Chapter I

INTRODUCTION

Machine Learning for Trading: From Idea to Execution

The rise of ML in the investment industry: -

The investment industry has evolved dramatically over the last several decades and continues to do so amid increased competition, technological advances, and a challenging economic environment. This section reviews several key trends that have shaped the investment environment in general, and the context for algorithmic trading more specifically, and related themes that will recur throughout this project.

The trends that have propelled algorithmic trading and ML to current prominence include:

Changes in the market microstructure, such as the spread of electronic trading and the integration of markets across asset classes and geographies

The development of investment strategies framed in terms of risk-factor exposure, as opposed to asset classes

The revolutions in computing power, data-generation and management, and analytic methods

The outperformance of the pioneers in algorithmic traders relative to human, discretionary investors

In addition, the financial crises of 2001 and 2008 have affected how investors approach diversification and risk management and have given rise to low-cost passive investment vehicles in the form of exchange-traded funds (ETFs). Amid low yield and low volatility after the 2008 crisis, cost-conscious investors shifted $2 trillion from actively-managed mutual funds into passively managed ETFs. Competitive pressure is also reflected in lower hedge fund fees that dropped from the traditional 2% annual management fee and 20% take of profits to an average of 1.48% and 17.4%, respectively, in 2017.

From electronic to high-frequency trading: -

Electronic trading has advanced dramatically in terms of capabilities, volume, coverage of asset classes, and geographies since networks started routing prices to computer terminals in the 1960s.

- High Frequency Trading: Overview of Recent Developments, Congressional Research Service, 2016

Factor investing and smart beta funds: -

The return provided by an asset is a function of the uncertainty or risk associated with the financial investment. An equity investment implies, for example, assuming a company's business risk, and a bond investment implies assuming default risk.

To the extent that specific risk characteristics predict returns, identifying and forecasting the behaviour of these risk factors becomes a primary focus when designing an investment strategy. It yields valuable trading signals and is the key to superior active-management results. The industry's understanding of risk factors has evolved very substantially over time and has impacted how ML is used for algorithmic trading.

The factors that explained returns above and beyond the CAPM were incorporated into investment styles that tilt portfolios in favour of one or more factors, and assets began to migrate into factor-based portfolios. The 2008 financial crisis underlined how asset-class labels could be highly misleading and create a false sense of diversification when investors do not look at the underlying factor risks, as asset classes came crashing down together.

Over the past several decades, quantitative factor investing has evolved from a simple approach based on two or three styles to multifactor smart or exotic beta products. Smart beta funds have crossed $1 trillion AUM in 2017, testifying to the popularity of the hybrid investment strategy that combines active and passive management. Smart beta funds take a passive strategy but modify it according to one or more factors, such as cheaper stocks or screening them according to dividend pay-outs, to generate better returns. This growth has coincided with increasing criticism of the high fees charged by traditional active managers as well as heightened scrutiny of their performance.

The ongoing discovery and successful forecasting of risk factors that, either individually or in combination with other risk factors, significantly impact future asset returns across asset classes is a key driver of the surge in ML in the investment industry and will be a key theme throughout this book.

Algorithmic pioneers outperform humans at scale: -

The track record and growth of Assets Under Management (AUM) of firms that spearheaded algorithmic trading has played a key role in generating investor interest and subsequent industry efforts to replicate their success.

Systematic strategies that mostly or exclusively rely on algorithmic decision-making were most famously introduced by mathematician James Simons who founded Renaissance Technologies in 1982 and built it into the premier quant firm. Its secretive Medallion Fund, which is closed to outsiders, has earned an estimated annualized return of 35% since 1982.

DE Shaw, Citadel, and Two Sigma, three of the most prominent quantitative hedge funds that use systematic strategies based on algorithms, rose to the all-time top-20 performers for the first time in 2017 in terms of total dollars earned for investors, after fees, and since inception.

ML driven funds attract $1 trillion AUM

Morgan Stanley estimated in 2017 that algorithmic strategies have grown at 15% per year over the past six years and control about $1.5 trillion between hedge funds, mutual funds, and smart beta ETFs. Other reports suggest the quantitative hedge fund industry was about to exceed $1 trillion AUM, nearly doubling its size since 2010 amid outflows from traditional hedge funds. In contrast, total hedge fund industry capital hit $3.21 trillion according to the latest global Hedge Fund Research report.

The emergence of quantamental funds: -

Two distinct approaches have evolved in active investment management: systematic (or quant) and discretionary investing. Systematic approaches rely on algorithms for a repeatable and data-driven approach to identify investment opportunities across many securities; in contrast, a discretionary approach involves an in-depth analysis of a smaller number of securities. These two approaches are becoming more similar as fundamental managers take more data-science-driven approaches.

Even fundamental traders now arm themselves with quantitative techniques, accounting for $55 billion of systematic assets, according to Barclays. Agnostic to specific companies, quantitative funds trade patterns and dynamics across a wide swath of securities. Quants now account for about 17% of total hedge fund assets, data compiled by Barclays shows.

<u>**ML and alternative data: -**</u>

Hedge funds have long looked for alpha through informational advantage and the ability to uncover new uncorrelated signals. Historically, this included things such as proprietary surveys of shoppers, or voters ahead of elections or referendums. Occasionally, the use of company insiders, doctors, and expert networks to expand knowledge of industry trends or companies crosses legal lines: a series of prosecutions of traders, portfolio managers, and analysts for using insider information after 2010 has shaken the industry.

In contrast, the informational advantage from exploiting conventional and alternative data sources using ML is not related to expert and industry networks or access to corporate management, but rather the ability to collect large quantities of data and analyse them in real-time.

Three trends have revolutionized the use of data in algorithmic trading strategies and may further shift the investment industry from discretionary to quantitative styles:

1. The exponential increase in the amount of digital data.
2. The increase in computing power and data storage capacity at lower cost.
3. The advances in ML methods for analysing complex datasets.

Can We Predict the Financial Markets Based on Google's Search Queries?, Perlin, et al, 2016, Journal of Forecasting

(https://onlinelibrary.wiley.com/doi/abs/10.1002/for.2446)

Design and execution of a trading strategy: -

ML can add value at multiple steps in the lifecycle of a trading strategy, and relies on key infrastructure and data resources. Hence, this book aims to addresses how ML techniques fit into the broader process of designing, executing, and evaluating strategies.

An algorithmic trading strategy is driven by a combination of alpha factors that transform one or several data sources into signals that in turn predict future asset returns and trigger buy or sell orders.

Sourcing and managing data: -

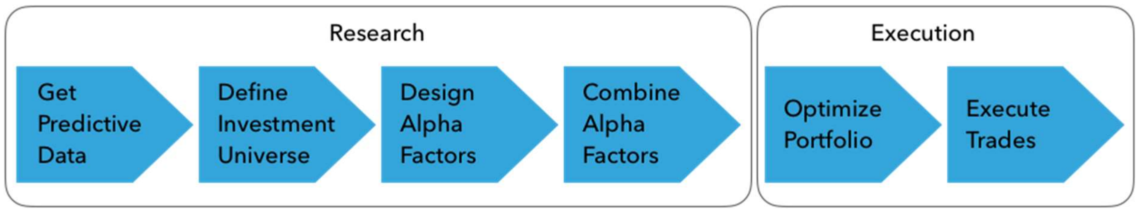
The dramatic evolution of data in terms of volume, variety, and velocity is both a necessary condition for and driving force of the application of ML to algorithmic trading. The proliferating supply of data requires active management to uncover potential value, including the following steps:

1. Identify and evaluate market, fundamental, and alternative data sources containing alpha signals that do not decay too quickly.
2. Deploy or access cloud-based scalable data infrastructure and analytical tools like Hadoop or Spark Sourcing to facilitate fast, flexible data access.

Carefully manage and curate data to avoid look-ahead bias by adjusting it to the desired frequency on a point-in-time (PIT) basis. This means that data may only reflect information available and know at the given time. ML algorithms trained on distorted historical data will almost certainly fail during live trading.

Alpha factor research and evaluation: -

Alpha factors are designed to extract signals from data to predict asset returns for a given investment universe over the trading horizon. A factor takes on a single value for each asset when evaluated, but may combine one or several input variables. The process involves the steps outlined in the following figure:



The Research phase of the trading strategy workflow includes the design, evaluation, and combination of alpha factors. ML plays a large role in this process because the complexity of factors has increased as investors react to both the signal decay of simpler factors and the much richer data available today.

Portfolio optimization and risk management: -

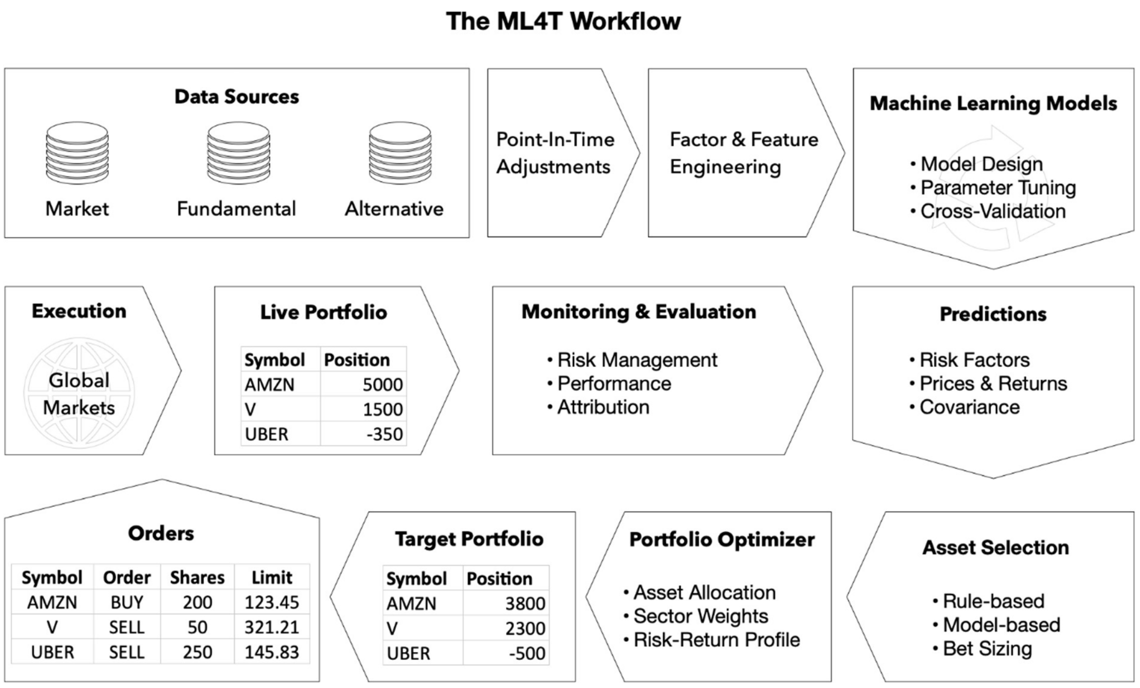
Alpha factors emit entry and exit signals that lead to buy or sell orders, and order execution results in portfolio holdings. The risk profiles of individual positions interact to create a specific portfolio risk profile. Portfolio management involves the optimization of position weights to achieve the desired portfolio risk and return a profile that aligns with the overall investment objectives. This process is highly dynamic to incorporate continuously-evolving market data.

Strategy Backtesting: -

The incorporation of an investment idea into an algorithmic strategy requires extensive testing with a scientific approach that attempts to reject the idea based on its performance in alternative out-of-sample market scenarios. Testing may involve simulated data to capture scenarios deemed possible but not reflected in historic data.

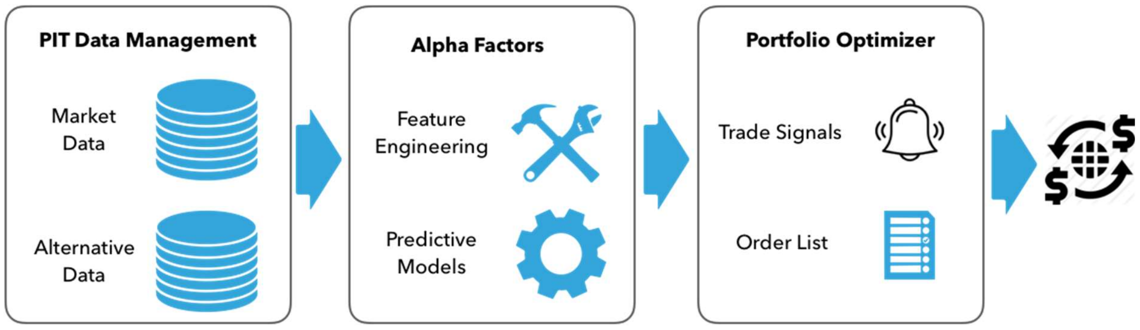
Chapter II

LITERATURE REVIEW



Proposed Solution: -

For generating better returns multiple alpha factors are combined together to generate set of portfolio weights we achieve this by using non overlapping voters classifier with random forest as the base estimator then after training the model on previous 3 years of data we generated alpha factors next year's data and then we used statistical risk model to optimize the entire portfolio's weights.



Chapter III

METHODOLGY

Market & Fundamental Data

How to work with Market Data: -

Market data results from the placement and processing of buy and sell orders in the course of the trading of financial instruments on the many marketplaces. The data reflects the institutional environment of trading venues, including the rules and regulations that govern orders, trade execution, and price formation.

Algorithmic traders use ML algorithms to analyse the flow of buy and sell orders and the resulting volume and price statistics to extract trade signals or features that capture insights into, for example, demand-supply dynamics or the behaviour of certain market participants.

Market microstructure: -

Market microstructure is the branch of financial economics that investigates the trading process and the organization of related markets. The following references provide insights into institutional details that can be quite complex and diverse across asset classes and their derivatives, trading venues, and geographies, as well as data about the trading activities on various exchanges around the world.

Access to Market Data: -

There are several options to access market data via API using Python. In this chapter, we first present a few sources built into the pandas library. Then we briefly introduce the trading platform Quantopian, the data provider Quandl (acquired by NASDAQ in 12/2018) and the backtesting library zipline that we will use later in the project, and list several additional options to access various types of market data. The data for this project is downloaded from Yahoo Finance using yfinance library for Python, there is a notebook associated with the downloading data for the project named “Alpha Research Dataset Preparation” in the project directory.

Efficient data storage with pandas: -

We'll be using many different data sets in this book, and it's worth comparing the main formats for efficiency and performance. In particular, we compare the following:

- CSV: Comma-separated, standard flat text file format.
- HDF5: Hierarchical data format, developed initially at the National Center for Supercomputing, is a fast and scalable storage format for numerical data, available in pandas using the PyTables library.
- Parquet: A binary, columnar storage format, part of the Apache Hadoop ecosystem, that provides efficient data compression and encoding and has been developed by Cloudera and Twitter. It is available for pandas through the pyarrow library, led by Wes McKinney, the original author of pandas.

Alternative Data for Trading

This chapter explains how individuals, business processes, and sensors produce alternative data. It also provides a framework to navigate and evaluate the proliferating supply of alternative data for investment purposes.

It demonstrates the workflow, from acquisition to pre-processing and storage using Python for data obtained through web scraping to set the stage for the application of ML. It concludes by providing examples of sources, providers, and applications.

More specifically, this chapter covers:

1. How the alternative data revolution has unleashed new sources of information.
2. How individuals, business processes, and sensors generate alternative data.
3. How to evaluate the proliferating supply of alternative data used for algorithmic trading.
4. How to work with alternative data in Python, such as by scraping the internet.
5. Important categories and providers of alternative data.

The Alternative Data Revolution: -

For algorithmic trading, new data sources offer an informational advantage if they provide access to information unavailable from traditional sources, or provide access sooner. Following global trends, the investment industry is rapidly expanding beyond market and fundamental data to alternative sources to reap alpha through an informational edge. Annual spending on data, technological capabilities, and related talent are expected to increase from the current $3 billion by 12.8% annually through 2020.

Today, investors can access macro or company-specific data in real-time that historically has been available only at a much lower frequency. Use cases for new data sources include the following:

1. Online price data on a representative set of goods and services can be used to measure inflation.

2. The number of store visits or purchases permits real-time estimates of company or industry-specific sales or economic activity.
3. Satellite images can reveal agricultural yields, or activity at mines or on oil rigs before this information is available elsewhere.

Useful references include: -

1. [The Digital Universe in 2020](https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf)
2. [Big data: The next frontier for innovation, competition, and productivity](https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation), McKinsey 2011
3. [McKinsey on Artificial Intelligence](https://www.mckinsey.com/featured-insights/artificial-intelligence)

Evaluating Alternative Datasets: -

The ultimate objective of alternative data is to provide an informational advantage in the competitive search for trading signals that produce alpha, namely positive, uncorrelated investment returns. In practice, the signals extracted from alternative datasets can be used on a standalone basis or combined with other signals as part of a quantitative strategy.

Key reference:

- [Big Data and AI Strategies](http://valuesimplex.com/articles/JPM.pdf), Kolanovic, M. and Krishnamachari, R., JP Morgan, May 2017

The Market for Alternative Data: -

The investment industry is going to spend an estimated $2bn-3bn on data services in 2018, and this number is expected to grow at double digits per year in line with other industries. This expenditure includes the acquisition of alternative data, investments in related technology, and the hiring of qualified talent.

Working with Alternative Data: -

This section illustrates the acquisition of alternative data using web scraping, targeting first OpenTable restaurant data, and then move to earnings call transcripts hosted by Seeking Alpha.

1. [Quantifying Trading Behaviour in Financial Markets Using Google Trends](https://www.nature.com/articles/srep01684), Preis, Moat and Stanley, Nature, 2013
2. [Quantifying StockTwits semantic terms' trading behaviour in financial markets: An effective application of decision tree algorithms](https://www.sciencedirect.com/science/article/pii/S0957417415005473), Al Nasseri et al, Expert Systems with Applications, 2015

Financial Feature Engineering: How to research Alpha Factors

Algorithmic trading strategies are driven by signals that indicate when to buy or sell assets to generate superior returns relative to a benchmark such as an index. The portion of an asset's return that is not explained by exposure to this benchmark is called alpha, and hence the signals that aim to produce such uncorrelated returns are also called alpha factors.

If you are already familiar with ML, you may know that feature engineering is a key ingredient for successful predictions. This is no different in trading. Investment, however, is particularly rich in decades of research into how markets work and which features may work better than others to explain or predict price movements as a result. This chapter provides an overview as a starting point for your own search for alpha factors.

This project also presents key tools that facilitate the computing and testing alpha factors. We will highlight how the NumPy, pandas and TA-Lib libraries facilitate the manipulation of data and present popular smoothing techniques like the wavelets and the Kalman filter that help reduce noise in data.

We also preview how you can use the trading simulator Zipline to evaluate the predictive performance of (traditional) alpha factors. We discuss key alpha factor metrics like the information coefficient and factor turnover. An in-depth introduction to backtesting trading strategies that use machine learning, which covers the ML4T workflow that we will use throughout the book to evaluate trading strategies.

In particular, this project will address the following topics:

1. Which categories of factors exist, why they work, and how to measure them.
2. Creating e alpha factors using NumPy, and pandas.
3. How to denoise data using wavelets and the Kalman filter.
4. Using e Zipline offline and on Quantopian to test individual and multiple alpha factors.
5. How to use Alphalens to evaluate predictive performance and turnover using, among other metrics, the information coefficient (IC).

On the shoulders of giants: meet the factor establishment: -

In an idealized world, categories of risk factors should be independent of each other (orthogonal), yield positive risk premia, and form a complete set that spans all dimensions of risk and explains the systematic risks for assets in a given class. In practice, these requirements will hold only approximately.

1. Dissecting Anomalies by Eugene Fama and Ken French (2008)
2. Explaining Stock Returns: A Literature Review by James L. Davis (2001)
3. Market Efficiency, Long-Term Returns, and Behavioral Finance by Eugene Fama (1997)
4. The Efficient Market Hypothesis and It's Critics by Burton Malkiel (2003)
5. The New Palgrave Dictionary of Economics (2008) by Steven Durlauf and Lawrence Blume, 2nd ed.
6. Anomalies and Market Efficiency by G. William Schwert25 (Ch. 15 in Handbook of the- "Economics of Finance", by Constantinides, Harris, and Stulz, 2003)
7. Investor Psychology and Asset Pricing, by David Hirshleifer (2001)

Alpha Factors in practice: from data to signals: -

Alpha factors are transformations of market, fundamental, and alternative data that contain predictive signals. They are designed to capture risks that drive asset returns. One set of factors describes fundamental, economy-wide variables such as growth, inflation, volatility, productivity, and demographic risk. Another set consists of tradeable investment styles such as the market portfolio, value-growth investing, and momentum investing.

There are also factors that explain price movements based on the economics or institutional setting of financial markets, or investor behaviour, including known biases of this behaviour. The economic theory behind factors can be rational, where the factors have high returns over the long run to compensate for their low returns during bad times, or behavioural, where factor risk premiums result from the possibly biased, or not entirely rational behaviour of agents that is not arbitraged away.

Based on a conceptual understanding of key factor categories, their rationale and popular metrics, a key task is to identify new factors that may better capture the risks embodied by the return drivers laid out previously, or to find new ones.

Seeking signals – how to use zipline: -

Historically, alpha factors used a single input and simple heuristics, thresholds or quantile cut-offs to identify buy or sell signals. ML has proven quite effective in extracting signals from a more diverse and much larger set of input data, including other alpha factors based on the analysis of historical patterns. As a result, algorithmic trading strategies today leverage a large number of alpha signals, many of which may be weak individually but can yield reliable predictions when combined with other model-driven or traditional factors by an ML algorithm.

The open source zipline library is an event-driven backtesting system maintained and used in production by the crowd-sourced quantitative investment fund Quantopian to facilitate algorithm-development and live-trading.

It automates the algorithm's reaction to trade events and provides it with current and historical point-in-time data that avoids look-ahead bias.

The architecture – event-driven trading simulation: -

1. A zipline algorithm will run for a specified period after an initial setup and executes its trading logic when specific events occur.
2. These events are driven by the trading frequency and can also be scheduled by the algorithm, and result in zipline calling certain methods.
3. The algorithm maintains state through a context dictionary and receives actionable information through a data variable containing point-in-time (PIT) current and historical data.
4. The algorithm returns a DataFrame containing portfolio performance metrics if there were any trades, as well as user-defined metrics that can be used to record, for example, the factor values.

The Pipeline API facilitates the definition and computation of alpha factors for a cross-section of securities from historical data. A pipeline defines computations that produce columns in a table with PIT values for a set of securities. It needs to be registered with the initialize() method and can then be executed on an automatic or custom schedule. The library provides numerous built-in computations such as moving averages or Bollinger Bands that can be used to quickly compute standard factors but also allows for the creation of custom factors as we will illustrate next.

Most importantly, the Pipeline API renders alpha factor research modular because it separates the alpha factor computation from the remainder of the algorithm, including the placement and execution of trade orders and the bookkeeping of portfolio holdings, values, and so on.

Separating signal and noise – how to use alphalens: -

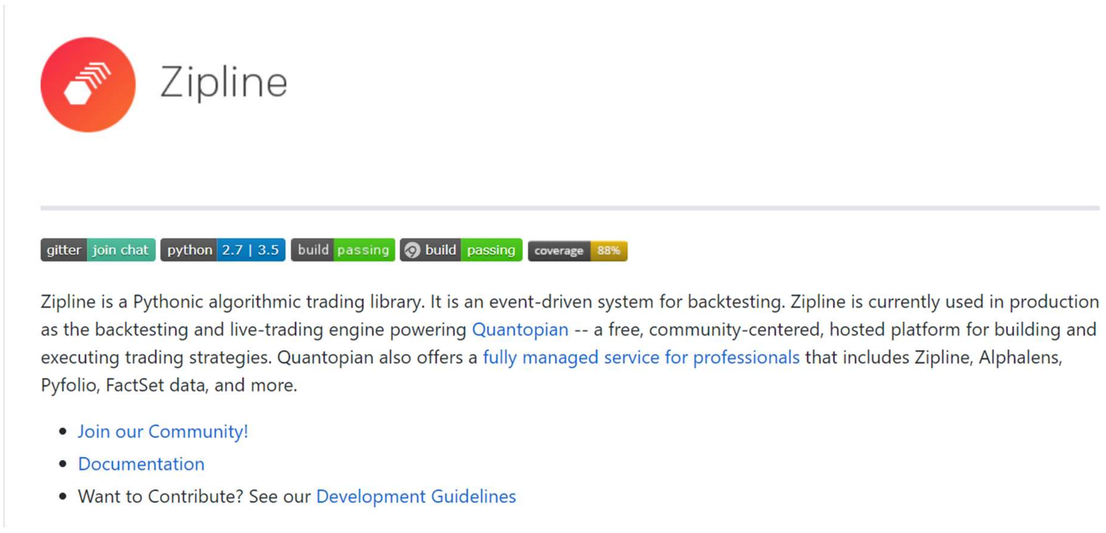
Quantopian has open sourced the Python library alphalens for the performance analysis of predictive stock factors that integrates well with the backtesting library zipline and the portfolio performance and risk analysis library pyfolio that we will explore in the next chapter.

alphalens facilitates the analysis of the predictive power of alpha factors concerning the:

1. Correlation of the signals with subsequent returns.
2. Profitability of an equal or factor-weighted portfolio based on a (subset of) the signals.
3. Turnover of factors to indicate the potential trading costs.
4. Factor-performance during specific events.
5. Breakdowns of the preceding by sector.

The analysis can be conducted using tearsheets or individual computations and plots.

Zipline: -



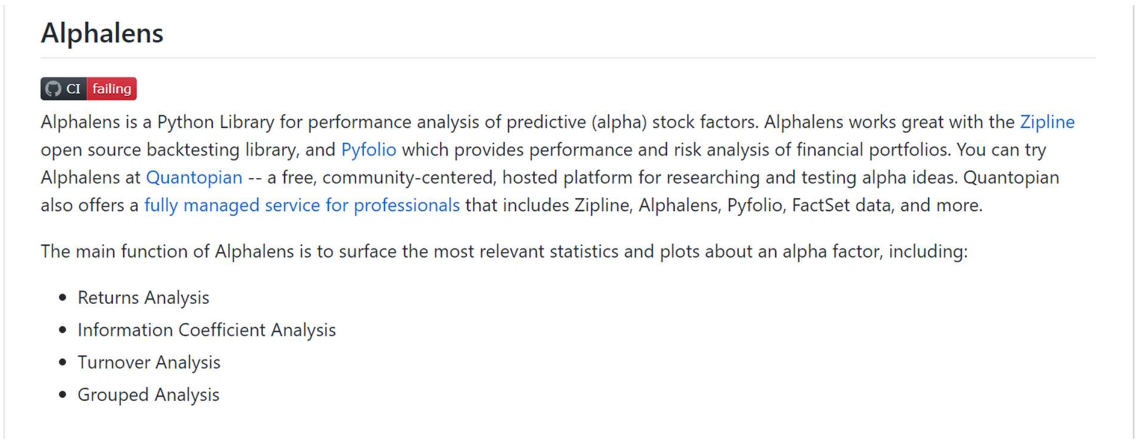
Zipline

gitter join chat | python 2.7 | 3.5 | build passing | build passing | coverage 88%

Zipline is a Pythonic algorithmic trading library. It is an event-driven system for backtesting. Zipline is currently used in production as the backtesting and live-trading engine powering Quantopian -- a free, community-centered, hosted platform for building and executing trading strategies. Quantopian also offers a fully managed service for professionals that includes Zipline, Alphalens, Pyfolio, FactSet data, and more.

- Join our Community!
- Documentation
- Want to Contribute? See our Development Guidelines

Alphalens: -



Alphalens

CI failing

Alphalens is a Python Library for performance analysis of predictive (alpha) stock factors. Alphalens works great with the Zipline open source backtesting library, and Pyfolio which provides performance and risk analysis of financial portfolios. You can try Alphalens at Quantopian -- a free, community-centered, hosted platform for researching and testing alpha ideas. Quantopian also offers a fully managed service for professionals that includes Zipline, Alphalens, Pyfolio, FactSet data, and more.

The main function of Alphalens is to surface the most relevant statistics and plots about an alpha factor, including:

- Returns Analysis
- Information Coefficient Analysis
- Turnover Analysis
- Grouped Analysis

Portfolio Optimization and Performance Evaluation

Alpha factors generate signals that an algorithmic strategy translates into trades, which, in turn, produce long and short positions. The returns and risk of the resulting portfolio determine the success of the strategy.

To test a strategy prior to implementation under market conditions, we need to simulate the trades that the algorithm would make and verify their performance. Strategy evaluation includes backtesting against historical data to optimize the strategy's parameters and forward-testing to validate the in-sample performance against new, out-of-sample data. The goal is to avoid false discoveries from tailoring a strategy to specific past circumstances.

In a portfolio context, positive asset returns can offset negative price movements. Positive price changes for one asset are more likely to offset losses on another the lower the correlation between the two positions. Based on how portfolio risk depends on the positions' covariance, Harry Markowitz developed the theory behind modern portfolio management based on diversification in 1952. The result is mean-variance optimization that selects weights for a given set of assets to minimize risk, measured as the standard deviation of returns for a given expected return.

The capital asset pricing model (CAPM) introduces a risk premium, measured as the expected return in excess of a risk-free investment, as an equilibrium reward for holding an asset. This reward compensates for the exposure to a single risk factor—the market—that is systematic as opposed to idiosyncratic to the asset and thus cannot be diversified away.

Risk management has evolved to become more sophisticated as additional risk factors and more granular choices for exposure have emerged. The Kelly criterion is a popular approach to dynamic portfolio optimization, which is the choice of a sequence of positions over time; it has been famously adapted from its original application in gambling to the stock market by Edward Thorp in 1968.

As a result, there are several approaches to optimize portfolios that include the application of machine learning (ML) to learn hierarchical relationships among assets and treat their holdings as complements or substitutes with respect to the portfolio risk profile.

In this project, we have covered the following topics:

1. How to measure portfolio risk and return.
2. Managing portfolio weights using mean-variance optimization and alternatives.
3. Using machine learning to optimize asset allocation in a portfolio context.
4. Simulating trades and create a portfolio based on alpha factors using Zipline.

How to build and test a portfolio with zipline: -

The open source zipline library is an event-driven backtesting system maintained and used in production by the crowd-sourced quantitative investment fund Quantopian to facilitate algorithm-development and live-trading. It automates the algorithm's reaction to trade events and provides it with current and historical point-in-time data that avoids look-ahead bias.

We used zipline to simulate the computation of alpha factors from trailing cross-sectional market, fundamental, and alternative data. Now we will exploit the alpha factors to derive and act on buy and sell signals.

We will postpone optimizing the portfolio weights until later in this chapter, and for now, just assign positions of equal value to each holding.

How to measure performance: -

ML is about optimizing objective functions. In algorithmic trading, the objectives are the return and the risk of the overall investment portfolio, typically relative to a benchmark (which may be cash, the risk-free interest rate, or an asset price index like the Nifty and Sensex).

The Sharpe Ratio: -

The ex-ante Sharpe Ratio (SR) compares the portfolio's expected excess portfolio to the volatility of this excess return, measured by its standard deviation. It measures the compensation as the average excess return per unit of risk taken. It can be estimated from data.

Financial returns often violate the iid assumptions. Andrew Lo has derived the necessary adjustments to the distribution and the time aggregation for returns that are stationary but autocorrelated. This is important because the time-series properties of investment strategies (for example, mean reversion, momentum, and other forms of serial correlation) can have a non-trivial impact on the SR estimator itself, especially when annualizing the SR from higher-frequency data.

- The Statistics of Sharpe Ratios, Andrew Lo, Financial Analysts Journal, 2002

The Fundamental Law of Active Management: -

A high Information Ratio (IR) implies attractive out-performance relative to the additional risk taken. The Fundamental Law of Active Management breaks the IR down into the information coefficient (IC) as a measure of forecasting skill, and the ability to apply this skill through independent bets. It summarizes the importance to play both often (high breadth) and to play well (high IC).

The IC measures the correlation between an alpha factor and the forward returns resulting from its signals and captures the accuracy of a manager's forecasting skills. The breadth of the strategy is measured by the independent number of bets an investor makes in a given time period, and the product of both values is proportional to the IR, also known as appraisal risk (Treynor and Black).

The fundamental law is important because it highlights the key drivers of outperformance: both accurate predictions and the ability to make independent forecasts and act on these forecasts matter. In practice, estimating the breadth of a strategy is difficult given the cross-sectional and time-series correlation among forecasts.

1. Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk by Richard Grinold and Ronald Kahn, 1999

2. How to Use Security Analysis to Improve Portfolio Selection, Jack L Treynor and Fischer Black, Journal of Business, 1973
3. Portfolio Constraints and the Fundamental Law of Active Management, Clarke et al 2002

How to Manage Portfolio Risk & Return: -

- Portfolio Selection, Harry Markowitz, The Journal of Finance, 1952
- The Capital Asset Pricing Model: Theory and Evidence, Eugene F. Fama and Kenneth R. French, Journal of Economic Perspectives, 2004

Mean-variance optimization: -

MPT solves for the optimal portfolio weights to minimize volatility for a given expected return, or maximize returns for a given level of volatility. The key requisite input is expected asset returns, standard deviations, and the covariance matrix.

Formulating Portfolio Optimization Problems: -

So far, we've discussed one way to formulate a portfolio optimization problem. We learned to set the portfolio variance as the objective function, while imposing the constraint that the portfolio weights should sum to 1. However, in practice you may *frame* the problem a little differently. Let's talk about some of the different ways to *set up* a portfolio optimization problem.

Common Constraints: -

There are several common constraints that show up in these problems. Earlier, we were allowing our portfolio weights to be negative or positive, as long as they summed to 1. If a weight turned out to be negative, we would consider the absolute value of that number to be the size of the *short* position to take on that asset. If your strategy does not allow you to take short positions, your portfolio weights will all need to be positive numbers. In order to enforce this in the optimization problem, you would add the constraint that every x_i in the **x** vector is *positive*.

No Short Selling: -

$$0 \leq x_i \leq 1, \qquad i = 1, 2, \ldots, n$$

You may choose to impose constraints that would limit your portfolio allocations in individual sectors, such as technology or energy. You could do this by limiting the sum of weights for assets in each sector.

Sector Limits: -

$$x_{\text{biotech1}} + x_{\text{biotech2}} + x_{\text{biotech3}} \leq M, \qquad M = \text{percent of portfolio to invest in biotech companies}$$

If your optimization objective seeks to minimize portfolio variance, you might also incorporate into the problem a goal for the total portfolio return. You can do this by adding a constraint on the portfolio return.

Constraint on Portfolio Return: -

$$\mathbf{x}^{\mathrm{T}}\mu \geq r_{\text{min}}, \qquad r_{\text{min}} = \text{minimum acceptable portfolio return}$$

Maximizing Portfolio Return: -

We can also flip the problem around by maximizing returns instead of minimizing variance. Instead of minimizing variance, it often makes sense to impose a constraint on the variance in order to manage risk. Then you could maximize mean returns, which is equivalent to minimizing the negative mean returns. This makes sense when your employer has told you, “I want the best return possible, but you must limit your losses to p*p* percent!”

objective: $\text{minimize}: \; -\mathbf{x}^{\mathrm{T}}\mu$

constraint: $\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x} \leq p, \quad p = \text{maximum permissible portfolio variance}$

Maximizing Portfolio Return and Minimizing Portfolio Variance: -

Indeed, you could also create an objective function that both maximizes returns and minimizes variance, and controls the trade-off between the two goals with a parameter, b*b*. In this case, you have two terms in your objective function, one representing the portfolio mean, and one representing the portfolio variance, and the variance term is multiplied by b*b*.

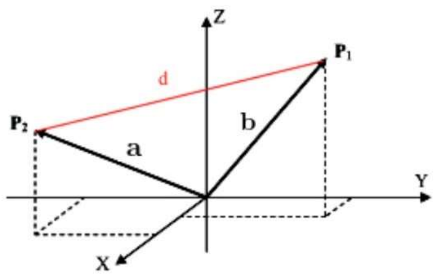
How does one determine the parameter b*b*? Well, it's very dependent on the individual and the situation, and depends on the level of risk aversion appropriate. It basically represents how much percent return you are willing to give up for each unit of variance you take on.

objective: $$\text{minimize :} \quad -\mathbf{x}^{\mathrm{T}}\mu + b\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}, \quad b = \text{tradeoff parameter}$$

A Math Note: the L2-Norm: -

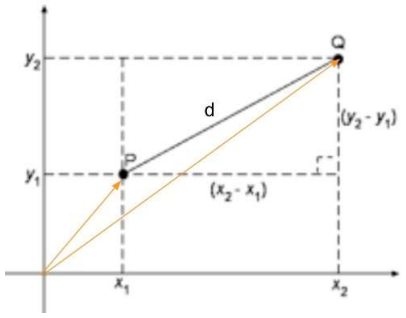
There's another way to formulate an optimization objective that relies on a new piece of notation, so I'll just take a moment to explain that now. Say we just want to minimize the difference between two quantities. Then we need a measure of the difference, but generalized into many dimensions. For portfolio optimization problems, each dimension is an asset in the portfolio. When we want to measure the distance between two vectors, we use something called the Euclidean norm or L2-norm. This is just the square root of the squared differences of each of the vectors' components. We write it with double bars and a 2 subscript.

$$d = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2} = \|\mathbf{a} - \mathbf{b}\|_2$$



Note that this reduces to the familiar Pythagorean theorem in 2 dimensions.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



Minimizing Distance to a Set of Target Weights: -

Back to portfolio optimization! One way to formulate an optimization problem is to use the L2 norm and minimize the difference between your vector of portfolio weights and a set of predefined target portfolio weights $\mathbf{x}^*$. The goal would be to get the weights as close as possible to the set of target weights while respecting a set of constraints. As an example, these target weights might be values thought to be proportional to future returns for each asset, in other words, an alpha vector.

objective: $\quad \text{minimize} : \quad \|\mathbf{x} - \mathbf{x}^*\|_2 , \quad \mathbf{x}^* = \text{a set of target portfolio weights}$

Tracking an Index: -

What if you want to minimize portfolio variance, but have the portfolio track an index at the same time? In this case, you would want terms in your objective function representing both portfolio variance and the relationship between your portfolio weights and the index weights, **q**. There are a few ways to set this up, but one intuitive way is to simply minimize the difference between your portfolio weights and the weights on the assets in the index, and minimize portfolio variance at the same time. The trade-off between these goals would be determined by a parameter, λ.

objective:

$$\text{minimize}: \quad \mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x} + \lambda \left\| \mathbf{x} - \mathbf{q} \right\|_2, \quad \mathbf{q} = \text{a set of index weights}, \quad \lambda = \text{a tradeoff parameter}$$

The Machine Learning Workflow

Developing an ML solution requires a systematic approach to maximize the chances of success while proceeding efficiently. It is also important to make the process transparent and replicable to facilitate collaboration, maintenance, and subsequent refinements.

The process is iterative throughout, and the effort at different stages will vary according to the project. Nonetheless, this process should generally include the following steps:

1. Frame the problem, identify a target metric, and define success.
2. Source, clean, and validate the data.
3. Understand your data and generate informative features.
4. Pick one or more machine learning algorithms suitable for your data.
5. Train, test, and tune your models.
6. Use your model to solve the original problem.

Frame the problem: goals & metrics: -

The starting point for any machine learning exercise is the ultimate use case it aims to address. Sometimes, this goal will be statistical inference in order to identify an association between variables or even a causal relationship. Most frequently, however, the goal will be the direct prediction of an outcome to yield a trading signal.

How to explore, extract and engineer features: -

Understanding the distribution of individual variables and the relationships among outcomes and features is the basis for picking a suitable algorithm. This typically starts with visualizations such as scatter plots, as illustrated in the companion notebook (and shown in the following image), but also includes numerical evaluations ranging from linear metrics, such as the correlation, to nonlinear statistics, such as the Spearman rank correlation coefficient that we encountered when we introduced the information coefficient. It also includes information-theoretic measures, such as mutual information.

<u>Design and tune the model: -</u>

The ML process includes steps to diagnose and manage model complexity based on estimates of the model's generalization error. An unbiased estimate requires a statistically sound and efficient procedure, as well as error metrics that align with the output variable type, which also determines whether we are dealing with a regression, classification, or ranking problem.

<u>Bias-Variance Trade-Off: -</u>

The errors that an ML model makes when predicting outcomes for new input data can be broken down into reducible and irreducible parts. The irreducible part is due to random variation (noise) in the data that is not measured, such as relevant but missing variables or natural variation.

Long-Short Signals with Decision Trees & Random Forests

In this project, we will learn how to use two new classes of machine learning models for trading: decision trees and random forests. We will see how decision trees learn rules from data that encode non-linear relationships between the input and the output variables. We will illustrate how to train a decision tree and use it for prediction for regression and classification problems such as asset returns and price moves. We will also visualize and interpret the rules learned by the model, and tune the model's hyperparameters to optimize the bias-variance trade-off and prevent overfitting.

Decision trees are not only important standalone models but are also frequently used as components in other models. In the second part of this chapter, we will introduce ensemble models that combine multiple individual models to produce a single aggregate prediction with lower prediction-error variance.

We will illustrate bootstrap aggregation, often called bagging, as one of several methods to randomize the construction of individual models and reduce the correlation of the prediction errors made by an ensemble's components. We will illustrate how bagging effectively reduces the variance, and learn how to configure, train, and tune random forests. We will see how random forests as an ensemble of a large number of decision trees, can dramatically reduce prediction errors, at the expense of some loss in interpretation.

Then we will proceed and build a long-short trading strategy that uses a Random Forest ensemble to generate profitable signals for Indian equities over the last 5 years. We will source and prepare the stock price data, tune the hyperparameters of a Random Forest model, and backtest trading rules based on the models' signals. The resulting long-short strategy uses machine learning rather than the cointegration relationship to identify and trade baskets of securities whose prices will likely move in opposite directions over a given investment horizon.

In short:

1. Use decision trees for regression and classification.
2. Gain insights from decision trees and visualize the rules learned from the data.
3. Understand why ensemble models tend to deliver superior results.
4. Use bootstrap aggregation to address the overfitting challenges of decision trees.
5. Train, tune, and interpret random forests.
6. Employ a random forest to design and evaluate a profitable trading strategy.

Decision trees: Learning rules from data: -

Decision trees are a machine learning algorithm that predicts the value of a target variable based on decision rules learned from data. The algorithm can be applied to regression and classification problems by changing the objective that governs how the algorithm learns the rules.

We will discuss how decision trees use rules to make predictions, how to train them to predict (continuous) returns as well as (categorical) directions of price movements, and how to interpret, visualize, and tune them effectively.

How to build a regression tree: -

Regression trees make predictions based on the mean outcome value for the training samples assigned to a given node and typically rely on the mean-squared error to select optimal rules during recursive binary splitting.

How to build a classification tree: -

A classification tree works just like the regression version, except that categorical nature of the outcome requires a different approach to making predictions and measuring the loss. While a regression tree predicts the response for an observation assigned to a leaf node using the mean outcome of the associated training samples, a classification tree instead uses the mode, that is, the most common class among the training samples in the relevant region. A classification tree can also generate probabilistic predictions based on relative class frequencies.

How to visualize a decision tree: -

You can visualize the tree using the graphviz library because sklearn can output a description of the tree using the .dot language used by that library. You can configure the output to include feature and class labels and limit the number of levels to keep the chart readable, as follows:

Random forests: -

Decision trees are not only useful for their transparency and interpretability but are also fundamental building blocks for much more powerful ensemble models that combine many individual trees with strategies to randomly vary their design to address the overfitting and high variance problems discussed in the preceding section.

Ensemble models: -

Ensemble learning involves combining several machine learning models into a single new model that aims to make better predictions than any individual model. More specifically, an ensemble integrates the predictions of several base estimators trained using one or more given learning algorithms to reduce the generalization error that these models may produce on their own.

How bagging lowers model variance: -

Bagging refers to the aggregation of bootstrap samples, which are random samples with replacement. Such a random sample has the same number of observations as the original dataset but may contain duplicates due to replacement.

Bagging reduces the variance of the base estimators by randomizing how, for example, each tree is grown and then averages the predictions to reduce their generalization error. It is often a straightforward approach to improve on a given model without the need to change the underlying algorithm. It works best with complex models that have low bias and high variance, such as deep decision trees, because its goal is to limit overfitting. Boosting methods, in contrast, work best with weak models, such as shallow decision trees.

How to build a random forest: -

The random forest algorithm expands on the randomization introduced by the bootstrap samples generated by bagging to reduce variance further and improve predictive performance. In addition to training each ensemble member on bootstrapped training data, random forests also randomly sample from the features used in the model (without replacement). Depending on the implementation, the random samples can be drawn for each tree or each split. As a result, the algorithm faces different options when learning new rules, either at the level of a tree or for each split.

Statistical Risk Models

A statistical risk model has its factors constructed using principal component analysis (PCA), which ensures that those factors have the maximum explanatory power by processing asset return time series. This technique dynamically selects factors based on the maximum commonality among asset returns rather than constraining the model on a set of pre-defined factors.

To elaborate further, the use of the PCA technique enables the formulation of statistical model factors (principal explanatory component, in this case) by clustering securities in sets in order to maximize asset return correlation within the cluster. At the same time, the clustered securities will have negligible correlations with the rest of the securities' returns, thus enabling the derived factors to capture maximum risk.

Mathematically, both fundamental and statistical risk models begin with the same linear factor model of asset returns:

R = Bf + u .... (1)

R is a vector of asset returns, *B* is a matrix of factor exposures, *f* is a vector of factor returns, and *u* is a vector of asset-specific, idiosyncratic returns.
While *R* is known, fundamental and statistical risk models approach the solution of the rest of the terms in this equation differently.

While fundamental models have the factors and their exposures (*B*) given, the equation is solved for the factor return, *f,* using regression; in macroeconomic models, the returns are available while the factor exposures are estimated. However, for statistical risk models, both the matrix of factor exposures, *B*, and the vector of factor returns, *f*, are solved for simultaneously to maximize the predictive power of the above equation.

In a nutshell, the factors of the statistical model showcase high adaptability of the factors, especially relevant in times when the market is predominantly driven by unexpected factors or extreme events.

Principal component analysis: -

Given a collection of points in two, three, or higher dimensional space, a "best fitting" line can be defined as one that minimizes the average squared distance from a point to the line. The next best-fitting line can be similarly chosen from directions perpendicular to the first. Repeating this process yields an orthogonal basis in which different individual dimensions of the data are uncorrelated. These basis vectors are called **principal components**, and several related procedures **principal component analysis** (**PCA**).

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It is often used to visualize genetic distance and relatedness between populations. PCA is either done by singular value decomposition of a design matrix or by doing the following 2 steps:

1. calculating the data covariance (or correlation) matrix of the original data
2. performing eigenvalue decomposition on the covariance matrix

Usually the original data is normalized before performing the PCA. The normalization of each attribute consists of *mean centering* – subtracting its variable's measured mean from each data value so that its empirical mean (average) is zero. Some fields, in addition to normalizing the mean, do so for each variable's variance (to make it equal to 1); see z-scores. The results of a PCA are usually discussed in terms of *component scores*, sometimes called *factor scores* (the transformed variable values corresponding to a particular data point), and *loadings* (the weight by which each standardized original variable should be multiplied to get the component score). If component scores are standardized to unit variance, loadings must contain the data variance in them (and that is the magnitude of eigenvalues). If component scores are not standardized (therefore they contain the data variance) then loadings must be unit-scaled, ("normalized") and these weights are called eigenvectors; they are the cosines of orthogonal rotation of variables into principal components or back.

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection of this object when viewed from its most informative viewpoint. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

PCA is also related to canonical correlation analysis (CCA). CCA defines coordinate systems that optimally describe the cross-covariance between two datasets while PCA defines a new orthogonal coordinate system that optimally describes variance in a single dataset.

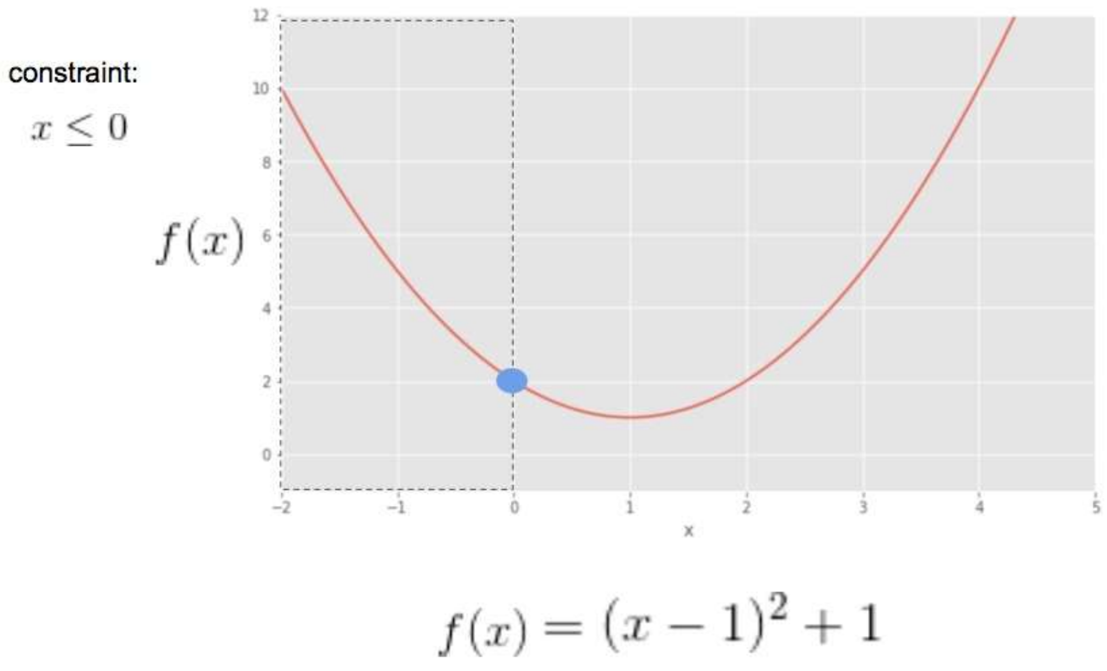
Robust and L1-norm-based variants of standard PCA have also been proposed.

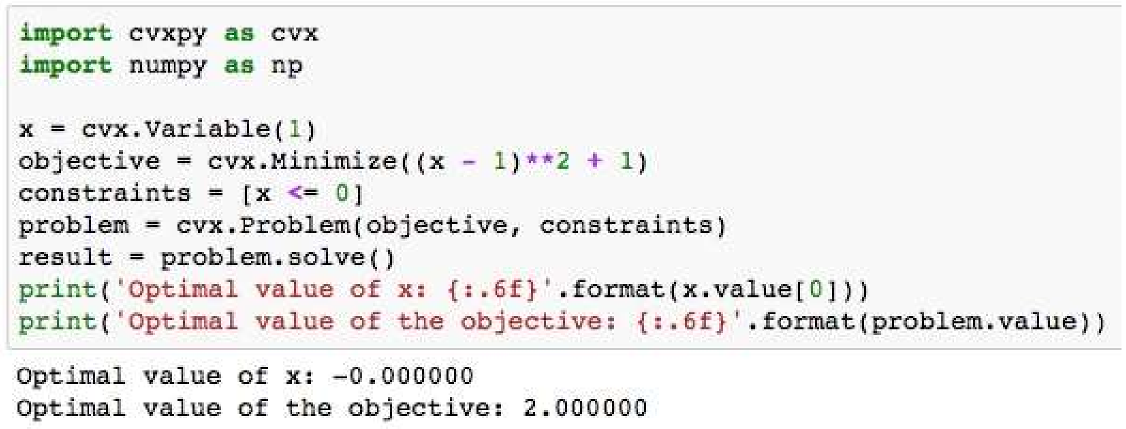
Cvxpy: -

What is cvxpy? **cvxpy** is a Python package for solving convex optimization problems. It allows you to express the problem in a human-readable way, calls a solver, and unpacks the results.

cvxpy Example

Below, please find an example of the use of cvxpy to solve the simple example we referred to in the lectures, optimizing the objective function $(x - 1)^2 + 1$ subject to the constraint $x \leq 0$.





```
import cvxpy as cvx
import numpy as np

x = cvx.Variable(1)
objective = cvx.Minimize((x - 1)**2 + 1)
constraints = [x <= 0]
problem = cvx.Problem(objective, constraints)
result = problem.solve()
print('Optimal value of x: {:.6f}'.format(x.value[0]))
print('Optimal value of the objective: {:.6f}'.format(problem.value))
```

```
Optimal value of x: -0.000000
Optimal value of the objective: 2.000000
```

How to use cvxpy

Import: First, you need to import the package: import cvxpy as cvx

Steps: Optimization problems involve finding the values of a *variable* that minimize an *objective function* under a set of *constraints* on the range of possible values the variable can take. So, we need to use cvxpy to declare the *variable*, *objective function* and *constraints*, and then solve the problem.

Optimization variable: Use cvx.Variable() to declare an optimization variable. For portfolio optimization, this will be **x**, the vector of weights on the assets. Use the argument to declare the size of the variable; e.g. x = cvx.Variable(2) declares that **x** is a vector of length 2. In general, variables can be scalars, vectors, or matrices.

Objective function: Use cvx.Minimize() to declare the objective function. For example, if the objective function $(\mathbf{x}-\mathbf{y})^2$, you would declare it to be: objective = cvx.Minimize((x - y)**2).

Constraints: You must specify the problem constraints with a list of expressions. For example, if the constraints are $\mathbf{x} + \mathbf{y} = 1$ and $\mathbf{x} - \mathbf{y} \geq 1$ you would create the list: constraints = [x + y == 1, x - y >= 1]. Equality and inequality constraints are elementwise, whether they involve scalars, vectors, or matrices. For example, together the constraints 0 <= x and x <= 1 mean that every entry of **x** is between 0 and 1. You cannot construct inequalities with < and >. Strict inequalities don't make sense in a real-world setting. Also, you cannot chain constraints together, e.g., 0 <= x <= 1 or x == y == 2.

Quadratic form: Use cvx.quad_form() to create a quadratic form. For example, if you want to minimize portfolio variance, and you have a covariance matrix **P**, the quantity cvx.quad_form(x, P) represents the quadratic form $\mathbf{x}^T\mathbf{P}\mathbf{x}$, the portfolio variance.

Norm: Use cvx.norm() to create a norm term. For example, to minimize the distance between **x** and another vector, **b**, i.e. $\|\mathbf{x}-\mathbf{b}\|_2$, create a term in the objective function cvx.norm(x-b, 2). The second argument specifies the type of norm; for an L2-norm, use the argument 2.

Constants: Constants are the quantities in objective or constraint expressions that are not Variables. You can use your numeric library of choice to construct matrix and vector constants. For instance, if x is a cvxpy Variable in the expression A*x + b, A and b could be Numpy ndarrays, Numpy matrices, or SciPy sparse matrices. A and b could even be different types.

Optimization problem: The core step in using cvxpy to solve an optimization problem is to specify the problem. Remember that an optimization problem involves minimizing an *objective function*, under some *constraints*, so to specify the problem, you need both of these. Use cvx.Problem() to declare the optimization problem. For example, problem = cvx.Problem(objective, constraints), where objective and constraints are quantities you've defined earlier. Problems are immutable. This means that you cannot modify a problem's objective or constraints after you have created it. If you find yourself wanting to add a constraint to an existing problem, you should instead create a new problem.

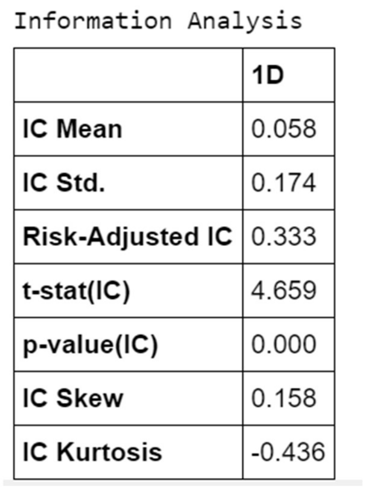
Solve: Use problem.solve() to run the optimization solver.

Status: Use problem.status to access the status of the problem and check whether it has been determined to be unfeasible or unbounded.

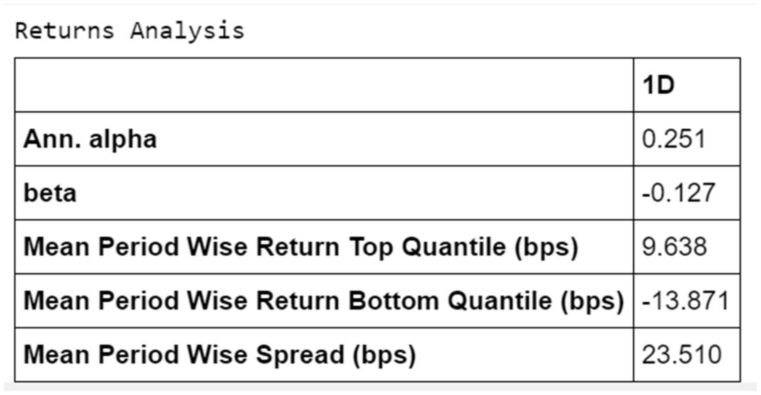
Results: Use problem.value to access the optimal value of the objective function. Use e.g. x.value to access the optimal value of the optimization variable.

Chapter IV

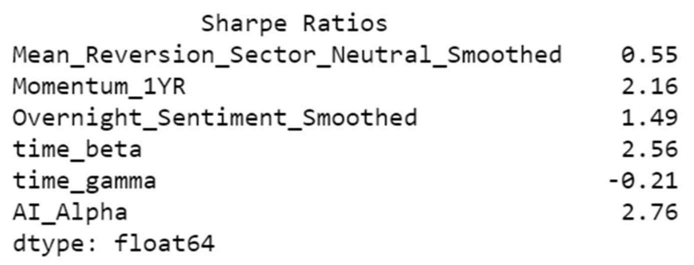
RESULTS AND PERFORMANCE ANALYSIS

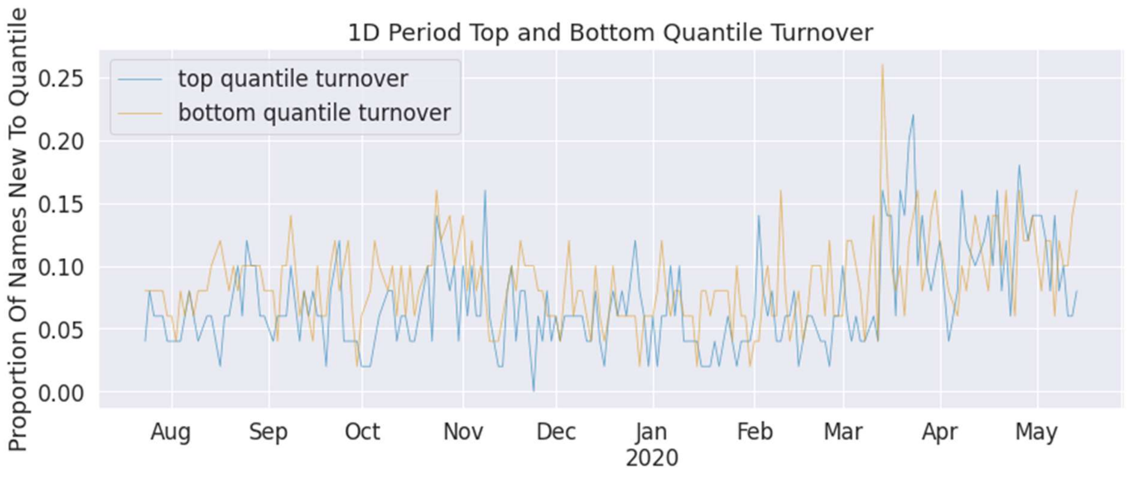

Information Analysis

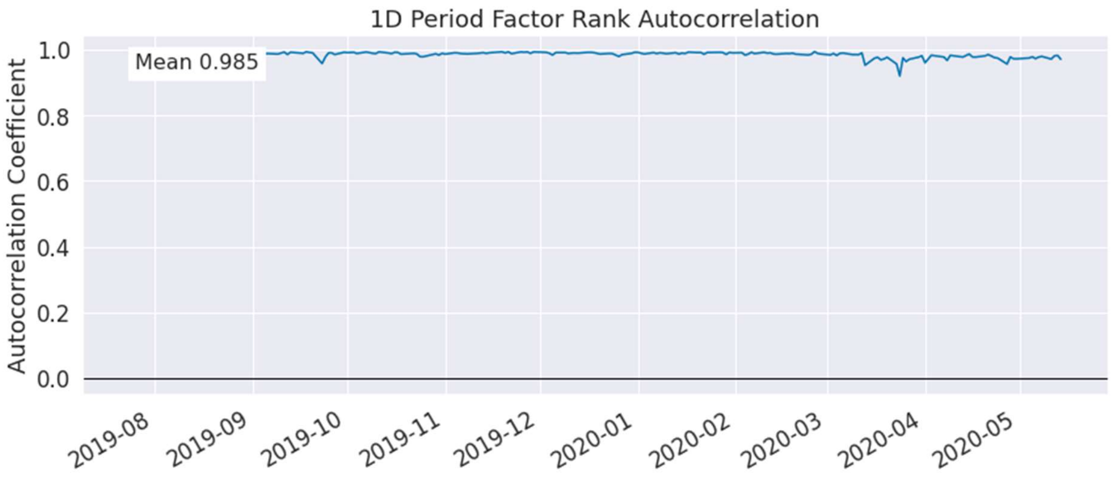
	1D
IC Mean	0.058
IC Std.	0.174
Risk-Adjusted IC	0.333
t-stat(IC)	4.659
p-value(IC)	0.000
IC Skew	0.158
IC Kurtosis	-0.436

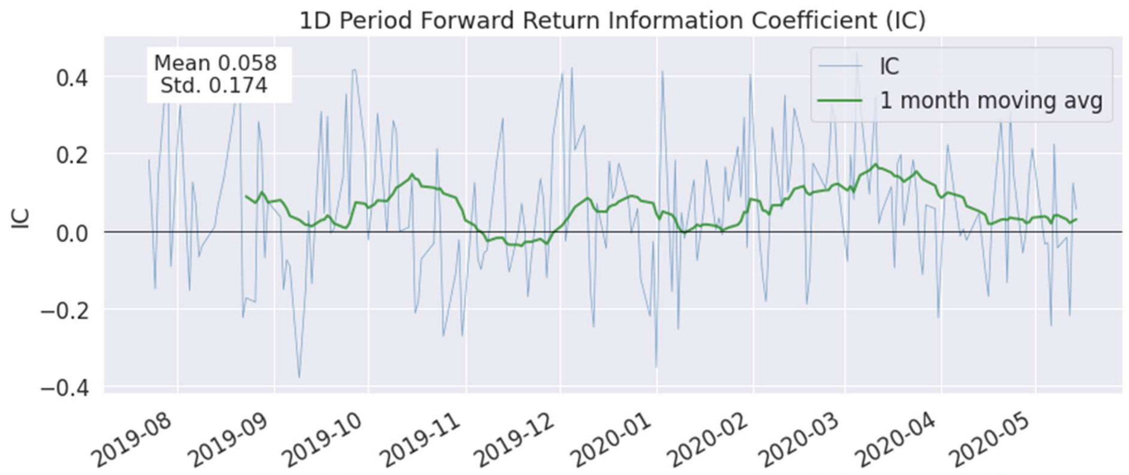

Returns Analysis

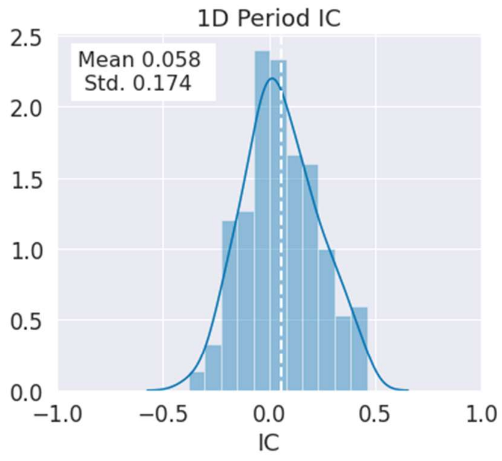
	1D
Ann. alpha	0.251
beta	-0.127
Mean Period Wise Return Top Quantile (bps)	9.638
Mean Period Wise Return Bottom Quantile (bps)	-13.871
Mean Period Wise Spread (bps)	23.510

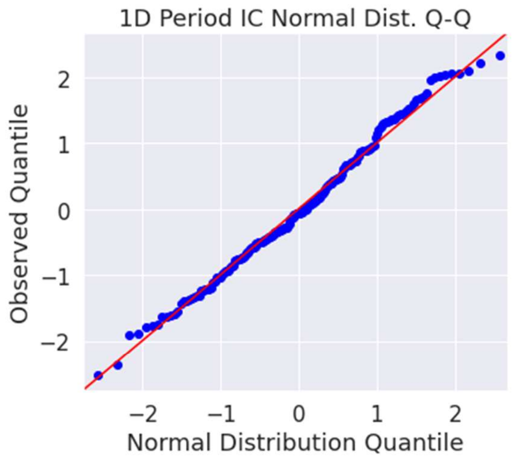

```
             Sharpe Ratios
Mean_Reversion_Sector_Neutral_Smoothed    0.55
Momentum_1YR                              2.16
Overnight_Sentiment_Smoothed              1.49
time_beta                                 2.56
time_gamma                               -0.21
AI_Alpha                                  2.76
dtype: float64
```

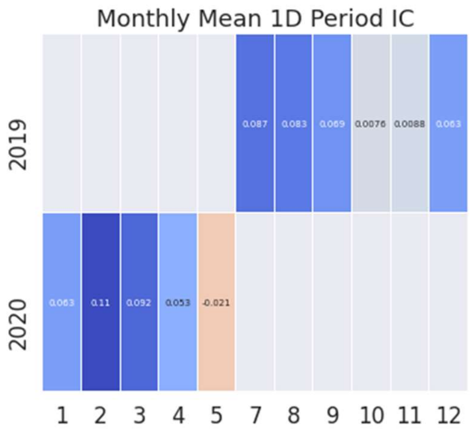

1D Period Top and Bottom Quantile Turnover
Proportion Of Names New To Quantile
top quantile turnover
bottom quantile turnover
0.25
0.20
0.15
0.10
0.05
0.00
Aug
Sep
Oct
Nov
Dec
Jan
2020
Feb
Mar
Apr
May

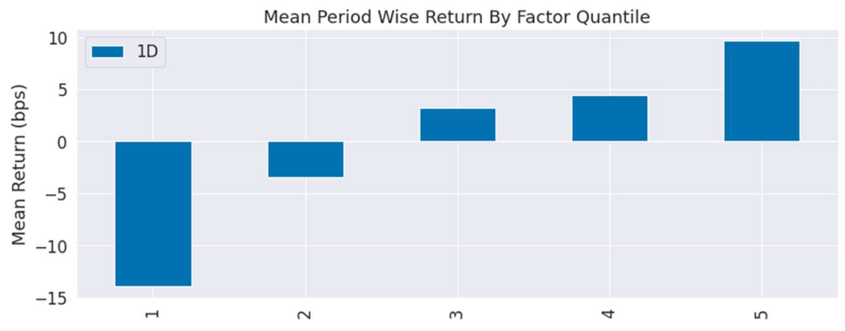

1D Period Factor Rank Autocorrelation
Autocorrelation Coefficient
Mean 0.985
1.0
0.8
0.6
0.4
0.2
0.0
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05

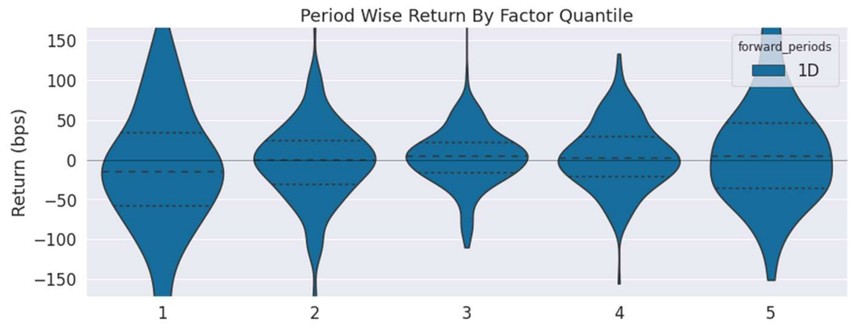

1D Period Forward Return Information Coefficient (IC)
Mean 0.058
Std. 0.174
IC
1 month moving avg
IC
0.4
0.2
0.0
−0.2
−0.4
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05

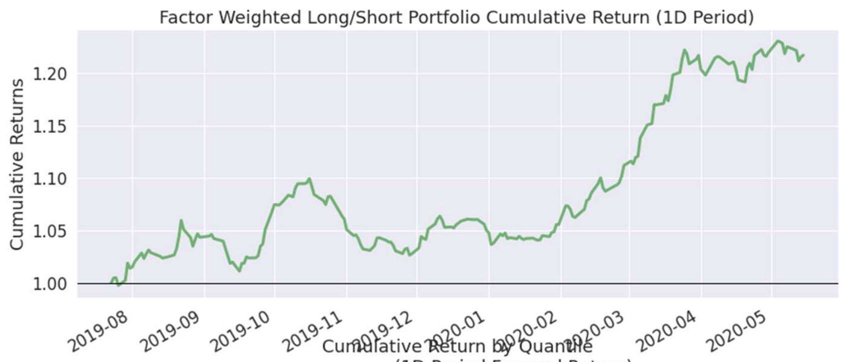

1D Period IC
Mean 0.058
Std. 0.174
2.5
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
0.5
1.0
IC

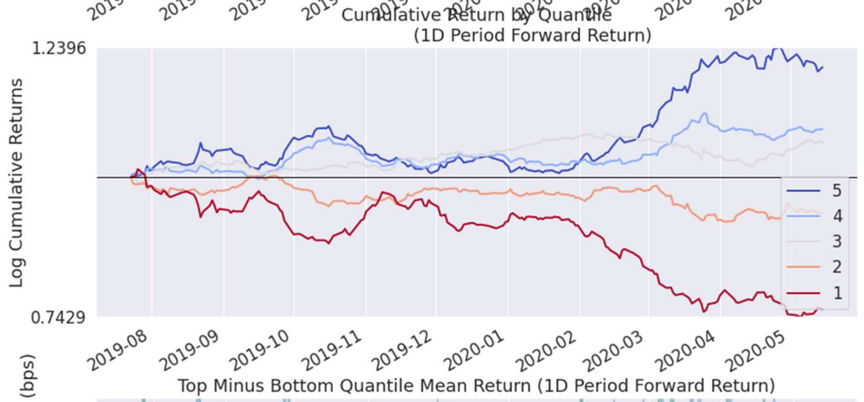

1D Period IC Normal Dist. Q-Q
Observed Quantile
2
1
0
−1
−2
−2
−1
0
1
2
Normal Distribution Quantile

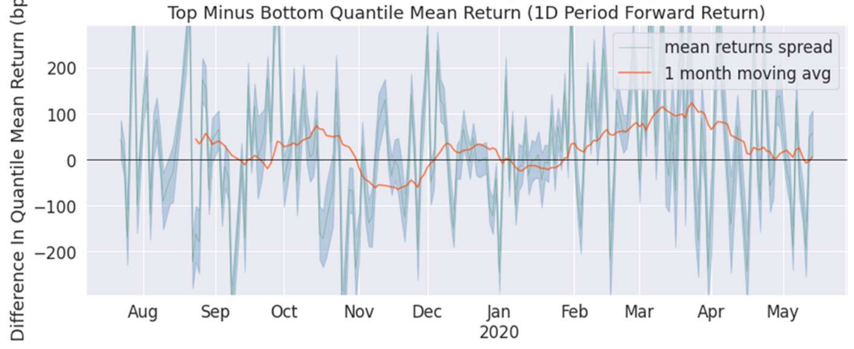

Monthly Mean 1D Period IC
2019
2020
0.087
0.083
0.069
0.0076
0.0088
0.063
0.063
0.11
0.092
0.053
-0.021
1
2
3
4
5
7
8
9
10
11
12

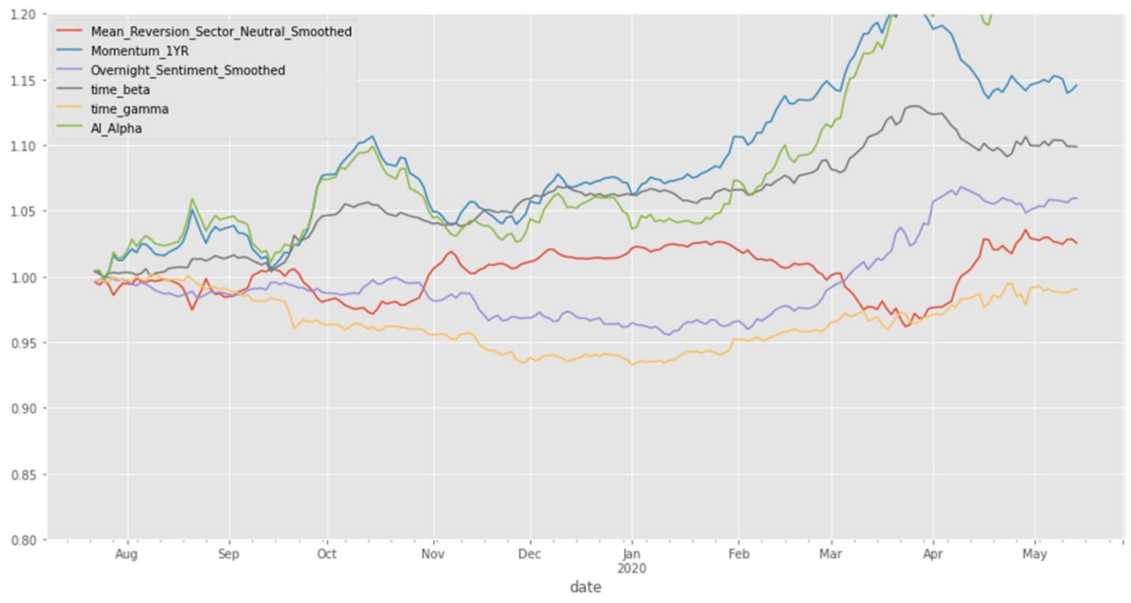

Mean Period Wise Return By Factor Quantile
1D
Mean Return (bps)
10
5
0
−5
−10
−15
1
2
3
4
5


Period Wise Return By Factor Quantile
forward_periods
1D
Return (bps)
150
100
50
0
−50
−100
−150
1
2
3
4
5

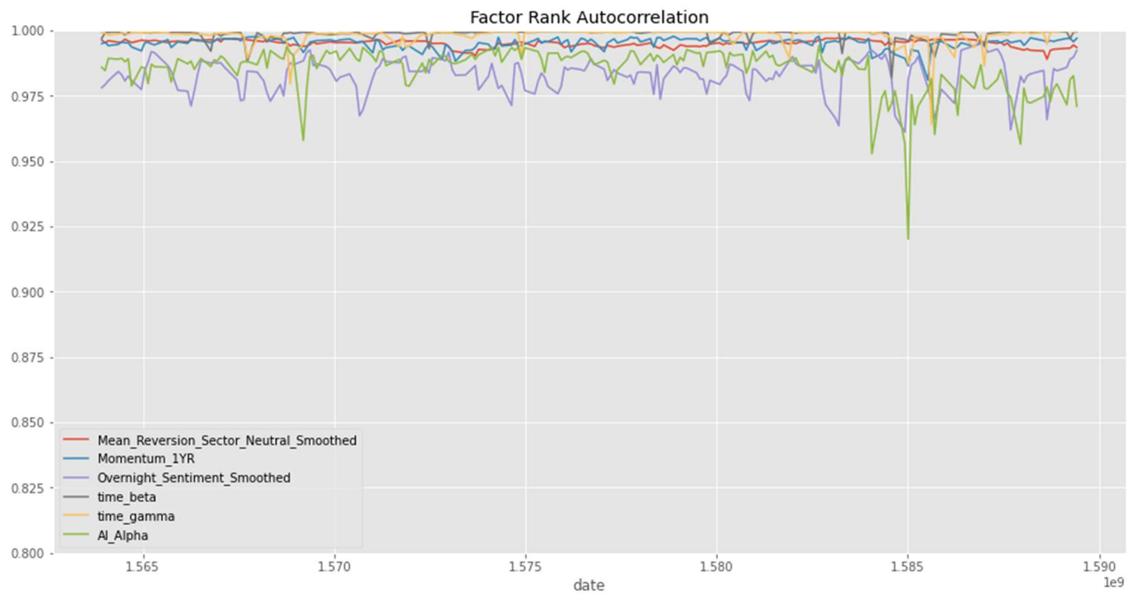

Factor Weighted Long/Short Portfolio Cumulative Return (1D Period)
Cumulative Returns
1.20
1.15
1.10
1.05
1.00
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05


Cumulative Return by Quantile
(1D Period Forward Return)
Log Cumulative Returns
1.2396
0.7429
5
4
3
2
1
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
2020-04
2020-05


Top Minus Bottom Quantile Mean Return (1D Period Forward Return)
mean returns spread
1 month moving avg
Difference In Quantile Mean Return (bps)
200
100
0
−100
−200
Aug
Sep
Oct
Nov
Dec
Jan
2020
Feb
Mar
Apr
May



Mean_Reversion_Sector_Neutral_Smoothed
Momentum_1YR
Overnight_Sentiment_Smoothed
time_beta
time_gamma
AI_Alpha
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
Aug
Sep
Oct
Nov
Dec
Jan
2020
Feb
Mar
Apr
May
date



Factor Rank Autocorrelation
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Mean_Reversion_Sector_Neutral_Smoothed
Momentum_1YR
Overnight_Sentiment_Smoothed
time_beta
time_gamma
AI_Alpha
1.565
1.570
1.575
1.580
1.585
1.590
1e9
date

Chapter V

CONCLUSION AND FURTHER WORK

Key Takeaways and Lessons Learned: -

1. Data is the single most important ingredient.
2. Domain expertise helps unlock value in data.
3. ML is a toolkit for solving problems with data.
4. Model diagnostics help speed up optimization.
5. Beware of backtest overfitting.
6. How to gain insights from black-box models.

Conclusion: -

Despite using some commonly used alpha factors we were able to get good performance on the testing set i.e. the portfolio's return was nearly 20% of the capital used to trade and we got a good Sharpe ratio of 2.76 also the mean period wise spread is much higher which is a trait of a good Alpha Factor. Despite when the Indian markets were falling this portfolio was able to generate significant returns.

Further Work: -

1. Using Text Data for Trading: Sentiment Analysis.
2. Topic Modelling for Earnings Calls and Financial News.
3. Using Deep Learning for Trading.
4. Using CNN: Time Series as Images and Satellite Image Classification.
5. Using RNN for Trading: Multivariate return series and text data.
6. Using Conditional Autoencoders for Asset Pricing and GANs.
7. Using Reinforcement Learning to Build a Trading Agent.
8. Using More Sophisticated Alpha Factors Rather Than Commonly Used Alpha Factors.
9. Using Fundamental Data for Computing Alpha Factors.
10. Using Boosting Methods in Place of Tree Based Methods.

Chapter VI

REFERENCES

1. https://github.com/stefan-jansen/machine-learning-for-trading
2. https://onlinelibrary.wiley.com/doi/abs/10.1002/for.2446
3. [The Digital Universe in 2020](https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf)
4. [Big data: The next frontier for innovation, competition, and productivity](https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation), McKinsey 2011
5. [McKinsey on Artificial Intelligence](https://www.mckinsey.com/featured-insights/artificial-intelligence)
6. [Big Data and AI Strategies](http://valuesimplex.com/articles/JPM.pdf), Kolanovic, M. and Krishnamachari, R., JP Morgan, May 2017
7. [Quantifying Trading Behaviour in Financial Markets Using Google Trends](https://www.nature.com/articles/srep01684), Preis, Moat and Stanley, Nature, 2013
8. [Quantifying StockTwits semantic terms' trading behaviour in financial markets: An effective application of decision tree algorithms](https://www.sciencedirect.com/science/article/pii/S0957417415005473), Al Nasseri et al, Expert Systems with Applications, 2015
9. Dissecting Anomalies by Eugene Fama and Ken French (2008)
10. Explaining Stock Returns: A Literature Review by James L. Davis (2001)
11. Market Efficiency, Long-Term Returns, and Behavioral Finance by Eugene Fama (1997)
12. The Efficient Market Hypothesis and It's Critics by Burton Malkiel (2003)
13. The New Palgrave Dictionary of Economics (2008) by Steven Durlauf and Lawrence Blume, 2nd ed.
14. Anomalies and Market Efficiency by G. William Schwert25 (Ch. 15 in Handbook of the- "Economics of Finance", by Constantinides, Harris, and Stulz, 2003)
15. Investor Psychology and Asset Pricing, by David Hirshleifer (2001)
16. https://github.com/quantopian/zipline
17. https://github.com/quantopian/alphalens

18. https://github.com/pandas-dev/pandas
19. https://github.com/numpy/numpy
20. The Statistics of Sharpe Ratios, Andrew Lo, Financial Analysts Journal, 2002
21. Active Portfolio Management: A Quantitative Approach for Producing Superior Returns and Controlling Risk by Richard Grinold and Ronald Kahn, 1999
22. How to Use Security Analysis to Improve Portfolio Selection, Jack L Treynor and Fischer Black, Journal of Business, 1973
23. Portfolio Constraints and the Fundamental Law of Active Management, Clarke et al 2002
24. Portfolio Selection, Harry Markowitz, The Journal of Finance, 1952
25. The Capital Asset Pricing Model: Theory and Evidence, Eugene F. Fama and Kenneth R. French, Journal of Economic Perspectives, 2004
26. https://github.com/cvxgrp/cvxpy
27. https://github.com/scikit-learn/scikit-learn