ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

DAY TRADING CLUSTERS BUILT FROM MULTI-FACTOR MODELS

FINANCIAL BIG DATA

William Martin

13th January 2021

Contents

Introduction **2**

Data **3**
2.1 Data collection 3
2.2 Exploratory data analysis 4
2.3 Data cleaning 6

Methodology **7**
3.1 Multi-factor model 7
3.2 Clustering 8
3.3 Trading strategy 8
3.4 Summary 9

Results **10**
4.1 Discussion 10

Conclusion **13**

Appendix **14**

INTRODUCTION

Building portfolios that outperform a certain benchmark while being exposed to limited risk has always been a challenge for portfolio managers, even more so when client constraints, sector-related or ESG-related (Environmental, Social, Governance), are introduced.

Nowadays, financial institutions each have their own methodology of investment and trading, ranging from qualitative to quantitative and pertaining to different technical and technological features. In this project, we would like to revisit the Capital Asset Pricing Model (CAPM) and more particularly its extended versions: the multi-factor models proposed by Fama and French [1][2] and Carhart [3]. These models all arise from the findings of Stephen Ross [4], which claim that excess returns are generated by a K-factor model

$$R_j^e = \alpha_j + \beta_{j1}F_1 + \beta_{j2}F_2 + ... + \beta_{jK}F_K + \epsilon_j \tag{1.1}$$

This is the Arbitrage Pricing Theory (APT) and it says in other words that expected returns are generated by exposures (factor betas) to several sources of risk times factor risk-premia [5]. In this prospect, we would like to explore the power of multi-factor models and determine if the factor betas can be categorized in order to build outperforming portfolios from the underlying securities. This categorization will be done using clustering algorithms and we will propose a simple day-trading long-only strategy of a portfolio built upon the clustering of these factor betas. A rolling-window calibration will be deployed in order to take into account the non-stationary nature of the stock market and annualized metrics will be computed in order to assess and compare the trading strategy using different combinations of factor models and clustering algorithms. This way, we hope to determine if an intelligently constituted portfolio can outperform the market or other cluster-related portfolios (e.g. sector-based portfolios).

DATA

For this project, we will consider a time frame between the 1st of January 2010 to the 31st of December 2019, totaling 10 years and the universe of the Russell 3000 to build our portfolios. This chapter describes the process of collecting, analyzing, and cleaning the data that will be needed throughout our study. We will also go through a quantitative study to assess the quality of the data collected.

2.1 DATA COLLECTION

We have almost solely made use of the Center for Research in Security Prices, LLC (CRSP) and the Fama-French Portfolios Factors databases through the Wharton Research Data Services (WRDS) [6]. *EPFL* credentials give free access to standard data such a stock prices, stock returns, number of shares outstanding, and basic company information. Across all tables of CRSP, the PERMNO feature is used as unique identifier for each security. Indeed, "PERMNO is a unique five-digit permanent identifier assigned by CRSP to each security in the file" [6].

The first step is to identify the constituents of the Russell 3000 index, which tracks the performance of the 3000 largest U.S.-traded stocks. Even though in reality the stocks of the index are reconstituted once a year [7], we will build a custom version of the Russell 3000 on a monthly basis. To do this, we query the stock price and number of shares outstanding of every security of the database at every end of month between 01.01.2010 and 31.12.2019, and compute the market capitalization as follows

$$MCAP_{t,p} = PRC_{t,p} \times SHROUT_{t,p} \quad \forall p \in P, \forall t \in T_{end} \tag{2.2}$$

where we denote P the set of all PERMNOs available on the database, T_{end} the end of months between 2010 and 2019, and finally $PRC_{p,t}$ and $SHROUT_{p,t}$ respectively the share price and number of shares outstanding of PERMNO/security p at time t. We then select the 3000 largest MCAPs and construct a matrix of binary values[1], which we will denote

$$S \in \mathbb{B}^{|T_{end}| \times |P|} \tag{2.3}$$

where $\mathbb{B}$ is the domain of Boolean values and $|T_{end}| = 120$. $S_{t,p} = 1$ if security p is a constituent of the Russell 3000 during month t and $S_{t,p} = 0$ otherwise. By discarding securities which have never entered the Russell 3000, we end up with $|P| = 5743$, thus saving space on memory. The results are stored in a zipped *.csv* file. From this point on, all references to the Russell 3000 express the custom-built Russell 3000.

Next, using all 5743 PERMNOs we acquire metadata[2] of each company such as company name, ticker symbol, and the North American Industry Class System (NAICS). The latter is a six-digit number where each digit gives information on the sector, the economic activity, and the industry group of the company [8]. The results are stored in a zipped *.csv* file.

The third category of data that needs to be collected is the Fama-French 5 factors [9]. This includes SMB (Small Minus Big), HML (High Minus Low), RMW (Robust Minus Weak), and CMA (Conservative

[1] Refer to *main_constituents.py* for the implementation
[2] Refer to *main_info.py* for the implementation

Minus Aggressive) factors as well as $R_m - R_f$: the excess return on the market[3]. Moreover, we also download the Momentum Factor (MOM) as well as the Short-Term and Long-Term Reversal Factors (ST and LT Rev). The results are stored in a zipped *.csv* file.

Finally, we query the daily returns of all stocks of interest from the WRDS data services. Since this will be the bulk of the data and that we plan on computing the risk loadings of a multi-factor model of each security on a month-wide sliding window, we opted in querying the returns on a monthly basis only of securities that make up the custom Russell 3000 (which is, we recall, reconstituted on a monthly basis as well). We also download returns of securities a month prior of becoming a constituent since we need a month-long window of historical data (as depicted in Figure 2.1).

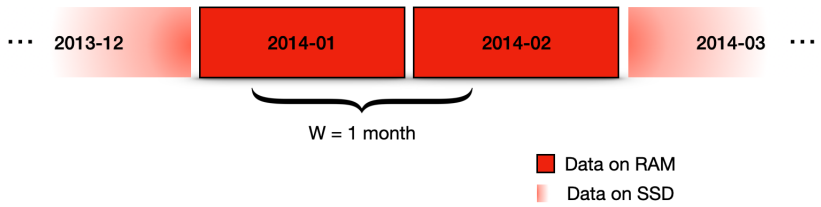

FIGURE 2.1
Data on RAM vs. SSD strategy.

Proceeding this way, we will limit the number of file readings and have only the necessary data on RAM when performing the rolling windows. Indeed, it is better to read 120 files (10 years times 12 months) rather than 5743 corresponding to the constituents. The results are saved in *parquet* files. Since we are dealing with files with a large number of columns, it is advantageous to use a column-oriented file format such as *parquet* for faster reading and writing.

2.2 EXPLORATORY DATA ANALYSIS

In this section, we explore the collected data and assess their quality[4]. Figure 2.2 depicts a heatmap of the binary matrix S (equation 2.3). Obviously, there are only values of 0s and 1s and all rows sum up to 3000. We observe an increasing number of companies entering the Russell 3000 and maintaining their status as a constituent on the long run. This explains the denser regions of 1s in the beginning of 2010 and denser regions of 0s towards the end of the time frame. This effect can be simply explained by the fact that new companies are being created and growing in value over time.

[3]Refer to *main_fama.py* for the implementation
[4]Refer to the *Jupyter Notebook eda.ipynb* for the implementation

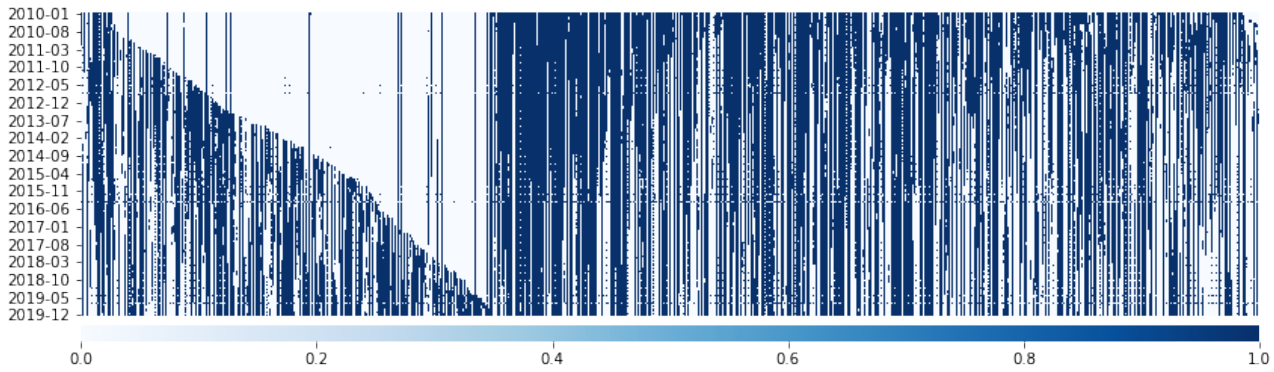

FIGURE 2.2
Binary matrix C of custom Russell 3000 constituents. X-axis: securities, Y-axis: month.

Figure 2.3 depicts a heatmap of missing values of daily returns for every month and security. We observe an interesting and logical phenomenon: valuable companies that have entered the Russell 3000 have done so by having announced an Initial Public Offering (IPO). Therefore, the day these companies enter the market, they will immediately enter the Russell 3000. Missing values are maximal (according to the colorbar limits) a month before entering the market since we have decided to download an additional month of historical data as per the explanation depicted in Figure 2.1. Therefore, these are not actually missing data from the database but data that simply do not exist. During the month of actually entering the Russell 3000 the number of missing values decreases (this depends if a company entered at the beginning or the end of the month). Overall and not taking into account the aforementioned phenomenon, there are very little missing values. Indeed, WRDS is known to host reliable and high quality data.

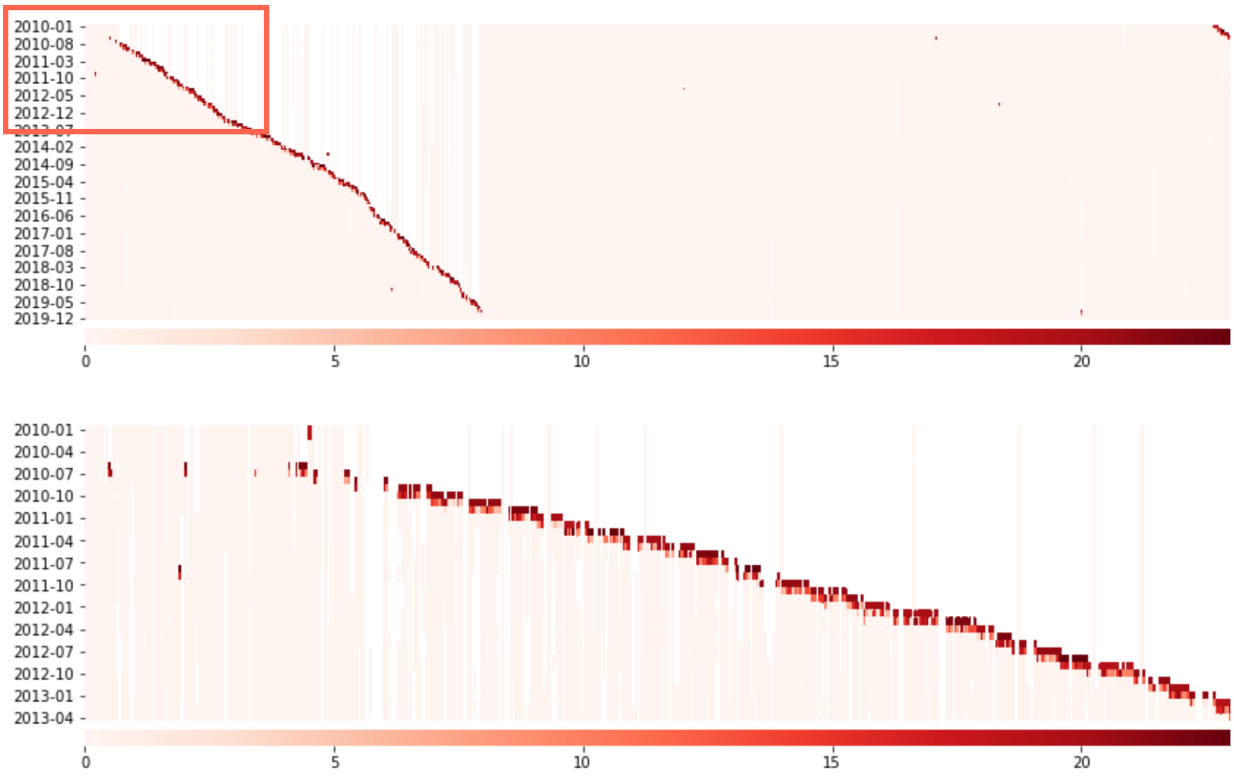

FIGURE 2.3
Missing values. X-axis: securities, Y-axis: month. Top: overall view. Bottom: zoomed version. White zones indicate no available data (data not downloaded if a company is not a Russell 3000 constituent).

Excluding the artificially added missing values, Figure 2.4 depicts the evolution of all missing values over the 3000 constituents over time (the constituents change every month). The evolution of missing values is random, i.e. there is not a particular period where there are more missing values than another, thus it is safe to assume that there are no major issues with the data collected and the database itself. Indeed, the average number of missing values per constituent per month is a mere 0.033. This result is clearly visualized in Figure 2.3 by the lack of dark regions.

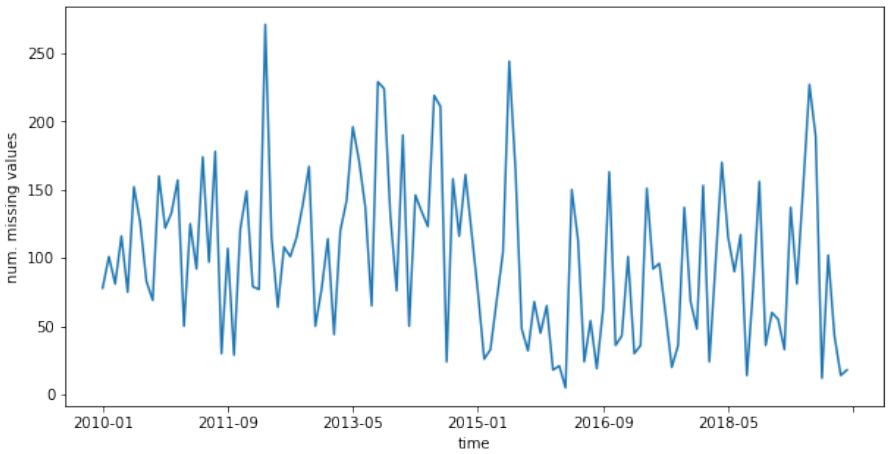

FIGURE 2.4
Evolution of missing values over time across all constituents of the Russell 3000.

In order parallelize processes, the *Python* library *Dask* [10] was used. For instance, using *Dask* to count missing values halved the processing time compared to running sequentially the code. Figure 2.5 depicts the task graph of counting missing values of returns. For visualization purposes, only 3 months of data was used but we can easily see that parallelism is very much possible.

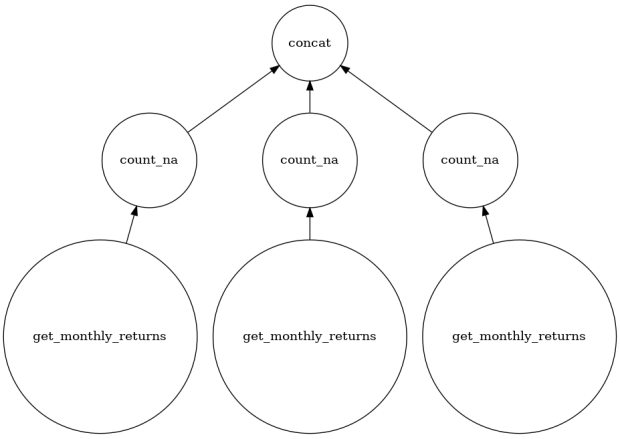

FIGURE 2.5
Visualization of the underlying task graph of counting the number of missing values of 3 months of data

2.3 DATA CLEANING

Missing values (artificially added included) represent approximately 0.25% of the total number of datapoints and the bulk of these comes from data that does not exist. Thus, we choose to omit data imputation and choose to not consider them into our methodology. Normally, small intermittent missing values of returns are replace by 0s since the most common method of replacing missing stock prices is a forward fill.

METHODOLOGY

3.1 MULTI-FACTOR MODEL

We will use multi-factor models as a way to explain stock returns. By doing this, we hope to discover which factors have the most impact on the returns of a security and assess if assets can be grouped according to their factor loadings. We consider the following multi-factor models that are built upon the well-known Capital Asset Pricing Model (CAPM)

$$R^e_{t,p} = \alpha_{t,p} + \beta^1_{t,p} R^e_{t,M} + \beta^2_{t,p} SMB_t + \beta^3_{t,p} HML_t + \epsilon_t \tag{3.4}$$

$$R^e_{t,p} = \alpha_{t,p} + \beta^1_{t,p} R^e_{t,M} + \beta^2_{t,p} SMB_t + \beta^3_{t,p} HML_t + \beta^4_{t,p} RMW_t + \beta^5_{t,p} CMA_t + \epsilon_t \tag{3.5}$$

$$R^e_{t,p} = \alpha_{t,p} + \beta^1_{t,p} R^e_{t,M} + \beta^2_{t,p} SMB_t + \beta^3_{t,p} HML_t + \beta^4_{t,p} RMW_t + +\beta^5_{t,p} CMA_t + \beta^6_{t,p} MOM_t + \epsilon_t \tag{3.6}$$

$$R^e_{t,p} = \alpha_{t,p} + \beta^1_{t,p} R^e_{t,M} + \beta^2_{t,p} SMB_t + \beta^3_{t,p} HML_t + \beta^4_{t,p} RMW_t + +\beta^5_{t,p} CMA_t + \beta^6_{t,p} MOM_t \tag{3.7}$$

$$+ \beta^7_{t,p} STRev_t + \beta^8_{t,p} LTRev_t + \epsilon_t \tag{3.8}$$

$\forall t \in T, \forall p \in P(t)$ and where $R^e_{t,p} = R_{t,p} - R^f_t$ and $R^e_{t,M} = R_{t,M} - R^f_t$ are respectively the excess returns on security p and the market M. The set T contains the days from the 1st of February 2010 to the 31st of December 2019. The first month of January is omitted since one month of data is needed to compute the first loadings. The set $P(t)$ contains the 3000 constituents at time t.

The first model proposed by Fama-French (equation 3.4) controls for market exposure and standard factors that capture the effects of size and value [1]. The second model (equation 3.5) is the 5-factor asset pricing model identified again by Fama-French more recently [2] where two additional factors (Robust-minus-Weak and Conservative-minus-Aggressive) are controlled by the regression. The third model proposed by Carhart [3] (equation 3.6) adds the momentum factor (MOM), which is a strategy of buying recent "winner" stocks and shorting recent "loser" stocks. Finally, the fourth model (equation 3.8) adds two more factors. The first is the short-term reversal factor identified by Schaumburg et al. [11], where they have identified that stocks performing poorly on the shorter-term horizon tend to better perform in the future. The long-term reversal factor is similar to its short-term counterpart but is characterized by a longer-term horizon than in the MOM factor.

The regression coefficients and intercept are computed using ordinary least squares (OLS) minimization[5] implemented in *Sklearn* [12], which is more robust but less fast than the method of *numpy*, *lstsq*. The computation is performed for every security that is a constituent of the Russell 3000 for every $t \in T$ using one month of historical data (t-1 day to t-1 month). A sliding month-wide window is used to accomplish this according to Figure 2.1, while only maintaining two months of daily returns

[5]Refer to the *Jupyter Notebook risk_model.ipynb* for the implementation

on RAM. A *parquet* file is written on disk for every time t and model m and contains a matrix that we denote $L_{t,m} \in \mathbb{R}^{n_{feat,m} \times 3000}$, where $n_{feat,m}$ is the number of factors of model m plus one for the intercept term. This results in data of regression loadings that have a memory size that is approximately 70-fold bigger than that of all the daily returns (taking into account all four models). Here, we also make use of *Dask* in order to carry out multi-processing.

3.2 CLUSTERING

Once the regression loadings $L_{t,m}$ have been computed, clustering is performed[6]. The goal here to group securities together by their regression coefficients into K clusters. For algorithms that require a predefined number of clusters K we chose $K = 24$, which is the number of sectors present in our dataset and also the numbers of existing sectors according to the North American Industry Classification System (NAICS) [8]. To our end, we make heavy use of the algorithms implemented in *Sklearn*. A summary of each algorithm used in our experiments is depicted in Table 3.1 [13].

Method name	Parameters	Geometry (metric used)
K-Means	number of clusters	Distances between points
Mean-shift	bandwidth	Distances between points
Spectral clustering	number of clusters	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Distances between points

TABLE 3.1
Summary of clustering algorithms used

For each clustering algorithm g, and model m, clustering is performed on the transpose of the matrix of loadings $L^T_{t,m}$ $\forall t \in T$ and we produce the resulting matrix $C_{m,g} \in \mathbb{N}^{+,|P|\times|T|}$, where $|P| = 5743$ and T is the set of days starting from the 1st of February 2010. The elements of the matrix are clusters/labels associated to security p at time t. Obviously, the matrix $C_{m,g}$ will contain NaNs since only 3000 securities enter the clustering algorithm at every time t. Each clustering result $C_{m,g}$ is saved in a *parquet* file. Again, we make use of *Dask* to speed up the lengthy process.

3.3 TRADING STRATEGY

The goal in this section is to implement a long-only trading strategy based on a trading signal[7]. The universe of the strategy is of course the Russell 3000 and the trading signal will allow us to select a cluster that will constitute the portfolio to long. We choose a simple signal

$$c^*_t = \underset{c}{\operatorname{argmax}}\, R^{EW}_{c,t-1} \tag{3.9}$$

where $R^{EW}_{c,t-1}$ is the return at time $t-1$ of the equally-weighted portfolio consisting of securities of cluster c. Thus at every time t, we retrieve the appropriate column of the matrix $C_{m,g}$ (containing the cluster associated to each security at time t), compute the equally-weighted returns of each cluster using returns of securities at $t-1$ and take the cluster c^*_t corresponding to the best portfolio return. The constituents of c^*_t thus allows to construct a portfolio for time t. Since we are using returns at time

[6]Refer to the *Jupyter Notebook clustering.ipynb* for the implementation
[7]Refer to the *Jupyter Notebook trading_strategy.ipynb* for the implementation

$t-1$ and since the regression loadings are computed using historical data, there is no look-ahead bias in the strategy.

Moreover, we assume the following strategy setting for conducting comparable simulations:

- Start with an initial capital of $\$1.00$ on the 1st of February 2010.
- Re-balance on a daily basis and fully re-invest.
- No transaction costs

Finally, we implement and simulate the trading strategy for all matrices $C_{m,g}$, i.e. for all models m and clustering algorithms g.

3.4 SUMMARY

A pipeline of the methodology and the flow of data for each choice of factor model m and clustering algorithm g is depicted in Figure 3.6.

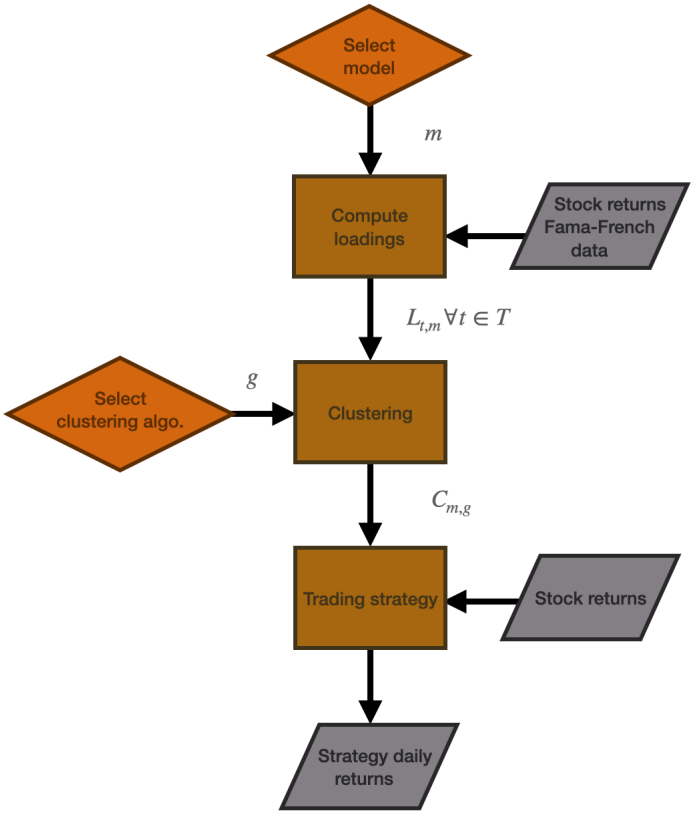

FIGURE 3.6
Flowchart of methodology

RESULTS

The pipeline of Figure 3.6 is applied for all models proposed in section 3.1 and clustering algorithms listed in Table 3.1, resulting in a time series of daily returns of the trading strategy over 10 years ($\forall m, \forall g$). Starting from an initial capital of $\$1.00$, we can compute the evolution of capital vs. time by simple cumulative product of returns (+1). From here, we can first compute our first metric, the overall gain of the strategy defined as the ratio between the capital at the end of the investment period Cap_T and the beginning Cap_0:

$$G_{m,g}^{overall} = \frac{Cap_T}{Cap_0} \quad \forall m, \forall g \tag{4.10}$$

Then annualized metrics of returns, standard deviation, and Sharpe ratio can be computed as follows

$$R_{m,g}^{ann.} = \left(\prod_{t=0}^{T} (1 + R_{t,m,g}^{net}) \right)^{252/T} - 1 \quad \forall m, \forall g \tag{4.11}$$

$$\sigma_{m,g}^{ann.} = \sigma(R_{t,m,g}^{net}) \sqrt{252} \quad \forall m, \forall g \tag{4.12}$$

$$SR_{m,g}^{ann.} = \frac{R_{m,g}^{ann.} - R^{f,ann.}}{\sigma_{m,g}^{ann.}} \quad \forall m, \forall g \tag{4.13}$$

where R^{net} is the return net of transaction costs and $R^{f,ann.}$ the annualized return of the risk-free rate. We assume there are 252 trading days in a year. The evolution of capital for each model and clustering algorithm is depicted in Figure 6.8 in the appendix. The annualized returns are listed in Table 6.2, the annualized standard deviations in Table 6.3, the annualized Sharpe ratios in Table 6.4, and finally the overall gains in Table 6.5 in the appendix. In our results, we define the following:

- Best cluster: this is the cluster of stocks that verifies equation 3.9.
- Worse cluster: this is the cluster of stocks for which the return of the equally-weighted portfolio is minimized ($argmin$ of equation 3.9).
- Russell 3000: portfolio holding the 3000 constituents of the index.
- Random strategy: we wished to test a portfolio with the same number of stocks as the best cluster against the best cluster. To do this we sample stocks at random from the Russell 3000 until we have the desired number of stocks in this portfolio.
- Market: this is the market performance from data of the Fama-French website [9].

4.1 DISCUSSION

We first observe that the performance of the market ($R_M - R_f$ downloaded from the Fama-French website [9] to which we've added R_f) and of the custom-built Russell 3000 (reconstituted on a monthly basis) are similar. The annualized standard deviation of the market is the lowest which can be explained by the fact that the market is built from all NYSE, AMEX, and NASDAQ firms resulting in a better

diversification than any other model. The portfolio containing the Russell 3000 still manages to have a Sharpe ratio better than that of the market and above 1.0 which is due to the larger returns (annualized of 18% vs. 14%). In concrete terms, investing $\$1000$ during 10 years in either indices will result in a final capital of $\$5160$ and $\$3696$ respectively for the Russell 3000 and the overall market.

With respect to our methodology, a 3-factor model (equation 3.4) does not seem to correctly identify outperforming/underperforming clusters of portfolios, regardless of the clustering algorithm used (Figure 6.8). Indeed, the worse cluster consistently outperforms the best cluster and the Sharpe ratios of either cluster are consistently worse than that of the market, making the latter or the Russell 3000 wiser choices of investment.

The mean-shift clustering algorithm gives undesirable results, regardless of the factor model chosen. The best cluster consistently underperforms the worse and the latter gives outstanding results relative to the market (higher Sharpe ratio). This phenomenon could be due to the fact that mean-shift clustering is the only algorithm where the number of clusters is not specified. Moreover, the nature of the metric used (nearest-neighbor graph) and the dimensionality of the problem induces a number of clusters greater than that fixed in the other algorithms ($K = 24$). Indeed in higher dimensions, data is more spread out due to the curse of dimensionality. Consequently, a smaller number of securities constitute the clusters, diversification is suboptimal, and there are more chances that a "bad apple" drives the performance of the portfolio down. Finally, one might identify a potential pairs trading strategy (long worse cluster, short best cluster) but the historical correlation of the returns between the portfolios is actually close to 0, canceling this hypothesis.

We can make more sense of our results when taking the 5-factor model originally proposed by Fama and French [1], with specified-K clustering algorithms. Indeed in each case, the best cluster outperforms the worse and consistently beats the market. Moreover, spectral clustering gives the best results with a Sharpe ratio of 1.608 and investing $\$1000$ for 10 years with this strategy would result in a final capital of $\$3'594'566$. This spectacular performance could be the result of successfully categorizing securities by their factor loadings and extracting an outperforming portfolio.

The 6- and 8-factor models seem to give diverse and inconsistent results. This can be a sign that momentum-related factors are poor cluster separators and cannot correctly differentiate good portfolios from bad. Another reason that could explain this phenomenon is that momentum factors are added to models that already count a significant number of coefficients. One method to properly assess the cumulative power of coefficients would be to conduct a bottom-up method of feature selection widely used in machine learning (e.g. forward/backward search). Furthermore, other common techniques such as data standardization could have been used since we are dealing with clustering algorithms using distance metrics but we assumed this was not needed since the regression coefficients have the same unit and that a coefficient significantly larger than any other other actually holds important information that should not be transformed.

With respect to the considered models, an exploration of different factors could be done. Sources such as *Google* trends, *Twitter* trends, or patent filing frequency for example could constitute conceivable long-short factors to take into consideration. We are however limited in terms of request rate on standard API endpoints and have therefore decided to not consider this path.

In addition, we have chosen to omit transaction costs but this is by far not a negligible factor. It is well-known that these costs can eat up a good percentage of returns of a good trading strategy, so far more than in the case of intraday trading. Figure 4.7 depicts the evolution of capital of a trading strategy using a 5-factor model with spectral clustering, and introducing transaction costs of a mere 0.02 basis points per trade. Already, investing $\$1000$ for 10 years results in a final capital of only $\$14'783$, which is more realistic than the case without costs ($\$3'594'566$). One can argue however that null or quasi-null transaction costs are becoming a reality with online brokers such as *Robinhood*

and *DEGIRO* but we have not delved into the legislation of using these platforms as a professional trader (or at least for high-frequency trading). The transaction costs are evidently high because we are dealing with intraday trading where at most 250 securities are exchanged ($3000/24$ stocks sold and bought) each day. These costs could be reduced by reducing the frequency of trading and/or taking into account a minimum allocation time per stock. Moreover on a side-note, we could further take into account capital gains and dividend taxes but this is considered outside the scope of this project and greatly depends on the place of residence of the trader.

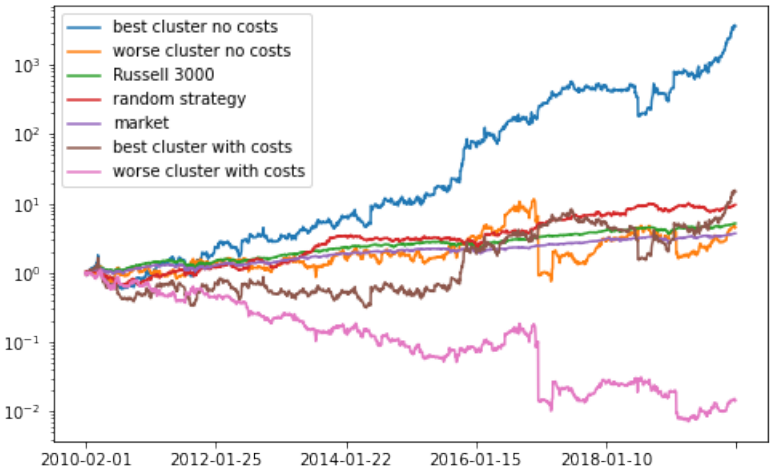

FIGURE 4.7
Evolution of capital, comparison of trading strategy with transaction costs (0.02 basis points per trade) vs. without.

We have assumed that 10 years of daily data was sufficient to assess the quality of our trading strategy. Indeed, having deployed a rolling window is a fair approach to take into account the non-stationary nature of the stock market. Nonetheless, more historical data could have been used but this would increase the computing overhead and space of data needed, which has already shown some limits[8] even with the use of multi-processing (*Dask*). Moreover, data space and computing overhead takes a heavier toll when the number of factors in a model increases (higher dimensions and more strenuous clustering). Two solutions can be proposed and used: either decrease the trading frequency (and reduce transaction costs at the same time) but this is not scalable or deploy big data tools such as Hadoop or Spark and take advantage of distributed file systems and computing power (scalable solution).

Finally, a comparison of all clusters could have been done to properly assess the quality of the methodology but for the purpose of simplicity and comparability, we have chosen to restrict ourselves to the best and worse clusters.

[8]Computations was performed on a custom-built computer using an AMD Ryzen 3600X processor.

CONCLUSION

In this project, we use popular multi-factor models as a way to explain stock returns and compute the daily regression loadings of these models using daily returns and a month-wide rolling window. These coefficients are then fed into a clustering algorithm in order to categorize these stocks and create portfolios from the clusters. This procedure is repeated on a daily basis on 10 years of data. Then, a simple trading strategy is implemented using a trading signal incorporating the equally-weighted returns of these portfolios and simulations are conducted in order to retrieve relevant annualized metrics. We finally compare trading strategies using different combinations of multi-factor models and clustering algorithms.

Our results reveal that a simple 3-factor model cannot encapsulate enough information in order for clustering algorithms to identify outperforming and underperforming portfolios. In addition, models incorporating momentum-related factors do not give consistent results, indicating that these factors are poor cluster separators or factors that worsen the proper categorization of portfolios. Consistent and suitable results are obtained by using the well-known 5-factor model of Fama and French with a clustering algorithm based on distance between points and parameterizable number of clusters. Using spectral clustering and implementing a simple trading signal resulted in a trading strategy that consistently outperformed the market by a large margin.

Finally, the results of this project only account for a small choice of tuning parameters. Indeed, different long-short factors, trading signals, width of the rolling window, number of clusters, and trading frequencies (only to name a few) are all axes of exploration that could be taken into consideration in future works.

APPENDIX

factor model	algorithm	best cluster	worse cluster	Russell 3000	random strategy	market
3	kmeans	0.160	0.271	0.18	0.141	0.141
	agglomerative_ward	0.205	0.286	0.18	0.189	0.141
	meanshift	-0.177	0.174	0.18	0.186	0.141
	spectral	-0.082	0.682	0.18	0.143	0.141
5	kmeans	0.928	0.248	0.18	0.069	0.141
	agglomerative_ward	0.932	0.297	0.18	0.092	0.141
	meanshift	-0.025	0.572	0.18	0.247	0.141
	spectral	1.284	0.165	0.18	0.256	0.141
6	kmeans	0.842	0.392	0.18	0.037	0.141
	agglomerative_ward	0.777	0.228	0.18	0.085	0.141
	meanshift	-0.208	1.126	0.18	0.219	0.141
	spectral	0.798	0.592	0.18	0.274	0.141
8	kmeans	0.620	0.087	0.18	0.102	0.141
	agglomerative_ward	0.655	0.571	0.18	0.132	0.141
	meanshift	-0.165	1.266	0.18	0.264	0.141
	spectral	0.746	-0.271	0.18	0.257	0.141

TABLE 6.2
Annualized returns for every model and clustering algorithm in percentage/100.

factor model	algorithm	best cluster	worse cluster	Russell 3000	random strategy	market
3	kmeans	0.814	0.697	0.162	0.208	0.152
	agglomerative_ward	0.809	0.745	0.162	0.212	0.152
	meanshift	1.127	1.184	0.162	0.318	0.152
	spectral	0.850	1.005	0.162	0.298	0.152
5	kmeans	0.894	0.760	0.162	0.230	0.152
	agglomerative_ward	0.891	0.791	0.162	0.229	0.152
	meanshift	1.162	1.265	0.162	0.333	0.152
	spectral	0.795	0.767	0.162	0.291	0.152
6	kmeans	0.894	0.889	0.162	0.230	0.152
	agglomerative_ward	0.887	0.754	0.162	0.250	0.152
	meanshift	1.186	1.243	0.162	0.339	0.152
	spectral	0.777	0.882	0.162	0.298	0.152
8	kmeans	0.870	1.030	0.162	0.251	0.152
	agglomerative_ward	0.867	0.977	0.162	0.262	0.152
	meanshift	1.210	1.269	0.162	0.332	0.152
	spectral	0.737	0.800	0.162	0.297	0.152

TABLE 6.3
Annualized standard deviations for every model and clustering algorithm in percentage/100.

factor model	algorithm	best cluster	worse cluster	Russell 3000	random strategy	market
3	kmeans	0.190	0.381	1.077	0.655	0.893
	agglomerative_ward	0.247	0.377	1.077	0.869	0.893
	meanshift	-0.162	0.142	1.077	0.568	0.893
	spectral	-0.102	0.673	1.077	0.463	0.893
5	kmeans	1.032	0.319	1.077	0.278	0.893
	agglomerative_ward	1.040	0.369	1.077	0.379	0.893
	meanshift	-0.026	0.448	1.077	0.728	0.893
	spectral	1.608	0.209	1.077	0.860	0.893
6	kmeans	0.935	0.435	1.077	0.140	0.893
	agglomerative_ward	0.870	0.296	1.077	0.320	0.893
	meanshift	-0.180	0.902	1.077	0.630	0.893
	spectral	1.020	0.665	1.077	0.903	0.893
8	kmeans	0.707	0.079	1.077	0.386	0.893
	agglomerative_ward	0.749	0.579	1.077	0.483	0.893
	meanshift	-0.141	0.994	1.077	0.778	0.893
	spectral	1.006	-0.346	1.077	0.850	0.893

TABLE 6.4
Annualized Sharpe ratios for every model and clustering algorithm (unitless).

factor model	algorithm	best cluster	worse cluster	Russell 3000	random strategy	market
3	kmeans	4.354	10.741	5.16	3.706	3.696
	agglomerative_ward	6.364	12.079	5.16	5.578	3.696
	meanshift	0.144	4.892	5.16	5.405	3.696
	spectral	0.428	173.019	5.16	3.775	3.696
5	kmeans	669.752	8.956	5.16	1.939	3.696
	agglomerative_ward	684.111	13.151	5.16	2.392	3.696
	meanshift	0.779	88.507	5.16	8.937	3.696
	spectral	3594.566	4.563	5.16	9.560	3.696
6	kmeans	425.600	26.471	5.16	1.438	3.696
	agglomerative_ward	298.268	7.685	5.16	2.249	3.696
	meanshift	0.099	1768.097	5.16	7.121	3.696
	spectral	335.038	100.461	5.16	11.013	3.696
8	kmeans	119.635	2.281	5.16	2.620	3.696
	agglomerative_ward	147.089	87.952	5.16	3.405	3.696
	meanshift	0.167	3326.825	5.16	10.183	3.696
	spectral	251.489	0.043	5.16	9.684	3.696

TABLE 6.5
Overall gains for every model and clustering algorithm (unitless).

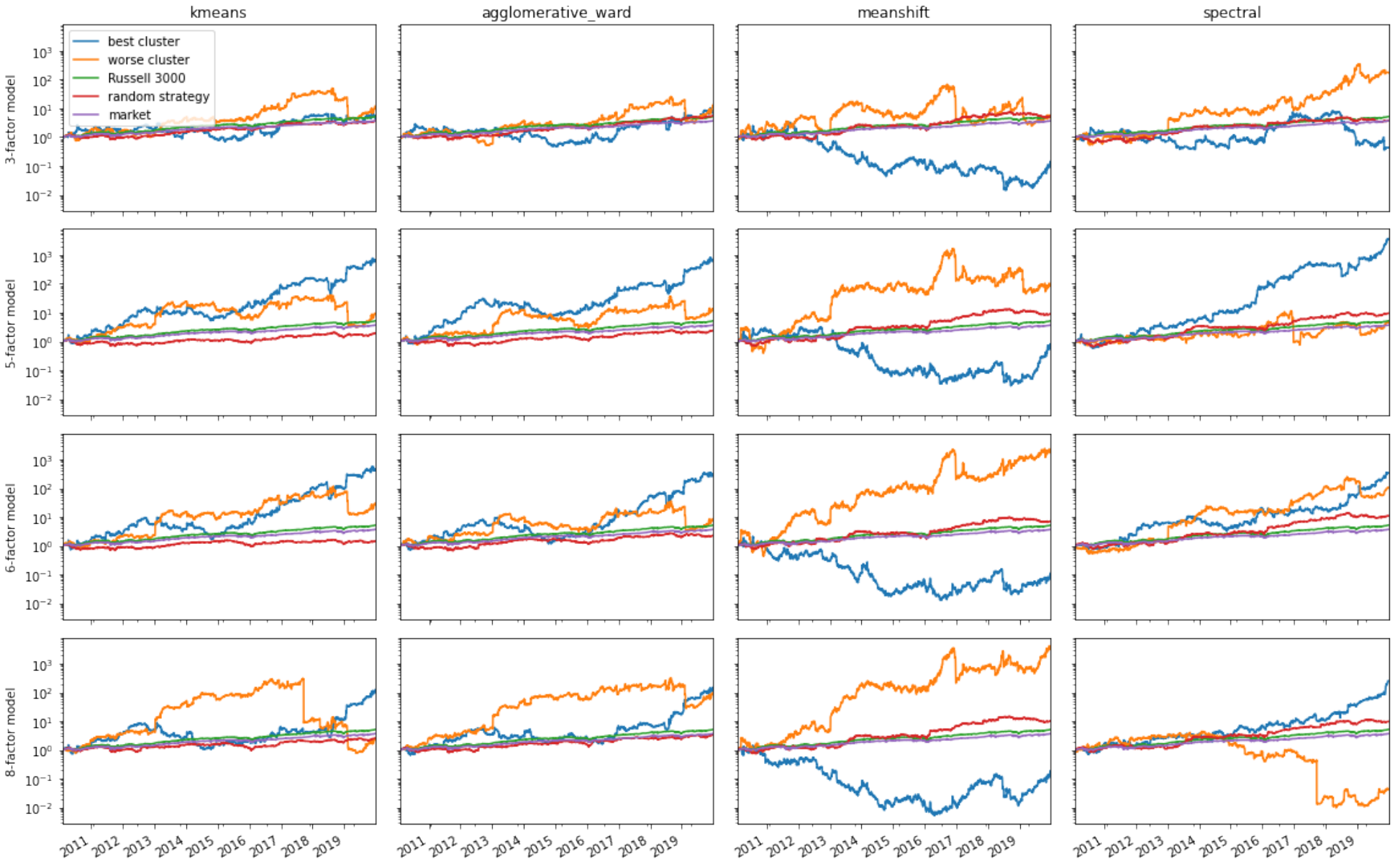

FIGURE 6.8
Evolution of capital for every model and clustering algorithm

BIBLIOGRAPHY

[1] Eugene F. Fama and Kenneth R. French. *Common risk factors in the returns on stocks and bonds*. Mar. 2002. URL: https://www.sciencedirect.com/science/article/pii/0304405X93900235.

[2] Eugene F. Fama and Kenneth R. French. *A five-factor asset pricing model*. Oct. 2014. URL: https://www.sciencedirect.com/science/article/pii/S0304405X14002323.

[3] Mark Carhart. *On Persistence in Mutual Fund Performance*. Jan. 1997. URL: https://econpapers.repec.org/article/blajfinan/v_3a52_3ay_3a1997_3ai_3a1_3ap_3a57-82.htm.

[4] Stephen A Ross. *The arbitrage theory of capital asset pricing*. July 2004. URL: https://www.sciencedirect.com/science/article/pii/0022053176900466.

[5] Pierre Collin-Dufresne. *Factor Models and the Arbitrage Pricing Theory*. Jan. 2021.

[6] Wharton Research Data Services. *Wharton Research Data Services*. URL: https://wrds-www.wharton.upenn.edu/.

[7] James Chen. *Learn About the Russell 3000 Index*. Dec. 2020. URL: https://www.investopedia.com/terms/r/russell_3000.asp.

[8] US Census Bureau. *2017 NAICS Manual*. Dec. 2018. URL: https://www.census.gov/library/publications/2017/econ/2017-naics-manual.html.

[9] *Kenneth R. French*. URL: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

[10] Dask Development Team. *Dask: Library for dynamic task scheduling*. 2016. URL: https://dask.org.

[11] Zhi Da, Qianqiu Liu and Ernst Schaumburg. 'A Closer Look at the Short-Term Return Reversal'. In: *Management Science* 60 (Mar. 2014), pp. 658–674. DOI: 10.1287/mnsc.2013.1766.

[12] *sklearn.linear_model.LinearRegression*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[13] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.