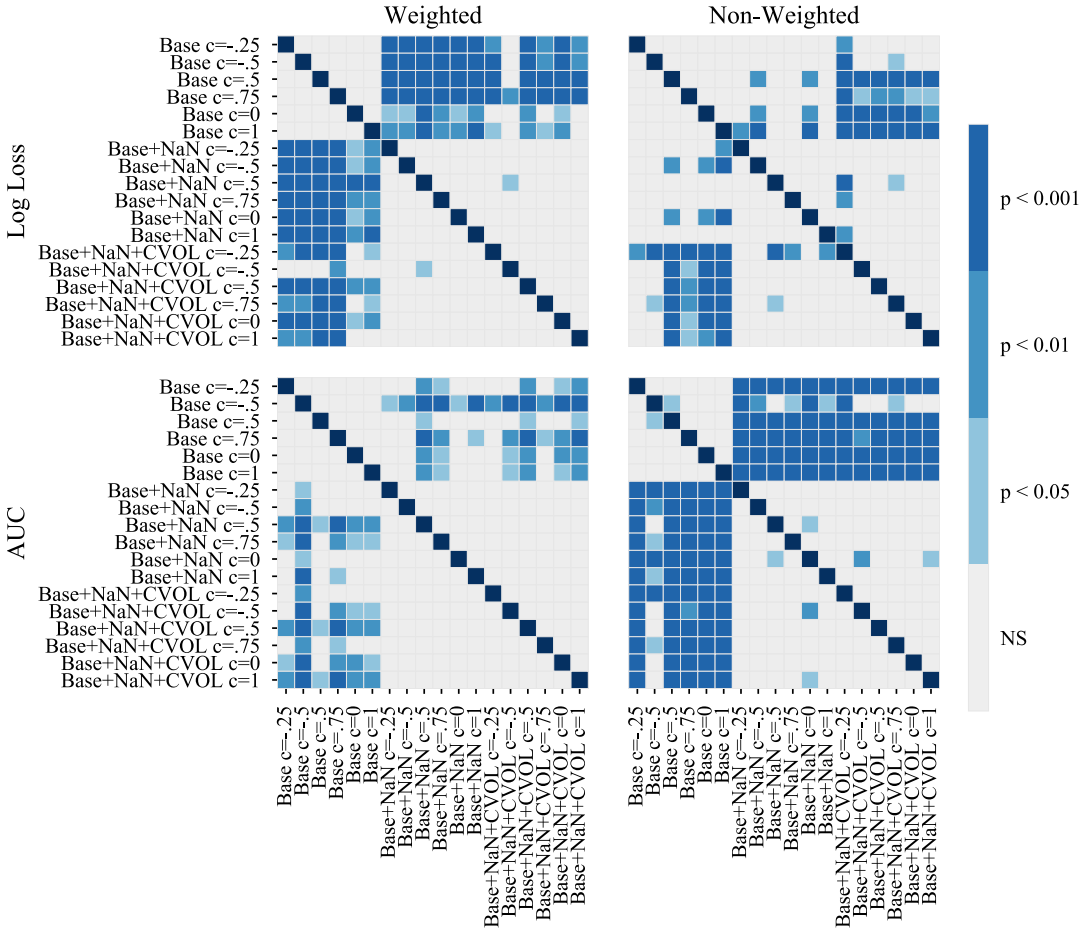

Figure 5.7: Comparison of training with missing values

Visualization of p-values from Nemenyi tests. Base models refers to previous growing window models.

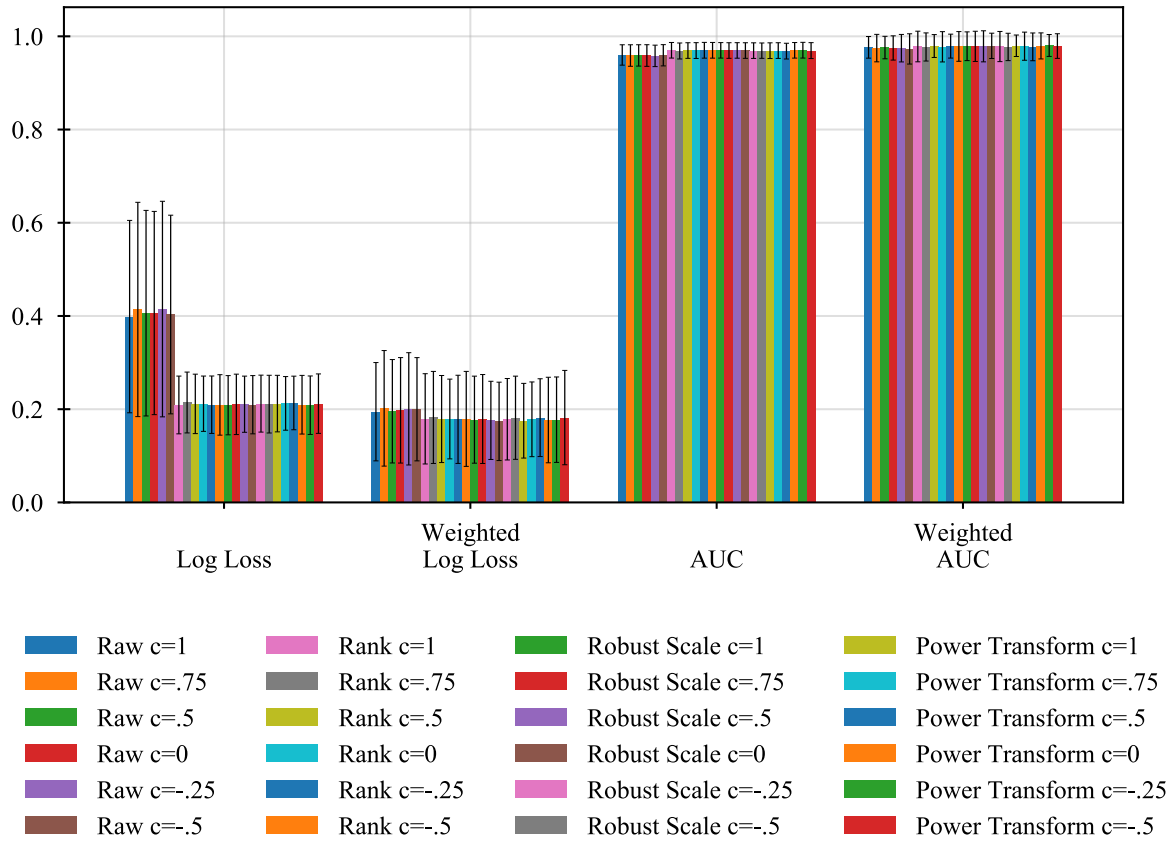

Figure 5.8: Comparison of training with rolling pre-processed data: Means and standard deviations of metrics computed monthly.

Raw models refer to previous models with raw data including all missing values and predictors

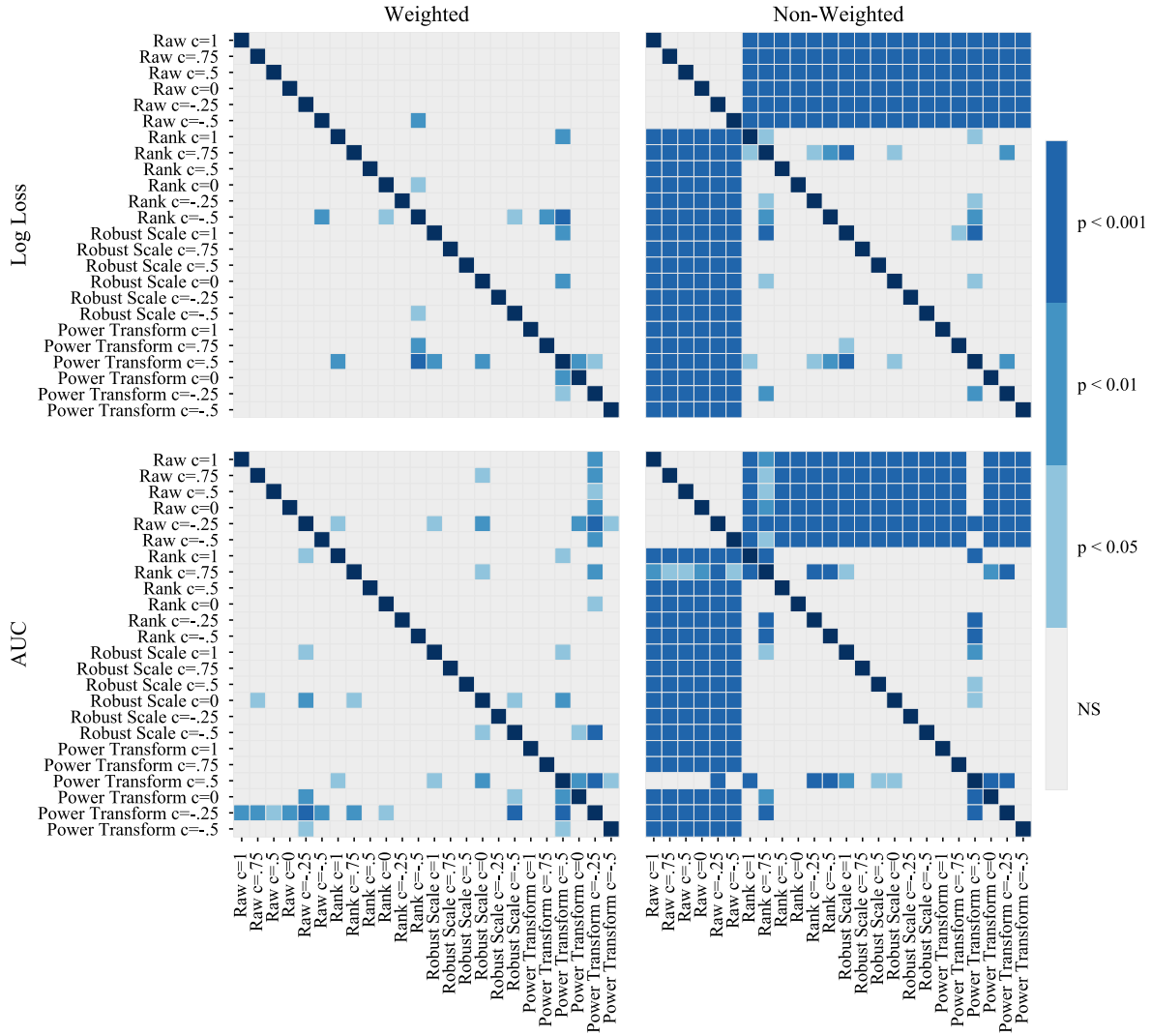

Figure 5.9: Comparison of training with rolling pre-processed data:

Visualization of p-values from Nemenyi tests. Raw models refer to previous models with raw data including all missing values and predictors

5.3 Feature Selection

The training methodology is defined so far as:

- Growing window
- Keeping missing values
- Rolling pre-processing

We are interested in removing irrelevant features and associated noise from the models. As described earlier, RFE and SFS are performed on all feature and two different feature subsets, previously filtered for correlation and multicollinearity. Results displayed in Figure 5.11 shows that GBT benefits from feature interaction between correlated features. Bests subsets proposed by RFECV and the three subsets that achieved best scores on SFS are selected for further investigation. Figure 5.10 displays selection percentage of predictors for theses subsets. Finally, these subsets are used for training and validation performances are reported in Figure 5.12. The subset `RFECV LL all` with 21 predictors is selected at this one the only having a lower mean and lower variance for weighted log loss compared to the base case.

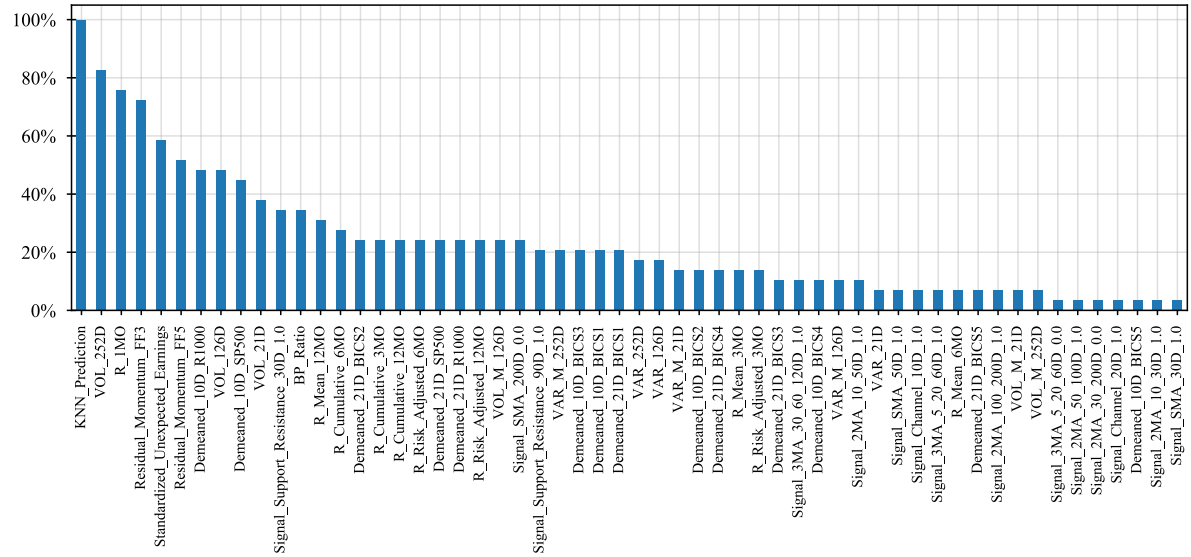

Figure 5.10: Predictor selection percentage for selected feature subsets.

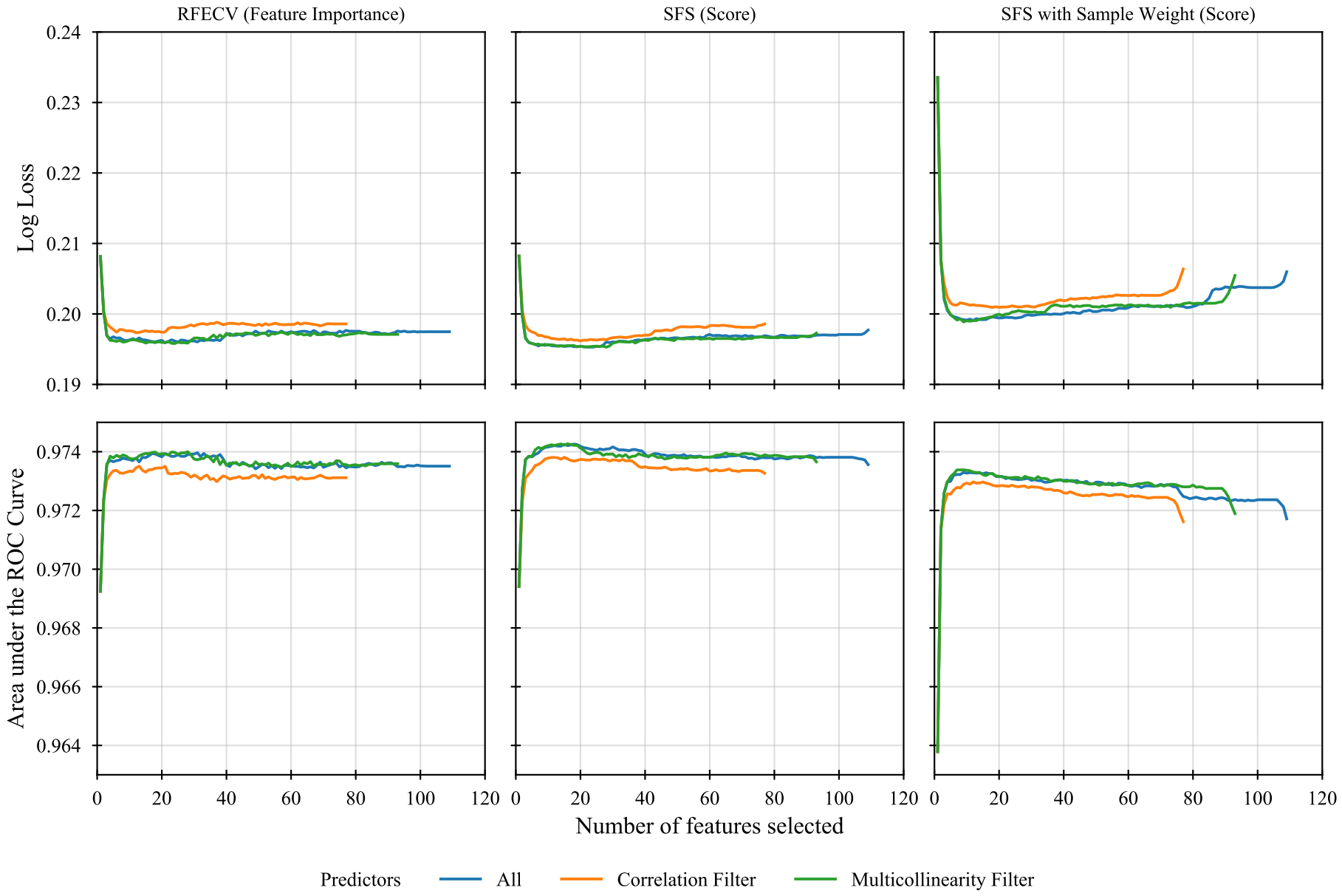

Figure 5.11: The cross-validated results for RFE and SFS using GBT.

Left pannel: RFECV selecting subset based on GBT feature importances. Middle and Right pannels: SFS selecting subset based on AUC/ LogLoss Score. Right pannels: GBT trained with sample weight.

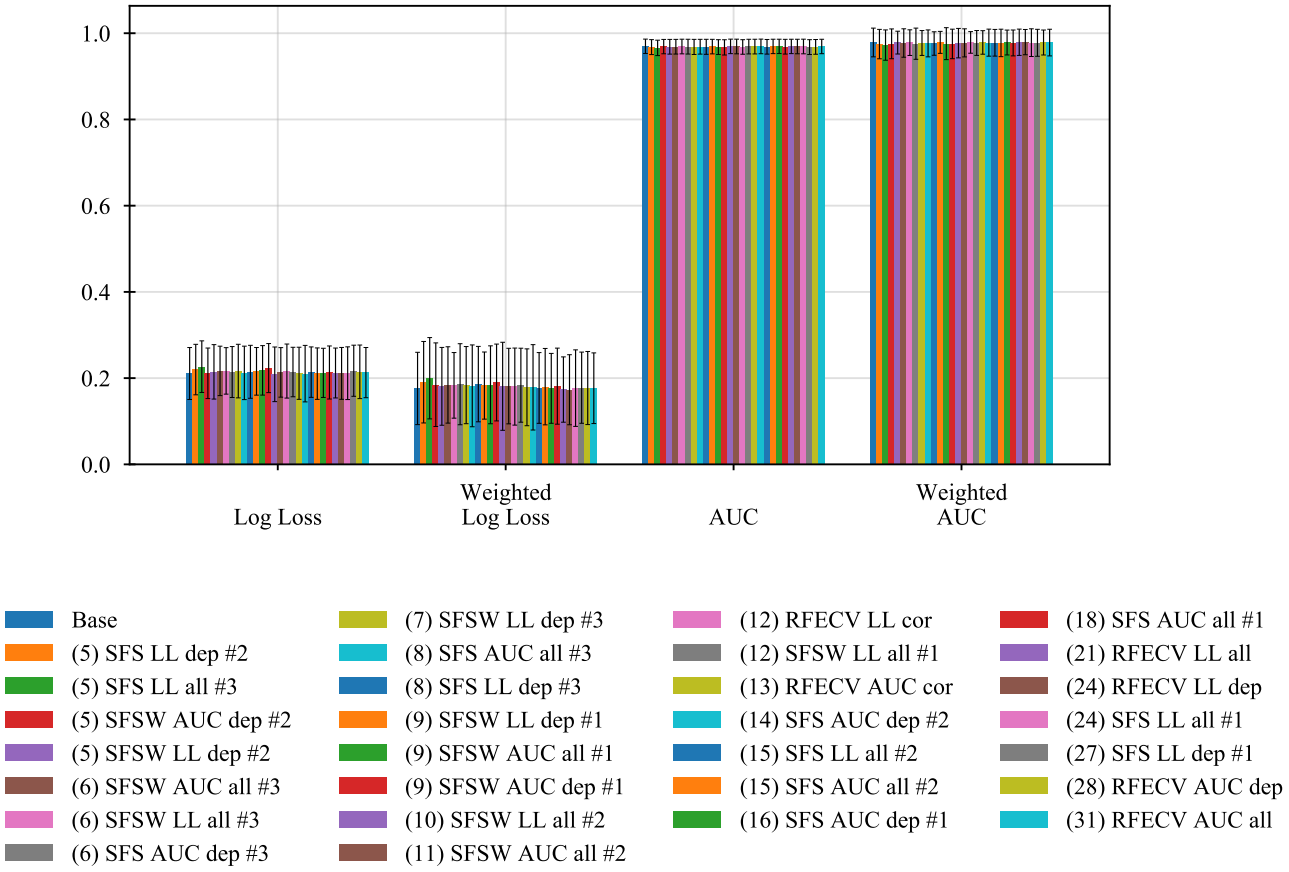

Figure 5.12: Comparison of training with different feature subsets: Means and standard deviations of metrics computed monthly.

Base models refers to model with RobustScaler() and a time decay factor of 0.5. Value in brackets represents the number of predictors included in the subset.

5.4 Data Leakage

Context

As early backtesting with the previously mentioned models, lead to extraordinary performances for a monthly strategy (Sharp Ratio above 5), the process modelling has been interrupted to investigate the possibility of data leakage. After investigations across the entire pipeline, it has been deduced that the predictions from the *k*-NN were probably the suspects for such leakage. This chapter is meant then to describe and to emphasize the subtility of data leakage (Kaufman et al., 2012) in financial machine learning application. It was enough to unintentionally leak data at one step to compromise the entire pipeline and invalidate results, despite rigorous procedures along the process:

1. **Feature Generation**: inclusion of delisted index components; usage of PIT data; rolling prediction for *k*-NN; alignment of sample features and targeted returns with one day gap; Treatments for missing values and outliers not considered;
2. **Feature Engineering**: Early feature selection without taking targets into account.
3. **Predictive Modelling**: Dataset splitting; Cross-Validation with one month purging on training set; Results validation with validation set; Locking of the testing set; Rolling feature scaling.

Leakage in financial application occurs when targets from training and testing sets are derived from overlapping points (López de Prado, 2018). The returns derivation illustrates the issue as they are defined from two different price points. Let y_t be the targeted s-days returns for two timestamps assigned to the training and testing sets as $t = t_1, t_2$:

$$y_{t_1} = \frac{P_{t_1+s}}{P_{t_1}} - 1$$

$$y_{t_2} = \frac{P_{t_2+s}}{P_{t_2}} - 1$$

For $t_2 = t_1 + 1$ and then $P_{t_1+s} = P_{t_2+s-1}$, information leakage necessary occurs as prices are shared. Another way more intuitive way to interpret is: By the time one should predict y_{t_2}, the estimator could not have been trained with y_t for $t = t_2 - 1, ..., t_2 - s$ as related prices were not

available yet. In order to reduce the likelihood of leakage (López de Prado, 2018) suggests two approaches:

1. **Purging and Embargoing the training set**: Removing data from the training set to ensure non-overlapping data between training and testing sets.
2. **Avoid overfitting**: Impeaching the classifier to profits from leakage even if it occurs. See (López de Prado, 2018) for description of adapted classifiers.

Regarding this paper, the prediction methodologies used for the *k*-NN strategy and the main framework are fundamentally different. The *k*-NN makes daily predictions of the return over the next 20 days (with one day gap) while the main framework makes monthly predictions only. In that sense, the main framework methodology is bias-free as we purge the last month of the training set. For the *k*-NN case, the situation is different as returns are continuously overlapping (see Figure 5.7). As the *k*-NN predictions were initially added to the feature set, leakage could not be prevented in the modelling process, even by locking the testing subset.

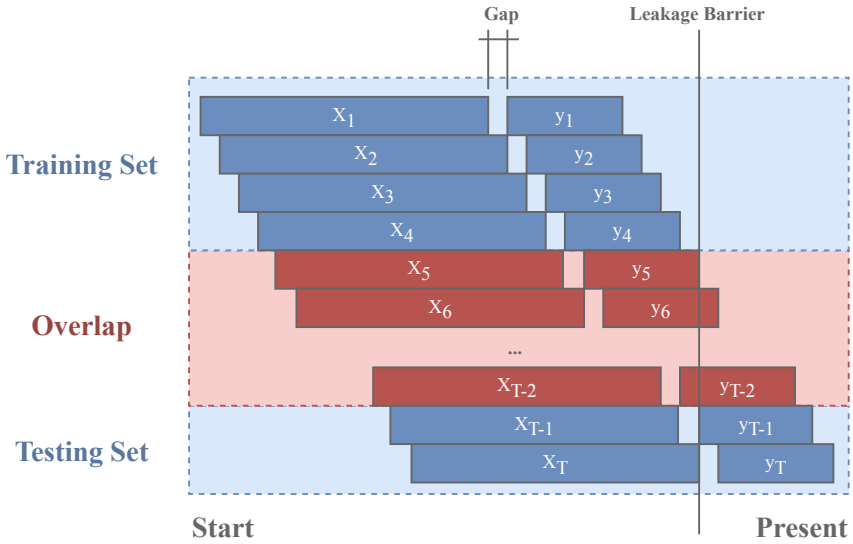

Figure 5.13: Graphical illustration of data leakage.

Lenghts of rectangle represent the timerange needed to compute feature and targets. The leakage barrier represents all training returns sharing data with the testing returns.

The situation could have been avoided with:

1. Larger purging or using a different framework for *k*-NN predictions.
2. Using one of the classifiers recommended by (López de Prado, 2018).

Due to its nature, data leakage is not trivial to detect and eliminate (Kaufman et al., 2012). The issue seemed obvious from backtests results. However, after reflexions, it could have been spotted earlier as constant flags were raised along the process:

1. **Feature Generation**: High performances for the regression problem (see Appendix F)
2. **Feature Engineering**: High discrimination power of the leaking feature in comparison to the others.
3. **Predictive Modelling**: High performances of learners; Constantly ranked first by feature selections and High feature importances attributed by models.

Redefined Pipeline

For ethical and scientific reasons, it has been decided to re-start the modelling processes performed until now. As data leakage was spotted in the latest days of this thesis, the primary model has been designed in a shorter way:

- *k*-NN have been retrained and forecasts re-computed with both a purging period of 21 days
- Models, pre-processing and hyper-tuning selection have been performed within one step with cross-validation.
- Feature selection is skipped due to the nature of DT models
- Analysis from Chapter 4, should remain robust for the rest of the predictors.

The cross-validation performances of the redesigned primary model dropped terribly to $L_{weighted} \simeq 0.55$. As it is not far from the performances of classifier with random guesses, a second pipeline has been designed parallelly with the primary model optimized for weighted recall. As described in (López de Prado, 2018), this approach aims to classify the most possible number of positive labels at the expense of false-positive increase. False-positive are then expected to be filtered by the secondary model.

5.5 Pipeline Performances

After re-configuring pipelines and optimizing them for their own metrics, the pipeline optimized for log loss selected GBTs for both models while recall selected an RF as primary and a GBT as a secondary model. Performances are compared on the validation set as observable from Figure 5.14. First, secondary models for both models seem to share some performances. Very close hyperparameters have been selected for both of them, interrogating the discrimination capacity of the first two models, as performances obtain very close to the equivalent of random guesses. Due to higher classification performances of secondary models in comparison to the first ones, it raises the issue of accepting a bet that has been predicted for the wrong side.

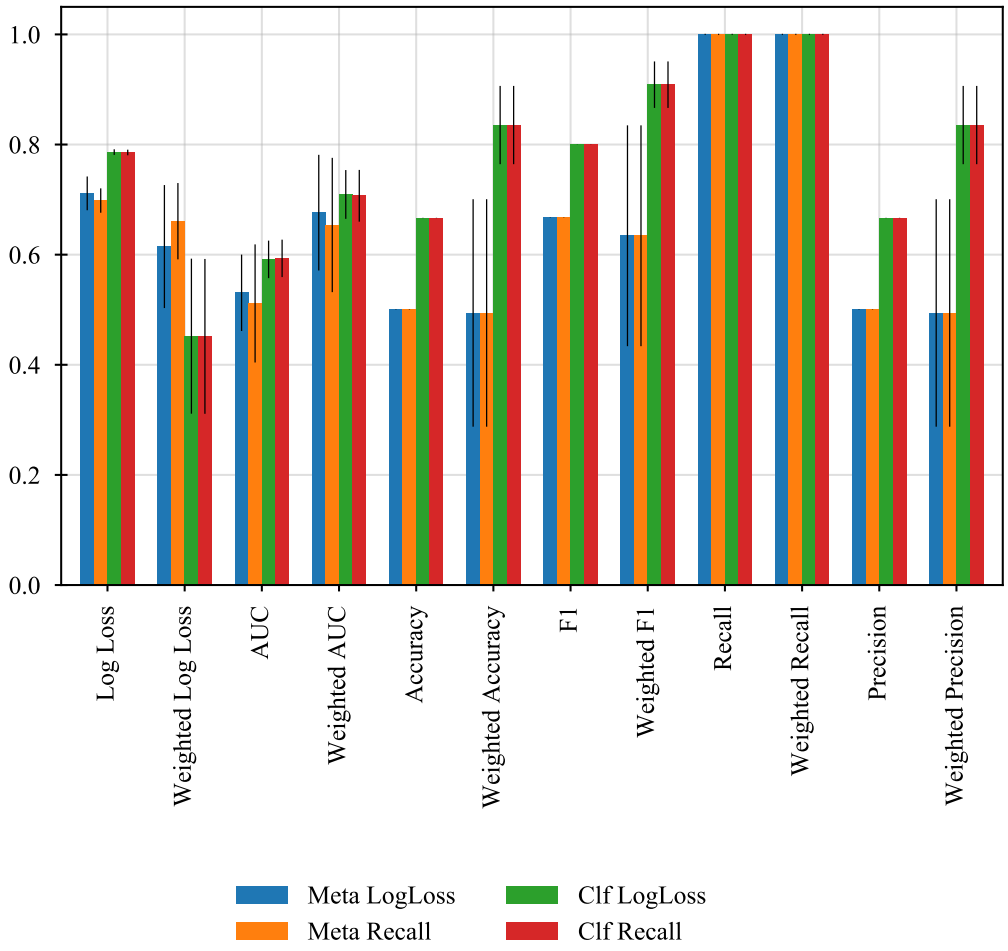

Figure 5.14: Comparison of pipeline performances on the validation set: Means and standard deviations of metrics computed monthly.

Meta refers to the primary model outputting meta-labels and *Clf* refers to the secondary model. The metric associated determines the pipeline for which the first model has been optimized for.

6 Portfolio Allocation and Portfolio Analysis

6.1 Context

Once ML forecasts have been outputted, it remains to convert them into trading signals and backtest the strategy. For these tasks, the in-sample and out-of-sample time ranges are defined respectively as:

- **In-Sample**: Training and Validation subsets defined in the predictive modelling process, i.e. from the 31st January 1995 to the 31st march 2015.
- **Out-of-Sample**: Testing subset also defined earlier and locked until now, i.e. from the 30th April 2015 to the 29th March 2019.

An Equity Long-Only Investment Strategy is attempted as shorting stocks are not relevant in a monthly setting. Finally, portfolio statistics include trading costs of 0.25% and computations used are presented in Appendix G.

Signals

Weights are defined from the combination of sides predicted earlier from the primary model and the size probabilities from the secondary one, with the *Bet Sizing* approach presented by (López de Prado, 2018). From the output of the secondary model, let define $p(x)$ as the probability that the label $x \in \{-1,1\}$ takes place. By computing testing the null hypothesis $H_0 : p(x=1) = \frac{1}{2}$ and the test static z:

$$z = \frac{p(x=1) - \frac{1}{2}}{\sqrt{p(x=1)(1-p(x=1))}} \sim Z, \text{ with } z \in (-\infty, \infty)$$

The bet size m is defined as:

$$m = 2\Phi(z) - 1 \text{ with } m \in (-1,1)$$

Where Z represents the Standard Normal distribution and Φ is the corresponding CDF.
Final signals can be obtained by multiplying the bet size with the predicted side.

Backtesting through Cross-Validation

The signals obtained needs to be translated to allocation weights as a long-only strategy assumes:

$$\sum_i w_{i,t} = 1 \text{ with } w_{i,t} \geq 0 \quad \forall t$$

Where $w_{i,t}$ stands for the normalized weight assigned to stock i at the timestamp t. As we are interested in the bets with the highest probabilities, the n highest signals are selected and normalized:

$$w_i = \frac{m_i}{\sum_i m_i} \text{ where } i \in I^*$$

Where I^* denotes the set of indices associated with the n highest signals. To prevent excess turnover induced by small weight differences at every prediction, the weights are further discretized:

$$w_i^* = \text{round}\left[\frac{w_i}{d}\right] d \text{ where } d \in (0,1]$$

The parameters n and d are determined by optimizing the Sharp Ratio strategy due to the pitfall of out-sample performances of strategy optimized for return only as observed by (Wiecki, Campbell, Lent, & Stauth, 2016). To reduce the likelihood of backtest overfitting and obtain the highest *true* Sharpe ratio possible, we rely on the Combinatorial Purged Cross-Validation (CPCV) (López de Prado, 2018). It recombines groups of observations into training and testing sets where learners can be applied to the forecast the testing sets. Finally φ backtest paths can be recombined from the forecasts and allows to derive an empirical distribution of the Sharp ratio. Let T observations be partitioned into N groups without shuffling of size $\lfloor T/N \rfloor$; k is the number of groups assigned to the testing set, the number of possible training/testing splits φ_{split} is:

$$\varphi_{split}[N,k] = \binom{N}{N-k} = \frac{\prod_{i=0}^{k-1}(N-i)}{k!}$$

The k testing sets from each combination can be re-grouped into φ_{path} paths to be backtested:

$$\varphi_{path}[N,k] = \frac{k}{N}\binom{N}{N-k} = \frac{\prod_{i=1}^{k-1}(N-i)}{(k-1)!}$$

From there, the empirical distribution of the strategy's Sharpe ratio can be derived.

6.2 Results

Empirical Distribution of Strategies' Sharpe Ratio

The CPCV is applied on the in-smaple dataset yo determine trading parameters. From the in-sample time range, $T = 243$ months can be observed. To let the k groups timerange matches the out-of-sample time range (~4 years), monthly samples are assigned to $N = 15$ groups with size of 16 months and $k = 3$. Finally, we obtain the number of splits and backtest paths to be computed:

$$\varphi_{split} = 455$$
$$\varphi_{path} = 91$$

The following trading parameters have been obtained by optimising the Sharpe Ration from the CPCV:

$$n_{\text{logloss}} = 26 \quad d_{\text{logloss}} = \frac{1}{1066}$$
$$n_{\text{recall}} = 101 \quad d_{\text{recall}} = \frac{1}{6363}$$

Empirical Sharpe Ratio distribution can be plotted as follow:

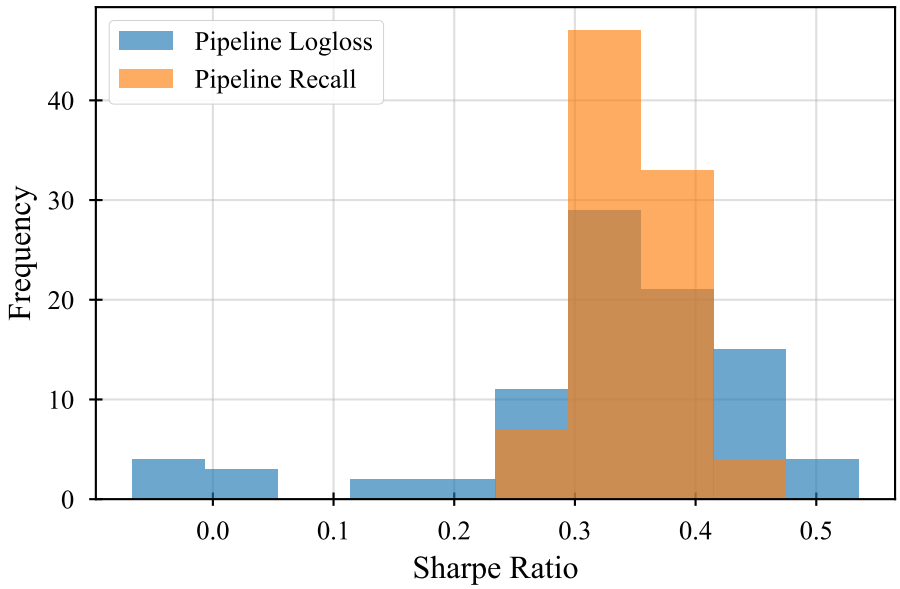

Figure 6.1 Empirical Sharpe Ratio distribution for strategies

Backtest Statistics

Strategies are backtested with selected parameters. Cumulative returns for strategies and benchmarks are reported in Figure 6.2. Cumulative returns Ratios of the strategies over the Russell 1000 Index can be visualized in Figure 6.3. Finally, Strategy performances are reported in Table 6.1. Despite similar Sharpe Ratio, the LogLoss pipeline managed to remain more profitable than the benchmark including trading costs while trading costs killed the performances of the Recall pipeline. By a higher differential between average returns from hits and misses, the log loss pipeline manages to recover quicker from drawdowns.

From the trading parameters obtained below, we retrieve the effects from the metrics. As the log loss pipeline seems more confident on its bets, he selects only 26 stocks per time, leading to a higher turnover. The recall pipeline has however kept on increasing false positives and must pick a higher number of stocks per time to reach profitability, leading to lower turnover. Finally as empirical distribution and strategy performances do not correspond, we can question if such performances are not purely due by luck.

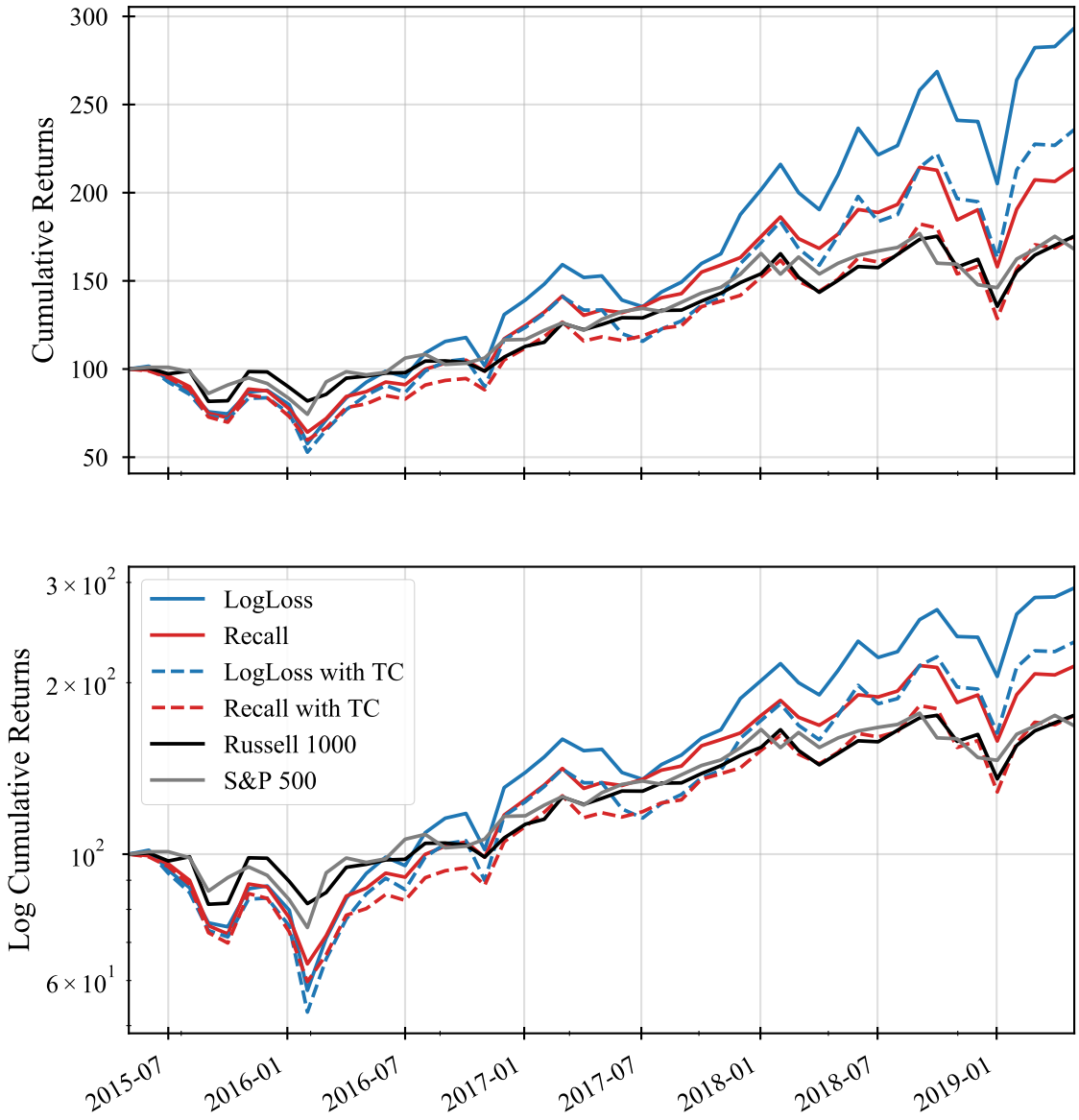

Figure 6.2 Top panel: Out-Sample cumulative returns for strategies and benchmarks.

Bottom panel: Out-Sample log cumulative returns for strategies and benchmarks.

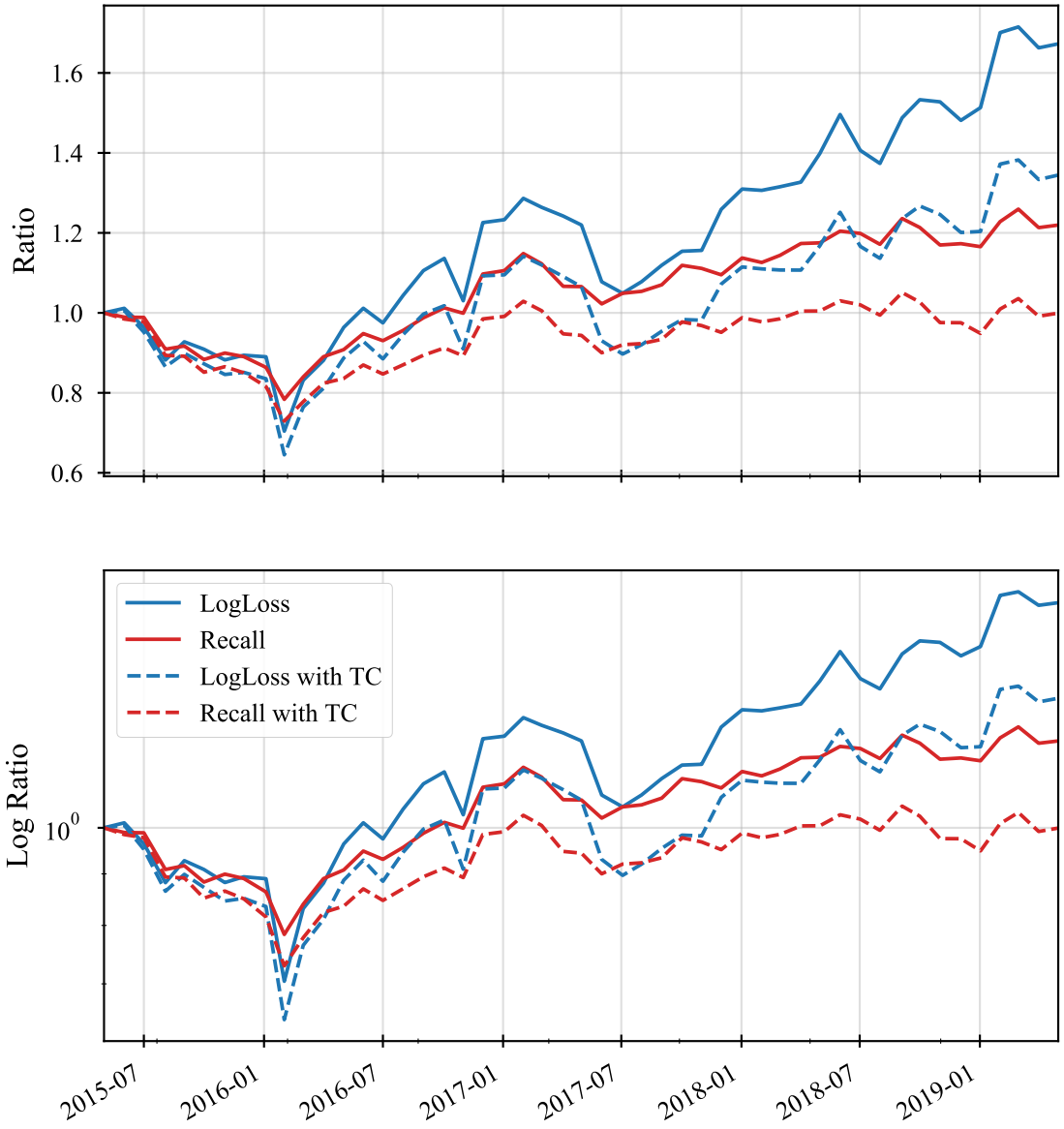

Figure 6.3 Top panel: Out-Sample ratio of the strategies cumulative returns over the Russell 1000 Index cumulative returns.

Bottom panel: Out-Sample ratio of the strategies cumulative returns over the Russell 1000 Index cumulative returns displayed in log values.

Table 6.1 Strategies performances for out-of-sample data.

Category	Measure	Pipeline LogLoss		Pipeline Recall		Russell 1000
		w/o TC	with TC	w/o TC	with TC	Index
General	Time range	from the 30th April 2015 to the 29th March 2019				
	Turnover	16		13		N/A
Performance	Cumulative return	193.3%	135.8%	113.8%	75.2%	75.3%
	CAGR	40.8%	35.3%	33.1%	28.8%	28.8%
	Alpha	7.4%	3.2%	2.1%	-1.2%	N/A
	Beta	1.35	1.35	1.20	1.20	N/A
	Stability	0.92	0.89	0.9	0.84	0.89
	Hit Ratio	53.8%		54.3%		N/A
	Average return from hits	0.34%		0.08%		N/A
	Average return from misses	-0.29%		-0.07%		N/A
Risk	Volatility	21.1%	21.1%	16.5%	16.5%	12.7%
	Skewness	0.20	0.19	-0.29	-0.31	-0.76
	Kurtosis	0.75	0.73	0.62	0.62	1.75
	Maximum drawdown	-21.8%	-24%	-18%	-20%	-14%
Efficiency	Sharpe ratio	0.91	0.72	0.77	0.57	0.69
	Information ratio	0.8	0.5	0.56	0.08	N/A
	Sortino ratio	1.63	1.23	1.23	0.87	1.01
	Calmar ratio	0.84	0.58	0.65	0.41	0.58
	Tail ratio	1.24	1.14	1.10	1.02	0.85

7 Conclusion and Further Research

Because of data leaking has been spotted in the last weeks of this thesis, most findings of this thesis remain invalid. However, practical use of financial machine learning concepts and related issues have been underlined here.

Section 3 reviewed essential concepts before starting to design any trading strategy.

In Section 4, Assumptions about missing data and outliers have been defined, and a FAMD has been applied to both quantitative a qualitative variables to extract information from 92 dimensions The interpretation of FAMD plots suggested many features were redundant. Correlated quantitative variables have also not been filtered. As shown further by RFECV and SFS, some feature interaction can occur between correlated feature, increasing model performance at the same time. Finally the uncertainty coefficient was used to remove 11 qualitative variables. As an example, exponential moving averages were found too associated with the respective length simple moving averages and were dropped, therefore.

Section 5 presented the modelling process and the meta-labelling approach. Non-parametric statistical tests were also used to compare classifier performances. However, after feature selection to determine a more efficient feature subset, data leakage has been suspected due to early out-of-sample backtest performances. The issue has been reviewed and fixed. New machine learning pipelines have been created to deliver more reasonable backtests.

The predictions were converted, in Chapter 6 to allocation weights with statistical tests. Besides, strategies have been optimized with the Combinatorial Purged Cross-Validation allowing to backtest the entire training time range without overfitting. Finally two monthly equity strategies based on Ensemble of Decision-Tree algorithms were proposed, and their performances analyzed. While trading costs annihilates profitability for the first one, the second one remains profitable and above the selected benchmark. These results were found similar to those of (Fischer & Krauss, 2018; Krauss, Do, & Huck, 2017) for whom, the performances of machine-learning based strategy were continually dropping over time. It can be assumed that financial markets agents have been using machine learning techniques long enough for simple models to capture any profits nowadays. Also, the inputs were derived from already well documented and might no longer offer valuable trading signals or as (López de Prado, 2018) suggests, traditional financial data have been so exploited that new alphas should be looked in alternative data.

Finally, two additional chapters would have been added, if data leakage presence not occurred:

- Machine Learning Interpretation with SHAP (Lundberg et al., 2018) as feature importances interpretation from decision trees algorithms cannot be considered consistent.
- AutoML and Deep Learning. Models have been initially developed before data leakage was noticed. Therefore, it did not seem to be a priority to pursue development.

Further research are suggested with the following five approaches.

1. **Features**: Increasing them, as a limited amount of fundamental data were considered here.
2. **Learners**: The field of AI/ML is currently one of the newest research areas, and it can be hard to keep pace with the constant flow of new learners developed. If the present topic should still be considered as a classification problem, one can consider more advanced deep learning models such as attention-based neural networks (H. Li, Shen, & Zhu, 2018; Youru Li, Zhu, Kong, Han, & Zhao, 2019; Vinayavekhin et al., 2018) or probabilistic deep learning (H. Wang & Yeung, 2016). Framing it as a stock ranking problem, LambdaRank or other ranking algorithms could be used (Burges, 2010; Song, Liu, & Yang, 2017; L. Wang & Rasheed, 2018). Finally, as trading strategies involve optimizing trading decisions, a deep-/reinforcement learning framework could be a natural framework (Sato, 2019).
3. **Asset classes**: The present methodology could be easily extended to other asset classes as Kakushadze & Serur (2018) also define strategies for options, fixed income, indexes, volatility, foreign exchange, commodities, futures and others. If different class portfolios were to be designed, a machine learning asset allocation such as the Hierarchical Risk Parity allocation (HRP) (López de Prado, 2016) could be considered.
4. **Strategy framework**: Compared to the proposed framework by (López de Prado, 2018), this paper works with the monthly fixed-time horizon. The returns are also labelled according to their cross-section percentile. In that sense, positive returns could be labelled negative for the case of a bullish market and vice-versa. If dynamic thresholds were to be defined from a risk management realistic point of view, the triple-barrier method (López de Prado, 2018) could have been used. Future researchers are also invited to explore different trading frequency, considering overlapping bets or long/short implementation. The strategy could also implement a layer of complexity such as stop-loss limits, market regime detection (Chakravorty & Awasthi, 2018; Cook & Smalter Hall, 2017; John M., 2016) or different allocation systems (Black &

Litterman, 1992; López de Prado, 2016; Rotando & Thorp, 1992). Finally, the content of (Cesa-Bianchi & Lugosi, 2006) was recommended, however, not considered in this paper due to time constraints.

5. **Portfolio metrics**: Due to time constraints, advanced backtests statistics (López de Prado, 2018) such as Probabilistic Sharpe Ratio and Deflated Sharpe Ratio, or a strategy risk review (López de Prado, 2018) have not been performed here.

Finally, using R is also a consideration suggested here. While python is better suited for a production environment, the lack of flexibility and features from its ML packages at this time raised along the thesis might question its use for research purposes. For instance, Kuhn & Johnson (2019) rely on the Caret R package (Kuhn, 2008) that includes feature selection with genetic algorithms.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Retrieved from https://www.tensorflow.org/

Adams, J., Hayunga, D., Mansi, S., Reeb, D., & Verardi, V. (2019). Identifying and treating outliers in finance. *Financial Management*, *48*(2), 345–384. https://doi.org/10.1111/fima.12269

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *KDD 2019: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631. https://doi.org/10.1145/3292500.3330701

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *American Statistician*, *46*(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

An, B. J., Ang, A., Bali, T. G., & Cakici, N. (2014). The Joint Cross Section of Stocks and Options. *Journal of Finance*, *69*(5), 2279–2337. https://doi.org/10.1111/jofi.12181

Ang, A., Hodrick, R. J., Xing, Y., & Zhang, X. (2006). The Cross-Section of Volatility and Expected Returns. *Journal of Finance*, *61*(1), 259–299. https://doi.org/10.1111/j.1540-6261.2006.00836.x

Ang, A., Hodrick, R. J., Xing, Y., & Zhang, X. (2009). High idiosyncratic volatility and low returns: International and further U.S. evidence. *Journal of Financial Economics*, *91*(1), 1–23. https://doi.org/10.1016/j.jfineco.2007.12.005

Avellaneda, M., & Lee, J. H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance*, *10*(7), 761–782. https://doi.org/10.1080/14697680903124632

Baker, M., Bradley, B., & Wurgler, J. (2011). Benchmarks as limits to arbitrage: Understanding the low-volatility anomaly. *Financial Analysts Journal*, *67*(1), 40–54. https://doi.org/10.2469/faj.v67.n1.4

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, *12*(7), 1–24. https://doi.org/10.1371/journal.pone.0180944

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research*, *18*(1998), 2653–2688. Retrieved from

http://arxiv.org/abs/1606.04316

Bengfort, B., Bilbro, R., Danielsen, N., Gray, L., McIntyre, K., Roman, P., … Krishna, G. (2018). *Yellowbrick v0.9*. https://doi.org/10.5281/ZENODO.1488364

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *NIPS 2011 Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2546–2554. Retrieved from http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimizat

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*, 281–305. Retrieved from http://jmlr.csail.mit.edu/papers/v13/bergstra12a.html

Bergstra, J., Yamins, D. L. K., & Cox, D. D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *ICML 2013: Proceedings of the 30th International Conference on International Conference on Machine Learning*, *28*, 115–123. Retrieved from http://proceedings.mlr.press/v28/bergstra13.html

Bernard, V. L., & Thomas, J. K. (1989). Post-Earnings-Announcement Drift: Delayed Price Response or Risk Premium? *Journal of Accounting Research*, *27*, 1. https://doi.org/10.2307/2491062

Black, F. (1986). Noise. *The Journal of Finance*, *41*(3), 528–543. https://doi.org/10.1111/j.1540-6261.1986.tb04513.x

Black, F., & Litterman, R. (1992). Global Portfolio Optimization. *Financial Analysts Journal*, *48*(5), 28–43. https://doi.org/10.2469/faj.v48.n5.28

Blitz, D., Huij, J., & Martens, M. (2011). Residual Momentum. *Journal of Empirical Finance*, *18*(3), 506–521. https://doi.org/10.1016/j.jempfin.2011.01.003

Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *The Journal of Finance*, *47*(5), 1731–1764. https://doi.org/10.1111/j.1540-6261.1992.tb04681.x

Burges, C. J. C. (2010). *From RankNet to LambdaRank to LambdaMART: An Overview*. Retrieved from https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L. F., Nobrega, J. P., & Oliveira, A. L. I. (2016). Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, *55*, 194–211. https://doi.org/10.1016/j.eswa.2016.02.006

Cerqueira, V., Torgo, L., & Mozetič, I. (2019). Evaluating time series forecasting models: An empirical study on performance estimation methods. *CoRR*, *abs/1905.1*. Retrieved from http://arxiv.org/abs/1905.11744

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, Learning, and Games* (1st ed.). Cambridge University Press.

Chakravorty, G., & Awasthi, A. (2018). Deep Learning for Global Tactical Asset Allocation. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3242432

Chan, E. P. (2009). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business* (1st ed.). John Wiley & Sons.

Chan, E. P. (2017). *Machine Trading: Deploying Computer Algorithms to Conquer the Markets* (1st ed.). John Wiley & Sons.

Chan, L. K. C., Jegadeesh, N., & Lakonishok, J. (1996). Momentum Strategies. *Journal of Finance*, *51*(5), 1681–1713. https://doi.org/10.1111/j.1540-6261.1996.tb05222.x

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *KDD 2016: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Chollet, F., & Others. (2015). *Keras*. Retrieved from https://keras.io

Cook, T., & Smalter Hall, A. (2017). Macroeconomic Indicator Forecasting with Deep Neural Networks. In *The Federal Reserve Bank of Kansas City Research Working Papers*. https://doi.org/10.18651/RWP2017-11

Cortes, Corinna, & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Daniel, G., Sornette, D., & Woehrmann, P. (2009). Look-Ahead Benchmark Bias in Portfolio Performance Evaluation. *The Journal of Portfolio Management*, *36*(1), 121–130. https://doi.org/10.3905/JPM.2009.36.1.121

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. Retrieved from https://dask.org

DeMiguel, V., Garlappi, L., Nogales, F. J., & Uppal, R. (2009). A Generalized Approach to Portfolio Optimization: Improving Performance by Constraining Portfolio Norms. *Management Science*, *55*(5), 798–812. https://doi.org/10.1287/mnsc.1080.0986

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30. Retrieved from http://www.jmlr.org/papers/v7/demsar06a.html

Donchian, R. D. (1960). High Finance in Copper. *Financial Analysts Journal*, *16*(6), 133–

142. https://doi.org/10.2469/faj.v16.n6.133

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

Ferri, F. J., Pudil, P., Hatef, M., & Kittler, J. (1994). Comparative study of techniques for large-scale feature selection. In *Pattern Recognition in Practice IV* (Vol. 16, pp. 403–413). https://doi.org/10.1016/B978-0-444-81892-8.50040-7

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 2962–2970). Retrieved from http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

Foster, G. (1977). Quarterly accounting data: Time-series properties and predictive-ability results. *The Accounting Review*, *52*(1), 1–21. https://doi.org/10.2307/2490556

Foster, G., Olsen, C., & Shevlin, T. (1984). Earnings Releases, Anomalies, and the Behavior of Security Returns. *Accounting Review*, *59*(4), 574. https://doi.org/10.2307/247321

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, *29*(5), 1189–1232. Retrieved from http://www.jstor.org/stable/2699986

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701. https://doi.org/10.2307/2279372

Fu, X., Du, J., Guo, Y., Liu, M., Dong, T., & Duan, X. (2018). *A Machine Learning Framework for Stock Selection*. Retrieved from http://arxiv.org/abs/1806.01743

Gama, J., Medas, P., & Rodrigues, P. (2004). Concept Drift in Decision Trees Learning from Data Streams. *EUNITE 2004: Proceedings of the 4th European Symposium on Intelligent Technologies and Their Implementation on Smart Adaptive Systems*, 218–225. Retrieved from www.eunite.org

Gibrat, R. (1978). L'analyse des données. *Journal de La Société Statistique de Paris*, *119*(3), 201–228. Retrieved from http://www.numdam.org/item?id=JSFS_1978__119_3_201_0

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (1st ed.). MIT Press.

Goodwin, T. H. (1998). The Information Ratio. *Financial Analysts Journal*, *54*(4), 34–43.

https://doi.org/10.2469/faj.v54.n4.2196

Green, D. M., & Swets, J. A. (1966). *Signal Detection Theory and Psychophysics* (1st ed.). John Wiley & Sons.

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2017). Correlation and variable importance in random forests. *Statistics and Computing*, *27*(3), 659–678. https://doi.org/10.1007/s11222-016-9646-1

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, *3*, 1157–1182. https://doi.org/10.1162/153244303322753616

Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (Eds.). (2006). *Feature extraction* (1st ed.). https://doi.org/10.1007/978-981-13-6098-5_3

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, *46*(1), 389–422. https://doi.org/10.1023/A:1012487302797

Hand, D. J., & Yu, K. (2001). Idiot's Bayes: Not So Stupid After All? *International Statistical Review*, *69*(3), 385–398. https://doi.org/10.1111/j.1751-5823.2001.tb00465.x

Harald, C. (1946). *Mathematical Methods of Statistics* (1st ed.). Princeton University Press.

Harris, D. E. (2017). The Distribution of Returns. *Journal of Mathematical Finance*, *07*(03), 769–804. https://doi.org/10.4236/jmf.2017.73041

Hastie, T., Robert, T., & Jerome, F. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). https://doi.org/10.1007/978-0-387-84858-7

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv E-Prints*, arXiv:1312.4569. Retrieved from https://arxiv.org/pdf/1207.0580.pdf

Ho, T. K. (1995). Random decision forests. *ICDAR 1995: Proceedings of the 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735

Hsu, P.-H., & Kuan, C.-M. (2005). Re-Examining the Profitability of Technical Analysis with White's Reality Check and Hansen's SPA Test. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.685361

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science and Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hutter, F., Lars, K., & Joaquin, V. (Eds.). (2019). *Automated Machine Learning: Methods, Systems, Challenges* (1st ed.). https://doi.org/10.1007/978-3-030-05318-5

Jansen, S. (2018). *Hands-On Machine Learning for Algorithmic Trading: Design and implement investment strategies based on smart algorithms that learn from data using Python* (1st ed.). Packt Publishing.

Japkowicz, N., & Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective* (1st ed.). https://doi.org/10.1017/CBO9780511921803

Joblib developers. (2011). *Joblib*. Retrieved from https://github.com/joblib/joblib

John M., M. (2016). *Applying Machine Learning to Identify Regimes For Asset Allocation and ALM*. Retrieved from https://www.northinfo.com/documents/719.pdf

Jones, E., Oliphant, T., Peterson, P., & Others, A. (2001). *SciPy: Open Source Scientific Tools for Python*. Retrieved from http://www.scipy.org/

Kakushadze, Z. (2016). 101 Formulaic Alphas. *Wilmott Journal*, *2016*(84), 72–81. https://doi.org/10.1002/wilm.10525

Kakushadze, Z., & Serur, J. A. (2018). *151 Trading Strategies* (1st ed.). https://doi.org/10.1007/978-3-030-02792-6

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in Data Mining: Formulation, Detection, and Avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *6*(4), 15. https://doi.org/10.1145/2020408.2020496

Ke, G., Meng, Q., Wang, T., Chen, W., Ma, W., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *NIPS 2017: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3149–3157. Retrieved from http://papers.nips.cc/paper/6907-a-highly-efficient-gradient-boosting-decision-tree.pdf

Kelly, M. G., Hand, D. J., & Adams, N. M. (1999). The Impact of Changing Populations on Classifier Performance. *KDD 1999: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 367–371. https://doi.org/10.1145/312129.312285

Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, *30*(1/2), 81–93. https://doi.org/10.2307/2332226

Kluyver, T., Ragan-kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., … Willing, C. (2016). Jupyter Notebooks: a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *ELPUB 2016: Proceedings of the 20th International Conference on Electronic Publishing* (pp. 87–90).

https://doi.org/10.3233/978-1-61499-649-1-87

Kofman, P., & Sharpe, I. (2000). Imputation Methods for Incomplete Dependent Variables in Finance. In *Research Paper Series 33*. Retrieved from Quantitative Finance Research Centre, University of Technology, Sydney website: https://ideas.repec.org/p/uts/rpaper/33.html

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, *259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031

Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of Statistical Software*, *28*(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* (1st ed.). https://doi.org/10.1007/978-1-4614-6849-3

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models* (1st ed.). CRC Press.

Lê, S., Josse, J., & Husson, F. (2008). FactoMineR: An R Package for Multivariate Analysis. *Journal of Statistical Software*, *25*(1), 1–18. https://doi.org/10.18637/jss.v025.i01

Lee, T.-H. (2007). Loss Functions in Time Series Forecasting. *University of California*. Retrieved from http://www.faculty.ucr.edu/~taelee/paper/lossfunctions.pdf

Li, H., Shen, Y., & Zhu, Y. (2018). Stock Price Prediction Using Attention-based Multi-Input LSTM. *PLMR: Proceedings of The 10th Asian Conference on Machine Learning*, 454–469. Retrieved from http://proceedings.mlr.press/v95/li18c/li18c.pdf

Li, Yaliang, Gao, J., Li, Q., & Fan, W. (2015). Ensemble Learning. In C. C. Aggarwal (Ed.), *Data Classification: Algorithms and Applications* (1st ed., pp. 483–503). CRC Press.

Li, Youru, Zhu, Z., Kong, D., Han, H., & Zhao, Y. (2019). EA-LSTM: Evolutionary Attention-based LSTM for Time Series Prediction. *Knowledge-Based Systems*, *181*, 104785. https://doi.org/https://doi.org/10.1016/j.knosys.2019.05.028

López de Prado, M. (2016). Building Diversified Portfolios that Outperform Out of Sample. *The Journal of Portfolio Management*, *42*(4), 59–69. https://doi.org/10.3905/jpm.2016.42.4.059

López de Prado, M. (2018). *Advances in Financial Machine Learning* (1st ed.). John Wiley & Sons.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., … Lee, S.-I. (2019). Explainable AI for Trees: From Local Explanations to Global Understanding. *ArXiv E-Prints*, arXiv:1905.04610. Retrieved from http://arxiv.org/abs/1905.04610

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent Individualized Feature Attribution for Tree Ensembles. *ArXiv E-Prints*, arXiv:1802.03888. Retrieved from http://arxiv.org/abs/1802.03888

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *NIPS 2017: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4765–4774. Retrieved from http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Luo, Y., Alvarez, M., Wang, S., Jussa, J., Wang, A., & Rohal, G. (2014). *Seven sins of quantitative investing*. Retrieved from http://newyork.qwafafew.org/wp-content/uploads/sites/4/2015/10/Luo_20150128.pdf

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *SciPy 2010: Proceedings of the 9th Python in Science Conference*, 51–56. Retrieved from https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf

Meucci, A. (2010). Quant Nugget 4: Annualization and General Projection of Skewness, Kurtosis and All Summary Statistics. *GARP Risk Professional - "The Quant Classroom,"* 59–63. Retrieved from https://ssrn.com/abstract=1635484

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A Unifying View on Dataset Shift in Classification. *Pattern Recognition*, *45*(1), 521–530. https://doi.org/10.1016/J.PATCOG.2011.06.019

Murphy, J. J. (1999). *Technical analysis of the financial markets : a comprehensive guide to trading methods and applications* (1st ed.). Prentice Hall Press.

Nemenyi, P. (1962). Distribution-free multiple comparisons. *Biometrics*, *18*(2), 263.

Ng, A. Y. (2019). *Machine Learning Yearning: Technical Strategy for AI Engineers, In the Era of Deep Learning* (1st ed.). Retrieved from https://www.deeplearning.ai/machine-learning-yearning/

Nielsen, D. (2016). Tree Boosting With XGBoost-Why Does XGBoost Win" Every" Machine Learning Competition? (Norwegian University of Science and Technology). https://doi.org/10.1111/j.1758-5899.2011.00096.x

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. *GECCO 2016: Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 485–492. https://doi.org/10.1145/2908812.2908918

Pages, J. (2004). Analyse factorielle de données mixtes. *Revue de Statistique Appliquée*, *52*(4), 93–111. Retrieved from http://www.numdam.org/item/RSA_2004__52_4_93_0

Pearson, K. (1900). X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *50*(302), 157–175. https://doi.org/10.1080/14786440009463897

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. https://doi.org/10.1080/14786440109462720

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(2/1/2011), 2825–2830. Retrieved from http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

Pérez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science and Engineering*, *9*(3), 21–29. https://doi.org/10.1109/MCSE.2007.53

Piotroski, J. D. (2006). Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers. *Journal of Accounting Research*, *38*, 1. https://doi.org/10.2307/2672906

Pumperla, M. (2016). *Hyperas*. Retrieved from https://github.com/maxpumperla/hyperas

Quantopian Inc. (2015). *Pyfolio*. Retrieved from https://github.com/quantopian/pyfolio

Quantopian Inc. (2017). *Empyrical*. Retrieved from https://github.com/quantopian/empyrical

R Development Core Team (R Foundation for Statistical Computing). (2008). *R: A Language and Environment for Statistical Computing*. Retrieved from http://www.r-project.org

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, *3*(24). https://doi.org/10.21105/joss.00638

Rashmi, K. V., & Gilad-Bachrach, R. (2015). DART: Dropouts meet Multiple Additive Regression Trees. *ArXiv E-Prints*, arXiv:1505.01866. Retrieved from http://arxiv.org/abs/1505.01866

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *KDD 2016: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. https://doi.org/10.1145/2939672.2939778

Robnik-Sikonja, M., & Kononenko, I. (2003). Theoretical and Empirical Analysis of ReliefF

and RReliefF. *Machine Learning*, *53*(1), 23–69. https://doi.org/10.1023/A:1025667309714

Roderick, J. A. L., & Donald, B. R. (2019). *Statistical Analysis with Missing Data* (3rd ed.). John Wiley & Sons.

Rosenberg, B., Reid, K., & Lanstein, R. (1985). Persuasive evidence of market inefficiency. *The Journal of Portfolio Management*, *11*(3), 9–16. https://doi.org/10.3905/jpm.1985.409007

Ross, B. C. (2014). Mutual Information between Discrete and Continuous Data Sets. *PLOS ONE*, *9*(2), e87357. Retrieved from https://doi.org/10.1371/journal.pone.0087357

Rotando, L. M., & Thorp, E. O. (1992). The Kelly Criterion and the Stock Market. *The American Mathematical Monthly*, *99*(10), 922–931. https://doi.org/10.1080/00029890.1992.11995955

Rule, A., Birmingham, A., Zuniga, C., Altintas, I., Huang, S.-C., Knight, R., … Rose, P. W. (2019). Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLOS Computational Biology*, *15*(7), e1007007. https://doi.org/10.1371/journal.pcbi.1007007

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Ryll, L., & Seidens, S. (2019). Evaluating the Performance of Machine Learning Algorithms in Financial Market Forecasting: A Comprehensive Survey. *ArXiv E-Prints*, arXiv:1906.07786. Retrieved from http://arxiv.org/abs/1906.07786

Salzberg, S. L. (1997). On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach. *Data Mining and Knowledge Discovery*, *1*(3), 317–328. https://doi.org/10.1023/A:1009752403260

Sato, Y. (2019). Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey. *ArXiv E-Prints*, arXiv:1904.04973. Retrieved from http://arxiv.org/abs/1904.04973

Schreiber, J. (2018). Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, *18*(164), 1–6. Retrieved from http://arxiv.org/abs/1711.00137

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. *SciPy 2010: Proceedings of the 9th Python in Science Conference*, (Scipy), 61. Retrieved from http://statsmodels.sourceforge.net/

Serneels, S., De Nolf, E., & Van Espen, P. J. (2006). Spatial sign preprocessing: A simple

way to impart moderate robustness to multivariate estimators. *Journal of Chemical Information and Modeling*, *46*(3), 1402–1409. https://doi.org/10.1021/ci050498u

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

Sharpe, W. F. (1966). Mutual Fund Performance. *The Journal of Business*, *39*(1), 119–138. Retrieved from https://www.jstor.org/stable/2351741

Sirotyuk, E. (2018). State of Machine Learning Applications in Investment Management. In T. Guida (Ed.), *Big Data and Machine Learning in Quantitative Investment* (1st ed., pp. 33–49). https://doi.org/10.1002/9781119522225.ch3

Song, Q., Liu, A., & Yang, S. Y. (2017). Stock portfolio selection using learning-to-rank algorithms with news sentiment. *Neurocomputing*, *264*, 20–28. https://doi.org/10.1016/j.neucom.2017.02.097

Sortino, F. A., & Price, L. N. (1994). Performance Measurement in a Downside Risk Framework. *The Journal of Investing*, *3*(3), 59–64. https://doi.org/10.3905/joi.3.3.59

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, *15*(1), 72–101. https://doi.org/10.2307/1412159

Stonebraker, M., & Rowe, L. A. (1986). The design of POSTGRES. *SIGMOD 1986: Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, 340–355. https://doi.org/10.1145/16894.16888

TeamHG-Memex. (2016). *ELI5*. Retrieved from https://eli5.readthedocs.io/en/latest/overview.html

Terpilowski, M. (2019). scikit-posthocs: Pairwise multiple comparison tests in Python. *The Journal of Open Source Software*, *4*(36), 1169. https://doi.org/10.21105/joss.01169

Theil, H. (1970). On the Estimation of Relationships Involving Qualitative Variables. *American Journal of Sociology*, *76*(1), 103–154. https://doi.org/10.1086/224909

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M., & Moore, J. H. (2018). Benchmarking Relief-Based Feature Selection Methods for Bioinformatics Data Mining. *Journal of Biomedical Informatics*, *85*(3), 168–188. https://doi.org/10.1016/j.jbi.2018.07.015

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure

for Efficient Numerical Computation. *Computing in Science and Engineering*, *13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Van Rossum, G., & De Boer, J. (1991). Interactively Testing Remote Servers using the Python Programming Language. *CWI Quarterly*, *4*(4), 283–304. Retrieved from https://ir.cwi.nl/pub/18204

Vinayavekhin, P., Chaudhury, S., Munawar, A., Agravante, J., De Magistris, G., Kimura, D., & Tachibana, R. (2018). Focusing on What is Relevant: Time-Series Learning and Understanding using Attention. *ICPR 2018: Proceedings of the 24th International Conference on Pattern Recognition*, 2624–2629. Retrieved from http://static.aixpaper.com/pdf/e/62/1806.08523.pdf

W. Lo, A. (2017). *Adaptive Markets: Financial Evolution at the Speed of Thought* (1st ed.). Princeton University Press.

Wang, H., & Yeung, D.-Y. (2016). Towards Bayesian Deep Learning: A Survey. *ArXiv E-Prints*, arXiv:1604.01662. Retrieved from http://arxiv.org/abs/1604.01662

Wang, L., & Rasheed, K. (2018). Stock Ranking with Market Microstructure, Technical Indicator and News. *ICAI 2018: Proceedings of the 20th International Conference on Artificial Intelligence*, 322–328. Retrieved from https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/ICA3687.pdf

Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Ostblom, J., Lukauskas, S., … Qalieh, A. (2018). *mwaskom/seaborn: v0.9.0 (July 2018)*. https://doi.org/10.5281/ZENODO.1313201

Wiecki, T., Campbell, A., Lent, J., & Stauth, J. (2016). All That Glitters Is Not Gold: Comparing Backtest and Out-of-Sample Performance on a Large Cohort of Trading Algorithms. *The Journal of Investing*, *25*(3), 69–80. https://doi.org/10.3905/joi.2016.25.3.069

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics*, *1*(6), 80. https://doi.org/10.2307/3001968

Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959. https://doi.org/10.1093/biomet/87.4.954

Young, T. W. (1991). Calmar ratio: A smoother tool. *Futures*, *20*(1), 40.

Žliobaitė, I. (2010). Learning under Concept Drift: an Overview. *ArXiv E-Prints*, arXiv:1010.4784. Retrieved from http://arxiv.org/abs/1010.4784

Zychlinski, S. (2018). *Dython*. Retrieved from http://shakedzy.xyz/dython/

Appendix A: Python and Dataset Files

Due to Bloomberg licensing, datasets are only available upon request, while the Python codes and Jupyter Notebooks can be found in the GitHub repository: https://github.com/lschneidpro

Besides, the following links are provided for Python development on the ETH clusters:

Python on the ETH clusters:

https://scicomp.ethz.ch/wiki/Python

Configuring submission command to the batch systems of ETH clusters:

https://scicomp.ethz.ch/lsf_submission_line_advisor/

Managing Python environmments with Miniconda on the ETH clusters:

https://www.tomstesco.com/euler-hpc-cluster/

http://kevinkle.in/jekyll/update/2019/02/28/leonhard.html

Jupyter Notebooks on ETH clusters:

https://gitlab.ethz.ch/sfux/Jupyter-on-Euler-or-Leonhard-Open

Appendix B: Downloading Bloomberg data with Python

The Bloomberg Application Programming Interface (API), namely BLPAPI[23], does not provide a native interface in Python, but in C++. Therefore, for a successful installation of the Python module, the following would be needed:

- A local installation of the Bloomberg C++ SDK;
- A C/C++ compiler to build the binary part of the Python module;
- Configuring the system environment variables to locate the C++ SDK.

More information can be found on this GitHub repository[24] or in the *WAPI* section of a Bloomberg Terminal. For easier processing of the data, it might also be wise to install another module. Indeed, the API responses are in JSON (JavaScript Object Notation). Different Python wrappers for BLPAPI are available. Tia[25] is used here in order to directly have the requested data in Pandas object. Depending on the need, the request to the API can include parameters for type of price adjustments[26]. Data can be now requested while a Bloomberg terminal is running in the background with the help of their provided functions. They are similar to Bloomberg's Excel functions are summarised in the table below.

Table B.1 Bloomberg Formulae.

Syntax	Description
`BDP('security','field')`	Bloomberg Data Point (BDP) returns data to a single cell. This formula contains ONLY one security and ONLY one field
`BDH('security','field(s)', 'start date', 'end date', 'opt arg 1', 'opt arg 2')`	Bloomberg Data History (BDH) returns the historical data for selected securities and timeframe.
`BDS('security', 'field', 'opt arg 1', 'opt arg 2')`	Bloomberg Data Set (BDS) returns multi-cell descriptive data.

[23] Bloomberg API: https://www.bloomberg.com/professional/support/api-library/
[24] Bloomberg Python API: https://github.com/msitt/blpapi-python
[25] Tia: https://github.com/bpsmith/tia
[26] How to get adjusted stock prices from BLPAPI: https://lichgo.github.io/2015/11/14/how-to-get-adjusted-stock-price-from-bloomberg-api.html

To complete this list, Bloomberg recently introduced a new syntax, Bloomberg Query Language[27] (BQL) with more advanced functionalities. In addition to retrieving both current and historical data, as BDH and BDP, it is capable of performing complex tasks before retrieval (screening, interval calculations, time series calculations and supports the declaration of custom fields). At the time of writing, BQL is only available through Excel and have been only used to retrieve Point-In-Time (PIT) fundamental data.

To identify a security, Bloomberg uses its own internal Security Identifier (SID) called *Ticker*, but other identifier types can also be passed to the functions, such as CUSIP or ISIN. Few examples to demonstrate how to request Bloomberg's data are provided below:

Retrieving the constituents of the Russel 1000 Index on the 31st January 2000:

```
from tia.bbg import LocalTerminal

resp = LocalTerminal.get_reference_data(sids=['RIY Index'],
                                        flds=['INDX_MWEIGHT_HIST'],
                                        END_DATE_OVERRIDE='2000/01/31')
res = resp.as_frame()
```

Retrieving the daily closing prices of the Apple stock from the 31st January 2000 to today:

```
from tia.bbg import LocalTerminal

resp = LocalTerminal.get_historical(sids=['AAPL US Equity'],
                                    flds=['PX LAST'],
                                    start='2000/01/31')
res = resp.as_frame()
```

Retrieving the PIT Price/Book ratio of the Apple stock for a range of date with BQL

```
=BQL("AAPL US Equity"; "PX_TO_BOOK_RATIO"; "AS_OF_DATE=RANGE(1990-01-
01,1995-01-31)")
```

[27] BQL: http://fintools.com/BQL/Introduction_to_BQL_for_Excel%20080718_f%20jb.pdf

Appendix C: List of Bloomberg Fields downloaded

Table C.1 Bloomberg Fields.

Field	Description
INDX_MWEIGHT_HIST	The list of all the equity members of the index. Historical members can be retrieved by utilising this field in conjunction with the date override field End Date Override.
NAME	The name of the company
EQY_INIT_PO_TYP	Initial Public Offer Type. The type of shares offered are common, class A, class B
EQY_INIT_PO_DT	Initial Public Offer Date, the date the company goes public.
LAST_UPDATE_DT	Date of Last Update. The date is only available if a trade has occurred in the past six weeks. If the most recent trade occurred before this, blank would be returned.
LAST_UPDATE_DATE_EOD	End of day value for Date of Last Update
MARKET_STATUS	Trading status of an equity. Some possible values are: Active, Delisted, Acquired, Unlisted, Suspended, Halted, Private Company, Expired or Postponed
ID_ISIN	The International Securities Identification Number (ISIN) consists of a two-letter country code, followed by the nine-character alphanumerical national security identifier, and a check digit.
ID_CUSIP	Security identification number for the U.S. and Canada. The Committee on Uniform Security Identification Procedures (CUSIP) number consists of nine alphanumeric characters. The first six characters identify the issuer, the following two identify the issue, and the final character is a check digit.
FUNDAMENTALS_TICKER	The ticker to access fundamental equity data for a company

BICS_LEVEL_1_SECTOR_NAME	The Bloomberg Industry Classification System (BICS) level 1 name
BICS_LEVEL_2_INDUSTRY_GROUP_NAME	The Bloomberg Industry Classification System (BICS) level 2 name
BICS_LEVEL_3_INDUSTRY_NAME	The Bloomberg Industry Classification System (BICS) level 3 name
BICS_LEVEL_4_SUB_INDUSTRY_NAME	The Bloomberg Industry Classification System (BICS) level 4 name
BICS_LEVEL_5_SEGMENT_NAME	The Bloomberg Industry Classification System (BICS) level 5 name
PX OPEN	Price at which the security first traded on trading day
PX HIGH	Highest price the security reached during the trading day
PX LOW	Lowest price the security reached during the trading day
PX LAST	Last price for the security on trading day
PX VOLUME	Total number of shares traded on a security on the trading day
EQY_WEIGHTED_AVG_PX	Volume Weighted Average Price: Trading benchmark calculated by dividing the total value traded (sum of price times trade size) by the total volume (sum of trade sizes), taking into account every qualifying transaction.
CUR_MKT_CAP	Total current market value of all of a company's outstanding shares stated in the pricing currency.
HIST_CALL_IMP_VOL	At the money call implied volatility of the 1st listed expiry that is at least 20 business days out from today, calculated from a weighted average of the volatilities of the closest out-of-the-money call option.
CALL_IMP_VOL_10D	Ten days at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.

CALL_IMP_VOL_30D	Thirty days at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.
CALL_IMP_VOL_60D	Sixty days at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.
3MO_CALL_IMP_VOL	Three months at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.
6MO_CALL_IMP_VOL	Six months at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.
12MO_CALL_IMP_VOL	Twelve months at the money call implied volatility based on the Listed Implied Volatility Engine (LIVE) calculator.
IS_DIL_EPS_BEF_XO	Diluted EPS Before Extraordinary Items: It excludes the effects of discontinued operations, accounting standard changes, and natural disasters. Uses weighted average shares figured as if all of the company's potentially dilutive securities had been changed into shares of common stock.
PX_TO_BOOK_RATIO	Ratio of the stock price to the book value per share.

Appendix D: Dataset Cleaning: a Detailled Description

High-data quality is critical in a robust strategy development process and obtaining it could be the most time-consuming step. This survey indicates that collecting and cleaning data can be accounted for respectively 19% and 60% of data-scientists work[28].

Downloading and Cleaning Historical Index Members

1. End-of-month index members are retrieved from 31st January 1995 (earliest availability on Bloomberg) and set in a binary matrix indicating whether the stock is a constituent of the index
2. Due to the override date function of Bloomberg, SIDs appears how they were for the given dates, i.e. PIT. As Bloomberg standardised equity data convention[29] over time, some old SIDs appear to be duplicate, referencing the same underlying security. For instance, *AAPL US* and *APPL UW* refer both to Apple's stocks but are distinct in our initial binary matrix. All SIDs are renamed with *US Equity* ending, and Booleans data from duplicate SIDs are merged such that continuous historical membership of stocks is preserved.
3. Few SIDs refer to different class of shares[30] of the same company. Depending on the company, their prices can be highly correlated as they are exposed to the same underlying factors. The aim of this master thesis is to select companies shares in a cross-sectional manner. For this reason, the most common share classes, referred by the field `FUNDAMENTAL_TICKER` are kept.
4. Fundamental data is not always available on Bloomberg, especially for older SIDs. As earlier fundamental data might not always be available, Bloomberg tends to refer theses data to the company that acquired the selected company if an acquisition happened. These cases are disregarded with the help of the field `FUNDAMENTAL_TICKER`.

[28] Forbes: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-en-joyable-data-science-task-survey-says/#4a599ddf6f63
[29] StackExchange: https://quant.stackexchange.com/questions/17157/difference-between-the-two-bloomberg-codes
[30] Investopedia: https://www.investopedia.com/terms/c/class.asp

Figure D.1 illustrates the effect of the cleaning process. For each month, maximum 1.5 % of the original SIDs are removed, and 2993 final SIDs are numbered across the selected timeframe.

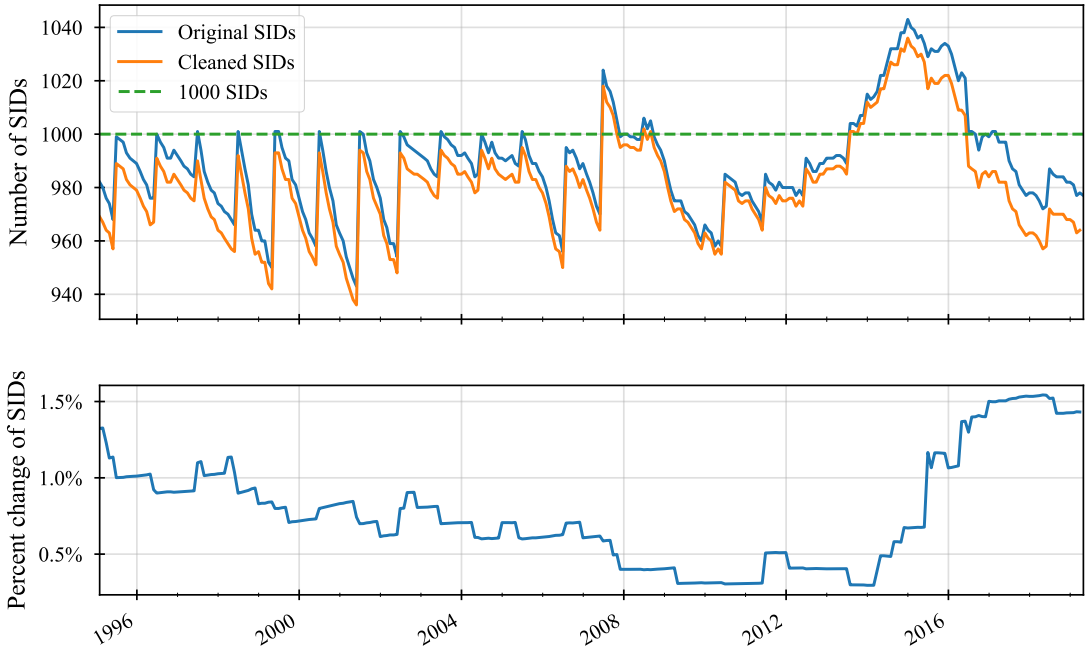

Figure D.1 Visualizations of the absolute and relative differences of SIDs over the years after SIDs cleaning.

Downloading and Cleaning Equity Data

Equity data can be finally downloaded from the cleaned SIDs as follows:

1. Fundamental and Market data are downloaded from Bloomberg. Stock prices are adjusted for dividends, corporate actions and stock splits.
2. Monthly Fama-French 3-/5-Factors are downloaded from Kenneth R. French's website[31] through the pandas-datareader library[32].
3. As trading days do not follow the business calendar of pandas due to American holidays and additional market closing days[33], non-trading days are removed with the NYSE stock exchange calendar from the library pandas-market-calendars[34].

[31] Kennet R. French's website: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
[32] Pandas datareader: https://pandas-datareader.readthedocs.io
[33] CNN: https://www.cnn.com/2018/12/01/business/markets-closed-george-h-w-bush/index.html
[34] Pandas market calendar: https://pandas-market-calendars.readthedocs.io/en/latest/index.html

4. If data is missing between earlier and future non-missing data, missing values are filled forward.
5. Fama-French factors and implied volatilities are converted from percentage to decimal to remain consistent with computed returns.
6. Rows full of missing values are dropped.
7. All Pandas objects are converted to type float32 and finally exported as HDF5[35]for storage efficiency purposes.

The missing data after cleaning are shown in Figure D.2 and spot the lack of availability of implied volatility before 2005.

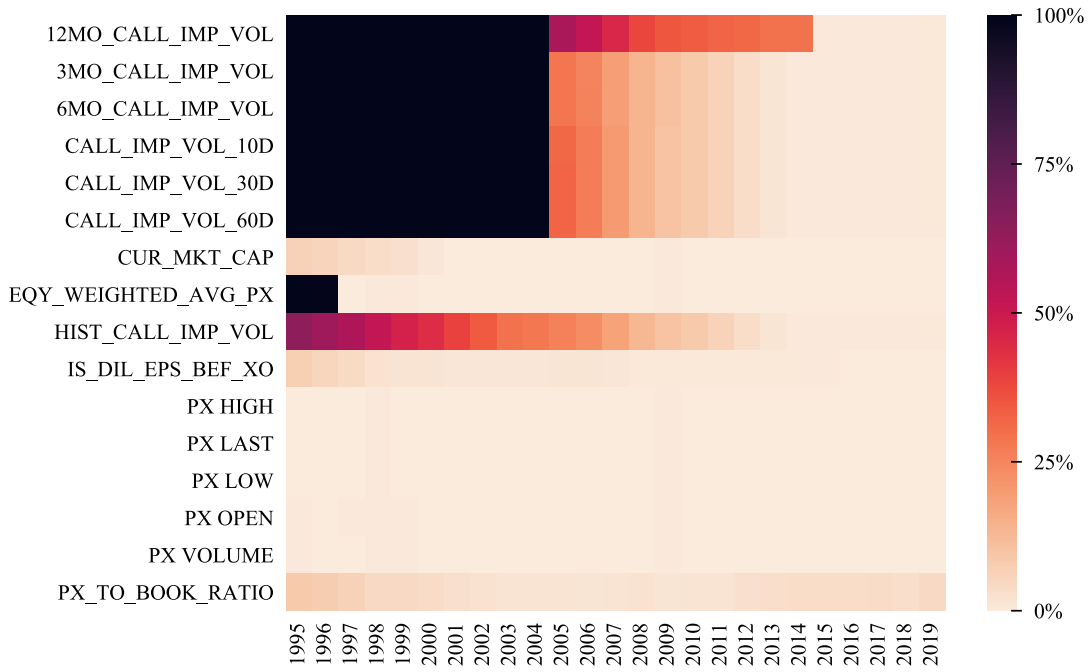

Figure D.2 Visualization of missing data paters over years.

[35] Pytables: https://www.pytables.org/

Appendix E: List of Features

The features are summarised in Table E.1. The computation of each of them and a description of related strategies can be found in Appendix F.

Table E.1 List of Feature.

Index	Variable Name	Description	Type	Strategy
1	R_1MO	Monthly return	Quantitative	Price-Momentum
2	R_Cumulative_3MO	Cumulative return with a 1-month skip period and a 3-month formation period	Quantitative	Price-Momentum
3	R_Mean_3MO	Mean monthly return computed over a 3-month formation period	Quantitative	Price-Momentum
4	R_Risk_Adjusted_3MO	Risk-adjusted mean monthly return over a 3-month formation period	Quantitative	Price-Momentum
5	R_Cumulative_6MO	Cumulative return with a 1-month skip period and a 6-month formation period	Quantitative	Price-Momentum
6	R_Mean_6MO	Mean monthly return computed over a 12-month formation period	Quantitative	Price-Momentum
7	R_Risk_Adjusted_6MO	Risk-adjusted mean monthly return over a 6-month formation period	Quantitative	Price-Momentum
8	R_Cumulative_12MO	Cumulative return with a 1-month skip period and a 12-month formation period	Quantitative	Price-Momentum
9	R_Mean_12MO	Mean monthly return computed over a 12-month formation period	Quantitative	Price-Momentum
10	R_Risk_Adjusted_12MO	Risk-adjusted mean monthly return over a 12-month formation period	Quantitative	Price-Momentum
11	Standardized_Unexpected_Earnings	Standardized unexpected earnings	Quantitative	Earnings-Momentum

12	Month_EPS	Labels for month EPS announcement	Qualitative	Earnings-Momentum
13	BP_Ratio	Book-to-Price (B/P) ratio	Quantitative	Value
14	VOL_21D	Daily volatility over a 21-day observation period	Quantitative	Low Volatility Anomaly
15	VOL_126D	Daily volatility over a 126-day observation period	Quantitative	Low Volatility Anomaly
16	VOL_252D	Daily volatility over a 252-day observation period	Quantitative	Low Volatility Anomaly
17	VOL_M_21D	Monthly volatility over a 21-day observation period	Quantitative	Low Volatility Anomaly
18	VOL_M_126D	Monthly volatility over a 126-day observation period	Quantitative	Low Volatility Anomaly
19	VOL_M_252D	Monthly volatility over a 252-day observation period	Quantitative	Low Volatility Anomaly
20	VAR_21D	Daily variance over a 21-day observation period	Quantitative	Low Volatility Anomaly
21	VAR_126D	Daily variance over a 126-day observation period	Quantitative	Low Volatility Anomaly
22	VAR_252D	Daily variance over a 252-day observation period	Quantitative	Low Volatility Anomaly
23	VAR_M_21D	Monthly variance over a 21-day observation period	Quantitative	Low Volatility Anomaly
24	VAR_M_126D	Monthly variance over a 126-day observation period	Quantitative	Low Volatility Anomaly
25	VAR_M_252D	Monthly variance over a 252-day observation period	Quantitative	Low Volatility Anomaly
26	CVOL_Diff_10D	First-difference of call implied volatilities for a maturity of 10 days	Quantitative	Implied Volatility

27	CVOL_Diff_30D	First-difference of call implied volatilities for a maturity of 30 days	Quantitative	Implied Volatility
28	CVOL_Diff_60D	First-difference of call implied volatilities for a maturity of 60 days	Quantitative	Implied Volatility
29	CVOL_Diff_6MO	First-difference of call implied volatilities for a maturity of 6 months	Quantitative	Implied Volatility
30	CVOL_Pct_10D	Percent change of call implied volatilities for a maturity of 10 days	Quantitative	Implied Volatility
31	CVOL_Pct_30D	Percent change of call implied volatilities for a maturity of 30 days	Quantitative	Implied Volatility
32	CVOL_Pct_60D	Percent change of call implied volatilities for a maturity of 60 days	Quantitative	Implied Volatility
33	CVOL_Pct_6MO	Percent change of call implied volatilities for a maturity of 6 months	Quantitative	Implied Volatility
34	CVOL_Spread_10D	Realised-implied volatility spread for a maturity of 10 days	Quantitative	Implied Volatility
35	CVOL_Spread_30D	Realised-implied volatility spread for a maturity of 30 days	Quantitative	Implied Volatility
36	CVOL_Spread_60D	Realised-implied volatility spread for a maturity of 60 days	Quantitative	Implied Volatility
37	CVOL_Spread_6MO	Realised-implied volatility spread for a maturity of 6 months	Quantitative	Implied Volatility

38	CVOL_Cross_10D	Cross-sectional innovation for a maturity of 10 days	Quantitative	Implied Volatility
39	CVOL_Cross_30D	Cross-sectional innovation for a maturity of 30 days	Quantitative	Implied Volatility
40	CVOL_Cross_60D	Cross-sectional innovation for a maturity of 60 days	Quantitative	Implied Volatility
41	CVOL_Cross_6MO	Cross-sectional innovation for a maturity of 6 months	Quantitative	Implied Volatility
42	Residual_Momentum_FF3	Risk-adjusted residual returns for 3 Fama-French Factors	Quantitative	Residual Momentum
43	Residual_Momentum_FF5	Risk-adjusted residual returns for 5 Fama-French Factors	Quantitative	Residual Momentum
44	Demeaned_10D_R1000	10-day demeaned returns from the Russell 100 Index	Quantitative	Mean-Reversion-Single cluster
45	Demeaned_10D_SP500	10-day demeaned returns from the S&P500	Quantitative	Mean-Reversion-Single cluster
46	Demeaned_21D_R1000	21-day demeaned returns from the Russell 100 Index	Quantitative	Mean-Reversion-Single cluster
47	Demeaned_21D_SP500	21-day demeaned returns from the S&P500	Quantitative	Mean-Reversion-Single cluster
48	Demeaned_10D_BICS1	10-day demeaned returns from BICS level 1	Quantitative	Mean-Reversion-Multiple Clusters
49	Demeaned_10D_BICS2	10-day demeaned returns from BICS level 2	Quantitative	Mean-Reversion-Multiple Clusters
50	Demeaned_10D_BICS3	10-day demeaned returns from BICS level 3	Quantitative	Mean-Reversion-Multiple Clusters
51	Demeaned_10D_BICS4	10-day demeaned returns from BICS level 4	Quantitative	Mean-Reversion-Multiple Clusters
52	Demeaned_10D_BICS5	10-day demeaned returns from BICS level 5	Quantitative	Mean-Reversion-Multiple Clusters
53	Demeaned_21D_BICS1	21-day demeaned returns from BICS level 1	Quantitative	Mean-Reversion-Multiple Clusters

54	Demeaned_21D_BICS2	21-day demeaned returns from BICS level 2	Quantitative	Mean-Reversion-Multiple Clusters
55	Demeaned_21D_BICS3	21-day demeaned returns from BICS level 3	Quantitative	Mean-Reversion-Multiple Clusters
56	Demeaned_21D_BICS4	21-day demeaned returns from BICS level 4	Quantitative	Mean-Reversion-Multiple Clusters
57	Demeaned_121D_BICS5	21-day demeaned returns from BICS level 5	Quantitative	Mean-Reversion-Multiple Clusters
58	Signal_SMA_30D	Signals for 30-day SMA	Qualitative	Single Moving Average
59	Signal_SMA_50D	Signals for 50-day SMA	Qualitative	Single Moving Average
60	Signal_SMA_100D	Signals for 100-day SMA	Qualitative	Single Moving Average
61	Signal_SMA_200D	Signals for 200-day SMA	Qualitative	Single Moving Average
62	Signal_EMA_30D	Signals for 30-day EMA	Qualitative	Single Moving Average
63	Signal_EMA_50D	Signals for 50-day EMA	Qualitative	Single Moving Average
64	Signal_EMA_100D	Signals for 100-day EMA	Qualitative	Single Moving Average
65	Signal_EMA_200D	Signals for 200-day EMA	Qualitative	Single Moving Average
66	Signal_2MA_10_30D	Signals for 10/30 day SMAs	Qualitative	Two Moving Averages
67	Signal_2MA_10_50D	Signals for 10/50 day SMAs	Qualitative	Two Moving Averages
68	Signal_2MA_30_50D	Signals for 30/50 day SMAs	Qualitative	Two Moving Averages
69	Signal_2MA_30_100D	Signals for 30/100 day SMAs	Qualitative	Two Moving Averages

70	Signal_2MA_30_200D	Signals for 30/200 day SMAs	Qualitative	Two Moving Averages
71	Signal_2MA_50_100D	Signals for 50/100 day SMAs	Qualitative	Two Moving Averages
72	Signal_2MA_50_200D	Signals for 50/200 day SMAs	Qualitative	Two Moving Averages
73	Signal_2MA_100_200D	Signals for 100/200 day SMAs	Qualitative	Two Moving Averages
74	Signal_3MA_5_10_20D	Signals for 5/10/20 day SMAs	Qualitative	Three Moving Averages
75	Signal_3MA_5_20_60D	Signals for 5/20/60 day SMAs	Qualitative	Three Moving Averages
76	Signal_3MA_10_30_50D	Signals for 10/30/50 day SMAs	Qualitative	Three Moving Averages
77	Signal_3MA_30_60_120D	Signals for 30/60/120 day SMAs	Qualitative	Three Moving Averages
78	Signal_3MA_30_120_200D	Signals for 30/120/200 day SMAs	Qualitative	Three Moving Averages
79	Signal_3MA_60_90_120D	Signals for 60/90/120 day SMAs	Qualitative	Three Moving Averages
80	Signal_3MA_60_120_200D	Signals for 60/120/200 day SMAs	Qualitative	Three Moving Averages
81	Signal_3MA_90_120_200D	Signals for 90/120/200 day SMAs	Qualitative	Three Moving Averages
82	Signal_Support_Resistance_10D	Signals for 10-day Support Resistance Indicator	Qualitative	Support and Resistance

83	Signal_Support_Resistance_20D	Signals for 20-day Support Resistance Indicator	Qualitative	Support and Resistance
84	Signal_Support_Resistance_30D	Signals for 30-day Support Resistance Indicator	Qualitative	Support and Resistance
85	Signal_Support_Resistance_60D	Signals for 60-day Support Resistance Indicator	Qualitative	Support and Resistance
86	Signal_Support_Resistance_90D	Signals for 90-day Support Resistance Indicator	Qualitative	Support and Resistance
87	Signal_Channel_10D	Signals for 10-day Channel Indicator	Qualitative	Channel
88	Signal_Channel_20D	Signals for 20-day Channel Indicator	Qualitative	Channel
89	Signal_Channel_30D	Signals for 30-day Channel Indicator	Qualitative	Channel
90	Signal_Channel_60D	Signals for 60-day Channel Indicator	Qualitative	Channel
91	Signal_Channel_90D	Signals for 90-day Channel Indicator	Qualitative	Channel
92	KNN_Prediction	Forecasts of monthly returns with *k*-NN	Quantitative	Machine Learning-Single-Stock KNN

Appendix F: Equity Strategies Computation

The following strategies were derived from the Equity section of (Kakushadze & Serur, 2018). For each strategy, an overview and a few variants are proposed based on academic papers to expand the predictors base. The mathematical notation of the authors is used here.

Table F.1 List of Equity Strategies.

Index	Strategy	Variants computed
1	Price-Momentum	10
2	Earnings-Momentum	2
3	Value	1
4	Low-Volatility Anomaly	12
5	Implied Volatility	16
6	Multifactor Portfolio	0
7	Residual Momentum	2
8	Pairs Trading	0
9	Mean-Reversion-Single Cluster	4
	Mean-Reversion-Multiple Clusters	10
10	Mean-Reversion-Weighted Regression	0
11	Single Moving Average	8
12	Two Moving Average	8
13	Three Moving averages	8
14	Support and Resistance	4
15	Channel	4
16	Event-Driven-M&A	0
17	Machine-Learning-Single-Stock KNN	1
18	Statistical Arbitrage-Optimization	0
19	Market-Timing	0
20	Alpha Combos	0

Strategy 1: Price-Momentum

The momentum describes the inertia effect in stock returns, where past performing stocks would continue to perform well and vice-versa. The rationale for such an effect points to behavioural biases behaviour (initial underreaction and delayed overreaction over information dissemination), persistent supply, demand imbalances, a positive feedback loop between risk assets and the economy, or market microstructure (Jansen, 2018). The strategy then amounts to buying the best past performer stocks and selling the worst past performers. The most recent month is usually skipped due to a mean-reversion effect empirically observed in monthly-returns (Kakushadze & Serur, 2018). Let t denote time measured in units of 1 month, with $t = 0$ corresponding to the most recent time; $P_i(t)$ is the time series of adjusted closing prices for stock i; $R_i(t)$ is the monthly return:

$$R_i(t) = \frac{P_i(t)}{P_i(t+1)} - 1$$

R_i^{cum} is the cumulative return computed over the T-month formation period (usually $T = 12$) skipping the most recent S-month skip period (usually $S = 1$):

$$R_i^{cum} = \frac{P_i(S)}{P_i(S+T)} - 1$$

R_i^{mean} is the mean monthly return computed over the formation period:

$$R_i^{mean} = \frac{1}{T} \sum_{t=S}^{S+T-1} R_i(t)$$

σ_i is the monthly volatility calculated over the formation period:

$$\sigma_i^2 = \frac{1}{T-1} \sum_{t=S}^{S+T-1} (R_i(t) - R_i^{mean})^2$$

And $R_i^{risk.adj}$ is the risk-adjusted mean monthly return over the formation period:

$$R_i^{risk.adj} = \frac{R_i^{mean}}{\sigma_i}$$

The predictors are subsequently computed for, $T \in \{1,3,6,12\}$ as shown in Table F-2:

Table F.2 Price-Momentum variants.

Variable name	Value	Description
R_1MO	$R_i(1)$	Monthly return
R_Cumulative_3MO	R_i^{cum} for $T = 3$	Cumulative return with a 1-month skip period and a 3-months formation period
R_Mean_3MO	R_i^{mean} for $T = 3$	Mean monthly return computed over 3-months formation period
R_Risk_Adjusted_3MO	$R_i^{risk.adj}$ for $T = 3$	Risk-adjusted mean monthly return over 3-months formation period
R_Cumulative_6MO	R_i^{cum} for $T = 6$	Cumulative return with a 1-month skip period and a 6-months formation period
R_Mean_6MO	R_i^{mean} for $T = 6$	Mean monthly return computed over 12-months formation period
R_Risk_Adjusted_6MO	$R_i^{risk.adj}$ for $T = 6$	Risk-adjusted mean monthly return over 6-months formation period
R_Cumulative_12MO	R_i^{cum} for $T = 12$	Cumulative return with a 1-month skip period and a 12-months formation period
R_Mean_12MO	R_i^{mean} for $T = 12$	Mean monthly return computed over 12-months formation period
R_Risk_Adjusted_12MO	$R_i^{risk.adj}$ for $T = 12$	Risk-adjusted mean monthly return over 12-months formation period

Strategy 2: Earnings-Momentum

Momentum strategies can also be transposed to Earnings Per Share (EPS). Due to the seasonal aspect of business, seasonally different quarterly earnings show a correlation from one quarter to the next (Foster, 1977). As such, whenever a quarterly EPS is up compared to the quarter of the prior year, the next quarter would generate higher EPS expectation from analysts. Seasonal random walks with a trend have been then proposed to model unexpected EPS (Bernard & Thomas, 1989; Foster, Olsen, & Shevlin, 1984)). To capture deviation from this serial correlation and associated future returns, the Standardized Unexpected Earnings (SUE) definition of (L. K. C. Chan, Jegadeesh, & Lakonishok, 1996) is used here. As before, the strategy consists of buying a top percentile and short a bottom percentile. Let t denote time measured in the units

of 1 quarter, with $t = 0$ corresponding to the most recent time; E_i is the most recently announced quarterly EPS of the stock i; E_i' is the EPS announced four quarters ago; σ_i is the standard deviation of the unexpected earnings $E_i - E_i'$ over the last eight quarters:

$$SUE_i = \frac{E_i - E_i'}{\sigma_i}$$

In addition, as the paper framework is monthly, and returns might be diminishing along holding time (Kakushadze & Serur, 2018), months are labelled according to the previous EPS announcement month:

$$Label_i \doteq \begin{cases} 0 \text{ if } E_i \text{ annonced the same month} \\ 1 \text{ if } E_i \text{ annonced one month ago} \\ 2 \text{ if } E_i \text{ annonced two months ago} \end{cases}$$

The predictors are subsequently computed, as shown in Table F.3:

Table F.3 Earnings-Momentum variants.

Variable name	Value	Description
`Standardized_Unexpected_Earnings`	SUE_i	Standardised unexpected earnings
`Month_EPS`	$Label_i$	Labels for EPS announcement month

Bloomberg's field `Diluted EPS Before Extraordinary Items`[36] is used here instead of basic `EPS` as a precautionary approach. Indeed, diluted EPS take into account securities that can be turned into share while extraordinary items might not have desired impacts.

Strategy 3: Value

Value strategies rely on selecting stocks based on accounting-based fundamental metrics and estimation of asset's fair values. Stocks with low prices relative to their fundamental value, i.e. cheap stocks called *value stocks*, tend to deliver higher returns than the market portfolio (Jansen, 2018). (Rosenberg, Reid, & Lanstein, 1985) shows that a portfolio consisting of buying stocks with high ratio of book value of common equity per share to the market price, i.e.

[36] Stockpedia: https://www.stockopedia.com/ratios/diluted-eps-excluding-extrordinary-items-4914/

Book-to-Price ratio (B/P ratio), and selling their highly valuated counterparts, called *growth stocks*, deliver abnormal returns and goes against market efficiency. Such a phenomenon can be explained by increased premiums, compensating for higher risk and low-performance expectations from analysts (Piotroski, 2006). As before, the strategy consists of buying a top percentile group of stock and short a bottom one. Let t denote time measured in units of 1 month, with $t = 0$ corresponding to the most recent time; the stock is labelled by i:

$$\text{B/P ratio}_i = \frac{\text{Book value per share}_i}{\text{P}_i}$$

The predictor `BP_Ratio` is subsequently computed, as shown in Table F.4:

Table F.4 Value variants.

Variable name	Value	Description
`BP_Ratio`	B/P ratio_i	Book-to-Price (B/P) ratio

Strategy 4: Low Volatility Anomaly

Simple anomalies can challenge the Efficient-Market Hypothesis (EMH) (Fama, 1970). Low-risk stocks exhibit significantly higher risk-adjusted returns than the market portfolio, while high-risk stocks underperform it in the U.S. and across the world (Ang, Hodrick, Xing, & Zhang, 2006, 2009). (Baker, Bradley, & Wurgler, 2011) attributes this anomaly to behavioural finance biases such as fund managers with incentive towards risky assets. As before, the strategy consists of buying a top percentile and short a bottom percentile. Let t denote time measured in units of 1 day, with $t = 0$ corresponding to the most recent time; $P_i(t)$ is the time series of adjusted closing prices for a stock i; R_i^{cum} is the cumulative daily return computed over the T-day formation period:

$$R_i^{cum}(t) = \frac{P_i(t)}{P_i(t+T)} - 1$$

R_i^{mean} is the mean return computed over W-day observation period:

$$R_i^{mean} = \frac{1}{W}\sum_{t=0}^{W-1} R_i^{cum}(t)$$

σ_i is the monthly volatility calculated over the observation period:

$$\sigma_i^2 = \frac{1}{W-1}\sum_{t=0}^{W-1}(R_i^{cum}(t) - R_i^{mean})^2$$

The predictors are subsequently computed for $T \in \{1,21\}$ and $W \in \{21,126,252\}$ as shown in Table F-5:

Table F.5 Low volatility variants.

Feature name	Value	Description
VOL_21D	σ_i for $T=1,\ W=21$	Daily volatility over 21-days observation period
VOL_126D	σ_i for $T=1,\ W=126$	Daily volatility over 126-days observation period
VOL_252D	σ_i for $T=1,\ W=252$	Daily volatility over 252-days observation period
VOL_M_21D	σ_i for $T=21,\ W=21$	Monthly volatility over 21-days observation period
VOL_M_126D	σ_i for $T=21,\ W=126$	Monthly volatility over 126-days observation period
VOL_M_252D	σ_i for $T=21,\ W=252$	Monthly volatility over 252-days observation period
VAR_21D	σ_i^2 for $T=1,\ W=21$	Daily variance over 21-days observation period
VAR_126D	σ_i^2 for $T=1,\ W=126$	Daily variance over 126-days observation period
VAR_252D	σ_i^2 for $T=1,\ W=252$	Daily variance over 252-days observation period
VAR_M_21D	σ_i^2 for $T=21,\ W=21$	Monthly variance over 21-days observation period
VAR_M_126D	σ_i^2 for $T=21,\ W=126$	Monthly variance over 126-days observation period
VAR_M_252D	σ_i^2 for $T=21,\ W=252$	Monthly variance over 252-days observation period

The realised variance is also included, as (Kakushadze & Serur, 2018) often mentions allocating stock proportionally to their volatility, and their variance as well.

Strategy 5: Implied Volatility

The option prices can contain predictive information about stock returns. Consequently, (An, Ang, Bali, & Cakici, 2014) observed that stocks with more significant increases in implied volatilities for call options over the previous month have higher future returns on average, while stocks with implied volatilities for put options increases have lower future returns. As before, the strategy consists of buying a top percentile and short a bottom percentile. Other measures of innovation volatility from (An et al., 2014) have were added to the set of predictors. Let t denote time measured in the units of month, with $t = 0$ corresponding to the most recent time; $CVOL_i(t)$ is the time series of call implied volatilities for a stock i for maturity M; $\Delta CVOL_{i,M}(t)$ is the first-difference of call implied volatilities:

$$\Delta CVOL_{i,M}(t) = CVOL_{i,M}(t) - CVOL_{i,M}(t+1)$$

$\%\Delta CVOL_{i,M}(t)$ is the percent change of call implied volatilities:

$$\%\Delta CVOL_{i,M}(t) = \frac{\Delta CVOL_{i,M}(t) - \Delta CVOL_{i,M}(t+1)}{\Delta CVOL_{i,M}(t+1)}$$

$\sigma_{i,M}$ is the daily volatility computed over M-day window; $RVOL\text{-}IVOL_{i,M}$ is the realised-implied volatility spread:

$$RVOL\text{-}IVOL_{i,M} = \sigma_{i,M} - CVOL_{i,M}$$

$CVOL_{i,M}^{cs/shock}$ is the cross-sectional innovation:

$$CVOL_{i,M}^{cs/shock} = \varepsilon_i$$

Where the residual ε_i is estimated from the cross-sectional regression:

$$CVOL_{i,M}(t) = \alpha(t) + \beta(t) CVOL_{i,M}(t+1) + \varepsilon_i(t)$$

The predictors are subsequently computed daily and monthly for maturities of 10, 30 and 60 days and of 6 months, as shown in Table F.6:

Table F.6 Implied volatility variants.

Variable name	Value	Description
`CVOL_Diff_10D`	$\Delta CVOL_{i,10D}$	First-difference of call implied volatilities for a maturity of 10 days

CVOL_Diff_30D	$\Delta CVOL_{i,30D}$	First-difference of call implied volatilities for a maturity of 30 days
CVOL_Diff_60D	$\Delta CVOL_{i,60D}$	First-difference of call implied volatilities for a maturity of 60 days
CVOL_Diff_6MO	$\Delta CVOL_{i,6M}$	First-difference of call implied volatilities for a maturity of 6 months
CVOL_Pct_10D	$\%\Delta CVOL_{i,10D}$	Percent change of call implied volatilities for a maturity of 10 days
CVOL_Pct_30D	$\%\Delta CVOL_{i,30D}$	Percent change of call implied volatilities for a maturity of 30 days
CVOL_Pct_60D	$\%\Delta CVOL_{i,60D}$	Percent change of call implied volatilities for a maturity of 60 days
CVOL_Pct_6MO	$\%\Delta CVOL_{i,6MO}$	Percent change of call implied volatilities for a maturity of 6 months
CVOL_Spread_10D	$RVOL\text{-}IVOL_{i,10D}$	Realised-implied volatility spread for a maturity of 10 days
CVOL_Spread_30D	$RVOL\text{-}IVOL_{i,30D}$	Realised-implied volatility spread for a maturity of 30 days
CVOL_Spread_60D	$RVOL\text{-}IVOL_{i,60D}$	Realised-implied volatility spread for a maturity of 60 days
CVOL_Spread_6MO	$RVOL\text{-}IVOL_{i,6MO}$	Realised-implied volatility spread for a maturity of 6 months
CVOL_Cross_10D	$CVOL^{cs/shock}_{i,10D}$	Cross-sectional innovation for a maturity of 10 days
CVOL_Cross_30D	$CVOL^{cs/shock}_{i,30D}$	Cross-sectional innovation for a maturity of 30 days
CVOL_Cross_60D	$CVOL^{cs/shock}_{i,60D}$	Cross-sectional innovation for a maturity of 60 days
CVOL_Cross_6MO	$CVOL^{cs/shock}_{i,6MO}$	Cross-sectional innovation for a maturity of 6 months

Put implied volatilities, $PVOL$, are disregarded as the difference between $PVOL$ and $CVOL$ downloaded from Bloomberg is null most of the time for the selected timeframe.

Strategy 6: Multifactor Portfolio

This strategy consists of ranking stocks based on multiple factors, such as the previously mentioned ones. However, it is disregarded here as the machine learning framework is expected to rank itself the strategies.

Strategy 7: Residual Momentum

The performance of Momentum strategies is linked to the Fama-French factors. These strategies are then exposed to losses whenever such factors switch signs. To remove momentum dependency on systemic factors, (Blitz, Huij, & Martens, 2011) propose a momentum strategy on the residuals. As before, the strategy consists of buying a top percentile and short a bottom percentile. Let t denote time measured in units of 1 month, with $t = 0$ corresponding to the most recent time; $R_i(t)$ is the monthly return for a stock i; $R_{free}(t)$ is the risk-free monthly rate; $X_{FF3}(t)$ and $X_{FF5}(t)$ are the 3-/5 Fama-French factors from Kenneth R. French's website[37]:

$$X_{FF3}(t) = \begin{pmatrix} MKT(t) \\ SMB(t) \\ HML(t) \end{pmatrix} \quad X_{FF5}(t) = \begin{pmatrix} MKT(t) \\ SMB(t) \\ HML(t) \\ RMW(t) \\ CMA(t) \end{pmatrix}$$

The serial regression is run over a first T_1-month observation period (here $T_1 = 36$) with a S-month skip period (here $S = 1$) to estimate the regression coefficients $\alpha_i, \beta_{1,i}, \ldots, \beta_{N,i}$ for $i = 1, \ldots, I$:

$$R_i(t) - R_{free}(t) = \alpha_i + \sum_{n=1}^{N} \beta_{n,i} X_n(t) + \varepsilon_i(t) \quad \text{for } t = S, S+1, \ldots, T_1$$

The residuals $\varepsilon_i(t)$ are estimated over a second T_2-month observation period (here $T_2 = 12$) with again a S-month skip period and by dropping the intercept α_i:

[37] Kenneth R. French's website: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

$$\varepsilon_i(t) = R_i(t) - R_{free}(t) - \sum_{n=1}^{N} \beta_{n,i} X_n(t) \quad \text{for } t = S, S+1, \ldots, T_2$$

The residuals are then used to compute the risk-adjusted residual returns $\tilde{R}_i^{risk.adj}$:

$$\varepsilon_i^{mean} = \frac{1}{T} \sum_{t=S}^{S+T-1} \varepsilon_i(t)$$

$$\tilde{\sigma}_i^2 = \frac{1}{T-1} \sum_{t=S}^{S+T-1} \left(\varepsilon_i(t) - \varepsilon_i^{mean} \right)^2$$

$$\tilde{R}_i^{risk.adj} = \frac{\varepsilon_i^{mean}}{\tilde{\sigma}_i}$$

The predictors are subsequently computed for the 3-and 5 Fama-French factors as shown in-Table F.7:

Table F.7 Residual momentum variants.

Variable name	Value	Description
`Residual_Momentum_FF3`	$\tilde{R}_i^{risk.adj}$ for $FF3$	Risk-adjusted residual returns for the 3 Fama-French Factors
`Residual_Momentum_FF5`	$\tilde{R}_i^{risk.adj}$ for $FF5$	Risk-adjusted residual returns for the 5 Fama-French Factors

Strategy 8: Pairs Trading

This strategy attempts to capture mispricing from a pair of historically correlated stocks. However, it is disregarded here as the stock universe keeps changing each month. However, it is extended to clusters, as explained below.

Strategy 9: Mean-Reversion-Single and Multiple Clusters

Mean-reversion strategies, or *contrarian* strategies, assume that prices of correlated stocks follow an underlying trend and prices would eventually revert if stocks are temporarily under- or over-priced with respect to the other related securities. Consequently, the strategy properties are opposite to those of momentum factors (Jansen, 2018) and are associated with market over-reaction (Avellaneda & Lee, 2010). It consists then to buy the underpriced securities and sell the overpriced ones. Let t denote time measured in units of 1 day, with $t = 0$ corresponding to

the most recent time; $P_i(t)$ is the time series of adjusted closing prices for a stock i; $R_i(t)$ is the logarithmic return computed over the T-day formation:

$$R_i(t) = \ln\left(\frac{P_i(t)}{P_i(t+T)}\right)$$

Single Cluster

For a single cluster of highly correlated stocks, the demeaned return $\tilde{R}_i$ is derived by removing the cluster's average return $\overline{R}$:

$$\tilde{R}_i = R_i - \overline{R}$$

The predictors are subsequently computed by subtracting returns from benchmarks for $T \in \{10,21\}$ as shown inTable F.8:

Table F.8 Single cluster variants.

Variable name	Value	Description
`De-meaned_10D_R1000`	$\tilde{R}_i^{R1000}$ for $T=10$	10-days demeaned returns from the Russell 100 Index
`De-meaned_10D_SP500`	$\tilde{R}_i^{SP500}$ for $T=10$	10-days demeaned returns from the S&P500
`De-meaned_21D_R1000`	$\tilde{R}_i^{R1000}$ for $T=21$	21-days demeaned returns from the Russell 100 Index
`De-meaned_21D_SP500`	$\tilde{R}_i^{SP500}$ for $T=21$	21-days demeaned returns from the S&P500

Multiple Clusters

The previous strategy can be extended to several clusters. For K clusters of historically highly correlated stocks (such as industry group), labelled $A = 1,...,K$, N stocks can be associated through the $N \times K$ binary loading matrix Λ where $\Lambda_{i,A} = 1$ for stock i belonging to cluster A; otherwise $\Lambda_{i,A} = 0$, such that:

$$N_A = \sum_{i=1}^{N} \Lambda_{i,A} > 0$$

G is the map between stocks and clusters:

$$G : \{1,...,N\} \to \{1,...,K\}$$

The set J_A comprises the stocks belonging to cluster A such that:

$$J_A = \left\{ i \middle| G(i) = A \right\} \subset \left\{ 1,...,N \right\}$$

$\overline{R}_A$ is the mean return for a cluster $A = G(i)$:

$$\overline{R}_A = \frac{1}{N_A} \sum_{j \in J_A} R_j$$

The demeaned return $\tilde{R}_i$ is obtained by subtracting the cluster mean return:

$$\tilde{R}_i = R_i - \overline{R}_{G(i)}$$

The predictors are subsequently computed with clusters defined as the five different levels of the Bloomberg Industry Classification System (BICS)[38] for $T \in \left\{ 10,21 \right\}$ as shown in Table F.9:

Table F.9 Multiple clusters variants.

Variable name	Value	Description
`Demeaned_10D_BICS1`	$\tilde{R}_i^{BICS1}$ for $T = 10$	10-days demeaned returns from BICS level 1
`Demeaned_10D_BICS2`	$\tilde{R}_i^{BICS2}$ for $T = 10$	10-days demeaned returns from BICS level 2
`Demeaned_10D_BICS3`	$\tilde{R}_i^{BICS3}$ for $T = 10$	10-days demeaned returns from BICS level 3
`Demeaned_10D_BICS4`	$\tilde{R}_i^{BICS4}$ for $T = 10$	10-days demeaned returns from BICS level 4
`Demeaned_10D_BICS5`	$\tilde{R}_i^{BICS5}$ for $T = 10$	10-days demeaned returns from BICS level 5
`Demeaned_21D_BICS1`	$\tilde{R}_i^{BICS1}$ for $T = 21$	21-days demeaned returns from BICS level 1
`Demeaned_21D_BICS2`	$\tilde{R}_i^{BICS2}$ for $T = 21$	21-days demeaned returns from BICS level 2
`Demeaned_21D_BICS3`	$\tilde{R}_i^{BICS3}$ for $T = 21$	21-days demeaned returns from BICS level 3

[38] More details in Appendix C

`Demeaned_21D_BICS4`	$\tilde{R}_i^{BICS4}$ for $T = 21$	21-days demeaned returns from BICS level 4
`De-meaned_121D_BICS5`	$\tilde{R}_i^{BICS5}$ for $T = 21$	21-days demeaned returns from BICS level 5

Strategy 10: Mean-Reversion–Weighted Regression

The previous strategy assumed a binary loading matrix and can be extended to any non-binary matrix. However, it is disregarded here as no relevant inputs could be found.

Strategy 11: Single Moving Average

This strategy and the following ones belong to the field of *technical analysis* which (Murphy, 1999) define as the forecasting of future prices trends, primarily with charts, with three principal sources of information available: price, volume and open interest. The early principles have been compiled initially at the end of 19th century by Charles H. Dow in the Dow theory[39], and can be summarised with the following three main assumptions:

- The market discounts everything and prices might be affected politically, fundamentally, psychologically, and others.
- The price move in trends and can be identified with trend-following indicators.
- The history repeats itself, and past successful patterns should continue based on human psychology.

Despite its success, the field remains controverted among academics (Brock, Lakonishok, & LeBaron, 1992) and provides to be less effective in more extended frequency trading strategies in opposition to *fundamental analysis*. As an example, (Hsu & Kuan, 2005) examine the profitability of such strategies in different markets and add proof to its lack of effectiveness in mature markets.

The Moving Averages (MA) are one of the most widely used technical indicators. In the statistics field, it is a type of finite impulse response filters used to analyse a set of data points by

[39] Wikipedia: https://en.wikipedia.org/wiki/Dow_theory

creating a series of averages of different subsets of the full data set. A moving average is commonly used with time-series data to smooth out short-term fluctuations and highlight longer-term trends or cycles. The threshold between short-term and long-term depends on the application, and the parameters of the moving average are set accordingly. Let t denote time measured in units of 1 day, with $t=1$ corresponding to the most recent time; $P(t)$ is the time series of adjusted closing prices for a stock; $SMA(T)$ is the simple moving average of length T-day:

$$SMA(T)=\frac{1}{T}\sum_{t=1}^{T}P(t)$$

$EMA_i(T)$ is the exponential moving average from the observation window:

$$EMA(T)=\frac{1-\lambda(T)}{1-\lambda(T)^T}\sum_{t=1}^{T}\lambda(W)^{T-1}P(t)$$

Where the smoothing factor $\lambda(T)$ can be derived from T[40]:

$$\lambda(T)=\frac{2}{T+1}\quad \text{for T}\geq 1$$

The strategy is finally based on the stock price crossing the moving average:

$$\text{Signal}=\begin{cases}\text{Establish long/liquidate short position if } P>MA(T)\\ \text{Establish short/liquidate long position if } P<MA(T)\end{cases}$$

The predictors are subsequently computed for $T\in\{30,50,100,200\}$ as shown inTable F.10:

Table F.10 Single moving average variants.

Variable name	Value	Description
`Signal_SMA_30D`	Signal_{SMA} for $T=30$	Signals for 30-days SMA
`Signal_SMA_50D`	Signal_{SMA} for $T=50$	Signals for 50-days SMA
`Signal_SMA_100D`	Signal_{SMA} for $T=100$	Signals for 100-days SMA
`Signal_SMA_200D`	Signal_{SMA} for $T=200$	Signals for 200-days SMA
`Signal_EMA_30D`	Signal_{EMA} for $T=30$	Signals for 30-days EMA
`Signal_EMA_50D`	Signal_{EMA} for $T=50$	Signals for 50-days EMA
`Signal_EMA_100D`	Signal_{EMA} for $T=100$	Signals for 100-days EMA

[40] Exponentially weighted windows: https://pandas.pydata.org/pandas-docs/stable/user_guide/computation.html#exponentially-weighted-windows

`Signal_EMA_200D`	Signal_{EMA} for $T = 200$	Signals for 200-days EMA

Strategy 12: Two Moving Averages

The previous strategy can be extended to two moving averages. For two lengths T_1 and T_2 where $T_1 < T_2$, the signal is given by:

$$\text{Signal} = \begin{cases} \text{Establish long/liquidate short position if } MA(T_1) > MA(T_2) \\ \text{Establish short/liquidate long position if } MA(T_1) < MA(T_2) \end{cases}$$

The predictors are subsequently computed for 30, 50, 100 and 200 days as shown in Table F.11:

Table F.11 Two moving average variants.

Variable name	Value	Description
`Signal_2MA_10_30D`	Signal for $T_1 = 10,\ T_2 = 30$	Signals for 10/30 days SMAs
`Signal_2MA_10_50D`	Signal for $T_1 = 10,\ T_2 = 50$	Signals for 10/50 days SMAs
`Signal_2MA_30_50D`	Signal for $T_1 = 30,\ T_2 = 50$	Signals for 30/50 days SMAs
`Signal_2MA_30_100D`	Signal for $T_1 = 30,\ T_2 = 100$	Signals for 30/100 days SMAs
`Signal_2MA_30_200D`	Signal for $T_1 = 30,\ T_2 = 200$	Signals for 30/200 days SMAs
`Signal_2MA_50_100D`	Signal for $T_1 = 50,\ T_2 = 100$	Signals for 50/100 days SMAs
`Signal_2MA_50_200D`	Signal for $T_1 = 50,\ T_2 = 200$	Signals for 50/200 days SMAs
`Signal_2MA_100_200D`	Signal for $T_1 = 100,\ T_2 = 200$	Signals for 100/200 days SMAs

Strategy 13: Three Moving Averages

The previous strategy can be extended to three moving averages. For three simple moving averages with lengths $T_1 < T_2 < T_3$, the signal is given by:

$$\text{Signal} = \begin{cases} \text{Establish long position if } MA(T_1) > MA(T_2) > MA(T_3) \\ \text{Liquidate long position if } MA(T_1) \leq MA(T_2) \\ \text{Establish short position if } MA(T_1) < MA(T_2) < MA(T_3) \\ \text{Liquidate short position if } MA(T_1) \geq MA(T_2) \end{cases}$$

The predictors are subsequently computed, as shown in Table F.12:

Table F.12 Three moving average variants.

Variable name	Value	Description
`Signal_3MA_5_10_20D`	Signal for $T_1 = 5,\ T_2 = 10,\ T_3 = 20$	Signals for 5/10/20 days SMAs
`Signal_3MA_5_20_60D`	Signal for $T_1 = 5,\ T_2 = 20,\ T_3 = 60$	Signals for 5/20/60 days SMAs
`Signal_3MA_10_30_50D`	Signal for $T_1 = 5,\ T_2 = 30,\ T_3 = 50$	Signals for 10/30/50 days SMAs
`Signal_3MA_30_60_120D`	Signal for $T_1 = 30,\ T_2 = 60,\ T_3 = 120$	Signals for 30/60/120 days SMAs
`Signal_3MA_30_120_200D`	Signal for $T_1 = 30,\ T_2 = 120,\ T_3 = 200$	Signals for 30/120/200 days SMAs
`Signal_3MA_60_90_120D`	Signal for $T_1 = 60,\ T_2 = 90,\ T_3 = 120$	Signals for 60/90/120 days SMAs
`Signal_3MA_60_120_200D`	Signal for $T_1 = 60,\ T_2 = 120,\ T_3 = 200$	Signals for 60/120/200 days SMAs
`Signal_3MA_90_120_200D`	Signal for $T_1 = 90,\ T_2 = 120,\ T_3 = 200$	Signals for 90/120/200 days SMAs

Strategy 14: Support and Resistance

Price of security will tend to stop and reverse at a certain level. The strategy here intends to capture such peaks, or *resistance*, and through, or *support*. (Kakushadze & Serur, 2018) provides a strategy for daily frequency trading. It is extended here to a monthly frequency to fit

this paper framework without high expectation. Let t denote time measured in the units of day, with $t=1$ corresponding to the most recent time; $P(t)$ is the time series of closing prices for a stock; B_{up} and B_{down} are the highest and lowest prices from the T-day observation period:

$$B_{up} = \max\left(P(t), P(t+1)\ldots, P(t+T)\right)$$
$$B_{down} = \min\left(P(t), P(t+1)\ldots, P(t+T)\right)$$

The pivot point, or centre, C is:

$$C = \frac{B_{up} + B_{down} + P}{3}$$

The support S and resistance R levels are:

$$R = 2 \times C - B_{down}$$
$$S = 2 \times C - B_{up}$$

The trading signal is defined as follows:

$$\text{Signal=} \begin{cases} \text{Establish long position if P} > C \\ \text{Liquidate long position if P} \le R \\ \text{Establish short position if P} \le C \\ \text{Liquidate short position if P} \le S \end{cases}$$

The predictors are subsequently computed for $T \in \{10,20,30,60,90\}$ as shown in Table F.13:

Table F.13 Support and resistance variants.

Variable name	Value	Description
`Signal_Support_Resistance_10D`	Signal for $T=10$	Signals for 10-days Support Resistance Indicator
`Signal_Support_Resistance_20D`	Signal for $T=20$	Signals for 20-days Support Resistance Indicator
`Signal_Support_Resistance_30D`	Signal for $T=30$	Signals for 30-days Support Resistance Indicator
`Signal_Support_Resistance_60D`	Signal for $T=60$	Signals for 60-days Support Resistance Indicator
`Signal_Support_Resistance_90D`	Signal for $T=90$	Signals for 90-days Support Resistance Indicator

Strategy 15: Channel

The *Donchian* channel indicator (Donchian, 1960) allows for a visualisation of the volatility of a market. It will be relatively narrow if prices are stable and vice-versa. Let t denote time measured in the units of day, with $t = 1$ corresponding to the most recent time; $P(t)$ is the time series of closing prices for a stock; B_{up} and B_{down} are the ceiling and floor from the T-day observation period:

$$B_{up} = \max\Big(P(t), P(t+1)\ldots, P(t+T)\Big)$$
$$B_{down} = \min\Big(P(t), P(t+1)\ldots, P(t+T)\Big)$$

The trading signal is defined as follows:

$$\text{Signal} = \begin{cases} \text{Establish long/liquidate short position if } P = B_{down} \\ \text{Establish short/liquidate long position if } P = B_{up} \end{cases}$$

The predictors are subsequently computed for $T \in \{10,20,30,60,90\}$ as shown in Table F.14:

Table F.14 Channel variants.

Variable name	Value	Description
`Signal_Channel_10D`	Signal for $T = 10$	Signals for 10-days Channel Indicator
`Signal_Channel_20D`	Signal for $T = 20$	Signals for 20-days Channel Indicator
`Signal_Channel_30D`	Signal for $T = 30$	Signals for 30-days Channel Indicator
`Signal_Channel_60D`	Signal for $T = 60$	Signals for 60-days Channel Indicator
`Signal_Channel_90D`	Signal for $T = 90$	Signals for 90-days Channel Indicator

Strategy 16: Event-driven-M&A

This strategy attempts to capture excess returns generated via corporate actions such as Mergers and Acquisitions (M&A). However, it is disregarded here as no relevant data could be found.

Strategy 17: Machine Learning–Single-Stock KNN

The authors (Kakushadze & Serur, 2018)present here a strategy to predict future returns based on moving averages of the price and volume of varying lengths with a *k*-nearest neighbour (*k*-NN) algorithm (Altman, 1992). A less formal description can be found here[41]. As opposed to the authors, a cross-sectional version is implemented here. Let t denote time measured in units of 1 day, with $t=0$ corresponding to the most recent time; $P(t)$ is the time series of closing prices for a stock; $V_i(t)$ is the time series of the volume; $R(t)$ is the daily return. The target variable $y(t)$ is defined as the cumulative return from the next trading T days. To prevent look-ahead bias and remain consistent with the rest of the thesis framework, a skip-day S is added:

$$y(t)=\frac{P(t-S-T)}{P(t)}-1$$

The predictor variables $X_a(t)$, $a=1,...,m$, are the moving averages for volumes, for prices and the volatility for the T_a-day observation window:

$$X_1(t)=\frac{1}{T_1}\sum_{s=1}^{T_1}V(t+s)$$

$$X_2(t)=\frac{1}{T_2}\sum_{s=1}^{T_2}P(t+s)$$

$$X_3(t)=\sqrt{\frac{1}{T_3-1}\sum_{s=1}^{T_3}\left(R(t+s)-\frac{1}{T_3}\sum_{s=1}^{T_3}R(t+s)\right)^2}$$

…

As opposed to training algorithms parallelly for each security i, as the authors suggest, all individual stock datasets $\{(\boldsymbol{X_i},\boldsymbol{y_i})\}_{i=1,...,I}$ are stacked into a single one $(\boldsymbol{X},\boldsymbol{y})$ in order to fit one classifier on the investment universe simultaneously The predictors are further scaled by rolling sample size T_{train} with scaling function φ:

$$\tilde{\boldsymbol{X}}_a(t:t+T_{train})=\varphi(\boldsymbol{X}_a(t:t+T_{train}))$$

[41] A Complete Guide to K-Nearest-Neighbors with Applications in Python and R: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

For an out sample subset at the timestamp t, the k nearest neighbours of $\tilde{\boldsymbol{X}}_a(t)$ among $\tilde{\boldsymbol{X}}_a(t'), t' = t+1, ..., t+T_{train}$ can be found with the Minkowski distance[42]:

$$[D(t,t')]^p = \sum_{a=1}^{m} \left| \tilde{\boldsymbol{X}}_a(t) - \tilde{\boldsymbol{X}}_a(t') \right|^p$$

Where the equation is the Manhattan-distance for $p = 1$, or the Euclidean distance for $p = 2$. Let $\tilde{\boldsymbol{X}}_a(t'_\alpha(t)), \alpha = 1, ..., k$, be the k nearest neighbours of $\tilde{\boldsymbol{X}}_a(t)$. Finally, the predicted return $\hat{\boldsymbol{y}}(t)$ is the weighted corresponding realised values:

$$\hat{\boldsymbol{y}}(t) = \delta(k, \boldsymbol{y}(t'_\alpha(t)))$$

The exercise was repeated, as presented in Table F.15, and the predictor `KNN_Prediction` is subsequently computed, after cross-validations, as:

Table F.15 KNN variants.

Variable name	Value	Description
`KNN_Prediction`	$\hat{\boldsymbol{y}}(t)$ for $\left\{ \begin{array}{c} S = 1 \\ T = 20 \\ m = 9 \\ T_a \in \{21, 63, 126\} \\ T_{train} = 100 \\ \varphi \equiv \psi(\lambda, x) \\ k = 3 \\ p = 1 \\ \delta \equiv \text{'distance'} \end{array} \right\}$	Forecasts of monthly returns with k-NN

Where $\psi(\lambda, x)$ is the Yeo–Johnson transformation[43] (Yeo & Johnson, 2000).

Application

The exercise settings are defined as follows: after computing the moving average of prices, of volumes and volatility for windows of 21-, 63- and 126 days (hence $a = 9$), targets returns are set monthly ($T = 20$) with a skip-day ($s = 1$). One of the major drawbacks of k-NN, is its scalability. For D predictors and the sample size N, the time complexity for the naïve approach

[42] Wikipedia: https://en.wikipedia.org/wiki/Minkowski_distance

[43] `PowerTransformer`: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PowerTransformer.html

can reach $O(DN^2)$ [44]. To mitigate this, the returns at each timestamp are continuously predicted based upon the predictors of the last $T_{train} = 100$ days. The rest of the hyper-parameters are set through `RandomizedSearchCV` with the parameter's distributions set at Table F.16. The regression performances are measured with the Median Absolute Error $MedAE$, the Mean Absolute Error MAE, the Mean Forecast Trading Returns $MFTR$ (Lee, 2007) and the Mean Correct Forecast Directions $MCFD$ (Lee, 2007):

$$MedAE(y,\hat{y}) = median\left(\left|y_1 - \hat{y}_1\right|,...,\left|y_n - \hat{y}_n\right|\right)$$

$$MAE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}\left|y_i - \hat{y}_i\right|$$

$$MFTR(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}\left(y_i - \text{sign}(\hat{y}_i)\right)$$

$$MCFD(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{1}_{\left(\text{sign}(y_i)-\text{sign}(\hat{y}_i)>0\right)}$$

The two first ones are losses to asses predictions range errors and have been chosen as outliers are expected, while the latter ones measure the trading returns and the correctness of side predictions.

Table F.16 Hyperparameters grid.

Symbol	Description	Distributions of parameters
φ	Scaler[45]	$\varphi \in \left\{\begin{matrix}\text{'MinMaxScaler','RobustScaler',}\\ \text{'StandardScaler','QuantileTransformer',}\\ \text{'PowerTransformer','Normalizer'}\end{matrix}\right\}$
k	Number of neighbours	$k \in \{3,5,11,25,40,100\}$
δ	Weight function[46]	$\delta \in \{\text{'uniform','distance'}\}$
p	Power parameter for distance metric	$k \in \{1,2,3,4\}$

he following methodology was used to prevent overfitting:

[44] Nearest Neighbor Algorithms: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbor-algorithms

[45] Preprocessing data: https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-data

[46] `KNeighborsRegressor`: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

Table F.17 Implementation of the hyperparameters research

```
Sample 200 hyper-parameters combination from the parameter grid;
Sample 5 portions of the first half of the dataset;
for each sample do
    Generate 40 splits of training and testing subset with Rep-Hold;
    for each split do
        for each combination of hyper-parameters do
            Fit model on the training subset;
            Predict the testing subset and report performances;
        end
    end
end
Average performances and rank combination of hyper-parameters;
Select the best combination;
```

The average metrics and their standard deviation across resample can be seen for all parameter's combination on Figure F.1. The lowest *MedAE* corresponds already to the lowest *MAE* and highest *MFTR* and highest *MCFD*. Consequently, the parameters combination can be easily selected as there is no presence of trade-offs.

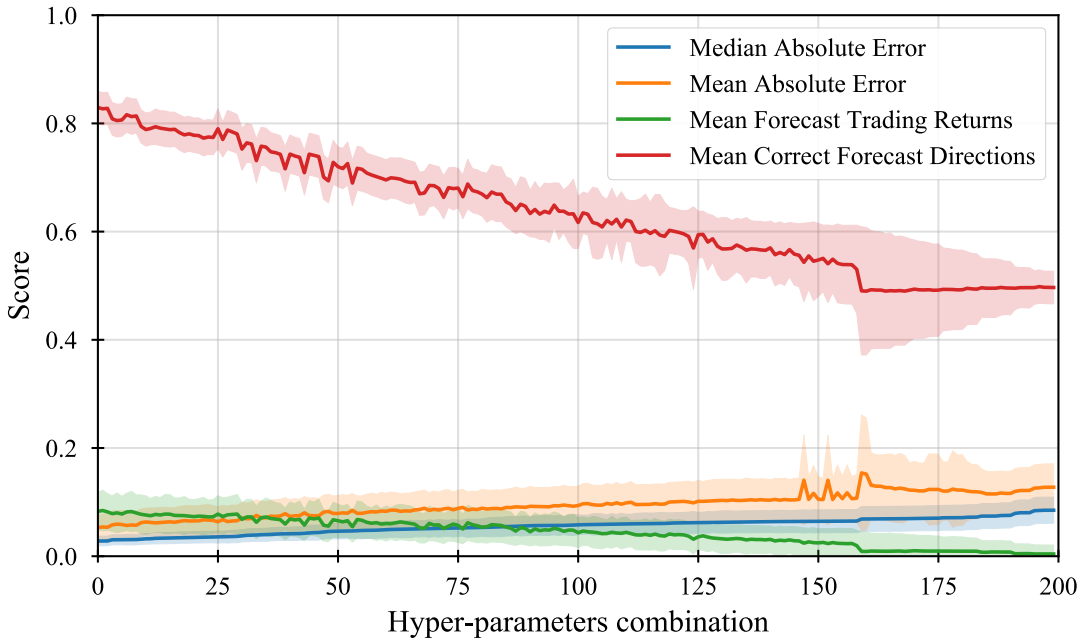

Figure F.1 Cross-Validation performances over hyper-parameters combination

The precedent methodology is biased, as hyper-parameters are not determined on an expanding rolling basis. The lookback windows T_{train} could also have been determined with cross-validation. However, this approach has been chosen for computational reasons (already 80'000 fits on matrices of size $120'000 \times 9$), and it is assumed it would not lead to significant look-ahead bias as the selected parameters continuously rank amongst the first ones across resamples and forecast remains predicted on a rolling basis. Figure F.2 shows the regression metric across time for the test subsets, i.e. forecasted returns. Despite outliers, 85% of side predictions are on average correct, and half of the predicted returns are wrong by less than 0.02 percent-point. Finally, predictions appear meaningful with and without outliers, as depicted by the scatterplots on Figure F.3. It should be remembered that returns have been predicted across all stocks and for the entire time range and thus, will not be all passed to the final learning algorithm. The individuals would be still selected according to their index memberships for each month, and this step is expected to filter outliers from stocks that have been dropped from the index.

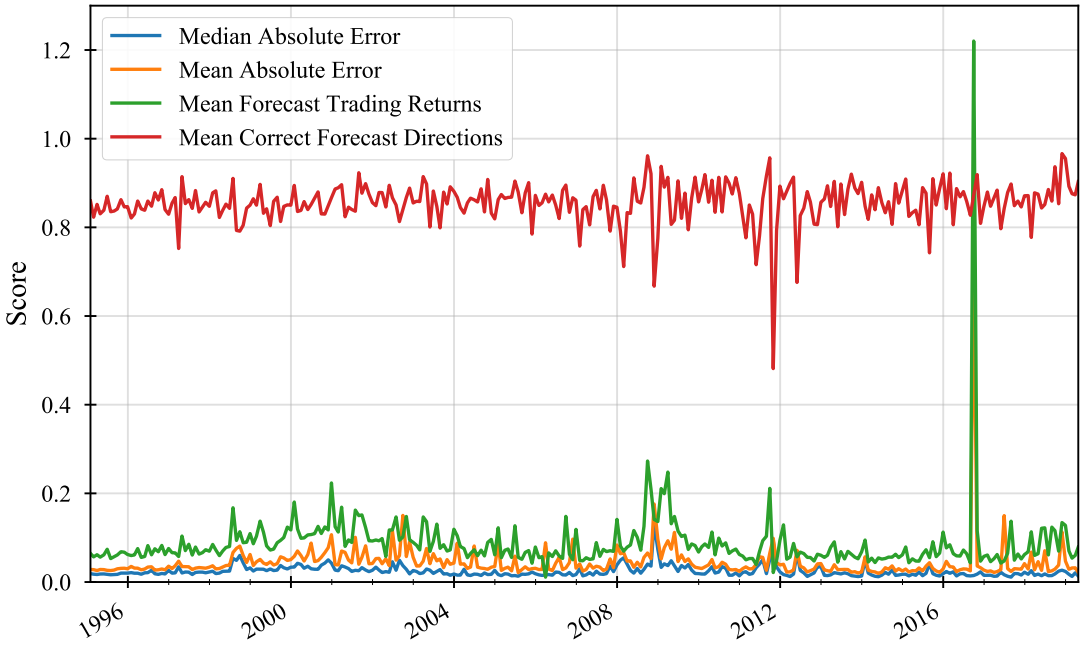

Figure F.2 Forecast performances over months

Besides, a different approach has been taken as *k*-NN can be sensitive to redundancy. Predictors with Pearson correlation coefficient above 0.9 have been dropped (see Figure F.4), and the same methodology was performed. However, this led to slightly worse performances across months such as $MedAE = 0.027 \pm 0.13$, in comparison to 0.024 ± 0.12 previously.

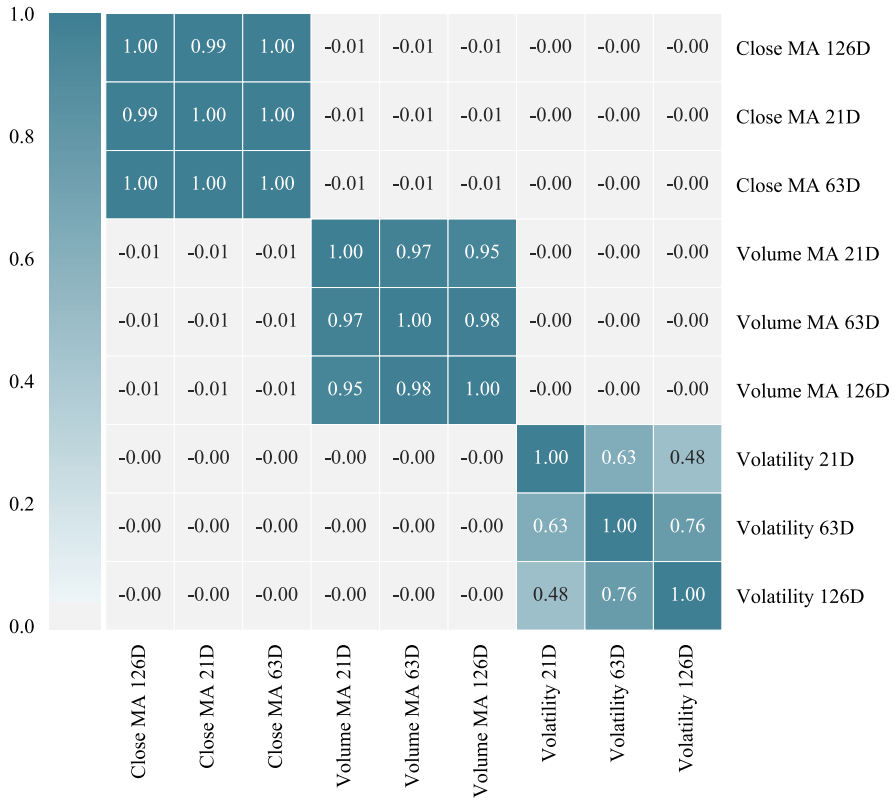

Figure F.4 Visualization of the predictors correlation matrix.

Strategy 18: Statistical Arbitrage-Optimization

This strategy allocates a portfolio requiring its Sharpe ratio to be maximised. However, it is disregarded here. Calculation of weights requires the inversion of a covariance matrix and would not be robust here for the following reasons:

- The investment universe comprises correlated stocks, and the inversion of its covariance matrix would lead to unstable solutions.
- It requires at least $0.5N(N+1)$ of IID observation in order to derive a non-singular covariance matrix of size N. Assuming 800 stocks must be allocated each month (assumed value of the Russell 1000 Index constituents after cleaning), it would require at least 1276 years of daily data (López de Prado, 2018).

Strategy 19: Market-Making

This strategy attempts to capture the bid-ask spread for a given stock (Kakushadze & Serur, 2018). However, it is disregarded here as it does not match with the frequency trading of this paper.

Strategy 20: Alpha Combos

With technological advances, this strategy attempts to mine a big dataset of expected returns, called *alphas* by (Kakushadze & Serur, 2018). On their own, alphas are ephemeral, and any profit would be taken away by trading costs. Therefore, the idea is to combine them into a large *alpha combo strategy*. However, it is disregarded here as:

- As the academic literature is minimal on the formulation of alphas in detail, researching such a large number of alphas is outside of the scope of this thesis.
- While (Kakushadze, 2016) provides formulas for 101 alphas, their average holding period approximately ranges between 0.6 to 6.4 days compared to the monthly framework defined here.
- In addition to the computational complexity, the 101 formulas presented are enigmatic and could compromise the interpretability of the final strategy.

Appendix G: Portfolio Statistics Computation

The following assumptions are defined. Let t denote time measured in units of 1 month for an observed period $t = 1,2,...,T$, with $t = 0$ corresponding to the initialization of the portfolio; N is the annualization factor ($N = 12$); the risk-free rate R_f is assumed to be null; $R_{i,t}$ is the monthly return for a stock labelled i from an investment universe $i = 1,2,...,I$; $R_{B,t}$ is the monthly return of a benchmark (defined here as the Russel 1000 Index); $R_{P,t}$ is the monthly return of the portfolio defined as the weighted average of the returns from the invested assets:

$$R_{P,t} = \sum_i w_{i,t-1} R_{i,t}$$

The portfolio return with transaction costs c (here $c = 0.25\%$ per trade) is defined as (E. P. Chan, 2009):

$$R_{P,t}^C = R_{P,t} - c \sum_i \left| w_{i,t} - w_{i,t-1} \right|$$

The *r*-moment for sample size *n* is given by:

$$m_r = \frac{1}{n} \sum_i (x_i - \bar{x})^r$$

The unbiased cumulant estimates as reported in (Harald, 1946):

$$k_2 = \frac{n}{n-1} m_2$$

$$k_3 = \frac{n^2}{(n-1)(n-2)} m_3$$

$$k_4 = \frac{n^2}{(n-1)(n-2)(n-3)} \left\{ (n+1) m_4 - 3(n-1) m_2^2 \right\}$$

Finally, the metrics from Table G.1 are derived (DeMiguel, Garlappi, Nogales, & Uppal, 2009; Goodwin, 1998; Meucci, 2010; Sharpe, 1964, 1966; Sortino & Price, 1994; Young, 1991). For metrics annualization, (Goodwin, 1998) compares four practices: usage of arithmetic-, geometric-, continuously compunded-mean and frequency-converted data. To remain consistant across metrics computations, the arithmetic mean is used.

Table G.1 Portfolio Statistics

Portfolio Statistics	Formula
General	
Turnover	$Turnover_{year} = \frac{N}{T}\sum_{t=1}^{T}\sum_{i=1}^{I}\left\|w_{i,t} - w_{i,t-1}\right\|$
Performance	
Cumulative return	$CR_t = \left(\prod_{i=1}^{t}\left(1+R_i\right)\right)-1$
CAGR	$CAGR = \left(\prod_{i=1}^{T}\left(1+R_i\right)\right)^{\frac{P}{T}}-1$
Alpha	$\alpha_{yearly} = \bar{\alpha}N$ where $\alpha_t = R_t - \beta R_{B,t}$ for $t = 1,...,T$
Beta	$\beta = \frac{\mathrm{cov}(R, R_B)}{\mathrm{var}(R_B)}$
Stability	R^2 from $\ln(CR)_t = a + bt$ for $t = 1,...,T$
Hit Ratio	$HR = \sum_t\sum_i \frac{\mathbf{1}_{(w_{i,t-1}R_{i,t}>0)}}{\mathbf{1}_{(w_{i,t-1}>0)}}$
Average return from hits	$\frac{\sum_t\sum_i \mathbf{1}_{(w_{i,t-1}R_{i,t}>0)} w_{i,t-1}R_{i,t}}{\sum_t\sum_i \mathbf{1}_{(\left\|w_{i,t-1}R_{i,t}\right\|>0)}}$
Average return from misses	$\frac{\sum_t\sum_i \mathbf{1}_{(w_{i,t-1}R_{i,t}<0)} w_{i,t-1}R_{i,t}}{\sum_t\sum_i \mathbf{1}_{(w_{i,t-1}R_{i,t}<0)}}$
Risk	
Volatility	$\sigma_{year} = k_2^{1/2}N^{1/2}$
Skewness	$g_{1,year} = \frac{k_3}{k_2^{3/2}}\frac{1}{N^{1/2}}$
Kurtosis	$g_{2,year} = \frac{k_4}{k_2^2}\frac{1}{N}$
Maximum drawdown	$MD = \max_{\tau\in(1,T)}(\max_{t\in(1,\tau)}(R_t - R_\tau))$

Efficiency

Sharpe ratio $$SR_{year} = \frac{k_1}{k_2^{1/2}} N^{1/2}$$

Information ratio $$IR_{year} = \frac{k_1 (R_p - R_B)}{k_2^{1/2} (R_p - R_B)} N^{1/2}$$

Sortino ratio $$Sortino_{year} = \frac{k_1}{dsr} N^{1/2} \quad \text{where } dsr = \sqrt{\frac{1}{T-1} \sum_t \min(0, R_t)}$$

Calmar ratio $$Calmar_{year} = \frac{k_1}{MD} N$$

Tail ratio $$Tail\ ratio = \frac{\left| \hat{F}^{-1}(0.95) \right|}{\left| \hat{F}^{-1}(0.05) \right|} \quad \text{where } \hat{F}^{-1}(x) \text{ is the } quantile\ function$$

Appendix H: AutoML and Deep Learning

As the process of designing and validating a machine learning pipeline can be time-consuming, the question of whether other faster-implemented tools could be used to compete with the work achieved so far has been raised. As such predictive performances are compared with AutoML and deep learning techniques.

AutoML

As machine learning pipeline performances can be very sensitive to numerous decisions, the field of automated machine learning (AutoML) emerged to ease the workflow of ML practitioners. These systems aim to automate the entire machine learning pipeline in a data-driven way: feature pre-processing, feature selection, model selection and hyper-parameters tuning. These techniques are now mature enough to compete with and even outperform human-machine learning experts (Hutter, Lars, & Joaquin, 2019). As such, Auto-sklearn and TPOT are used here. While Auto-sklearn optimizes pipeline decision with Bayesian optimization, TPOT relies on Genetic Algorithm (GA). In addition to be very computationally demanding, the drawbacks of using such tools is their parameters complexity and their lack of flexibility as they provide to be only wrappers of other machine learning libraries such as Scikit-learning. As such, sample weights can be passed only to the estimator using TPOT, but not for Auto-sklearn. They also both imputes automatically missing values using the median.

Deep learning

As opposed to shallow learning, deep learning depends less on the representation of data they are given and can avoid the time-consuming step of feature hand-designing (Goodfellow, Bengio, & Courville, 2016). For this reason, a deep learning implementation has been considered here. Among the family of Artificial Neural Networks (ANN), the Recurrent Neural Networks (RNN) (Rumelhart, Hinton, & Williams, 1986) are a group that can model the relationship between sequences instead of considering each sample independently and are thus suitable for application in Natural Language Processing (NLP) or Time Series. A significant issue with early RNNs implementations was their *vanishing gradient* problem. Neural networks work mainly by adjusting their internal weights by *backpropagating* the gradient computed from the loss function. Vanishing gradient illustrates the case where neural networks cannot converge as gradient drops or explodes. As solution, emerged the sub-group of Gated RNN, such as Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). In addition to share weights

across sequences, Gated RNNs include an internal gates mechanism to accumulate information and forget it when useful. Finally, the choice of using an LSTM has also been motivated by (Ryll & Seidens, 2019) for LSTM overall better performances for stock forecasting in comparison to more classic ANNs. For more information about LSTM architecture, the reader should consider (Goodfellow et al., 2016) or this less formal description[47]. The LSTM is implemented with Keras, a high-level API running on top of TensorFlow. This set up enables fast experimentation in a research environment, instead of relying directly on TensorFlow, more suitable for a production environment. For faster training of the LSTM, a GPU has been used. As deep learning involves mainly matrix operations computation, GPU offers a solution for scalability due to their architectures. For other ends than video processing, computing with GPU is referred as General-Purpose GPU programming (GPGPU). Finally, the content of (Chollet & Others, 2015; Ng, 2019) has been followed for deep learning development good practices.

[47] Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/