UNIVERSITY OF TURKU

NEURAL NETWORK -GARCH-COPULA PORTFOLIO OPTIMIZATION WITH MULTIFACTOR DATA

Master's
thesis plan

Author:
Johannes Salonen

Supervisor:
Ph.D. Luis Alvarez Esteban

May 20, 2024
Turku

UNIVERSITY OF TURKU

ABSTRACT

☐ Bachelor's thesis
☑ Master's thesis
☐ Licenciate's thesis
☐ Doctor's thesis

Subject	Finance	Date	20.05.2024
Author(s)	Johannes Salonen	Student number	1800638
		Number of pages	58+33
Title	Neural network -GARCH-copula portfolio optimization with multifactor data		
Supervisor(s)	Ph.D. Luis Alvarez Esteban		

Abstract

Return forecasting and portfolio selection have fascinated financial academics and practitioners alike for a long time. With the wake of artificial neural networks, and as importantly the computational capacity to take advantage of such models, financial academics and practitioners have turned their attention to more and more complex models to better understand and predict the behaviour of financial markets.

In literature, often one, seldom two, and rarely more categories of variables are utilized in return prediction. In this thesis, eight categories of variables are considered in return prediction, as data has become more available and immediate, and as such taking a multifactor approach was hypothesized to improve prediction accuracy. In literature when sophisticated neural network predictions have been considered, portfolio selection has often been simplified to equally weighted or similar approaches. In this thesis an error-GARCH-copula approach was utilized with the multifactor neural network to improve portfolio performance, as risk measures are just as important as returns in portfolio performance.

The combined methodology failed to consistently outperform market portfolios or portfolios based on simple linear regression predictions. The main issue with performance was seen to stem more from lack of informative predictions rather than the performance of the copula. Although as the return prediction relied heavily on dispersion measures, the NN-GARCH-copula portfolios also had worse risk measures than market portfolios. Based on these findings, it is suggested that input selection, more sophisticated architectures, and dynamic informative prediction intervals should be considered when using multifactor NN-copula approach.

Keywords	Neural network, BiLSTM, CNN-BiLSTM, CNN-BiLSTM-Attention, Copula, Portfolio
Further information	

The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin OriginalityCheck service.

Contents

Contents

1 Introduction

1.1 Background

1.1.1 Modern Portfolio Theory

Before 1952, finance literature considered the portfolio optimization problem of risk-return tradeoffs from an ad hoc perspective. In 1952 Harry Markowitz introduced what came to be known as modern portfolio theory (MPT) in his research paper Portfolio Selection (Markowitz 1952). MPT provided the notion that portfolio selection should be viewed as a problem of mean-variance-optimization (MVO). In MVO the portfolio selection problem states that for a given target return, there exists a portfolio allocation that achieves this target return in mean, while minimizing the variance of the portfolio. In the MPT framework, all allocations with the same mean return, but with higher variances, are deemed inefficient. (Kolm et al. 2014) Later Markowitz reflected on "why mean and variance" as the measures of expected return and risk (1999) and noted alternative approaches to MVO, including the use of other measures of risk and/or return (2010).

Reflecting on MPT, several opportunities for methodological improvement can be seen. Firstly, the use of historical mean, variance, and covariance as approximators of future mean, variance, and covariance behaviour is problematic in both short- and long-term. In the short-term, this has been shown numerous times, for example by Ledoit et al. (2003) who conducted portfolio optimization between US, North America, European, and World indices using a multivariate GARCH model and derived expected returns, variances, and covariances, and found that the model outperformed a (sliding window) historical mean, variance, and covariance approach. In the long term, the stationarity assumptions are simply logically flawed for real-world financial instruments, as the pricing dynamics of financial instruments are consequences of human society, which throughout history has shown to be dynamic. As an example of this, Awokuse et al. (2008) studied the structural changes in market dependencies between Asian emerging markets and the markets of more developed Japan, UK, and US, and they, unsurprisingly, found that the cointegration structure of the markets was affected, both positively and negatively, by financial liberalization policies.

Secondly, even if we could better define the expected return, variance, and covariance structure, as Markowitz himself reflected, why specifically use them in portfolio selection? In the traditional MPT-MVO framework, the portfolio level variance is an additive function of variances and covariances in the selection group. Therefore if non-linear dependencies exist they are incorrectly factored in the model. As an example of non-linear dependencies existing, Kenourgious et al. (2010) investigated the dependency structures between four emerging markets Brazil, Russia, India, and China and two developed markets US and UK. They found statistically significant non-linear and asymmetric dependencies between the markets. Therefore, while covariance-based MVO may be an easy implementation both analytically and computationally, it very well may not be optimal. Related to these covariance structures, there is the assumption that the returns are normally distributed, which has also shown to be false with stock-returns, as they are commonly seen to be fat-tailed and positively skewed (e.g. Kon 1984).

1.1.2 Alternative approaches to portfolio optimization

Given the limitations of MPT, alternative approaches have been proposed to return prediction and portfolio optimization.

Return prediction and forecasting has fascinated researchers and practitioners alike. One of the most commonly used regression models is the Auto-Regressive Moving Average (ARMA), where the prediction is a linear additive function of the previous lags and estimated errors. (Shah et al. 2019) Another often used model is the Threshold Auto-Regressive (TAR), which is a non-linear model, where the prediction is conditioned on some lagged variable(s) being within a certain range. The TAR-model is able to capture non-linearity that a linear AR-model can not. (Hansen 2011) On top of these statistical approaches, newer machine learning techniques have arised. These techniques include, among others, template matching, support vector machines, random forests, deep neural networks (DNNs), and recurrent neural networks (RNNs). (Shah et al. 2019) The input data for these predictions can be divided into structured and unstructured. Structured information includes market information i.e. price movement, bid/ask spreads and volume information, technical indicators, which can be divided into oscillators and momentum indicators, and economic indicators, which include both macroeconomic and financial statement information. Unstructured information can be derived from news channels, social networking sites and (micro)blogs. (Bustos and Pomares-Quimbaya 2020)

The MVO despite its flaws continues to see application in literature. However, there have been advances, which include the inclusion of transaction costs, specified constraints, quantifying the impact of estimations errors and making the optimization problem intertemporal. (Kolm et al. 2014) Artificial neural networks (ANNs) have also been utilized in portfolio selection. ANN strategies have been combined with other machine learning methods, such as genetic algorithms, stochastic algorithms, Q-learning, and fuzzy logic. There are two generally holding findings here. Firstly, ANN-based strategies perform better than traditional models such as MVO. Second, hybrid models, i.e those in which more than one machine learning technique is adopted, perform even better than simpler machine learning models. (Bahrammirzaee 2010) Another newer approach to this problem is the copula-based approach. A copula is a function that is able to map the marginal distributions of a vector of variables into a joint distribution. Copulas can be used in portfolio optimization, as they are able to reproduce non-normal multivariate distributions. (Patton 2012) Highly related to the use of copulas is the estimation of Value at risk (VaR) and Conditional VaR (CVaR) risk measures, as copulas are able to give better predictions of the tail-end behaviour of portfolio distributions. (Low 2018)

1.2 Motivation

Given that financial literature has created the foundations for robust portfolio optimization, I would like to expand on this and contribute to the growing domain. My aim is to investigate the possible performance of a neural network model that leverages multi-sourced information, with a GARCH-copula-based portfolio optimization. The motivations of this thesis are the following.

The first is the expansion of the NN-copula literature, which as of now is still fairly limited, despite taking advantage of two methods which have been shown great interest in financial literature over the past two decades. Previous studies have either focused on

a small number of indices (e.g. Zhao et al. 2018) or on a limited number of stock assets (e.g. Li and Muwafak 2021). In this thesis all stock which were a part of the S&P 500 during the entire period of 2015-2023 were used, totaling 329 stocks. The application of the NN-copula framework to a large number of underlying instruments informs of the applicability and generalizability of the methodology. The second major contribution to the NN-copula literature is the examination of transaction costs, and their dynamic effect on the model's performance.

The second motivation is the investigation of a multifactor input space with state-of-the-art neural network architectures. Previous studies using neural networks in stock prediction have ignored this holistic approach, instead often focusing on one variable category. While this is obviously a valuable approach, in a world of increased availability of data, computational capacity, and artificial neural network knowledge, gaining insight into possible methods to take advantage of that the vast data becomes of key importance, to both academics and practitioners alike.

1.3 Research question

The research questions of this thesis is the following:

- How can a neural network model be effectively deployed to integrate multiple data sources for predicting stock returns?
- How can copula-based mean-CVaR optimization be deployed using neural network predictions for stock returns?

The secondary research questions are:

- How do transaction costs impact the returns of an NN-copula portfolio?
- How do the returns of NN-copula portfolios compare to those of market and risk-naive portfolios?

2 Return forecasting

Whether forecasting future asset returns can be done is an everlasting and underlying question for both practitioners and academics of finance. For practitioners, return forecasting provides an avenue to create overperformance compared to market competitors. For academics, the forecastability of returns has implications on market modelling and efficiency. (Rapach and Zhou, 2013)

2.1 Return forecasting and the Efficient Market Hypothesis

The efficient market hypothesis (EMH) states that "A market is efficient with respect to information set Ω_t, if it is impossible to make economic profits by trading on the basis of Ω_t." Here it is important to note that economic profits are defined by risk adjusted returns. The EHM can be divided into three different hypothesis: the weak form of the Efficient Market Hypothesis (weak EMH). the semi-strong form of the Efficient Market Hypothesis (semi-strong EMH), and the strong form Efficient Market Hypothesis (strong EMH) In the

weak EMH, thr information set Ω_t is is assumed to be solely the information of past price history available at time t. In the semi-strong EMH, Ω_t is expanded to include all publicly available information at time t. In the strong-EMH, Ω_t is yet again expanded, this time to include all information available at time t, including insider information. (Jensen, 1978)

Certainly one could argue that if return forecasting were possible, the EMH would not even hold. This however may not be the case. The reasons for this stem from the information set used in forecasting, time-varying risk premia, transaction costs and trading restriction, intrinsic asset values and self-destruction of predictability. (Timmermann and Granger, 2004)

2.1.1 Information set

Any forecasting model is an implicit test of the EMH. The form of the EMH tested depends on the information set Ω_t used when making predictions. If the information set used only contains information on the past and current trading information, including price, return and volume, the forecasting model tests the weak-EMH. If the information set used is expanded to contain other variables not explicitly included in the trading information, the model can be seen as a test for the semi-strong EMH. In the most rare case, where insider information is taken into account, the strong-EMH. In general, there has been acceptance of either weak- or semi-strong EMH, while strong-EMH has seen less support. From a forecaster's point of view, the cost of acquiring and difficulties in measuring insider information is often considered to be reasons not to attempt to forecast with the full information set. (Timmermann and Granger, 2004)

An underlying assumption regarding the information set is the general rationality and symmetrically distributed irrationality of investors. The uniformity of information sets held by market participants also requires uniformity in the processing of that information for meaning. This however may not be the case, as studies in behavioural finance have shown psychological biases to be present among investors, and these biases affect the ways in which information is processed. Herding, the behaviour of imitating and following the actions of others instead of conducting independent analysis. This may lead to market prices failing to reflect all relevant information, resulting in inefficiency. Another bias that impacts information processing is the availability heuristic, which refers to the establishment of future scenario likelihoods based on the ease at which similar scenarios can be recalled. Given that the ease of recalling is correlated among market participants, the availability heuristic leads to investors overweighting more recent information and underweighting less recent information. (Hon-Snir et al. 2012) Given that this availability heuristic is similarily distributed amongst market participants, which seems reasonable as all market participants are always operating in the same time period, this may very well result in overall market inefficiency.

2.1.2 Time-varying risk premia

While EMH implies that there exists no arbitrage opportunity, this statement is equivalent to there being no predictability in returns, the main reason being time-varying risk premia of return. Time-varying risk premia means that there exists variation in the expected excess returns over a risk-free rate. Here the difference between the EMH and the random walk hypothesis becomes evident. The random walk hypothesis simply postulates that

$P_{t+1} = \mu + P_t + e$, where e is $i.i.d.$ with mean zero, μ is a drift constant, and $\mathbb{E}(R_{t+1}) = \mu$, while in the EMH framework given a stochastic discount factor Q_{t+1} this does not necessarily apply.

$$\mathbb{E}[R_{t+1}|\omega_t] = \frac{-\text{COV}(R_{t+1}, Q_{t+1}|\omega_t)}{\mathbb{E}(Q_{t+1}|\omega_t)}.$$

This implies that the forecasted return need not be 0 or the mean drift coefficient, like the random walk hypothesis implies, and e is not $i.i.d.$. (Timmermann and Granger, 2004)

However time varying risk premia fails to adequately explain return anomalies. Lewellen and Nagel (2004) conducted a rolling window conditional CAPM analysis on two well known anomalies, momentum and book-to-market. To test the momentum anomaly, they constructed a portfolio that goes long on the top decile of stocks, and shorts the bottom decile of stocks ranked by their performance over the past six months. To test the book-to-market anomaly, they constructed a portfolio which goes long on the top decile of stocks, and shorted the bottom decile of stocks ranked by their book-to-market ratio. The stocks were from the NYSE and Amex during the period 1964 to 2001. While they found considerable time-variability in the risk premium, the risk premiums failed to fully explain the performance of two anomalous portfolios. Both of the portfolios created unexplained positive returns when the portfolios were updated at a quarterly, semiannual and annual frequency. All unexplained returns were above a level of two standard deviations.

Adrian and Franzoni (2009) expanded the conditional CAPM and return anomaly literature, by creating a mean reverting Kalman conditional beta, and applying the model to book-to-market portfolios constructed by pentiles of book-to-market and market capitalisation. With their new conditional CAPM, 9 of the 25 portfolios created statistically significant (at over two deviations from zero) alpha, of which one, the portfolio from the smallest pentile of both book-to-market and market capitalization, was negative. The standard OLS conditional CAPM resulted in 12 of the 25 producing statistically significant alphas. This means that while the beta can be better explained by more sophisticated modelling, this does not remove the anomalous evidence for significant alpha. They also provided conditioning variables, which included the consumption-wealth ratio (CAY), high minus low (HML), term spread (TERM), and lagged market index (MRKT). What they found was that the pricing error, alpha, did not deviate by over two standard deviations from zero at any point during their testing period, when conditioning the beta. When taking into account both the new model and the conditioning variables, only one of the 25 portfolios had statistically significant alpha. While the conditioning variables obviously reduce the unexplained anomaly, the conditioning should not be necessary, as the anomalies are explained by non-market risk factors.

Thus while time-varying market risk premia can be seen to explain some of the anomalous excess returns, it fails to fully explain them in the cases of momentum and book-to-market alpha.

2.1.3 Transaction costs and trading restrictions

Taking into account transaction costs and trading restrictions makes outperforming forecasting less likely even with significant predictable components in returns. Transaction costs place a limit on the degree to which predictability in returns can be utilized. For example, let's say there were an accurate forecasting model, which would predict an excess

return of 0.5 %, and the investor were to experience an expected 0.5 % transaction cost in completing the trade, the inefficiency would not be exploitable, as the expected overperformance would be zero. This means transaction costs limit the temporal inefficiency in the market, as any inefficiency has to be greater than the transaction costs incurred when entering trades where one would benefit from that inefficiency. Realistic trade restrictions also limit market inefficiency. These include limits to short-selling and position sizes. Based on these, market inefficiencies which cannot be taken advantage of without realistically infeasible positions in practice do not exist. (Timmermann and Granger, 2004)

An argument could be that the significance of transaction costs is constantly decreasing, especially in high volume markets, like the stock market. Menkveld (2016) studied the change in NYSE and NASDAQ transaction costs by comparing the years 2001 and 2011, which were meant to represent periods before and after high frequency trading became common in the mentioned stock markets. What they found was that the effective spread more than halved from around 20 basis points in 2001 to around 5 basis points in 2011. The commission experienced by retail investors also more than halved from around 45 to 20 basis points, and for institutional investors commission reduced from around 12 to 5 basis points. Between the periods, total volume also more than doubled. As Frazzini et al. (2018), among others, have shown that transaction costs logarithmically increase as a function of trade size compared to total daily volume, this would be expected.

Simply based on the increasing volume in the markets, without even accounting for more efficient trading technology, one could surmise that transaction costs are constantly decreasing, which decreases the level of overperformance required from predictions for them to be economically viable. This would indicate that with transaction costs decreasing, the opportunity to actually see predictive models utilized would be higher.

2.1.4 Intrinsic asset value

Intrinsic asset value is the theoretical notion of a correct valuation for an asset. Therefore one could argue that if the price of an asset becomes dislocated from its intrinsic value, the market is inefficient. Using this definition, the market does not align with the EMH if the market price deviates from the true value in either degree or persistance. There is however a major problem with this notion, which is that the intrinsic value, if even existing, is non-observable or measurable. Therefore one could attempt to test this inefficiency, based on whether there are persistent changes in the difference between economic output and asset prices. Bubbles, where differences between asset prices and economic output come apparent however do not necessarily disprove the EMH, given that the time-varying risk premia reflect the market environment. (Timmermann and Granger, 2004)

While intrinsic asset value is unobservable, this has not stopped finance practitioners and academics from trying to do so via fundamental analysis. In order for intrinsic asset value based strategies conforming to efficient markets, the views of practitioners must be both rational and symmetrically biased across investors. This however may not be the case. For example the anomaly of post-earnings announcement drift, the continued same direction price change drift following earning news, can be attributed to limited attention. When investors fail to update their models based on the newest information, instead opting to keep short-term estimates the same based on earlier analysis. When there exists a cost to attention, this can result in investors not taking into account certain,

especially new, information. (Hirshleifer et al. 2011) Another possible explanation to the post-earning announcement drift is the relative activity of non-institutional investors. This would indicate that non-sophisticated investors hold non-symmetric irrational views which results in market inefficiency. (Bartov et al. 2000)

Thus there exists significant differences in the real long-term intrinsic values of assets and the market valuation of those assets. This can especially be seen in over- and underreactions to earnings data, which may be caused by either lack of attention, attention costs, or lack of investor sophistication.

2.1.5 Self-destruction of predictability

Even when assuming that the underlying reality driving price changes and dividends or coupons remain stationary, the forecasting models applied need not be. Once the technology and the capabilities to use that technology exist in forecasting, the possible outperforming model is unlikely to stay unique and likely becomes distributed amongst forecasting market participants. This distribution then translates to changes in demand of assets based on shared forecasting outcomes, which dissipates any outperformance related to the utilization of that model. Out-of-sample testing, where the model is trained on a separate sample and then its performance is tested on the sample not within the training sample, is generally thought to be a robust way in which to evaluate the possible real or future performance of the model. However out-of-sample testing does not remove the possibility of the self-destruction of predictability, as other market participants are able to create the very same models, which then in the deployment phase in the future, together eliminate any prediction power the model had caused by changes in the demand from following the model. Thus given enough symmetry in forecasting models across capital, forecasting excess return is not feasible. (Timmermann and Granger, 2004)

McLean and Pontiff (2016) investigated the effect of academic publication on characteristic portfolios had on the underlying characteristic portfolio post-publication performance. They found that the post-publication excess return decay was about 35 %. The decay of excess return was especially evident in characteristics, which had less arbitrage costs. These included strategies, which included high market value, high liquidity, low idiosyncratic risk and dividend paying stocks. Post-publication, the volume, variance, short interest, and correlation with other published characteristics increased. The study however only looked at published research, which means it omits the great amount of privately conducted research done by market participants. It it also significant that while the excess returns decreased, they did so on average only by roughly by a third, meaning there are significant limits to the self-destruction of predictability, one of the reasons being limits to arbitrage.

Despite the proposed self-destruction, as of 2010, only 22 % of equity mutual funds were explicitly indexed, meaning the rest, 78 %, was being actively managed. (Cremers et al. 2016) There is a great evidence that actively managed funds produce no different return than the benchmarks on average (e.g. Fama and French 2010), and underperform after expenses. There is however also evidence that supports active management. Cremers and Petajisto (2009) examined the effect of the active share, the share of which an equity mutual fund differs from benchmark index holdings, and found a positive relationship between active shares and the overperformance of the funds. Elton et al. (2012) found autocorrelation in the overperformance of mutual funds, which should not exist, if

prediction were not possible. Lin (2014) studied equity sector mutual funds operated by Fidelity Investments, and found that most funds which practiced active selection produced both higher nominal returns and risk-adjusted returns.

While there is great evidence to support the self-destructibility of active management, there is certainly a clear argument to be made that active management should exist in equilibrium with passive management. As less active management there is in the market, the less efficient the markets should be, and the more active management there is, the less inefficiency there is and therefore the opportunity to create overperformance lessens, making active management economically non-productive. Thus the existence of active management is in itself evidence of predictability in the markets.

2.2 Historical price and return data

The two most prevalent historical return related forecasting strategies are mean-reversion and momentum. Mean-reversion of returns refers to negative autocorrelation of returns, and momentum refers to the phenomenon that assets which have done well historically will also do so in the future. While these may appear contradictory the two are generally apparent on different time horizons (Balvers and Wu, 2006)

One would assume that a dynamic autocorrelation structure, which would fit both of these patterns, would be one that showcases more negative autocorrelation following negative returns and more positive autocorrelation following positive returns. This however may not the case, at least at the first lag, as quantile regression models have shown this to be either correct (Barnes and Hughes, 2002 (small companies)) or the very opposite (Barnes and Hughes, 2002 (large companies); Baur et al., 2012). As mentioned, these quantile analyses only consider the latest lag, which means uncertainty in the coefficients does not indicate that the phenomena do not exist. Another way in which the phenomena can be tested is via characteristic based portfolio selection. Balvers and Wu (2006) conducted a portfolio selection strategy analysis on 18 developed country indices. They constructed long and long-short portfolios made up of country indices based on the ranking of expected return from momentum and mean-reversion. All portfolios produced created excess mean returns, that were not explained by market risk or SMB and HML factors. In their study, the key difference to the quantile regression models was the number of lags, as well as holding period used. Holding period ranged from one month to two years, while observation periods (i.e. lags used) ranged from three months to two years. Therefore while the momentum and mean-reversion may not hold at the one period level, they are robust when utilizing longer past observation periods.

Volume data is often readily available with price data, and thus can be considered part of basic historical price data. Chen et al. (2001) found evidence that detrended volume Granger caused seven of nine developed country indices. Lee and Rui (2000) investigated the Granger causality in Chinese stock indices, and found no significant relationship. However, they found some evidence for cross-index Granger causality between volume and returns. There was also evidence that volume Granger causes volatility, and this relationship is bidirectional.

2.3 Technical indicators

Technical indicators are compositions of available market data. (Shynkevich et al., 2017). With increase in difficulty of statistically significant forecasting and greater computing power, machine technical analysis in the form of using technical indicators has seen great interest in financial literature (Kumbure et al. 2022). Shynkevich et al. (2017) utlized ten different technical indicators to predict return direction. These ten indicators were

- Simple moving average (SMA)
- Exponential moving average (EMA)
- Average True Range (ATR)
- Average Directional Movement Index (ADMI)
- Commodity Channel Index (CCI)
- Price rate-of-change (ROC)
- Relative Strength Index (RSI)
- William's %R oscillator
- Stochastic %K
- Stochastic %D

Using support vector machine (SVM) and neural network approaches, they found significant overperformance when the horizon and input window time span were roughly symmetrical. However, it is important to mention they used in-sample prediction, which reduces the significance of this results. A literature review on the use of technical indicators as input variables in forecasting return direction using deep neural networks found no significant overperformance out-of-sample. The out-of-sample classification however was still significant for some of the models out-of-sample, with a maximum accuracy around 65%. (Peng et al. 2021) Han et al. (2013) investigated the performance of portfolios constructed by volatility deciles, and then either buying the decile if its equally weighted price is above the moving average, and otherwise investing in a T-bill. They found that strategies based on the ten day moving average performed especially well, creating excess returns across all volatility deciles. This pattern was reproduced by 20 and 50 day moving averages, although the significance of excess returns decreased.

Technical indicators provide statistical and machine learning models with preprocessed information to help the model in forecasting. While there is still some doubt on the efficaciousness of using technical indicators, they have seen great interest especially in financial machine learning literature (Kumbure et al., 2022) A fairly exhaustive list of technical indicators from the Ta-lib is shown in the Appendix.

2.4 Pattern matching

Another aspect also utilized in technical analysis is chart pattern matching. Price change patterns are distinctive formations of price movement, which are proposed to identify future price movements. (Investopedia, 2023) While pattern matching is much less common in financial machine learning literature, evidenced by lack of mention in literature reviews on the topic, e.g. Kumbure et al. (2022), Pahwa et al. (2017), Strader et al. (2020), worldwide Google engine searches for chart patterns have been greater than searches for fundamental analysis since the April of 2022 (Google Trends).This may indicate a significant retail investor interest on the topic.

Nayak and Braak (2007) investigated the pattern matching strategies by using the current window of price behaviour as template, and finding the closest matches by euclidian distance to that sequence from historical data. The past query results under a given threshold then provide the current sequence an expected return. They the applied a long-short strategy given prediction over 1% and under -1 %. They however failed to produce overperformance in their out-of-sample testing, getting negative performance in a highly positive market. They attribute a part of this failure to testing each sequence, instead of known patterns. Diggs and Povinelli (2003) used a genetic algorithm to find temporal patterns, and ranked them on their ability to predict stock price movement. They applied this method to predict five-day returns, and found after transaction costs weekly return in excess of one percent, indicating great predictive and economic power. Povinelli (2000) previously applied the same model to daily return, and also found great overperformance, although this was without factoring in transaction costs, which they proposed to need to under 0.02 % in order for the model to be feasible. Leight et al. (2007) tested the bull flag chart pattern, and also found statistically significant excess returns.

While pattern matching literature is more sparse compared to technical analysis in the form of technical indicators, there is promising results in the possibility to create alpha using pattern matching. A fairly exhaustive list of candlestick pattern from Ta-lib is shown in the Appendix.

2.5 Fundamental data

Fundamental analysis traditionally refers to the examination of a security's intrinsic value using both quantitative and qualitative information of the underlying asset. Quantitative information generally refers to the reported financial information, whereas qualitative information on the other hand utilizes human evaluation of such factors as competitive advantage, management, and corporate governance. (Investopedia, 2023)

There is good evidence to show both qualitative and quantitative fundamentals can be used to produce abnormal returns. In terms of qualitative fundamental analysis, Paratoukas (2012) investigated the relationship between customer-base concentration of supplier and economic output and stock returns. They found that higher customer-base concentration was causally linked to higher rates of return, and eventually that the markets have not fully appreciated this information, leading to abnormal returns over a 30-year period studied. Friede et al. (2015) completed a meta-analysis investigating the relationship between ESG (environmental, social and governance) factors and financial performance. They found that the majority of studies showed positive relationships with both corporate financial performance and returns. However, the relationship with returns was more

positive with non-portfolio return studies, with a majority (56.7 %) of studies showing a positive effect, where as with portfolio studies the majority of studies showed either neutral (36.1 %) or mixed (37.4 %) results. Mayew and Venkatachalam (2012) studied the relationship between manager affect on analyst calls, analyst recommendations, and future stock earnings. They found that both were affected by managerial affect, estimated with vocal emotion analysis. Both positive and negative affect of managers was statistically significant in predicting the next one and the next two quarter returns.

In terms of quantitative fundamental analysis, Bartram and Grinblatt (2018) investigated the performance of peer-implied mispricings derived from regression analysis of financial statement information and market prices. They termed this analysis agnostic fundamental analysis, as it could be done by a statistician without sophisticated knowledge of the underlying companies. They then constructed portfolios out of pentiles of mispricing. They found that the differential alpha after factoring in an eight factor model consisting of market risk free return, small-minus-big, momentum, short-term reversal, long-term reversal, conservative minus aggressive and robust minus weak, the most undervalued pentile continued to overperform the most overvalued pentile up to observation delay of 34 months. While the most undervalued portfolio created consistently the highest alpha, these alphas were not statistically significant. However, for all constructed portfolios and factor models utilized, the differential alpha between the most over- and most undervalued portfolios was statistically significant for all constructions and factor models. For overvalued assets, there was statistically significant negative alphas. Hanauer et al. (2022) expanded on this analysis with a machine learning approach, and found that the modelling of non-linearities allowed by machine learning approaches improved the result of the original Bartram and Grinblatt (2018) paper. They were able to find standalone statistically significant alphas after normalization with a six factor model. The difference between the over- and undervalued portfolios remained significant. Thus there is good evidence to show that financial statement based misevaluation estimates can produce outperformance in stock markets, and this overperformance is robust to risk factors.

Yan and Zheng (2017) used a data-mining approach, and constructed 18 113 fundamental signals, i.e. individual pieces of accounting information some of which transformed based on other accounting information and evaluated their effects on performance. A total of 362 signals had a four-factor alpha with a t-statistic greater than 3. The most significant signal was change in invested capital over lagged market capitalization, with an alpha value of -0.75 percent per month with a t-statistic of -5.31.

Fundamental analysis, done both quantitatively and qualitatively, has shown to be significant in producing information on future economic performance and future return behaviour in stocks. This makes sense, considering fundamental analysis aims to gain insight into the instrinsic value of the underlying company, and according to the efficient market hypothesis the market valuation should reflect the intrinsic value.

2.6 Macroeconomic data

Considering that assets are generally a part of the larger economy, using macroeconomic data to predict returns is reasonable. Rapach et al. (2005) investigated the relationship between nine different macroeconomic variables and stock market returns in twelve industrialized countries. They conducted out-of-sample testing for their model, with predictive time horizons of 1, 3, 12, and 24 months, and found statistically significant predictive

power in 10 of the 12 countries for at least one of the horizons. Of the nine macroeconomic variables, none produced out-of-sample significant predictive power across all countries. However, relative government bond yield produced statistically significant out-of-sample predictive power in 10 of the 12 countries at the 1 month predictive horizon. Broad money growth (first difference in log-levels of broadly defined money stock) produced statistically significant predictive power in 7 of 11 available countries for at least one of the horizons.

Avramov and Chordia (2006) investigated the combined use of firm and macroeconomic variables across 3123 NYSE-AMEX stock and cash. The macroeconomic variables were dividend yield, default spread, term spread, and the T-bill yield. They found that their model produced abnormal return after adjusting for Fama-French plus momentum factors. What they found was that utilizing the macroeconomic variables, that during recessionary periods, the optimal predicted portfolio lessens exposure to momentum and increases exposure to small-cap stocks. Tsai et al. (2011) used a machine learning approach also with firm and economic variables, except in the Taiwan stock market. Using a classifying ensemble, they found their model overperformed a buy and hold index strategy.

Macroeconomic based return forecasting has seen interest in fund portfolios. Avramov et al. (2011) conducted an analysis of macroeconomic factor based hedge fund return prediction. They used small cap minus large cap return, change in 10-year treasury yield, change in spread between Moody's Baa and 10-year treasury adjusted for duration, PTFs bond, currency PTFS, and commodity PTFS, where PTFS is a primitive trend following strategy. They found significant overperformance before adjusting for managerial skill. However after adjusting for managerial skill in funds, the overperformance was even more significant.

While the use of singular macroeconomic indicators is unlikely to have predictability of returns, combining multiple indicators has been seen to create statistically significant overperformance when applied trading strategies and portfolio construction. (Cakmakli and van Dijk, 2010). Macroeconomic indicators have also seen usage with combinations of other variables, like firm fundamental variables.

2.7 Sentiment data

Sentiment has a few different definitions, including "level of noise [in] traders' beliefs relative to Bayesian beliefs" (Tetlock, 2007), "common opinion amongst market participants" (Yu, 2014), and "belief about future cash flows and investment risks that is not justified by the facts at hand" (Baker and Wugler, 2007). Sentiment thus indicates held beliefs concerning future returns, that may or may not be indicated by market data. Sentiment and its usage in stock return prediction thus opposes the belief of EMH unemotional and rational market participants (Baker and Wugler, 2007). One key part in sentiment based forecasting, is that sentiment is not directly observable, and therefore different proxies have been utilized to reflect sentiment.

Schmeling (2009) used consumer confidence as a proxy for sentiment across 18 industrialized countries. While they found sentiment Granger caused returns across markets, and vice versa, interestingly the coefficient for returns was significantly negative. This held for different factor portfolios, including large stocks, small stocks and size premium. While the Granger causality applied when using the whole dataset, aggregate market re-

turns were not Granger caused in seven of the 18 countries. However for all statistically significant countries, the mean coefficient for sentiment was negative, further supporting their findings for the overall data. They considered the differences between countries to stem from differences in herding behaviour across cultures.

Huang et al. (2015) used six factors to proxy sentiment, including close-end fund discount rate, share turnover, number of IPOs, first day returns of IPOs, dividend premium, and equity share in new issues. They also found negative coefficients between their best sentiment combination method from 1 to 12 month prediction horizons. Using a 25 year out-of-sample period, they found their model created statistically significant certainty equivalent returns and Sharpe ratios. They also found a statistically significant negative relationship between their sentiment measure and future accounting information up to a 12-month prediction horizon.

Non-financial sentiment proxies have also been utilized. Joseph et al. (2011) used Google search intensity of S&P 500 stocks as a proxy for sentiment. With a weekly investment horizon, they created portfolios of pentiles of search intensity, and found statistically significant four-factor alpha in the top four pentiles, with the highest pentile containing producing the most overperformance. They also created deciles of volatility sorted portfolios, and found that for the lower half the coefficient for sentiment was negatively statistically significantly, while in top half the sentiment coefficient was positively statistically significant. While the sentiment coefficient for the next week was positive, the sentiment coefficient was negative when predicting returns over 5 to 8 week horizons, exclusive of the first four. This indicates that there are overreaction and correction dynamics at play relating to sentiment. Ren et al. (2018) investigated the forecasting of SSE 50 index using basic historical price data and sentiment analysis of user generated content on forums related to the discussion of the stock market. They predicted market direction, and found that adding sentiment data improved the accuracy of forecasting by 18.6 %, which translated into improved performance of the trading model, which overperformed the market.

Analyst recommendations can be thought of as a special case sentiment, as it represents the sentiment held by specific finance professionals. Bordalo et al. (2019) found that there was a negative relationship between analyst recommendations and one month portfolio returns, constructed with deciles of analyst recommendations. They also found that analyst recommendation deciles were positively related to market beta.

The relationship between future returns and sentiment is highly dependent on the proxy being used. Traditional market and analyst based proxies for sentiment analyses appear to be negatively relational to future returns, where as proxies of general (retail) investor sentiment appear to be positively relational to future returns.

3 Copulas

A copula is a multivariate distribution function, that has standard uniform marginal distributions. Much of the literature of copulas has been limited to the inspection of the bivariate case, as modelling multivariate copulas has proven to be a difficult task. With this knowledge of the behaviour of bivariate copulas, an approach to creating multivariate copulas via a approach of combining bivariate copulas called pair-copula construction (PCC). (Aas and Berg 2010) The following presentation of PCC follows Aas et al. (2009).

3.1 Pair-copula decomposition of multivariate distributions

Let there be a vector $\mathbf{X} = (X_1, ..., X_n)$ of n random variables, which have the joint density function $f(x_1, ..., x_n)$. This density can be factorised in the following way:

$$f(x_1, ..., x_n) = f_n(x_n) \cdot f(x_{n-1} \mid x_n) \\ \cdot f(x_{n-2} \mid x_{n-1}, x_n) \cdots f(x_1 \mid x_2, ..., x_n)\},$$

which is a unique composition for every re-labelling or the ordering of variables. According to Sklar's theorem (Sklar 1959), for every multivariate distribution F with marginals $F_1(x_1), ..., F_n(x_n)$ there exists a copula function C such that

$$C\{F_1(x_1), ..., F_n(x_n)\} = F(x_1, ..., x_n),$$

which can be written with respect to the marginals u as

$$C(u_1, ..., u_n) = F\{F_1^{-1}(u_1), ..., F_n^{-1}(u_n)\},$$

Given that F is absolutely continuous and has strictly increasing and continuous marginal densities $F_1, ..., F_n$, applying the chain rule $f(x_1, ..., x_n$ can be decomposed as

$$f(x_1, ..., x_n) = c_{1...n}\{F_1(x_1), ..., F_n(x_n)\} \cdot f_1(x_1) \cdots f_n(x_n),$$

When $n = 3$, this can be decomposed as, for example

$$f(x_1 \mid x_2, x_3)) = c_{12|3}\{F(x_1 \mid x_3), F(x_2 \mid x_3)\} \cdot f(x_1 \mid x_3) \\ c_{13|2}\{F(x_1 \mid x_2), F(x_3 \mid x_2)\} \cdot f(x_1 \mid x_2) \\ c_{13|2}\{F(x_1 \mid x_2), F(x_3 \mid x_2)\} \cdot c_{12}\{F(x_1), F(x_2)\} \cdot f_1(x_1)$$

From this it is clear that a conditional marginal density can be decomposed with the general formula

$$f(x \mid \upsilon) = c_{x\upsilon_j|\upsilon_{-j}}\{F(x \mid \upsilon_{-j}), F(\upsilon_j \mid \upsilon_{-j}\} \cdot f(x \mid \upsilon_{-j}),$$

where j is an arbitrarily chosen conditioning component from υ, and $-j$ indicates the vector excluding it.

From this it can concluded, that under appropriate regularity conditions of the F and f functions, a multivariate density can be expressed via pair-copulae. Joe (1996) showed that the marginal conditional distribution for every j is

$$F(x \mid \upsilon) = \frac{\partial C_{x,\upsilon_j|\upsilon_{-\mathbf{j}}}\{F(x \mid \upsilon_{-\mathbf{j}}), F(\upsilon \mid \upsilon_{-\mathbf{j}})\}}{\partial F(\upsilon \mid \upsilon_{-\mathbf{j}})},$$

3.2 Vines

Considering that the number of possible pair-copulae constructions increases factorially with dimensions, a method of graphically organizing them, has been introduced. In literature, three types of vines have gotten the most attention, D-vines and canonical (C-) vines and regular (R-) vines.

D-vine structure is illustrated in Figure 8. Following the example in the figure, a five-dimensional D-vine. A D-vine consists of $n-1$ trees, denoted by T_j, $j = 1,...,4$. Tree T_j has $n+1-j$ nodes and $n-j$ edges. Each edge corresponds to a pair-copula density, e.g. edge $13 \mid 2$ corresponds to $c_{13|2}(\cdot)$. The density of D-vine can be written as

$$f(x_1,...x_n) = \prod_{k=1}^{n} f(x_k) \prod_{j=1}^{n-1} \prod_{i=1}^{n-j} c\{F(x_i \mid x_{i+1},...,x_{i+j-1}), F(x_{i+j} \mid x_{i+1},...,x_{i+j-1})\},$$

where j is identifies the tree index, and i the edge index.

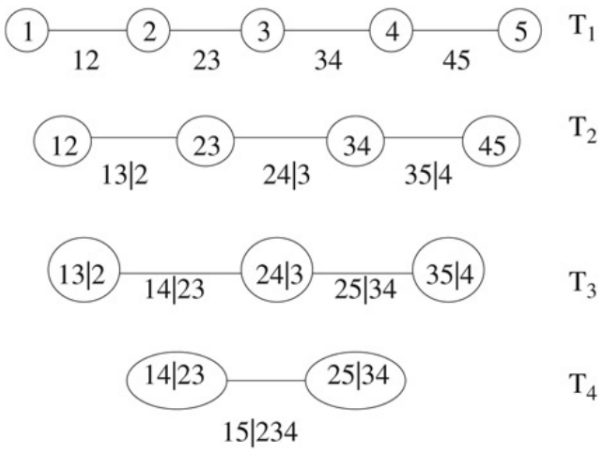

Figure 1: D-vine structure (Aas et al. 2009)

Inference of a D-vine copula in the empirical case, such as in finance, where the expected values and variances are autocorrelated, in order to model the D-vine structure, instead of modelling the the time series copulas, it makes sense to model the standardized residuals. Standardized residual are calculated by first retrieving conditional expected values and variances, normalizing residuals based on these. Assuming the modelling is successful, there should be no autocorrelations present in the error data, and their relationships can be modelled with the D-vine. The second step is to transform the error data to an (approximately) uniform distribution, with the empirical distribution function

$$U(x) = \frac{1}{n} \sum_{j=1}^{n} I(x_j < x),$$

where $I(\cdot)$ is an indicator function.

After this, the construction of the D-vine has to be determined. Considering that the number of possible vines increases factorially, this has to be done with heuristics for even moderate dimensions. A common approach is to use absolute Kendall's tau to find the most dependent structure, introduced by Dissmann et al. (2013). Considering that a D-vine is defined by its first tree, this simplifies the selection process by quite a lot. However with a given structure, the selection of the copula families has to be made. The selection of these copula families can be done from a set of preselected copula families, such as normal or T, or in a non-parametric way, showcased in Nagler et al. (2017). The inference of a D-vine is done with Algorithm 1, in which $\Theta_{j,i}$ are the parameters of copula density

Algorithm 1 Likelihood Evaluation for a D-vine

log-likelihood $= 0$
for $i = 1$ to n **do**
  $v_{0,i} = x_i$
end for
for $i = 1$ to $n-1$ **do**
  log-likelihood $=$ log-likelihood $+ L(v_{0,i}, v_{0,i+1}, \Theta_{1,i})$
end for
$v_{1,1} = h(v_{0,1}, v_{0,2}, \Theta_{1,1})$
for $k = 1$ to $n-3$ **do**
  $v_{1,2k} = h(v_{0,k+2}, v_{0,k+1}, \Theta_{1,k+1})$
  $v_{1,2k+1} = h(v_{0,k+1}, v_{0,k+2}, \Theta_{1,k+1})$
end for
$v_{1,2n-4} = h(v_{0,n}, v_{0,n-1}, \Theta_{1,n-1})$
for $j = 2$ to $n-1$ **do**
  for $i = 1$ to $n-j$ **do**
    log-likelihood $=$ log-likelihood $+ L(v_{j-1,2i-1}, v_{j-1,2i}, \Theta_{j,i})$
  end for
  if $j == n-1$ **then**
    Stop
  end if
  $v_{j,1} = h(v_{j-1,1}, v_{j-1,2}, \Theta_{j,1})$
  if $n > 4$ **then**
    for $i = 1, 2, \ldots, n-j-2$ **do**
      $v_{j,2i} = h(v_{j-1,2i+2}, v_{j-1,2i+1}, \Theta_{j,i+1})$
      $v_{j,2i+1} = h(v_{j-1,2i+1}, v_{j-1,2i+2}, \Theta_{j,i+1})$
    end for
  end if
  $v_{j,2n-2j-2} = h(v_{j-1,2n-2j}, v_{j-1,2n-2j-1}, \Theta_{j,n-j})$
end for

Algorithm 2 Simulating from D-vine

Generate one sample $X_1, \ldots, X_n$ from the vine.
Sample $W_1, \ldots, W_n$ independently from a uniform distribution on $[0,1]$.
$X_1 = V_{1,1} = W_1$
$X_2 = V_{2,1} = H^{-1}(W_2, V_{1,1}, \Theta_{1,1})$
$V_{2,2} = H(V_{1,1}, V_{2,1}, \Theta_{1,1})$
for $I = 3$ to N **do**
  $V_{I,1} = W_I$
  for $K = I-1, I-2, \ldots, 2$ **do**
    $V_{I,1} = H^{-1}(V_{I,1}, V_{I-1,2K-2}, \Theta_{K,I-K})$
  end for
  $V_{I,1} = H^{-1}(V_{I,1}, V_{I-1,1}, \Theta_{1,I-1})$
  $X_I = V_{I,1}$
  if $I == N$ **then**
    Stop
  end if
  $V_{I,2} = H(V_{I-1,1}, V_{I,1}, \Theta_{1,I-1})$
  $V_{I,3} = H(V_{I,1}, V_{I-1,1}, \Theta_{1,I-1})$
  if $I > 3$ **then**
    for $J = 2, 3, \ldots, I-2$ **do**
      $V_{I,2J} = H(V_{I-1,2J-2}, V_{I,2J-1}, \Theta_{J,I-J})$
      $V_{I,2J+1} = H(V_{I,2J-1}, V_{I-1,2J-2}, \Theta_{J,I-J})$
    end for
  end if
  $V_{I,2I-2} = H(V_{I-1,2I-4}, V_{I,2I-3}, \Theta_{I-1,1})$
end for

$c_{i,1+j|i+1,\ldots,i+j-1}(\cdot,\cdot)$, and $h(\cdot)$ is the given h-function of the distribution. Simulating from the given D-vine can be seen in Algorithm 2.

Another possible vine construction is the canonical (C-) vine, which be used when a particular variable is known to be key in governing interactions between other variables. In Figure 2 this is done with variable 1 as the "root" of the vine.

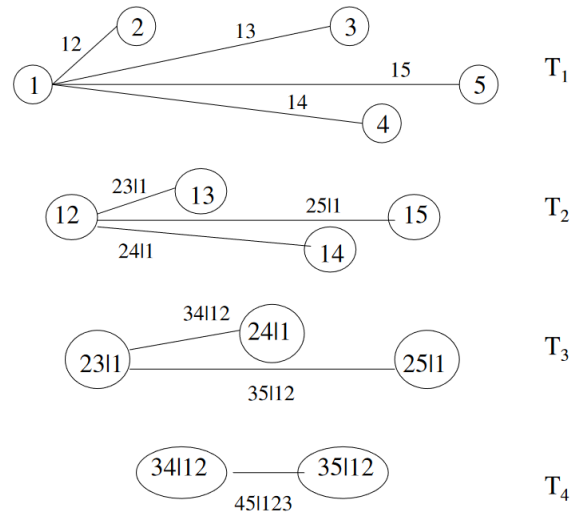

Figure 2: C-vine structure (Aas et al. 2009)

As can be seen, compared to the D-vine, in the C-vine variable 1 is used in each pair-copula construction within the vine, and as such its importance is highlighted.

Lastly, a regular (R-) vine decomposition, that is not D- nor C-vine, can be used when one wants flexibility in the vine decompositions. An example of a R-vine that is neither a D- nor C- vine is illustrated in Figure 3.

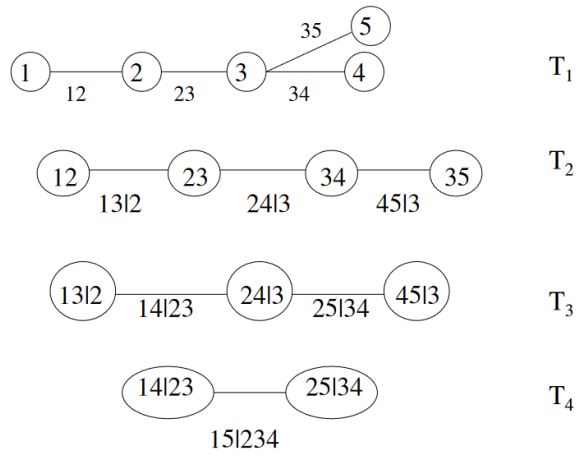

Figure 3: R-vine structure (Aas et al. 2009)

The example is neither a D-vine as there is a node that is connected to more than two edges, or a C-vine, as node 3 in T_1 is connected to 3 edges instead of 4. The flexibility of D-vine decomposition is illustrated by the fact that in a five-dimensional case, both

D- and C-vines have 60 possible decompositions, where as with a R-vine, there are 240 decompositions, including both the D- and C-vine decompositions.

The fitting of the best C- and R-vine largerly follows the same pattern as with a D-vine, with the absolute Kendall's tau heuristic proposed by Dissman et al. (2013), however with the given limitations of C- or R-vine.

3.3 Parametric and non-parametric copulas

When modelling pair-copulas the most well researched option is to use parametric pair-copulas to model dependence between variables. Below some of the most researched copulas are described (see Aas et al., 2009)). Bivariate Gaussian copula is used when the dependencies are symmetrical across the distribution. Bivariate Student's t-copula is used when there is symmetrically concentrated increased dependence in the tail-ends of the distribution. Bivariate Clayton copula on the other hand is used when there is asymmetric dependence with higher dependence in the lower tail, where as the Gumbel copula is the opposite, with high dependence in the positive tail.

With non-parametric copulas, no assumptions are made in regards to the dependence structure of the copulas. Instead, density estimators are made based on the available data, allowing for greater flexibility in modelling. Non-parametric estimators for pair-copulas include the empirical Bernstein copula, penalized Bernstein polynomials and B-splines, and kernel weighted local likelihood. (Nagler et al., 2017)

4 Machine learning

4.1 Machine learning models

4.1.1 Multi Layer Perceptron

A Multi Layer Perceptron (MLP) is a basic multilayer neural network architecture. An MLP learns to map from a fixed-size input to a fixed size output. When moving from one layer to another, a set of units calculates a weighted sum from the units of the previous layer. These units themselves include an activation function, which determines the units output. The current dominating activation function in MLPs is the rectified linear unit function (ReLU), which has the function form $f(z) = max(z, 0)$, where z is the input from the previous layer. The MLP has the basic form of an input layer, which is made of input units, an output layer, which is made of output units, and the possible layers between them, called hidden layers, are made up of hidden units. (LeCun et al. 2015) This basic architecture with two hidden layers can be seen illustrated in Figure 4.

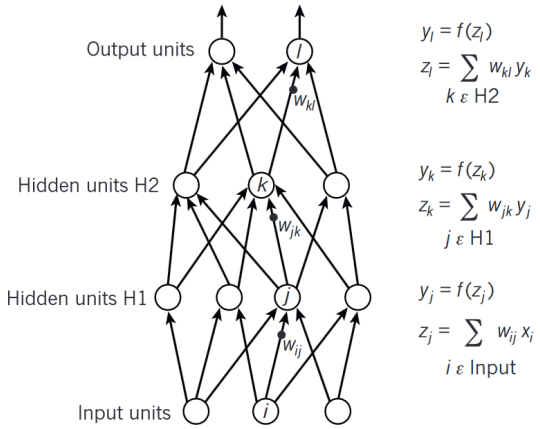

Figure 4: DMLP architecture (LeCun et al. 2015)

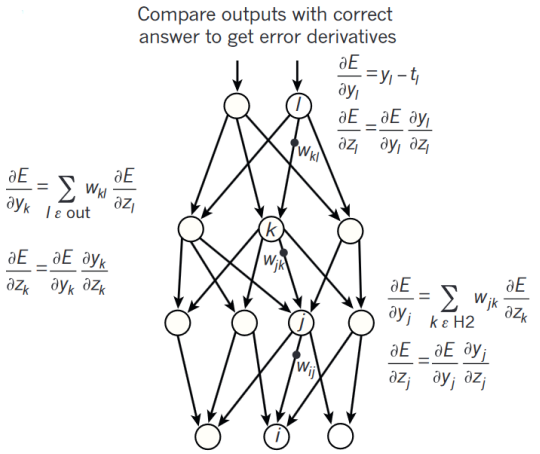

Figure 5: MLP backpropagation (LeCun et al. 2015)

An MLP can be trained via stochastic gradient descent. This can be done by back-propagating through the network from the received output errors, and partially derivating the errors first with respect to the outputs of the units, and from this deriving the error partial derivative with respect to weights of the network. With known partial derivatives with respect to weights, a gradient descent, meaning absolute minimization of the partial derivatives can be conducted. This process is stochastic, as the derivatives are computed at random instances of the weight updating process. (LeCun et al. 2015) This process is illustrated in Figure 5.

4.1.2 Convolutional Neural Network

Convolutional neural networks (ConvNets or CNNs) are a subtype of Feed Forward Neural Networks (FFNNs), but instead of all adjacent layer units having full connectivity, as seen with MLPs, the architecture includes filters, which allocate units to local patches of the feature map. A ConvNet architecture is typically structured in a series of stages. The first two stages are called convolutional layers and pooling layers. A convolutional layer involves the organization of units of a new layer into feature maps from the previous layers local patches via a set of weights called the filters. The local weighted sum is then

processed through a non-linear activation function, such as ReLU. After these separate feature maps are created from the previous layer, a pooling layer is applied. A pooling layer merges similar feature maps. A typical pooling unit takes the maximum of a local patch of units in a feature map(s) from the preceding convolutional layer. Neighboring pooling units also take input from shifted local patches. This configuration of convolution, non-linearity, and pooling is then often stacked to create a deep ConvNet. At the end of a ConvNet is usually a fully-connected layer to produce outputs. Optimization of a ConvNet is done the same way as in MLPs through backpropagation. (LeCun et al. 2015) ConvNet architecture is illustrated in Figure 6.

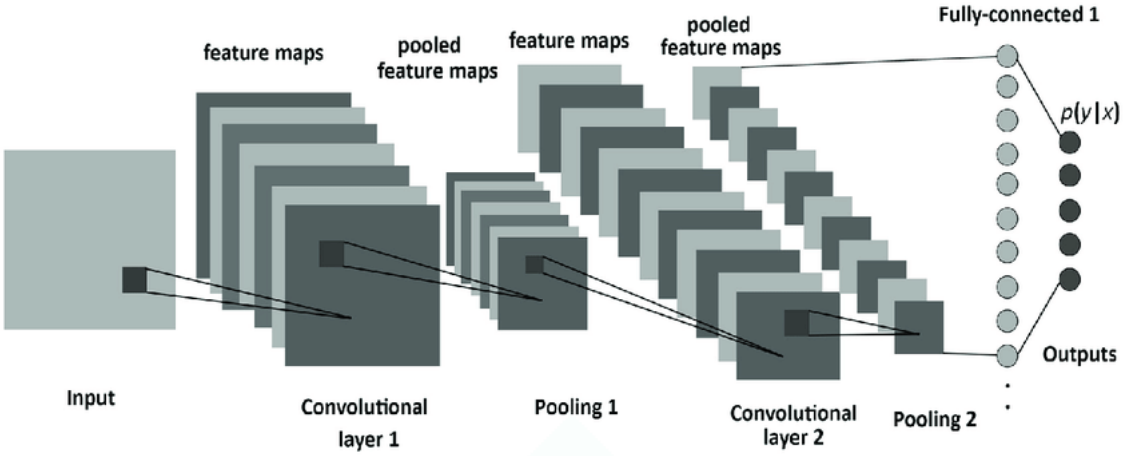

Figure 6: 2D ConvNet architecture (Albelwi and Mahmood 2017)

4.1.3 Recurrent Neural Network

Recurrent neural networks (RNNs) process an input statement one element at a time, while storing in their hidden units a state vector that contains information about past states of the sequence. RNNs are applicable when the input and output vectors are sequential in nature. RNNs can also be trained with the backpropagation method. A popular RNN architecture that creates an explicit memory process for the state vector is called the long short-term memory (LSTM) network. The LSTM model uses memory filters to train the model on how it should memorize past information. (LeCun et al. 2015) The structure of the LSTM unit can be seen in Figure 7.

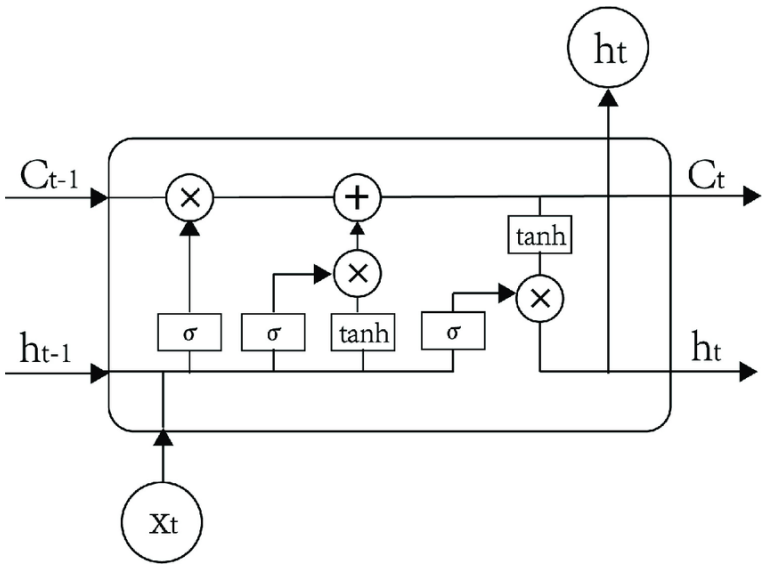

Figure 7: LSTM architecture (Qiu et al. 2020)

Following Qiu et al. (2020), the LSTM structure can be written out in the following way, starting with the first step

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f),$$

where f_t is the first step of process, σ is a logistic sigmoid function, W_f is the weight vector for f_t, h_{t-1} is the output from the previous time step, x_t is the new information at time t and $[h_{t-1}, x_t]$ a concatenated long vector combining them, and b_f is a bias vector. The next step i_t is calculated by

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i),$$

after which a candidate vector $\hat{C}_t$, where C_t is the reserved information vector to be passed on to the next moment, is calculated by

$$\hat{C}_t = \tanh(W_c * [h_{t-1}, x_t] + b_c),$$

where tanh is the hyperbolic tangent function. After this, the actual C_t is computed by

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t.$$

Now with C_t defined, the output h_t can be computed as:

$$h_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) * \tanh(C_t)$$

A Bidirectional LSTM (BiLSTM) is essentially a dual LSTM architecture, that does a forwards pass through the data, just like a regular LSTM, however an additional and seperate backwards pass through the data is made. Bidirectional RNNs were originally proposed by Schuster and Paliwal (1997). They proposed that the bidirectional model is better able to minimize objective functions by taking advantage of immeadiate information. The BiLSTM architecture is shown in Figure 8.

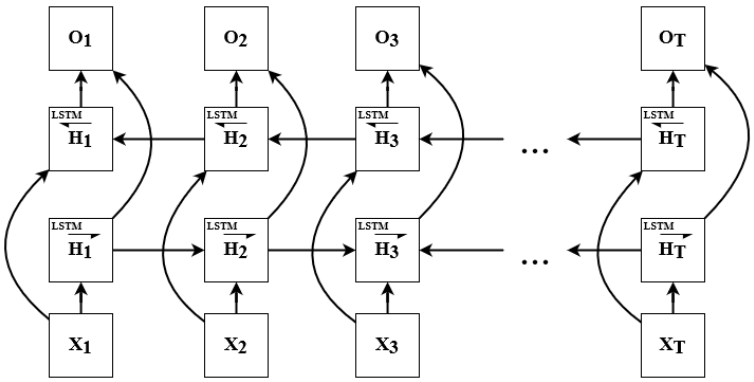

Figure 8: BiLSTM architecture (Adytia et al., 2022)

4.1.4 Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs) are an artificial neural network model used to learn probability distributions of the input space. An RBM is a a bipartite and undirected graph, with two layers, a visible and a hidden one. The units of layer are not connected to one another. The probability distribution of an RBM with binarily valued visible and hidden units can be defined using the energy function

$$E(v,h) = -a^T \upsilon - b^T h - \upsilon^T W h,$$

where υ and h denote the vectors of the visible and hidden units, respectively, W denotes the matrix of weights between the layers and a and b denote the biases of the visible and hidden units respectively. Using the energy function, the partition function Z and the probability semantics of the RBM can be calculated:

$$Z = \sum_{\upsilon} \sum_{h} exp(-E(\upsilon,h)),$$

$$P(\upsilon,h) = \frac{1}{Z} exp(-E(\upsilon,h)).$$

The RBM is trained by the contrastive divergence algorithm, which gives the stochastic approximation of the distance between the reconstruvted and actual probabbility distributions. (Sezer et al. 2020) An RBM architecture can be seen visualized in Figure 9.

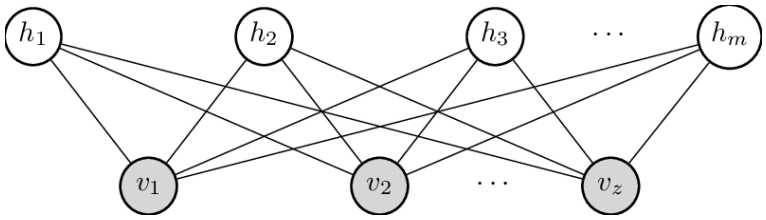

Figure 9: RBM architecture (Qiu et al. 2014)

4.1.5 Deep Belief Network

Deep Belief Networks (DBNs) are a stacked RBM architecture. DBNs training is often conducted in two phases, first the unsupervised phase where the probability distribution is approximated using the RBMs, and then a supervised learning phase, where the network is trained in a classification problem. The probability of generating a given visible vector is:

$$p(\upsilon) = \sum_{h} p(h \mid W) p(\upsilon \mid h, W).$$

The optimization is done first via the contrastive divergence algorithm of the RMBs and then via backpropagation. (Sezer et al. 2020) The architecture is illustrated in Figure 10.

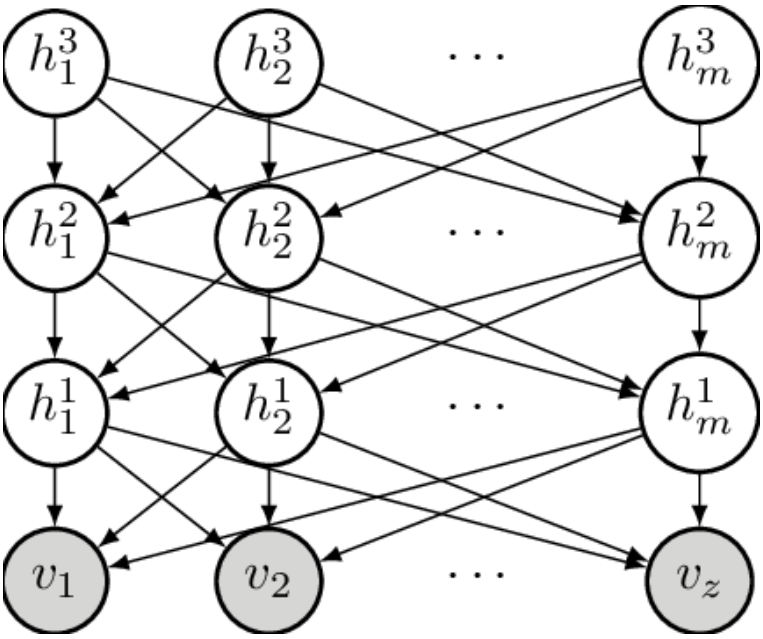

Figure 10: RBM architecture (Sezer et al. 2020)

4.1.6 Autoencoders

Autoencoder (AE) networks are an unsupervised learning model, that is used to reduce dimensionality and exract features of the original input dataset. AEs contain an input layer, output layer, and a one or more hidden layers. Between the input and a hidden layer there is an encoder encoder,which is reductive in dimensions, and between the hidden layer and an output layer there is a decoder, which return the encoded matrix into its original dimension. A simple AE can be defined by the following process:

$$h = f(x) = \sigma_1(W_1 x + b_1),$$

$$r = g(h) = \sigma_2(W_2 x h + b_1),$$

where $f(x)$ is the encoding function, $g(h)$ is the decoding function, W_i is the weight matrix, x is the input matrix, and b_i are bias vectors. The loss function

$$L(x, r) = \|x - r\|^2$$

is minimized in order to retain the most information. (Sezer et al. 2020) AE structure can be seen illustrated in Figure 11.

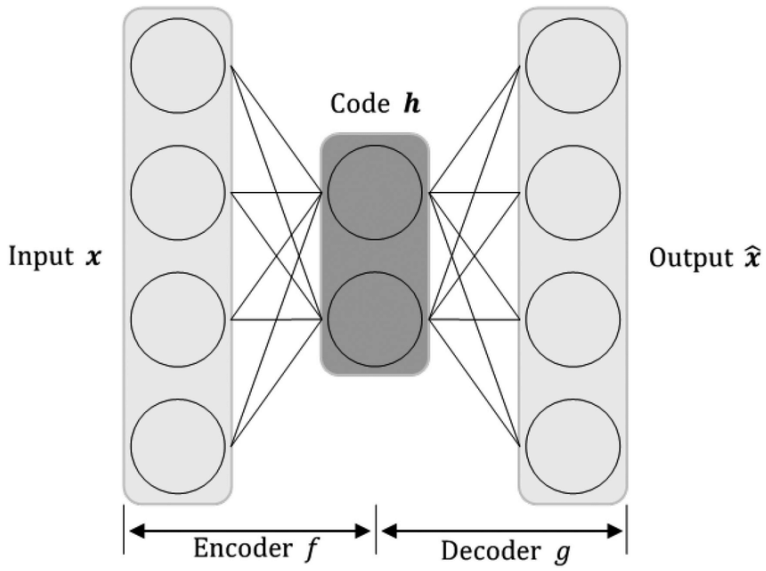

Figure 11: AE architecture (Liu et al. 2019)

(de Santana Correia and Colombini, 2022)

4.1.7 Deep reinforcement learning

Reinforcement learning (RL) is a machine learning method, that aims to train an agent to learn the action that gives the maximum reward given some environment. RL is based on the Markov Decision Process (MDP), where the process is made up of five tuples: state, action, reward, state probability transition matrix, and a discount factor. The agent's expected outcomes can be described by two function, the state-value function $v_\pi(s)$, which is the expected return given a state s and following policy π, and the action-value function $q_\pi(s,a)$, which is the expected value of taking action a:

$$v_\pi(s) = E_\pi[G_t \mid S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s],$$

$$q_\pi(s,a) = E_\pi[G_t \mid S_t = s, A_t = a],$$

where G_t is the discounted reward function, R_t is the reward at time t and γ is the discount factor of future rewards.

A popular method of RL is Q-learning, where Q refers to the action value function. Q-learning is a model-free and value-based RL method, where instead of a predetermined policy, it learns from applying different actions. The Q-learning model is SARSA model (state S_t, action A_t, reward R_t, (future) state S_{t+1}, and (future) action A_{t+1}). The updating of a SARSA action value is done with the function

$$Q(S_t,A_t) = Q(S_t,A_t) + \alpha\left[R(t+1) + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)\right],$$

where α is the learning rate. In the Q-learning method, the model is updated with

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R(t+1) + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t) \right],$$

Where a denotes action and the best action is denoted by max_a. (Sezer et al. 2020)

4.1.8 Attention

Attention is a key part of human cognition. The concept of attention has also been applied to neural networks, where attention modules are able to guide the neural network to examine data in greater detail. (Zagoruyko and Komodakis, 2016) Attention mechanisms can be categorized into three categories: Global attention, local attention, and self-attention.

Global attention, or soft attention, assigns inputs a weight between 0 and 1. This done with a softmax function

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

which means global attention can easily be applied to models trained by stochastic differentation. Global attention can be used for both temporal and spatial data. (de Santana Correia and Colombini, 2022)

Local attention, or hard attention, assigns inputs a weight of either 0 or 1, and thus determines wheter a input is either used or not. Local attention thus is effectively a conditional indicator function, which is non-differentiable. Thus when applying local attention to a model, the model has to be trained through reinforcement learning. Like global attention, local attention is can be applied to both temporal and spatial data. (de Santana Correia and Colombini, 2022)

Self-attention uses simple matrix calculations to quantify interdependencies between input elements. Some of the main advantages of self-attention are computational cost and availability of parallel computing, as self-attention uses relatively simple matrix operators. (de Santana Correia and Colombini, 2022) The matrix operators used include, but are not limited to dot product, Hadamard product, star product, and concatenation. It is also possible to add an additional activation layer(s) to the activation function. (Zhao et al. 2020)

4.1.9 Deep learning models in finance

Deep machine learning models have gotten great attention in financial literature. The areas of application include stock market forecasting and algorithmic trading, credit risk assessment of individuals, companies and banks, fraud detection, portfolio allocation, derivative pricing, and sentiment analysis, although this list is not exhaustive. (Ozbayoglu et al. 2020) Considering the scope of thesis, the focus here will be on the stock forecasting and portfolio allocation applications.

Stock price forecasting is the most frequent application of deep learning in financial literature. In literature stock forecasting has been studied at different time-scales from high frequencies to month ahead predictions. Use of different deep learning methodologies has also had great scope, as MLP, CNN, RNN, AE, and DBNs have all seen application in forecasting. There is however a clear trend of RNN being the most popular methodology, with the LSTM architecture dominating as the chosen specific architecture.

It is also noteworthy however, that hybrid models, where more than one deep learning technique is used, have also seen some prominence, with examples such as hybrid CNN-LSTM (e.g. Liu et al. 2017), and DBN-MLP (e.g. Batres-Estrada 2015) models. Input data used has also varied, as raw price data, technical indicators, (macro)economic data, news, and investor sentiment have all seen usage in literature. In general, deep learning methods outperformed traditional statistical models and other benchmarks, although in many cases the difference in performance has been seen to either negligible, or even worse than in traditional machine learning models. (Sezer et al. 2020)

Recent portfolio management literature has increasingly focused on deep reinforcement learning (RL) as the preferred method of machine learning. This can be seen from comparing a fairly recent 2020 literature review by Sezer et al. to a 2022 literature review by Olorunnimbe and Viktor (OV). While there are differences in the research paper filtration process, as OV include the inclusion of the term "back-testing" in their criteria, they found three out of four papers used RL, while Sezer et al. indicate only one of 24 portfolio management research papers included the use of RL. Comparing the two reviews, Sezer et al. reference such papers as Takeuchi et al. (2013), Grace (2017), Zhou (2019), where the portfolio weights are determined by the price change prediction, meaning these are clearly more heuristic approaches to the portfolio selection problem, rather than finding optimums. (It is noteworthy that research papers, which also analyze the dependence structure are present in the literature review.) This is contrasted by the RL research papers highlighted by OV, Park et al. (2020), Liang et al. (2018), and Guo et al. (2018), which directly try to learn the best portfolio weight change actions via their RL methodologies. Overall however, both literature reviews, Sezer et al. (2020) and Olorunnimbe and Viktor (2022) find that deep learning approaches to portfolio selection provide overperformance compared to benchmarks.

4.2 Machine learning-Copula models in portfolio management

Searching with the term "(neural AND network) OR (deep AND learn*) AND copula AND portfolio" in Scopus (scopus.com) nets four (from nine total) relevant search results. Here relevancy was determined by the presence of a hybrid model, where returns were predicted with a machine learning model and portfolio optimization was done utilizing said predictions and copula-based dependencies.

The first research paper combining deep learning with copulas in portfolio selection was Zhao et al. (2018). Their dataset included three American ETFs. They predicted return by either a special kind of FFNN called Psi-Sigma network introduced by Shin and Ghosh (1991) or a RNN, and a multivariate skew T-distribution with correlation structure predicted by either dynamic conditional correlation (DCC), asymmetric conditional dynamic correlation, or the generalized autoregrssive model (GAS). They conducted portfolio optimization either through MVO or with mean-CVaR optimization. They compared their prediction methods, with ARMA-based return predictions and found that the NN approaches beat the ARMA-approaches in all return and risk measures, except maximum drawdown, with two to four fold realized returns.

Yu et al. (2019) created an MLP with economic indicators to predict the returns of six Vanguard indices. They then employed a GARCH-copula model to get normalized errors, and using a simulations determined the best mean-CVaR portfolio. They compared their results, with an equally weighted portfolio, a historical return-based mean-CVaR opti-

mization and the GARCH-copula model without the MLP return predictions. They found that the MLP-GARCH-copula hybrid model performed the best in their dataset lasting from 2007 to 2018. This was true for all used measures, annualized returns, volatility, Sharpe ratio, maximum drawdown and 99% VaR. Yu and Chang (2020) was essentially an identical paper, except the testing time period was from 2002 to 2019. The results were similar, except that the historically-based and the GARCH-copula model itself had lower volatility than the combined MLP-GARCH-copula. The MLP-GARCH-copula model was still clearly overall the best performing model.

Li and Muwafak (2021) used a GARCH-copula nested in a hidden Markov model (HMM) to study risk dependencies in four financial industries. Their HMM had two states, a high and low dependency state. They then studied the implied tail dependence structures of the model, and using these created simulation studies of VaR for equally weighted portfolios. They also then separately created stock price predictions based on a RNN model constructed from their HMM. They conclude that the HMM-copula approach is effective in modelling risk contagion and stock predictions.

Xu and Cao (2023) used a hybrid LSTM-based variational AE model and modelled the dependence structure of latent variables with a weighted partial regular vine copula. They used eight indices and compared the performance with an ARMA-GARCH, a Gaussian process volatility model, CNN, LSTM, and a variational LSTM models, and found that their WPVC-VLSTM model outperformed the controls in annualized return, precision, recall, and accuracy.

Overall, the combination of deep learning and copulas, despite variations in specific methodology has produced better approximations of risk and better return and risk behaviour when applied to the portfolio selection process.

5 Empirical analysis

5.1 Data

The stock index investigated is the S&P 500. The start of the training period is 2016 and the end of testing is the end of 2023. This timespan is proposed to be reflective of a stock market, where machine learning is already present. While the study of machine learning methods began in the 1950s (Fradkov, 2020), a literature review on machine learning in the stock market conducted by Kumvure et al. (2022) found that more than 50% of relevant articles between 2000 and 2019 were published between 2015 and 2019. Based on this, and the principle of conservatism, the year 2016 reflects a starting point which is still far back enough that there is a significant amount of data, but not so far back that machine learning did not affect the markets, making the data less representative for training purposes. The training period is set to be from 2016 to 2021. The validation period is the year 2022 and testing is conducted with the year 2023. This represents a 75-12.5-12.5 split.

The data includes all stocks which are present at the beginning of 2015 and the end of 2023. The year 2015 is used, as at the beginning of 2016, lagged variables are based on data from 2015. This elimination has the benefit of all stocks being available throughout the dataset, which makes computations easier, however with the drawback of introducing bias. To mitigate the effect of this bias, index benchmarking is conducted with the set of selected stocks, as opposed to the actual index. The survivorship bias is also reduced by the fact, that the elimination of stocks could be done as an ad hoc rule that states that methodology only applies to stocks which have been present in the index for six years, although this does not factor in survivorship during the last year, the test period. The total number of stocks included based on this elimination is 329. The list of stocks was retrieved from https://github.com/fja05680/sp500. The full list of stocks included is reported in the appendix.

Price and volume is represented by five variables, close, open, high, low, and volume. These variables are transformed as percentage change, calculated by $(X_t - X_{t-1})/X_{t-1}$. Price and volume data is retrieved from Yahoo Finance (2024) using the yfinance (Aroussi, 2024) python library.

Technical indicators are used to potentially aid the model in extracting information from the price and volume history, The indicators considered are the following:

- Smooth Moving Average (SMA)
- Exponential Moving Average (EMA)
- Relative Strength Index (RSI)
- Normalized Average True Range (NATR)
- Average Directional Movement Index (ADX)
- Commodity Channel Index (CCI)
- Chande Momentum Oscillator (CMO)
- William's %R oscillator (%R)
- Aroon Oscillator (AO)
- Weighted Moving Average (WMA)
- Money Flow Index (MFI)
- Chaikin A/D Oscillator (CAD)
- Resistance level (RL)
- Support level (SL)
- Bollinger Bands (BB)
- Volatility (V)

All of the above technical indicators have the number of lookback periods as a hyperparameter. For each variable, the number of look back periods for which the indicator is computed are 5, 7, 14, 21, 50, 100, 252. The SMA, EMA, WMA, RL, SL, and BB variables are non-stationary, and a price relational transformation is calculated by $(X_t - C_t)/C_t$, where X_t is the technical indicator and C_t is the closing price. Volatility is normalized as X_t/C_t. Chailkin A/D Oscillator is Z-score normalized with the training data stock-wise.

Pattern matching is conducted with the Ta-lib (2024) Python library. All available candlestick patterns are utilized. The patterns used are listed in the appendix. Ta-lib checks if a pattern in the data matches that of a given pattern and then outputs 100 if the pattern matched indicates a bullish signal and -100 if it indicates a bearish signal. 0 indicates no match at all. Considering the number of patterns, and the relative rarity of signals, the outputs are transformed to summarizing variables. These transformations are percentage of indicators bullish and percentage of indicators bearish.

Fundamental data used comes from the income statement, balance sheet and cash flow statement. This is done to give a full view of a company's quantitative fundamentals. Fundamental variables used are listed below:

- Market cap
- Revenue
- Gross profit
- Net income
- Research and development costs
- Interest expense
- Current assets
- Non-current assets
- Current liabilities
- Long-term debt
- Operating cash flow
- Investing cash flow
- Financing cash flow

Fundamental data was retrieved from Polygon.io (2024).

Macroeconomic data used is listed below:

- Index return
- Change in CBOE crude oil volatility index
- Change in CBOE gold volatility index
- CBOE Volatility Index
- Change in CBOE Interest Rate 10 Year T No
- Change in CBOE 13 Week Treasury Bill Yield Index
- Change in unemployment
- Change in consumer price index
- Change in GDP

- Change in industrial production

The macroeconomic data is based on the United States, as the S&P 500 is as well. Unemployment, consumer price index, and industrial production are lagged two months from observation, and GDP two quarters. Index return refers to the equally weighted average return across the 329 stocks. Macroeconomic data was retrieved from FRED (2024), and the index prices were retrieved from Yahoo Finance (2024) using the yfinance (Aroussi, 2024) python library.

Sentiment:

- Google Trends

Market sentiment:

- American Association of Individual Investors Bearish index
- American Association of Individual Investors Bullish Index
- Percentage index advances daily
- Percentage index declines daily
- Percentage index new highs past five days over one year
- Percentage index new lows past five days over one year

Google Trends data is based on the company data, as opposed to search term or topic and lagged one week. The American Association of Individual Investors (AAII) conducts weekly surveys amongst its members. The survey question asks participants how they view the stock market to move in the next six months. There are three possible answers, bullish, neutral or bearish. Only the bearish and bullish indexes, referring to the percentage of answers being bearish or bullish respectively, are included, as the neutral index can be constructed from the two other indices. AAII data is lagged one day from day of reporting. Index movements are calculated with the price data for the 329 stocks.

A dummy variable indicating the sector the stock belongs to is used. Sector data is retrieved from Yahoo Finance (2024) using the yfinance (Aroussi, 2024) Python library, and the sector for each stock is assumed to be the current one. This means if a stock's sector has changed, this is not reflected in the data.

5.2 Model

The model construction is done in four phases. In the first phase agnostic fundamental analysis is conducted with reference to Bartram and Grinblatt's (2018), yielding implied returns from over/under-valuation. In the second phase, an evolutionary algorithm is used to determine the best set of technical indicators. In the third phase regression neural networks are trained to predict returns. In the fourth phase, copulas are fitted on GARCH-standardized return errors. Given simulated returns, the optimal portfolio weights are determined using mean-CVar optimization. The methodology is summarized below.

1) **Input:** Gather and format data.

a) Conduct agnostic fundamental analysis to get mispricing signals.

b) Conduct genetic algorithm selection over technical indicators and their hyperparameters.

2) **Neural Networks:** Train neural networks to predict five-day ahead returns using input data.

a) Obtain prediction errors.

3) **GARCH-Copula Portfolio Optimization:**

a) Train GARCH models to predict five-day volatility

(i) Normalize neural network prediction errors with GARCH-volatility

b) Fit a copula on normalized prediction errors.

c) Simulate 1000 instances from the fitted copula.

d) Conduct mean-CVaR optimization:

(i) Mean is defined by neural network predictions.

(ii) Losses for CVaR are defined by return predictions and simulated errors denormalized with predicted volatilities.

5.2.1 Agnostic fundamental analysis

In the first phase, data preparation regarding agnostic fundamental analysis is conducted. The rationale for this is both the univariate predictive power, demonstrated by Bartram and Grinblatt (2018), and the hypothesized cross-dependence with other time-series variables.

The analysis is conducted with a MLP, differing from both the Bartram and Grinblatt's (2018) and Hanauer et al. (2022). Input variables are the two most current reported periods of accounting numbers, as well as a sector dummy. Accounting numbers include variables from income statements, balance sheets and cash flow statements. The output variable is the market capitalization of the firm. The model is trained monthly with data available at the beginning of the month. Each training of the model is independent, which allows the model to capture time-variability in market pricing behaviour. The estimated outputs are then compared with the actual market valuation and the difference is computed as the percentage difference to the real market capitalization. The real market capitalization is updated daily using closing prices.

In choosing the model hyperparameters, a secondary regression analysis is conducted using predictions and the following five-day returns throughout the training data. The hyperparameter set is chosen by maximizing the variance explained by the model, consisting of a constant and the model implied mispricing. This is done as opposed to validation error minimization, as the purpose of the prediction is to be economically meaningful, rather than necessarily the one with lowest error, and due to the relatively low amount of dependent data compared to the size of the input space. The model and mean square computation is computed with Keras API (Chollet et al., 2015) and linear regression is conducted with statsmodels Python library (Seabold and Perktold, 2010).

The best model chosen for the agnostic fundamental analysis was a two hidden layer MLP, with 100 neurons in the first hidden layer, and 50 in the second, with rectified

linear unit activation and no dropout. The output layer contained one neuron, with linear activation. The model was trained on an Adam optimizer with a mean average percentage error, with a maximum of 1000 epochs and a loss patience of 10. Batch size used was 110, with random sampling from the data. All variables, except for dummies, were transformed by multiplying them by 1e-11. This was done to help numerical convergence. The model architecture is described in Table 1.

Table 1: FFNN for agnostic fundamental analysis

Layer	Output shape	Param	Connected to
Input	(None, 34)	0	
Hidden 1	(None, 100)	3500	Input
Hidden 2	(None, 50)	5050	Hidden 1
Output	(None, 1)	51	Hidden 2

5.2.2 Technical indicator input selection

Considering that there are 16 technical indicators with seven different input parameters, totalling 112 variables, reducing dimensions is necessary for optimal model fitting and the removal of redundant information. This input selection is done with a genetic algorithm. A genetic algorithm is inspired by evolutionary processes seen in nature. The procedure used largely follows that described by Haldurai et al. (2016):

Algorithm 3 Genetic Algorithm

Initialization:
Create a random population of chromosomes
Stop condition is not reached **Fitness:**
Fit predictive models of return using random forest regression based on the chromosomes
Calculate average validation error for each chromosome
Selection:
Select p parent chromosomes with the lowest prediction error
Crossover:
Create offspring chromosomes using uniform crossover from parent chromosomes Resulting in o number of offspring
Mutation:
Apply mutation operator to each chromosome with a certain probability Change the gene at any given locus across all its genes
Replace:
Replace the old population with the population of offspring
Loop:
Restart the process from the Fitness step until a stop condition is reached Choose the best chromosome throughout the procedure

The genetic algorithm is conducted with the scikit-learn library (Pedgerosa et al. 2011).

Technical indicator selection was conducted with an evolutionary algorithm utilizing a random forest regression. The evolutionary algorithm had a population size of 250, with a maximum of 20 features, and independent crossover probability of 0.9 and a mutation probability of 0.025. A random population of 10 chromosomes was used after population generation. Five-fold cross-validation within the training data was utilized in choosing the best fit model, with a patience for no change in cross-validation error of two. The random forest regressor used 5 estimators and had a minimum leaf size of 10 percent of all data.

5.2.3 Neural network models

There are three possible model architectures utilized which are BiLSTM, CNN-BiLSTM, and CNN-BiLSTM-Attention. BiLSTM is used instead of the LSTM, as it has shown better performance in stock prediction (Siami-Namini et al., 2019).

The BiLSTM model has three layers: the BiLSTM layer, a dense layer, and an ouput layer. For the dense layer, a ReLU activation function is used, and the output layer uses a linear activation function. For the models utilizing CNN, before the BiLSTM layer, a 1-dimensional CNN layer is applied to the input, followed by a pooling layer. This pooled ouput is then given as the input to the BiLSTM layer. In the CNN-BiLSTM-Attention model after the BiLSTM layer, a scaled dot-product attention layer is applied before the final dense layer. Hyperparameters are optimized over validation period mean square error.

All neural network models are trained with Keras API (Chollet et al., 2015), and the OLS estimation is conducted with the statsmodels Python library (Seabold and Perktold, 2010).

The models were trained on all stocks at once. All variables, except for dummies, were Z-score normalized based on training mean and standard deviation, including the returns. Return outputs were afterwards inversely transformed. The input space included data from periods t to $t-9$. Return output was calculated as $(C_{t+5}-C_t)/C_t$. All models were trained using mean square error optimization and the Adam optimizer with a learning rate of 0.001. All models were trained using a batch size of 10 percent of all training data. After choosing the best hyperparameters based on validation mean square error, an added selection criteria was considered. This was that the models produced at least one prediction over one percent in 100 of the 252 validation period days. This was done to ensure a balance between accuracy and usefullness in portfolio selection. All models were trained five times, and final output was determined by the average of those prediction outputs.

The BiLSTM models were trained with one BiLSTM layer, followed by a dropout layer, a dense layer, and the final output layer. The BiLSTM layer had 54 neurons with dropout and recurrent dropout of 0.7. The layer outputted 54 variables, to which a 0.9 dropout rate was used before a 54 neuron ReLU layer. The final output layer had one neuron with linear activation. The models were trained with 50 epochs. The model architecture is described in Table 2.

Table 2: BiLSTM architecture

Layer	Output shape	Param	Connected to
Input	(None, 10, 54)	0	
BiLSTM	(None, 54)	17712	Input
Hidden	(None, 54)	2970	BiLSTM
Output	(None, 1)	55	Hidden

The CNN-BiLSTM models were trained with convolutional layers with 54 filters, kernel size of four and same padding. The activation function used was a leaky rectified linear unit, with an alpha value of four. Average pooling with a pool size of three and same padding was used. A dropout of 0.1 was applied to the output. Following this, a BiLSTM layer with 54 neurons, with dropout and recurrent dropout of 0.6 was used. A dropout of 0.9 was applied on the output of the layer before a dense layer of 54 neurons, with ReLU activation. The final output layer used linear activation. The models were trained with 50 epochs. The model architecture is described in Table 3.

Table 3: CNN-BiLSTM architecture

Layer	Output shape	Param	Connected to
Input	(None, 10, 54)	0	
Conv1D	(None, 4, 54)	11718	Input
Average Pooling	(None, 4, 54)	0	Conv1D
BiLSTM	(None, 54)	17712	Average Pooling
Hidden	(None, 54)	2970	BiLSTM
Output	(None, 1)	55	Hidden

The CNN-BiLSTM-Attention model was otherwise identical to the CNN-BiLSTM model, except after the BiLSTM layer, an attention layer was applied. However much higher dropouts were used, with a 0.85 dropout rate after the pooling layer, 0.9 dropout and recurrent dropout in the BiLSTM layer, and after the attention layer a dropout of 0.9 was used. The models were trained with 25 epochs.

Table 4: CNN-BiLSTM-Attention architecture

Layer	Output shape	Param	Connected to
Input	(None, 10, 54)	0	
Conv1D	(None, 4, 54)	11718	Input
Average Pooling	(None, 4, 54)	0	Conv1D
BiLSTM	(None, 54)	17712	Average Pooling
Reshape 1	(None, 54, 1)	0	BiLSTM
Attention	(None, 54, 1)	0	Reshape 1
Reshape 2	(None, 54)	0	Attention
Hidden	(None, 54)	2970	Reshape 2
Output	(None, 1)	55	Hidden

On top of the three architectures used, a fourth model was created from using all three of the model outputs, named the All ensemble model.

5.2.4 Feature importance

Feature importance was done by categorizing the variables into eight categories: Basic variables, technical indicators, candlestick indicators, agnostic fundamental analysis, sentiment variables, market sentiment variables, macroeconomic indicators, and sector dummies. Permutation feature importance analysis was conducted by randomly permuting the variables within a category during the testing period, and comparing the permuted mean square error to the actual mean square error. The feature importance analysis follows that proposed by Fisher et al. (2019), decribed in Algorithm 4, where $\hat{f}$ is the function for an NN-model.

Algorithm 4 Feature importance analysis

1: Estimate the original model error $e_{original} = MSE(y, \hat{f}(X))$
2: **for** each feature $i \in= \{1, ..., n\}$ **do**
3: Generate a a new feature matrix $X_{permutation}$ from $X_{original}$, by permuting feature i
4: Calculate error $e_{permuation} = MSE(y, \hat{f}(X_{permuation}))$
5: **end for**

Feature importance was also analyzed with a linear regression model. To keep the analysis informative, means are taken across the ten lags used in the original models for each datapoint.

$$\hat{f}(\mathbf{y}) = \alpha + \beta \mathbf{X}_{means} + \varepsilon,$$

where $\hat{f}(\mathbf{y})$ are the NN model predictions and $\mathbf{X}_{means}$ is the mean transforemed matrix of the original dataset.

Both analyses were conducted with data from the testing period.

5.2.5 Copula construction and portfolio optimization

A regular copula was fitted on the train prediction return errors (train and validation for the test period), normalized with predicted conditional volatility from T-distributed GARCH(1,1) models

$$\begin{aligned} r_t &= \varepsilon_t, \\ \varepsilon_t &= \sigma_t z_t, \\ \sigma_t^2 &= \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2, \\ z_t &\sim \text{Student-t}(\nu). \end{aligned}$$

which were fit for each stock separately. A zero-mean GARCH model was chosen due to its simplicity, and lack of assumptions about the underlying time-series. Assuming an autoregressive or empirical mean were utilized, this would have to be assumed stable out-of-sample. Once the NN models were been trained on the training data, standardized return errors were computed as

$$\frac{r_{t+5} - \mathbb{E}(r_{t+5})}{\mathbb{E}(\sigma_{t+5})}.$$

Using the errors, regular vine copulas were fitted om the training data. Using standardized return errors allows for the usage of time-variability in volatility, and returns could be computed from simulations as $\mathbb{E}(r_{t+5}) + \mathbb{E}(\sigma_{t+5}) * e_s$, where e_s is a simulated normalized error from the copula. The copula is fitted using the pyvinecopulalib library (Nagler and Vatter, 2023) using Algorithm 5 (Dissman et al. 2013).

Algorithm 5 Sequential R-Vine model selection (Dissman et al. 2013)

1: Calculate the empirical Kendall's tau $\hat{\tau}_{j,k}$ for all possible variable pairs $\{j,k\}, 1 \leq j < j \leq n$.
2: Select the spanning tree that maximizes the sum of the absolute empirical Kendall's taus.
3: For each edge $\{j,k\}$ in the selected spanning tree, select a copula on and estimate the corresponding parameters(s). Then transform $\hat{F}_{j|k}(x_{\imath j}|x_{\imath k}$ and $\hat{F}_{l|j}(x_{\imath j}|x_{\imath k}, \imath = 1,...,N$ using the fitted copula $\hat{C}_{jk}$.
4: **for** $i = 2,...,n-1$ **do**
5: Calculate the empiritical Kendall's tau $\hat{\tau}_{j,k|D}$ for all conditionoal cariable pairs $j,k|D$ that can be part of tree T_i.
6: Among these edges, select tge spanning tree that maximizes the sum of absolute empirical Kendall's taus.
7: For each edge $j,k|D$ in the selected spanning tree, select a conditional copula and esimate the corresponding parameter(s). Then transform $\hat{F}_{j|k\cup D}(x_{\imath j}|x_{\imath k}, x_{\imath D})$ and $\hat{F}_{k|j\cup D}(x_{\imath k}|x_{\imath j}, x_{\imath D})$, $\imath = 1,...,N$, using the fitted copula $\hat{C}_{jk|D}$.
8: **end for**

Only one parameter pair-copulas were used in the copula fitting, which includes the following copulas: Gaussian, Clayton, Gumbel, Frank, and Joe. This was due to runtime limitations, and the high dimensionality of the data.

Portfolio optimization was done with mean-CVAR optimization using the PyPortfolioOpt library (Martin, 2021). The mean-CVAR optimization was computed with linear programming based on Uryasev and Rockafellar's (2001) method, which follows the

equation

$$\begin{aligned} \text{Minimize} \quad & \alpha + \frac{1}{1-\beta}\frac{1}{T}\sum_{i=1}^{T} u_i \\ \text{subject to} \quad & u_i \geq 0 \\ & u_i \geq -w^T r_i - \alpha, \end{aligned}$$

where u_i is the shortfall of for scenario i. The mean r_i is defined by the outputs from the NN models minus the considered transaction cost. Shortfalls are defined by the normalized copula error and predicted volatility from GARCH(1,1) model $\mathbb{E}(\sigma_t) * e_s$, where e_s is an instant simulated normalized errors. The portfolios are long only portfolios, which means an additional constraint $w_k \geq 0$, where $w_k \in w$ was added to the original formulation of the problem. For the mean-CVar optimization, 1000 simulations were conducted from the copula, and an α value of 0.05 ($\beta = 0.95$) and a CVaR value of 0.1 was used. Transaction costs of 0%, 0.5%, and 1% were considered. Transaction costs were equally divided between the buying date and selling date. Buying and selling was conducted at close, and it is assumed that all information available at close was available to the model.

5.2.6 Performance evaluation

The evaluation of the models investigates the informative value of fundamental analysis, technical indicators chosen, the performance of NN-copula model and the importance of variables used in the NN.

The performance of the agnostic fundamental analysis is analysed with portfolio performance of pentiles of the mispricing signal in monthly updated portfolios. The predictive power of the mispricing signal is also evaluated with the linear regression model

$$r_{t+5} = \alpha + \beta M,$$

where r_{t+5} is the five-day return and M is the mispricing signal.

The predictive accuracy of the different NN-models is evaluated with validation and test period mean square error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

and mean absolute error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|.$$

Technical indicators selected by the genetic algorithm and used in the NN-models are reported.

An additional linear regression model is estimated for benchmarking purposes, using data from training period, which utilizes the same set of variables as the NN-models

$$\mathbf{r}_{t+5} = \alpha + \beta \mathbf{x},$$

where $\mathbf{x}$ is the flattened vector of the input data.

The performance of the NN-copula models is compared with OLS-copula, equally-weighted (EW) and value-weighted (VW) buy-and-hold portfolios, as well as prediction weighted portfolios. The prediction weighted portfolios perform risk-naive selection based on the sum of of positive predictions

$$w_i = \frac{[\mathbb{E}(r_{t+5,i})]^+}{\sum_{k=0}^{N}[\mathbb{E}(r_{t+5,k})]^+},$$

where $\mathbb{E}(r_{t+5}, k)$ is the prediction of one of the models and N is the number of stocks.

Comparing the NN-copula performance to OLS-copula and market portfolios informs about the economic value of the NN predictions, where as comparing the performance of the copula portfolios to the prediction weighted portfolios informs about the successfulness of the copula and mean-CVaR optimization, as the used return predictions stay the same.

In evaluation the portfolios, five measures were used:

$$\text{Annualized Return} = 252 \times \text{Average Daily Log Return},$$

$$\text{Annualized Volatility} = \sqrt{252} \times \text{Average Daily Log Volatility},$$

$$\text{Sharpe ratio} = \frac{\text{Annualized Return}}{\text{Annualized Volatility}},$$

$$\text{CVaR}_{0.05} = \frac{1}{\lceil \alpha \times n \rceil} \sum_{i=1}^{\lceil \alpha \times n \rceil} L_{(n-i+1)},$$

where n is the number of days, L is the sorted list of losses, and $\lceil \alpha \times n \rceil$ is the number of losses included.

$$\text{Maximum drawdown} = \max_{t \leq u} \left(\frac{P(t) - P(u)}{P(t)} \right)$$

6 Results

6.1 Agnostic fundamental analysis

The summary statistics of the mispricing signal is reported in Table 5. The mean is clearly negative, and the distribution is highly positively skewed and significantly leptokurtic.

Table 5: Agnostic mispricing signal descriptive statistics

Statistic	Mean	Standard deviation	Skewness	Kurtosis
Variance	-0.0911	0.2894	2.880	199.8

The linear predictive performance of the agnostic pricing signal deteriorated during the test phase. Three OLS regression results of mispricing and the following five-day

returns without dividends are produced in Table 6 for the training, validation and testing periods.

Table 6: Agnostic mispricing signal and five-day returns

Period	Training	Validation	Test
	Coefficients		
Constant	0.0036***	-0.0005***	0.0020***
Mispricing signal	0.0048***	0.0118***	-0.0050***
R-squared	0.00078	0.0020	0.00049

The first describes the performance during the training period of 2016-2021. As can be seen, the mispricing has a statistically significant positive coefficient of 0.0048, with a constant of 0.0036. This was the highest R-squared producing model with a R-squared of 0.00078. During the validation period of 2022, the coefficient increased to 0.0118, with a constant of -0.0005. The R-squared value during the validation period was 0.0020. However during the test period, the performance of the signal not only disappeared, but became negative, with a coefficient of -0.0050, and a constant of 0.0020. However during the testing phase, the R-squared was also the lowest at 0.00049. This indicates that the performance of the mispricing signal is not constant, and may even become negative during periods.

There is also evidence that the relationship between the mispricing signal and actual returns is either non-linear or non-existent. This is evidenced by non-ordered performance of monthly updated pentile equally weighted portfolios based on the mispricing signal. During the training period of 2016-2021, the two portfolios with the highest annualized return performance with dividends were the highest and lowest pentiles of mispricing, with 18.6 percent and 17.9 percent respectively, compared to the equally weighted market return of 17.0 percent. During the validation period of 2022, the highest pentile performed the best with an annualized return of -5.3 percent, however the worst performing pentile was the second highest pentile with an annualized return of -8.4 percent, compared to the market's -6.4 percent. During the test period of 2023, the lowest pentile of mispricing actually performed the best, with an annualized return of 18.4 percent, while the highest pentile performed the worst, with an annualized return of 11.8 percent, compared to the equally weighted market return of 14.6 percent. The performance of the portfolios for the entire dataset is shown in Figure 12.

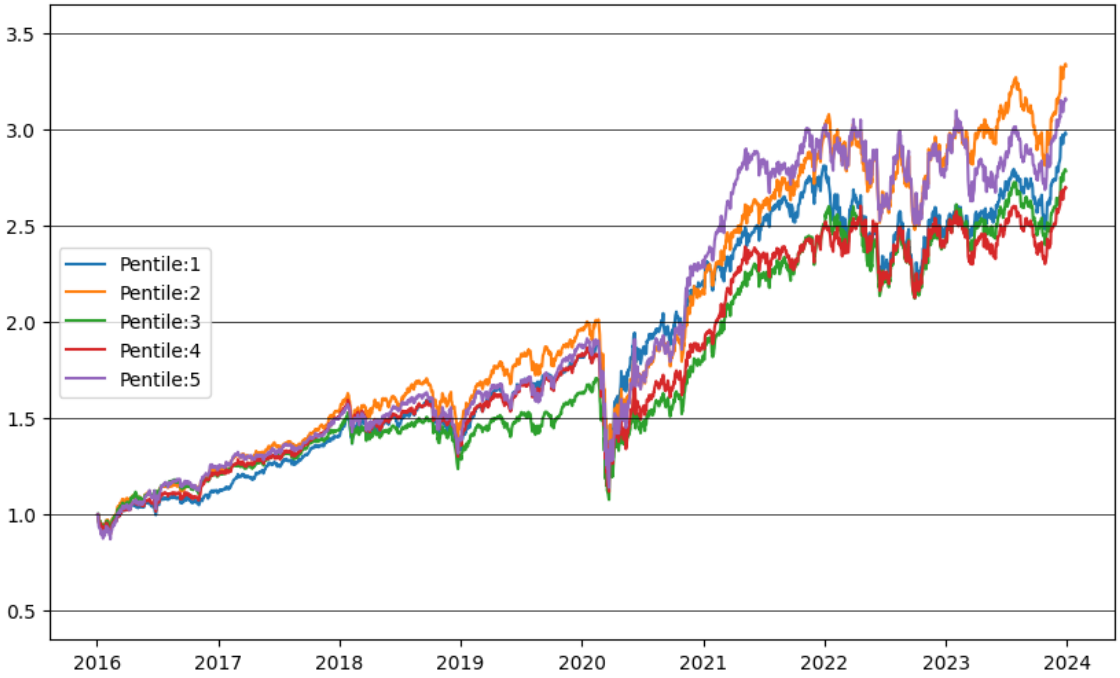

Figure 12: Agnostic mispricing portfolios

6.2 Technical indicator input selection

The best cross-validation MSE was 0.992. Considering that predictimg the mean would have had an MSE of 1 by definition, as the return values were Z-score normalized, the best Random forest regression model does not offer significant insight in the prediction of returns. The algorithm ended up choosing 19 variables, given a maximum of 20, which were the following technical indicators:

- Smooth Moving Average 100-day
- Relative Strength Index 7-day
- Relative Strength Index 21-day
- Normalized Average True Range 252-day
- Commodity Channel Index 7-day
- William's R% 100-day
- Aroon Oscillator 14-day
- Volume Weighted Average Price 14-day
- Volume Weighted Average Price 21-day
- Volume Weighted Average Price 50-day
- Volume Weighted Average Price 252-day
- Minimum 252-day
- Maximum 14-day
- Standard deviation 5-day
- Standard deviation 21-day
- Standard deviation 50-day
- Standard deviation 252-day
- High Bollinger Band 5-day
- High Bollinger Band 252-day

After adding the technical indicators, the final dataset contained 54 input variables.

6.3 Neural networks and portfolio optimization

The volatility and variance of five-day returns is described in Table 7, and MSE and MAE of the models is described in Tables 8 and 9 for the validation and test period respectively.

In terms of prediction accuracy the best model during the validation period was the BiLSTM model according to mean square error and the CNN-BiLSTM and CNN-BiLSTM-Attention models according to mean absolute error. The worst NN-model based on MSE was the CNN-BiLSTM-Attention model, and based on MAE the All ensemble. During the testing period both accuracy methods indicated that the CNN-BiLSTM-Attention model performed the best. The BiLSTM performed the worst in terms of both measures. However, the differences were quite small, and not differentiating until the third or fourth, or even fifth significant figure. The OLS model performed by far the worst across both periods and measures.

Table 7: Five-day return dispersion measures, x100

Period	Validation	Test
Variance	0.2371	0.1436
Standard deviation	4.869	3.790

Table 8: Validation prediction error, x100

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS
Mean square error	0.2382	0.2383	0.2391	0.2386	0.2666
Mean absolute error	3.651	3.650	3.650	3.653	3.908

Table 9: Test prediction error, x100

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS
Mean square error	0.1433	0.1431	0.1430	0.1431	0.1664
Mean absolute error	2.748	2.748	2.746	2.747	3.028

Portfolio return behaviour for the models with no transaction costs, 0.5% transaction costs and 1% transaction costs for the validation and test period are shown in Tables 10-21. Figures showing the return behaviour are presented in the Appendix.

With a CVaR of 0.1, during the validation period no model consistently outperformed the market. With 0% transaction costs, all of NN models performed slightly less worse, in terms of annualized returns than the market portfolios, with the BiLSTM model being the only one with a positive return. However in doing so, all NN model -based portfolios experienced nearly double the volatility and CVaR compared to the market portfolios. This is shown in Table 10. With 0.5% transaction costs, all models underperformed the market in terms of both return and risk measures. Compared to the other transaction costs

alternatives, the NN model performance was clearly the worst with 0.5% transaction costs. This is shown in Table 11. With 1% transaction costs, all NN models again outperformed the market portfolios in terms of returns, with BiLSTM and CNN-BiLSTM producing positive annualized returns. The CNN-BiLSTM produced the highest return and Sharpe ratio across the portfolios. While risk measures improved over the previous transaction cost, both volatility and CVaR were still higher than that of market portfolios. This is shown in Table 12. The OLS model performed by far the worst out of the available strategies in each transaction cost scenario both in terms of returns and risk measures.

During the validation period the prediction weighted portfolios overall performed worse than the copula-based portfolios. However when given 0% transaction costs, the portfolios had higher returns and better risk measures compared to both market portfolios and the 0% transaction cost copula-based portfolios. This is shown in Tables 13-15.

Table 10: Validation portfolio results, 0% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	5.72%	-7.55%	-6.15%	-6.45%	-47.4%	-8.59%	-15.3%
Annualized volatility	37.1%	36.3%	38.6%	38.2%	55.1%	21.9%	22.4%
Sharpe ratio	0.154	-0.208	-0.159	-0.169	-0.859	-0.393	-0.685
CVaR	4.73%	4.42%	4.88%	4.93%	7.92%	3.06%	3.16%
Maximum drawdown	23.6%	25.9%	27.8%	29.3%	51.2%	19.5%	22.8%

Table 11: Validation portfolio results, 0.5% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	-19.5%	-19.8%	-23.8%	-28.6%	-71.8%	-8.59%	-15.3%
Annualized volatility	37.3%	35.7%	39.7%	38.4%	55.1%	21.9%	22.4%
Sharpe ratio	-0.521	-0.555	-0.599	-0.745	-1.30	-0.393	-0.685
CVaR	5.02%	4.75%	5.18%	5.10%	8.01%	3.06%	3.16%
Maximum drawdown	32.6%	31.9%	34.2%	34.8%	59.2%	19.5%	22.8%

Table 12: Validation portfolio results, 1% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	13.7%	0.636%	-4.32%	-3.58%	-88.4%	-8.59%	-15.3%
Annualized volatility	26.8%	24.6%	26.0%	24.6%	58.9%	21.9%	22.4%
Sharpe ratio	0.509	0.259	-0.166	-0.146	-1.50	-0.393	-0.685
CVaR	3.88%	3.41%	3.96%	3.52%	8.47%	3.06%	3.16%
Maximum drawdown	15.7%	18.2%	25.7%	16.1%	63.3%	19.5%	22.8%

Table 13: Validation portfolio results, 0% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	4.01%	7.35%	-0.93%	4.76%	-3.15%	-8.59%	-15.3%
Annualized volatility	7.24%	8.41%	4.61%	6.48%	23.2%	21.9%	22.4%
Sharpe ratio	0.554	0.874	0.201	0.735	-0.136	-0.393	-0.685
CVaR	0.919%	1.09%	0.680%	0.813%	3.47%	3.06%	3.16%
Maximum drawdown	5.44%	4.76%	-5.25%	4.02%	18.6%	19.5%	22.8%

Table 14: Validation portfolio results, 0.5% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	-43.6%	-46.8%	-39.4%	-41.6%	-28.0%	-8.59%	-15.3%
Annualized volatility	39.7%	41.5%	32.2%	38.4%	29.1%	21.9%	22.4%
Sharpe ratio	-1.09	-1.13	-1.22	-1.08	-0.960	-0.393	-0.685
CVaR	5.39%	5.82%	4.52%	5.38%	4.43%	3.06%	3.16%
Maximum drawdown	42.9%	46.0%	39.8%	42.2%	31.8%	19.5%	22.8%

Table 15: Validation portfolio results, 1% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	-42.5%	-56.4%	-13.5%	-58.1%	-70.6%	-8.59%	-15.3%
Annualized volatility	49.4%	49.8%	46.6%	50.1%	39.4%	21.9%	22.4%
Sharpe ratio	-0.862	-1.13	-0.289	-1.16	-1.79	-0.393	-0.685
CVaR	6.61%	6.89%	6.60%	6.87%	6.75%	3.06%	3.16%
Maximum drawdown	48.5%	48.1%	27.3%	46.6%	52.1%	19.5%	22.8%

With a CVar of 0.1, during the testing period, no model consistently outperformed the market. With 0% transaction costs, all models outperformed the market in terms of annualized returns. While volatility and CVaR were elevated, all models had higher Sharpe ratios than the EW-portfolio, and the BiLSTM model had a higher Sharpe ratio than both the EW- and VW-portfolios. This is shown in Table 16. With 0.5% transaction costs, three of the NN models, the BiLSTM, the CNN-BiLSTM-Attention, and the All ensemble had higher returns than both market portfolios, while CNN-BiLSTM and the All ensemble produced higher returns than the EW-portfolio. Higher returns were again at the cost of higher risk measures, as only the BiLSTM model had a higher Sharpe ratio than the EW-portfolio and all NN models produced significantly lower Sharpe ratios than the VW-portfolio. This is shown in Table 17. With 1% transaction costs, no NN model outperformed both market portfolios in any of the measures, however the CNN-BiLSTM-Attention model produced higher returns than the EW-portfolios. The BiLSTM model was the only other model to produce positive annualized returns. While CVaR remained at the same level as with lower transaction costs, volatility measures improved, although still significantly inflated compared to market portfolios. This is shown in Table 17. In sharp contrast the performance of the OLS predictor in the validation data, the OLS model was the only model to consistently have higher returns than both market portfolios, and in two of the three transaction cost scenarios had the highest return compared of all portfolios. However, risk measures were consistently higher with the OLS model compared to the NN models. This is shown in Tables 16-18.

The performance of the prediction weighted portfolios again was significantly worse than the copula-based portfolios, although the difference was not as large as during the validation period. With 0% transaction costs, the prediction weighted portfolios again had by far the best risk measures compared to both market portfolios and copula-based portfolios, however at the costs of significantly lower returns. This is shown in Table 19. With 0.5% transaction costs, the BiLSTM, CNN-BiLSTM-Attention and OLS portfolios had higher annualized returns than both market portfolios, while the All ensemble had higher returns than the EW-portfolio. However the returns were consistently lower for the prediction weighted portfolios compared to copula-based portfolios. Risk measures were also worse for the prediction weighted portfolios compared to both market and copula-based portfolios. This is shown in Table 20. With 1% transaction costs, All models had negative returns, with significantly worse risk measures than both the copula-based and market portfolios. This is shown in Table 21.

Table 16: Test portfolio results, 0% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	58.6%	37.9%	46.3%	39.7%	72.6%	12.1%	18.9%
Annualized volatility	29.4%	30.6%	29.9%	30.4%	35.9%	13.6%	12.1%
Sharpe ratio	1.995	1.239	1.547	1.307	2.026	0.892	1.57
CVaR	3.16%	3.41%	3.41%	3.43%	4.07%	1.71%	1.48%
Maximum drawdown	13.4%	19.3%	18.9%	19.3%	16.6%	12.3%	9.84%

Table 17: Test portfolio results, 0.5% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	31.6%	17.5%	23.6%	17.3%	27.2%	12.1%	18.9%
Annualized volatility	29.4%	30.0%	29.8%	29.9%	35.4%	13.6%	12.1%
Sharpe ratio	1.07	0.586	0.791	0.580	0.768	0.892	1.57
CVaR	3.37%	3.50%	3.50%	3.55%	4.00%	1.71%	1.48%
Maximum drawdown	16.0%	21.6%	22.6%	23.5%	26.7%	12.3%	9.84%

Table 18: Test portfolio results, 1% transaction cost, CVaR = 0.1

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	2.66%	-8.26%	15.6%	-11.6%	26.1%	12.1%	18.9%
Annualized volatility	21.3%	27.1%	17.2%	23.5%	34.3%	13.6%	12.1%
Sharpe ratio	0.125	-0.305	0.906	-0.492	0.762	0.892	1.57
CVaR	3.28%	3.69%	2.55%	3.61%	4.01%	1.71%	1.48%
Maximum drawdown	19.9%	33.7%	13.2%	31.2%	23.8%	12.3%	9.84%

Table 19: Test portfolio results, 0% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	7.01%	7.40%	7.74%	6.94%	40.9%	12.1%	18.9%
Annualized volatility	5.24%	5.98%	5.51%	5.37%	18.2%	13.6%	12.1%
Sharpe ratio	1.339	1.239	1.407	1.293	2.249	0.892	1.57
CVaR	0.621%	0.697%	0.640%	0.640%	1.90%	1.71%	1.48%
Maximum drawdown	3.25%	3.66%	3.93%	3.61%	8.40%	12.3%	9.84%

Table 20: Test portfolio results, 0.5% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	21.4%	8.46%	19.9%	14.0%	19.1%	12.1%	18.9%
Annualized volatility	36.7%	37.2%	30.8%	35.3%	22.3%	13.6%	12.1%
Sharpe ratio	0.583	0.227	0.649	0.398	0.856	0.892	1.57
CVaR	4.46%	4.35%	3.66%	4.20%	3.00%	1.71%	1.48%
Maximum drawdown	32.5%	35.1%	24.6%	31.2%	17.1%	12.3%	9.84%

Table 21: Test portfolio results, 1% transaction cost, prediction weighted

Strategy	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All	OLS	EW	VW
Annualized return	-10.2%	-26.4%	-2.64%	-29.3%	-10.6%	12.1%	18.9%
Annualized volatility	39.0%	44.9%	39.1%	43.0%	29.2%	13.6%	12.1%
Sharpe ratio	-0.262	-0.587	-0.676	-0.681	-0.363	0.892	1.57
CVaR	5.84%	6.10%	5.37%	6.06%	3.73%	1.71%	1.48%
Maximum drawdown	48.7%	50.6%	39.8%	51.0%	36.9%	12.3%	9.84%

6.4 Feature importance

The results of the permutation feature importance analysis are shown in Figure 13. The values were computed as

$$Error = \frac{e_{permutation} - e_{original}}{e_{original}},$$

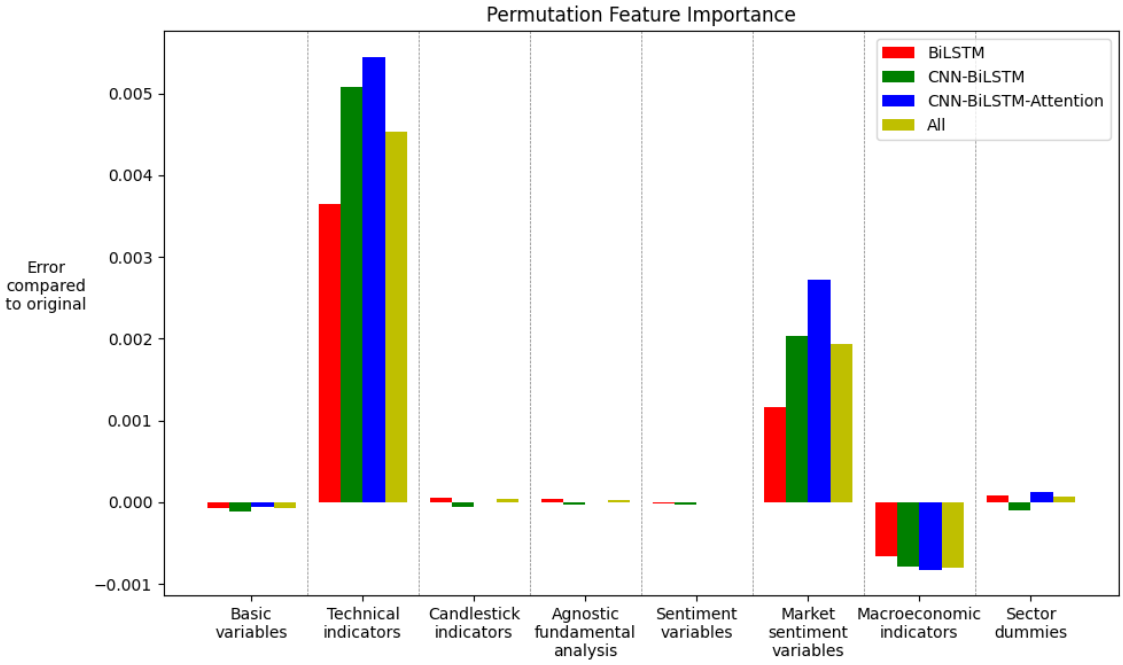

Figure 13: Permutation feature importance

Variable importance linear regression results for all models are shown split into Tables 22-24.

Table 22: Model variable output regression (1/3)

Variable	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All
Constant	0.0028***	0.0028***	0.0036***	0.0031***
Close	-0.0005***	-0.0002	4.409e-05	-0.0002***
High	7.002e-06	-0.0004***	-6.272e-05	-0.0001**
Low	-0.0003***	-0.0003**	-0.0004***	-0.0003***
Open	8.732e-05*	0.0001	-1.704e-05	7.141e-05
Volume	-0.0016***	-0.0018*	0.0007	-0.0009**
SMA 100-day	0.0010***	0.0015***	0.0007***	0.0011***
RSI 7-day	-0.0004***	-0.0005***	-0.0002***	-0.0004***
RSI 21-day	0.0009***	0.0012***	0.0005***	0.0008***
NATR 252-day	0.0003***	0.0003***	0.0005***	0.0003***
CCI 7-day	7.28e-05**	8.816e-05***	-3.226e-05*	4.29e-05***
William's R% 100-day	0.0002***	0.0004***	5.855e-07	0.0002***
Aroon osc 14-day	-7.228e-05***	-5.964e-05***	-6.243e-05***	-6.478e-05***

Table 23: Model variable output regression continued (2/3)

Variable	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All
VWAP 14-day	-0.0010***	-0.0017***	-0.0001	-0.0010***
VWAP 21-day	0.0002***	0.0008***	0.0007***	0.0003***
VWAP 50-day	0.0008***	0.0008***	0.0007***	0.0008***
VWAP 252-day	-0.0001***	-0.0006***	0.0003***	-0.0003***
Min 252-day	-0.0003***	-0.0004***	2.282e-05	-0.0002***
Max 14-day	0.0004***	0.0003***	0.0003***	0.0003***
STD 5-day	5.336e-05	0.0007***	-0.0008***	-1.243e-05
STD 21-day	-0.0001***	6.599e-05***	0.0002***	4.982e-05***
STD 50-day	0.0005***	0.0003***	0.0002***	0.0003***
STD 252-day	0.0006***	0.0012***	0.0007***	0.0008***
High BB 5-day	0.0001***	0.0006***	-0.0009 ***	-4.527e-05
High BB 252-day	0.0004***	0.0009***	-0.0002***	0.0004***
Candlestick bullish	4.899e-05***	-8.399e-06	-0.0001***	-3.027e-05***
Candlestick bearish	0.0001***	5.913e-05***	9.174e-06	6.946e-05***
Mispricing signal	-2.253e-06	-4.865e-05***	8.893e-05***	1.268e-05***
Google Trends	1.782e-05***	-3.966e-05***	-0.0001***	-4.192e-05***
AAII Bearish	-9.563e-05***	-0.0001***	4.03e-05***	-5.577e-05***
AAII Bullish	0.0001***	2.628e-05	0.0007***	0.0003***
Index advances	-0.0024***	0.0008	-0.0096***	-0.0037***
Index declines	-0.0026***	0.0006	-0.0093***	-0.0038***
Index new highs	6.538e-07	-5.784e-05***	0.0002***	3.808e-05***
Index new lows	2.819e-05***	0.0002***	0.0002***	0.0001***
Index return	-0.0003***	-0.0004***	0.0002***	-0.0002***

Table 24: Model variable output regression continued (3/3)

Variable	BiLSTM	CNN-BiLSTM	CNN-BiLSTM-Attention	All
VIX	-0.0001***	-0.0002***	0.0002***	-5.013e-05***
OVX	0.0007***	0.0019***	0.0107***	0.0044***
GVZ	8.096e-05***	-7.107e-05***	-9.652e-05***	-2.888e-05*
TNX	0.0002***	0.0001***	-0.0003***	1.629e-05
IRX	-5.649e-06	0.0001***	-0.0004***	-8.727e-05***
Unemployment	6.91e-05***	-9.222e-05***	-0.0002***	-8.81e-05***
Inflation	-7.043e-05***	-0.0002***	-0.0002***	-0.0002***
Industrial prod	-6.762e-05***	9.384e-05***	-0.0001***	-2.871e-05***
GDP	-0.0002***	-0.0002***	0.0005***	1.887e-05
Communication Services	-0.0002***	-0.0002***	-3.349e-06	-0.0001***
Consumer Cyclical	-0.0002***	5.195e-05***	-6.394e-05***	-5.879e-05***
Consumer Defensive	-5.357e-05***	7.286e-05***	-1.493e-05	1.455e-06
Energy	-0.0002***	-0.0002***	3.976e-05***	-0.0001***
Financial Services	-9.616e-05***	-0.0002***	-4.115e-06	-0.0001***
Healthcare	-0.0001***	-9.851e-05***	-0.0001***	-0.0001***
Industrials	-8.415e-05***	-5.494e-05***	-4.122e-05***	-6.01e-05***
Real Estate	-0.0002***	-0.0003***	-7.883e-05***	-0.0002***
Technology	-2.251e-05**	-9.662e-05***	7.006e-05***	-1.635e-05
Utilities	-2.251e-05**	-1.849e-05	-2.212e-05*	-4.693e-05***
R-squared	0.842	0.755	0.869	0.872

7 Discussion

7.1 Agnostic fundamental analysis

There are several possible reasons for the unstable and relative underperformance of the used model compared to those seen in literature (Bartram and Grinblatt's (2018) and Hanauer et al. (2022)). The first one is lack of data. While 329 is not a small number, Bartram and Grinblatt used all listed U.S. stocks, while Hanauer et al. used 8,121 European companies. 329 might not be enough data for the task. Wheter this is the case however may not be the true, as there exists literature, that utilizes the agnostic fundamental approach successfully with less than 329 stocks (e.g. Shah et al. 2023; Navas et al. 2023).

The second reason could be the time period. Hanauer et al.'s analysis spans the period in 1987–2019, while Bartram and Grinblatt used data from 1987–2012. The possibility of using a relatively big data approach with new technology may simply have made the approach non-viable given a more modern market period. In the periods studied in literature, such an approach was not viable and as such brings in to question the validity of the results seen.

Relating to the time period, a third possible reason is the complexification of the world and as such companies. In 1990, the utilities materials, and industrial sectors accounted for roughly 30 percent of the S&P 500 market capitalization, and by 2016, this share had fallen to 20.1 percent. Meanwhile the technology sector by itself had risen to 20.7 percent by 2016. (Bespoke, 2016) This change is reflective of the rise of the service economy, where more skilled workers produce more complex goods (Buera and Kaboski, 2012). This change and evolution of the economy lends itself to more complex companies, and the valuation of such companies becomes increasingly divorced from current accounting figures. Because of this change, agnostic fundamental analysis may have already become less reliable.

Despite the apparent failure, the variable was used, as while it did not have value alone, it may still have cross dependent or conditional value, which the nonlinear models could in theory still factor in.

7.2 Technical indicator input selection

Technical indicator selection was done independent of final models, with a random forest approach. The reason for this was computational costs, as running thousands of regressions with the final models would require significantly more computational power when compared to the random forest approach. While using the final models was feasible in theory, in practice the tradeoff between computational costs and selection optimization was considered in making this decision.

Relating to computational costs, not all stocks were used in the selection process, as this again required significant resources. A random sample of 10 stocks, which for the training period signified 15 110 data points, was used.

The selection was also done without taking into account the other variables used in the final model. This means while perhaps the indicators selected were optimal or close to optimal independently, the selection of technical indicators could have been influenced

by other data used later on, and therefore may not have been optimal for the final models, which did utilize the other non-technical indicator variables.

The selection was also conducted without lags and only the most recent data point was considered. This was done to simplify the selection process, and based on the hypothesis that the most recent state of the variable was the most significant. This means changes in the underlying technical indicators were not considered, which likely would have been informative, as the final models did utilize lags beneficially.

Based on this, the technical indicator selection could be described as good enough, however with a high degree of probability not optimal. The choice of technical indicators should thus be evaluated taking into account the above mentioned limiting factors.

The population of technical indicators included hyperparameters, which indicated the period used to calculate each indicator, were 5, 7, 14, 21, 50, 100, 252. All periods were selected which indicates that for weekly returns, both long and short term behaviours of stock price are informative.

Five of the indicators were measures of historical dispersion, either in the form of normalized average true range or standard deviation. These dispersion indicators used both relatively short-term and long-term lags. This likely stems from the well founded concept of volatility clustering, which informs return prediction in at least two ways. The first is that because past volatility informs future volatility, this informs return prediction by defining the likely range of those returns, meaning extremely high or low predictions are less unlikely during less volatile periods and vice versa. The secondary reason is that there might be a direct relationship between volatility and return prediction accuracy. Marquering and Verbeek (2004) found that return predictability was higher during periods of high volatility.

Volume weighted average price (VWAP) was selected four times, with both short and long term lags. VWAP taken together with current price indicates deviation from the current market sentiment over a give period. Literature on VWAP has especially focused on VWAP orders, which are buy or sell orders conditioned on the VWAP, which are done especially by institutional investors to limit market impact and commission costs. (Madhavan, 2002). VWAP has also been used successfully in trading strategies, (e.g. Colliri, and Ferreira, 2012). However, there may be an issue with combining VWAP and assumed constant transaction costs. While price deviations from the VWAP can be used to predict returns, orders placed farther away from the VWAP have been seen to have a larger market impact. (Madhavan, 2002) Based on this, the real transaction costs would be higher when the price is farther from VWAP, and not constant, as was used in the methodology.

7.3 Neural network predictions and portfolio optimization

Based on the prediction accuracies of the models and application of those predictions in portfolio selection, there appears to be some relationship between prediction accuracy and economic value. However this relationship was not constant across different transaction cost scenarios. The reasons for this may stem from the difference between prediction accuracy and prediction significance. This can be illustrated by a simplified example. Let's say there is a binary distribution, which outputs either -1 or 1 with equal probability, and we have two predictors: Predictor 1, which always outputs 0 and Predictor 2, which when it outputs 1 there is 75% likelihood that the actual value is 1, but otherwise outputs

-10. Clearly, the prediction accuracy of Predictor 1 is much better than Predictor 2's. Let's also say there is an agent which has a yes or no option on incurring the unrealized value or, and a goal of maximizing the sum of incurred values. Using Predictor 1 the agent would have expected outcome of zero, where as using predictor 2, and only incurring the outcome if the prediction is 1, the agent would have have an expected outcome of $0.5np$, where n is the sample size and p the probability Predictor 2 outputs 1. This example illustrates that instead of unconditional accuracy, conditional accuracy is the determining factor in choosing the best predictor for an agent who can opt in or out in using the predictor. This concept also applies to portfolio selection.

However as the differing validation and test period results show, in the trained models the prediction interval which is optimal is clearly not constant. In the validation period, the interval of $[1\%, \infty)$ appears to be needed in creating overperforming returns, while during the test period this interval was $[0, \infty)$. During the test period, this interval was not due to lack of conditional accuracy for the $[1\%, \infty)$ interval, but due to the low frequency of predictions in the interval. This resulted in lack of market participation. In subperiods during the where market participation was possible, the pool of stocks to create an optimized portfolio was too low for the $[1\%, \infty)$ interval, resulting in inefficient portfolios. This indicates that while using transaction costs as the threshold would make sense from an intuitive perspective, the distribution of predictions and the conditional accuracy appears to be non-constant, and as such, a dynamic threshold should be utilized instead of the only transaction costs. The second alternative would be to use no threshold conditioning on the prediction, which appeared to provide market beating results in the testing period, however the models trained outputted nearly only positive values, and thus using no threshold would likely in general result in negative returns, as seen during the validation period.

The highest variance in performance was seen with the OLS model, which makes sense considering it was the most overfit based on prediction errors. The least variance in portfolio performance was seen with the BiLSTM model, as it was never the worst performing portfolio across both periods and transaction cost scenarios. The BiLSTM model had the highest return in five of the six copula-based portfolios and two of six prediction weighted portfolios across both periods and transaction cost scenarios. While the BiLSTM model was the overall best performing model, it still performed worse than the market portfolios given 0.5% transaction costs during the validation period, and 1% transaction costs during the test period. As such it cannot be said to consistently either produce positive returns or even market outperforming returns, even without considering the elevated risk measures that accompanied its usage. Thus while NN models were able to produce higher returns than benchmarks in both periods under several transaction cost scenarios, this came at the cost of increased risk measures. And even given higher risk measures, no model consistently produced positive returns, or even higher returns compared to benchmarks.

The portfolio optimization used a CVaR of 0.1, as it was deemed the lowest possible value to achieve consistent market participation during the validation period. However during the test period it is noteworthy that due to lower risk during the period, a lower CVaR could have been used, which would have resulted in achieving similar returns with lower risk measures. Based on this, instead of a constant CVaR being utilized, optimal portfolio selection should have utilized a dynamic CVaR level. This was noted after the test results, and thus not used post hoc.

Compared to the risk naive approach of the prediction weighted portfolios, the copula-based CVaR-optimization consitently gave higher returns with lower risk measures across both periods given non-zero transaction costs. This indicates that the copula-based approach was able to better utilize the predictions in achieving higher returns, while also factoring in risk. However due to the relatively high CVaR used, market portfolios consistently had better risk measures compared to the copula-based portfolios.

7.4 Feature importance

Based on feature importance, the technical indicator selection strategy utilized was successful, as the permutation impact on prediction error was the largest for all models with technical indicators. However while the impact was the largest, the error was still moderately small, between 0.3% and 0.6%. This indicates that while the models rely most on technical indicators, there is a great amount of reliance on the whole array of variables and a great level of regularization.

The second most significant category of variables were the market sentiment variables, however again with relatively small errors added from permutation, between 0.1% and 0.3%. Interestingly the error added from permutation was the highest in the CNN-BiLSTM-Attention model for both technical indicators and market sentiment variables, which indicates that the attention layer likely highlighted the variables within the two categories more than architectures without an attention layer were able to.

All other categories of variables did not appear to significantly affect prediction accuracy, except for macroeconomic indicators, permuting of which actually improved the predictive accuracy of all models during the test period. A possible reason for this was the low frequency of updates in some of the variables and during permuting, the 10-day variable lags would likely be of different values than the current value, which was impossible to occur with actual data. A secondary reason was the change in interest rate behaviour post COVID, most of which was missing from the training period.

The rest of the categories of variables did not appear to impact permutation prediction accuracy significantly. Sector dummies did not significantly affect prediction accuracy, which indicates that the behaviour of stocks was similar across all sectors. This indicates that the choice of a model utilizing all stocks instead of dividing models by sector or company may be justified, especially after considering the increased amount of data available.

Permuting basic variables appeared to not significantly affect prediction error. This is extremely surprising, as basic variables, especially close-to-close returns, are one of the most commonly used variables in stock prediction (Kumbure et al., 2022, Shah et al. 2019). Considering that technical indicators, which are calculated based on these basic variables, are the most significant group of variables, one would expect basic variables to also be significant in prediction. This could indicate that the technical indicators helped the model extract information from the basic variables. The technical indicators also have the benefit of using data prior to the ten-day window, for example the 100-day SMA, without the need to use as many total lags. The technical indicators also have the benefit of dimensionality reduction, as some of the variables utilize more than one variable, for example the VWAP. Thus its possible that because of dimensionality reduction, information extraction, and increased lags, the models emphasized technical indicators rather than the underlying basic variables.

The linear regression results show that most variables, even when mean transformed variable-wise, have statistically significant linear coefficients on the model predictions. However most coefficients are relatively small, and as such the significance to predictions appears to be very small. The largest coefficients for the All ensemble, apart from the constant, are seen with index advances and declines, OVX change, 100-day SMA, and 14-day VWAP. While there is a loss of information from using means, the R-squared values of the models were relatively high, with CNN-BiLSTM having the lowest of 0.755, and the All ensemble having the highest of 0.872. This indicates that there was a great level of linearity in the models, as even with simplified variables the level of variance in the output explained by a linear transformation of the inputs was more than 75%.

8 Conclusion

The main research questions of the thesis were:

- How can a neural network model be effectively deployed to integrate multiple data sources for predicting stock returns?
- How can copula-based mean-CVaR optimization be deployed using neural network predictions for stock returns?

The secondary research question were:

- How do transaction costs impact the returns of an NN-copula portfolio?
- How do the returns of NN-copula portfolios compare to those of market and risk-naive portfolios?

In regards to how to effectively integrate multiple data sources for predicting stock returns, four NN models, a BiLSTM, CNN-BiLSTM, CNN-BiLSTM-Attention, and an ensemble which utilized predictions of all three were trained using eight categories of data: basic variables, technical indicators, candlestick indicators, agnostic fundamental mispricing, a Google Trends sentiment variable, market sentiment variables, macroeconomic indicators, and sectors dummies. All of the categories have been used in literature, either utilized independently, or with variables from one other category, but limited literature exists were such a significant cross-section of variables has been utilized. Based on prediction errors, there appeared no significant differences between the models based on how they utilized the data, nor in the prediction accuracies of the models out-of-sample periods of 2022 and 2023. Using a permutation feature importance analysis, all models similarly showed that the two categories of variables which affected prediction errors the most were the technical indicators and market sentiment variables. Permuting the other categories of variables did not seem to significantly affect prediction errors, except the macroeconomic indicators, permuting of which interestingly improved prediction accuracy across all models.

Based on portfolios constructed with the predictions only, none of the models were able to successfully outperform the market given different transaction cost scenarios or portfolio construction methods. The best model in terms portfolio performance was seen to be the CNN-BiLSTM-Attention model, as it was the worst performing portfolio in

only one of the six scenarios across both periods and transaction costs, while having the highest return in four of the six scenarios. However despite being the best of the NN models, it failed to give higher returns than at least one of the market portfolios (EW and VW) in five of the six scenarios, and had positive returns in only five of the six portfolio scenarios. While the NN models generally performed much better than the OLS model during the validation period, during the test period, the OLS model was seen to have similar performance to the NN models. Based on the failure of the NN prediction weighted models to consistently outperform the market and OLS prediction weighted portfolios, the models cannot be said to be reliable.

However the second major part of the portfolio selection model used alongside the NN models was the GARCH(1,1) standardized error copula and mean-CVaR optimization conducted using the NN predictions and simulated errors from said copula. Comparing the performance of prediction weighted portfolios and mean-CVaR portfolios showed that the portfolios constructed with the use of copulas consistently outperformed the prediction weighted portfolios in terms of returns and in the cases where transaction costs were taken into account, also in terms of risk measures. Interestingly, when the copula modelling and mean-CVaR portfolio selection was added, the most consistent model in terms of return performance across the two periods and three transaction cost scenarios changed, as the simplest of the models, BiLSTM was seen to perform the best. Despite the improvement in performance, the best model, BiLSTM, with the copula-based optimization failed to outperform both market portfolios in two of the six scenarios. On top of the lack of return outperformance, the NN GARCH-copula models consistently had worse risk measures compared to the market portfolios. Thus despite the improvement in portfolio performance compared the prediction weighted and market portfolios, the proposed model fails to consistently outperform the market benchmarks in terms of returns, and once risk measures are factored in, the performance of the model only comparatively worsens.

Transaction costs had a significant effect on the portfolio performances. The first obvious reason for this was that transaction costs lowered returns. However this effect was not consistently seen. During the validation period, with a transaction cost of 0.5%, the annualized return of the BiLSTM-GARCH-copula model was -19.5%, whereas with a 1% transaction cost the annualized return increased to 13.7%. During the test period, there was a negative relationship with transaction costs and annualized returns, however this not only related to the higher transaction costs. The reason for this was seen to be that the optimal prediction interval used was set to be $[\text{transaction cost}, \text{inf}]$, which from a intuitive perspective makes sense, as the prediction would be negative when it is lower the transaction cost. However, the range of predictions was seen to be non-constant due to the model relying highly on dispersion measures, which means there were long subperiods during both validation and test periods when either low or only a small number of predictions actually were positive after subtracting the transaction cost. This however was seen to not necessarily mean that the predictions under the given threshold were non-useful. This was indicated by the finding that during the validation period, the optimal interval appeared to somewhere in the intersection of $[0.5\%, \infty]$ and $[1\%, \infty]$, whereas during the test period the optimal interval appeared to be in the intersection $[0, \infty]$ and $[0.5\%, \infty]$. Based on these findings, the optimal prediction interval for portfolio selection in the model was variable and not necessarily connected to transaction costs.

The failure of the model brings attention to the limitations of the proposed method-

ology, that resulted in lack of performance. The limitations of the methodology are both significant and numerous.

The first limitation of the methodology comes from the variable selection process. One of the main purposes of the used methodology was to see how well NNs are able to utilize data from a multitude of sources. However, based on feature importance permutation analysis conducted on the models, only two categories of variables were seen to be the significant for model predictions, technical indicators and market sentiment variables. while the other six categories of variables, basic variables, candlestick patterns, agnostic fundamental mispricing, stock sentiment, and macroeconomic did not appear to significantly affect the predictions. This indicates that the models were not able to utilize the full scope of the data. There are two possible solutions to this. The first solution is to apply an input variable selection process, like an evolutionary algorithm on the whole variable set. With the current methodology, only technical indicators underwent this selection. Based on the finding that technical indicators were the most significant category of prediction and that it was the only category that underwent empirically guided selection, it may have been useful to apply this to the entire set of variables. The second possible solution would be to experiment with model architectures, in order to better extract information from the different categories of variables. These architectures could be for example separate modules that only use one category of the variables. This could help the model better extract information from the smaller sets of variables, as opposed to the the used approach where all data is available to the model at once.

The second limitation is also in the same realm of variable choice, as instead of using values that are proxies for intrinsic value, like price ratio, the variable chosen to represent fundamental data was a mispricing signal based on Bartram and Grinblatt's (2018) agnostic fundamental analysis. However the analysis failed (,due to possible reasons described in 6.1) and as such the model had no good descriptors of intrinsic value. As the purpose of the fundamental variables is to proxy intrinsic value, the use of agnostic fundamental analysis could still be useful for NN models, however steps required to make the mispricing signal significant in predicting returns should be taken. Alternatively simply using the underlying data or other proxies for intrinsic value, such as price ratios, instead of the pre-analysing of this data could allow the NN model to better take advantage of the information.

The third limitation of the NN-models were the hyperparameters. As an exhaustive search of the hyperparameter space is not computationally possible, a grid search guided by the author was conducted. During the hyperparameter optimization, it was noted that all of the model were extremely prone to overfitting, and as such the dropout rates utilized were relatively high, and only increased with model complexity. The used batch size was large in terms of number of data points, as each batch contained 10% of the data, which translated to roughly 50 000 data points. Smaller batch sizes were utilized with lower learning rates, however the control over overfitting was significantly harder. The reason for this may stem from the scope of the data and its processing, as the ratio of signal to noise in the data may have simply been too high for the models as is. There is also the near certainty that the optimal hyperparameter set was not found, either due to it being within the search range, although not a considered multiple, or possible due to it being outside the search range, as the use of normal batches sizes, ≤ 1000, would have required significantly lower learning rates and the computational time would have been significantly prolonged.

The fourth limitation was the way in which final outputs were computed. A simple stacking approach was utilized by taking the mean of the outputs of the underlying models. While this approach is simple, it may not optimally utilize the information given by the underlying predictors. Alternative approaches could be a meta-learner, which is a separately taught model to make optimal predictions based on the predictions of another model. An interesting and possibly useful approach to thr meta-learner would be to use the full set of stock predictions at time t, as the model itself was limited in its knowledge of the simultationsly made prediction, apart from sharing the variables of market sentiment and macroeconomic indicators. This approach could at the same time benefit from the increased dataset and training time as the original model, while still using temporal knowledge from other prediction made at the same time.

The fifth limitation was the set of possible pair-copulas used in the copula selection. The choice was made to use only one parameter copulas, meaning two parameter and nonparametric copulas were excluded. This was done due to the significant differences in computational time, as with the one parameter set of copulas the copula fitting took roughly two minutes, where as including two parameter or nonparametric copulas resulted in the selection taking over two hours. Although the actual time could have been much longer, as the execution was never ran to completion. Excluding two parameter copulas resulted in the exclusion of, among others, the T-distribution, one of the most common distributions used in financial literature. All copula computations were done in the Google Colaboratory environment with eight Intel Xeon 2.20 GHz CPUs. Based on the limited set of pair-copulas, a copula fitted with all possible pair-copulas could not have been worse than the one-parameter only copula utilized in the model.

The sixth limitation was the volatility modeling used. A simple GARCH(1,1) approach was utilized, however the use of neural networks has been seen to improve volatility prediction accuracy compared to ARCH models(e.g Bucci, 2020; Donaldson and Kamstra, 1997). Based on this, the second use of NNs also in the volatility phase could improve the performance of the model.

The seventh limitation was the lack of dynamic dependence used in the modelling. Stocks and related financial time-series have been to be dynamic in nature in terms of correlation structures (Case et al., 2012; Arai et al. 2015). Using predicted volatilities should capture some of this cross-dependence, however the relationship between dependencies and volatility is also dynamic in nature (Ramchand and Susmel, 1998). This is not captured by a single copula model. Thus a time-varying copula approach, such a regime switching copula could be utilized.

Based on these findings and limitations future work could be in regards to some of the following:

- Investigate input selection and neural network model architectures to optimize the use of multi-sourced data in stock prediction.
- Investigate the optimization of dynamic prediction interval and CVaR in the NN-copula framework.
- Investigate the use of NNs in volatility prediction in the NN-copula framework.
- Investigate the use of a meta-learner in the NN-copula framework.
- Investigate the use of a time-varying copula in the NN-copula framework.

The prediction of stock returns remains in large part illusive, despite advances in technology. The approach used, having multi-sourced inputs in a NN-GARCH-copula model did not result in consistent overperformance compared to market portfolios. While the GARCH-copula approach was shown to improve in portfolio selection given NN-based return prediction, the NNs did not give significant enough predictions to for the model to be reliable. Further research and methodology development is required to make the approach viable.

References

Aas, K., Berg, D. (2010). Modeling dependence between financial returns using pair-copula constructions. In Dependence modeling: Vine copula handbook (pp. 305-328).

Aas, K., Czado, C., Frigessi, A., Bakken, H. (2009). Pair-copula constructions of multiple dependence. Insurance: Mathematics and economics, 44(2), 182-198.

Acar, E. F., Czado, C., Lysy, M. (2019). Flexible dynamic vine copula models for multivariate time series data. Econometrics and Statistics, 12, 181-197.

Adrian, T., Franzoni, F. (2009). Learning about beta: Time-varying factor loadings, expected returns, and the conditional CAPM. Journal of Empirical Finance, 16(4), 537-556.

Adytia, D., Saepudin, D., Pudjaprasetya, S. R., Husrin, S., Sopaheluwakan, A. (2022). A deep learning approach for wave forecasting based on a spatially correlated wind feature, with a case study in the Java Sea, Indonesia. Fluids, 7(1), 39.

Albelwi, S., Mahmood, A. (2017). A framework for designing the architectures of deep convolutional neural networks. Entropy, 19(6), 242.

Arai, Y., Yoshikawa, T., Iyetomi, H. (2015). Dynamic stock correlation network. Procedia Computer Science, 60, 1826-1835.

Aroussi, R. (2024). https://github.com/ranaroussi/yfinance

Avramov, D., Chordia, T. (2006). Predicting stock returns. Journal of Financial Economics, 82(2), 387-415.

Avramov, D., Kosowski, R., Naik, N. Y., Teo, M. (2011). Hedge funds, managerial skill, and macroeconomic variables. Journal of Financial Economics, 99(3), 672-692.

Awokuse, T. O., Chopra, A., Bessler, D. A. (2009). Structural change and international stock market interdependence: Evidence from Asian emerging markets. Economic Modelling, 26(3), 549-559.

Bahrammirzaee, A. (2010). A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems. Neural Computing and Applications, 19(8), 1165-1195.

Baker, M., Wurgler, J. (2007). Investor sentiment in the stock market. Journal of economic perspectives, 21(2), 129-151.

Balvers, R. J., Wu, Y. (2006). Momentum and mean reversion across national equity markets. Journal of Empirical Finance, 13(1), 24-48.

Barnes, M. L., Hughes, A. T. W. (2002). A quantile regression analysis of the cross section of stock market returns.

Bartov, E., Radhakrishnan, S., Krinsky, I. (2000). Investor sophistication and patterns in stock returns after earnings announcements. The accounting review, 75(1), 43-63.

Bartram, S. M., Grinblatt, M. (2018). Agnostic fundamental analysis works. Journal of Financial Economics, 128(1), 125-147.

Batres-Estrada, B. (2015). Deep learning for multivariate financial time series.

Baur, D. G., Dimpfl, T., Jung, R. C. (2012). Stock return autocorrelations revisited: A quantile regression approach. Journal of Empirical Finance, 19(2), 254-265.

Bespoke. (2016). https://www.bespokepremium.com/interactive/posts/think-big-blog/historical-sp-500-sector-weightings

Bordalo, P., Gennaioli, N., Porta, R. L., Shleifer, A. (2019). Diagnostic expectations and stock returns. The Journal of Finance, 74(6), 2839-2874.

Brooks, C. (1998). Predicting stock index volatility: can market volume help?. Journal of forecasting, 17(1), 59-80.

Bucci, A. (2020). Realized volatility forecasting with neural networks. Journal of Financial Econometrics, 18(3), 502-531.

Buera, F. J., Kaboski, J. P. (2012). The rise of the service economy. American Economic Review, 102(6), 2540-2569.

Cakmakli, C., van Dijk, D. J. (2010). Getting the most out of macroeconomic information for predicting stock returns and volatility.

Case, B., Yang, Y., Yildirim, Y. (2012). Dynamic correlations among asset classes: REIT and stock returns. The Journal of Real Estate Finance and Economics, 44, 298-318.

Chen, G. M., Firth, M., Rui, O. M. (2001). The dynamic relation between stock returns, trading volume, and volatility. Financial Review, 36(3), 153-174.

Chollet, F. others, 2015. Keras. Available at: https://github.com/fchollet/keras.

Colliri, T., Ferreira, F. F. (2012, October). Stock prices assessment: proposal of a new index based on volume weighted historical prices. In 2012 Third Brazilian Workshop on Social Simulation (pp. 44-51). IEEE.

Cremers, M., Ferreira, M. A., Matos, P., Starks, L. (2016). Indexing and active fund management: International evidence. Journal of Financial Economics, 120(3), 539-560.

Cremers, K. M., Petajisto, A. (2009). How active is your fund manager? A new measure that predicts performance. The review of financial studies, 22(9), 3329-3365.

de Santana Correia, A., Colombini, E. L. (2022). Attention, please! A survey of neural attention models in deep learning. Artificial Intelligence Review, 55(8), 6037-6124.

Diggs, D. H., Povinelli, R. J. (2003). A temporal pattern approach for predicting weekly financial time series. In Artificial Neural Networks in Engineering (pp. 707-712).

Dissmann, J., Brechmann, E. C., Czado, C., Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. Computational Statistics Data Analysis, 59, 52-69.

Donaldson, R. G., Kamstra, M. (1997). An artificial neural network-GARCH model for international stock return volatility. Journal of Empirical Finance, 4(1), 17-46.

Elton, E. J., Gruber, M. J., Blake, C. R. (2012). Does mutual fund size matter? The relationship between size and performance. The Review of Asset Pricing Studies, 2(1), 31-55.

Fama, E. F., French, K. R. (2010). Luck versus skill in the cross-section of mutual fund returns. The journal of finance, 65(5), 1915-1947.

Federal Reserve Economic Data. (2024). https://fred.stlouisfed.org/

Fisher, A., Rudin, C., Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research, 20(177), 1-81.

Fradkov, A. L. (2020). Early history of machine learning. IFAC-PapersOnLine, 53(2), 1385-1390.

Frazzini, A., Israel, R., Moskowitz, T. J. (2018). Trading costs. Available at SSRN 3229719

Friede, G., Busch, T., Bassen, A. (2015). ESG and financial performance: aggregated evidence from more than 2000 empirical studies. Journal of sustainable finance investment, 5(4), 210-233.

Fu, T. C., Chung, F. L., Luk, R., Ng, C. M. (2007). Stock time series pattern matching: Template-based vs. rule-based approaches. Engineering Applications of Artificial Intelligence, 20(3), 347-364.

Gaio, L. E., Pimenta Júnior, T., Lima, F. G., Passos, I. C., Stefanelli, N. O. (2018). Value-at-risk performance in emerging and developed countries. International Journal of Managerial Finance, 14(5), 591-612.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., Suganthan, P. N. (2022). Ensemble deep learning: A review. Engineering Applications of Artificial Intelligence, 115, 105151.

Google Trends. (2024) trends.google.com

Grace, A. (2017). Can deep learning techniques improve the risk adjusted returns from enhanced indexing investment strategies.

Guo, Y., Fu, X., Shi, Y., Liu, M. (2018). Robust log-optimal strategy with reinforcement learning. arXiv preprint arXiv:1805.00205. Hansen, B. E. (2011). Threshold autoregression in economics. Statistics and its Interface, 4(2), 123-127

Haldurai, L., Madhubala, T., Rajalakshmi, R. (2016). A study on genetic algorithm and its applications. Int. J. Comput. Sci. Eng, 4(10), 139-143.

Han, Y., Yang, K., Zhou, G. (2013). A new anomaly: The cross-sectional profitability of technical analysis. Journal of Financial and Quantitative Analysis, 48(5), 1433-1461.

Hanauer, M. X., Kononova, M., Rapp, M. S. (2022). Boosting agnostic fundamental analysis: Using machine learning to identify mispricing in European stock markets. Finance Research Letters, 48, 102856.

Hardy, M. R. (2001). A regime-switching model of long-term stock returns. North American Actuarial Journal, 5(2), 41-53.

Hirshleifer, D., Lim, S. S., Teoh, S. H. (2011). Limited investor attention and stock market misreactions to accounting information. The Review of Asset Pricing Studies, 1(1), 35-73.

Hon-Snir, S., Kudryavtsev, A., Cohen, G. (2012). Stock market investors: Who is more rational, and who relies on intuition. International Journal of Economics and Finance, 4(5), 56-72.

Huang, D., Jiang, F., Tu, J., Zhou, G. (2015). Investor sentiment aligned: A powerful predictor of stock returns. The Review of Financial Studies, 28(3), 791-837.

Huang, M., Bao, Q., Zhang, Y., Feng, W. (2019). A hybrid algorithm for forecasting financial time series data based on DBSCAN and SVR. Information, 10(3), 103.

Investopedia. (2023). https://www.investopedia.com/articles/technical/112601.asp. Accessed on 28.01.2024.

Investopedia. (2023). https://www.investopedia.com/terms/f/fundamentalanalysis.asp. Accessed 31.01.2024.

Jensen, M. C. (1978). Some anomalous evidence regarding market efficiency. Journal of financial economics, 6(2/3), 95-101.

Joe, H. (1996). Families of m-variate distributions with given margins and m (m-1)/2 bivariate dependence parameters. Lecture notes-monograph series, 120-141.

Kenourgios, D., Samitas, A., Paltalidis, N. (2011). Financial crises and stock market contagion in a multivariate time-varying asymmetric framework. Journal of International Financial Markets, Institutions and Money, 21(1), 92-106.

Kon, S. J. (1984). Models of stock returns—a comparison. The Journal of Finance, 39(1), 147-165.

Kolm, P. N., Tütüncü, R., Fabozzi, F. J. (2014). 60 years of portfolio optimization: Practical challenges and current trends. European Journal of Operational Research, 234(2), 356-371.

Kothari, S. P., Lewellen, J., Warner, J. B. (2006). Stock returns, aggregate earnings surprises, and behavioral finance. Journal of Financial Economics, 79(3), 537-568.

Kumbure, M. M., Lohrmann, C., Luukka, P., Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. Expert Systems with Applications, 197, 116659.

Lateko, A. A., Yang, H. T., Huang, C. M., Aprillia, H., Hsu, C. Y., Zhong, J. L., Phng, N. H. (2021). Stacking ensemble method with the RNN meta-learner for short-term PV power forecasting. Energies, 14(16), 4733.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

Ledoit, O., Santa-Clara, P., Wolf, M. (2003). Flexible multivariate GARCH modeling with an application to international stock markets. Review of Economics and Statistics, 85(3), 735-747.

Lee, C. F., Rui, O. M. (2000). Does trading volume contain information to predict stock returns? Evidence from China's stock markets. Review of quantitative finance and accounting, 14, 341-360.

Leigh, W., Frohlich, C. J., Hornik, S., Purvis, R. L., Roberts, T. L. (2007). Trading with a stock chart heuristic. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(1), 93-104.

Lewellen, J., Nagel, S. (2006). The conditional CAPM does not explain asset-pricing anomalies. Journal of financial economics, 82(2), 289-314.

Li, L., Muwafak, B. M. (2021). Adoption of deep learning Markov model combined with copula function in portfolio risk measurement. Applied Mathematics and Nonlinear Sciences, 7(1), 901-916.

Liang, Z., Chen, H., Zhu, J., Jiang, K., Li, Y. (2018). Adversarial deep reinforcement

learning in portfolio management. arXiv preprint arXiv:1808.09940.

Lin, C. Y. (2014). Does active management work? Evidence from equity sector funds. Financial Services Review, 23(3).

Liu, N., Chen, C. B., Kumara, S. (2019). Semi-supervised learning algorithm for identifying high-priority drug–drug interactions through adverse event reports. IEEE journal of biomedical and health informatics, 24(1), 57-68.

Liu, S., Zhang, C., Ma, J. (2017). CNN-LSTM neural network model for quantitative strategy analysis in stock markets. In Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24 (pp. 198-206). Springer International Publishing.

Low, R. K. Y., Alcock, J., Faff, R., Brailsford, T. (2018). Canonical vine copulas in the context of modern portfolio management: Are they worth it?. Asymmetric Dependence in Finance: Diversification, Correlation and Portfolio Management in Market Downturns, 263-289.

Madhavan, A. (2002). VWAP strategies. Trading, 1, 32-39.

Markowitz, H.M. (1959) Portfolio Selection, Efficient Diversification of Investments. Yale University Press, Newhaven and London.

Markowitz, H. M. (1999). The Early History of Portfolio Theory: 1600-1960. Financial Analysts Journal, 55(4), 5–16. http://www.jstor.org/stable/4480178

Markowitz, H. M. (2010). Portfolio theory: as I still see it. Annu. Rev. Financ. Econ., 2(1), 1-23.

Marquering, W., Verbeek, M. (2004). The economic value of predicting stock index returns and volatility. Journal of Financial and Quantitative Analysis, 39(2), 407-429.

Martin, R., (2021). PyPortfolioOpt: portfolio optimization in Python. Journal of Open Source Software, 6(61), 3066, https://doi.org/10.21105/joss.03066

McLean, R. D., Pontiff, J. (2016). Does academic research destroy stock return predictability?. The Journal of Finance, 71(1), 5-32.

Mendes-Moreira, J., Soares, C., Jorge, A. M., Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. Acm computing surveys (csur), 45(1), 1-40.

Menkveld, A. J. (2016). The economics of high-frequency trading: Taking stock. Annual Review of Financial Economics, 8, 1-24.

Nagler, T., Schellhase, C., Czado, C. (2017). Nonparametric estimation of simplified vine copula models: comparison of methods. Dependence Modeling, 5(1), 99-120.

Nagler, T., Vatter, T. (2023). pycopulib: High Performance Algorithms for Vine Copula Modeling. https://pypi.org/project/pyvinecopulib/

Navas, R. D., Gama, A. P. M., Bentes, S. R. (2023). The relevance of using accounting fundamentals in the Euronext 100 index. Revista Brasileira de Gestão de Negócios, 25, 456-479.

Nayak, R., Te Braak, P. (2007). Temporal pattern matching for the prediction of stock prices. In Proceedings of the 2nd International Workshop on Integrating Artificial Intelligence and Data Mining (AIDM 2007) (pp. 95-104). Australian Computer Society.

Ogbuabor, G., Ugwoke, F. N. (2018). Clustering algorithm for a healthcare dataset using silhouette score value. Int. J. Comput. Sci. Inf. Technol, 10(2), 27-37.

Ozbayoglu, A. M., Gudelek, M. U., Sezer, O. B. (2020). Deep learning for financial applications: A survey. Applied Soft Computing, 93, 106384.

Pahwa, N., Khalfay, N., Soni, V., Vora, D. (2017). Stock prediction using machine learning a review paper. International Journal of Computer Applications, 163(5), 36-43.

Park, H., Sim, M. K., Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. Expert Systems with Applications, 158, 113573.

Patatoukas, P. N. (2012). Customer-base concentration: Implications for firm performance and capital markets: 2011 American accounting association competitive manuscript award winner. The accounting review, 87(2), 363-392.

Patton, A. J. (2012). A review of copula models for economic time series. Journal of Multivariate Analysis, 110, 4-18.

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

Peng, Y., Albuquerque, P. H. M., Kimura, H., Saavedra, C. A. P. B. (2021). Feature selection and deep neural networks for stock price direction forecasting using technical analysis indicators. Machine Learning with Applications, 5, 100060.

Polygon.io. (2024). polygon.io

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., Amaratunga, G. (2014, December). Ensemble deep learning for regression and time series forecasting. In 2014 IEEE symposium on computational intelligence in ensemble learning (CIEL) (pp. 1-6). IEEE.

Qiu, J., Wang, B., Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. PloS one, 15(1), e0227222.

Ramchand, L., Susmel, R. (1998). Volatility and cross correlation across major stock markets. Journal of Empirical Finance, 5(4), 397-416.

Rapach, D. E., Wohar, M. E., Rangvid, J. (2005). Macro variables and international stock return predictability. International journal of forecasting, 21(1), 137-166.

Rapach, D., Zhou, G. (2013). Forecasting stock returns. In Handbook of economic forecasting (Vol. 2, pp. 328-383). Elsevier.

Ren, R., Wu, D. D., Liu, T. (2018). Forecasting stock market movement direction using sentiment analysis and support vector machine. IEEE Systems Journal, 13(1), 760-770.

Schmeling, M. (2009). Investor sentiment and stock returns: Some international evidence. Journal of empirical finance, 16(3), 394-408.

Schuster, M., Paliwal, K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing, 45(11), 2673-2681.

Seabold, S. Perktold, J., 2010. statsmodels: Econometric and statistical modeling with python. In 9th Python in Science Conference.

Sezer, O. B., Gudelek, M. U., Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing, 90, 106181.

Shah, D., Isah, H., Zulkernine, F. (2019). Stock market analysis: A review and taxonomy of prediction techniques. International Journal of Financial Studies, 7(2), 26.

Shah, D., Shah, B., Sawarkar, H., Raja, V., Godbole, A. (2023, December). Proposing Investments Based on Fundamental Analysis. In 2023 Global Conference on Information Technologies and Communications (GCITC) (pp. 1-10). IEEE.

Shin, Y., Ghosh, J. (1991, July). The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. In IJCNN-91-Seattle international joint conference on neural networks (Vol. 1, pp. 13-18). IEEE.

Shynkevich, Y., McGinnity, T. M., Coleman, S. A., Belatreche, A., Li, Y. (2017). Forecasting price movements using technical indicators: Investigating the impact of varying input window length. Neurocomputing, 264, 71-88.

Siami-Namini, S., Tavakoli, N., Namin, A. S. (2019). A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. arXiv preprint arXiv:1911.09512.

Sklar, M. (1959). Fonctions de répartition à n dimensions et leurs marges. In Annales de l'ISUP (Vol. 8, No. 3, pp. 229-231).

Strader, T. J., Rozycki, J. J., Root, T. H., Huang, Y. H. J. (2020). Machine learning

stock market prediction studies: review and research directions. Journal of International Technology and Information Management, 28(4), 63-83.

Takeuchi, L., Lee, Y. Y. A. (2013). Applying deep learning to enhance momentum trading strategies in stocks. In Technical Report. Stanford, CA, USA: Stanford University.

Ta-lib. (2024) ta-lib.org. Accessed on 28.01.2024.

Tetlock, P. C. (2007). Giving content to investor sentiment: The role of media in the stock market. The Journal of finance, 62(3), 1139-1168.

Timmermann, A., Granger, C. W. (2004). Efficient market hypothesis and forecasting. International Journal of forecasting, 20(1), 15-27.

Tsai, C. F., Lin, Y. C., Yen, D. C., Chen, Y. M. (2011). Predicting stock returns by classifier ensembles. Applied Soft Computing, 11(2), 2452-2459.

Tu, J. (2010). Is regime switching in stock returns important in portfolio decisions?. Management Science, 56(7), 1198-1215.

Uryasev, S., Rockafellar, R. T. (2001). Conditional value-at-risk: optimization approach. Stochastic optimization: algorithms and applications, 411-435.

Yahoo Finance (2024). https://finance.yahoo.com/

Yan, X., Zheng, L. (2017). Fundamental analysis and the cross-section of stock returns: A data-mining approach. The Review of Financial Studies, 30(4), 1382-1423.

Yang, D., Zhang, Q. (2000). Drift-independent volatility estimation based on high, low, open, and close prices. The Journal of Business, 73(3), 477-492.

Yu, X. (2014). Analysis of new sentiment and its application to finance (Doctoral dissertation).

Yu, J., Le, T., Chang, K. C., Guharay, S. (2019, July). Fusing Economic Indicators for Portfolio Optimization-A Simulation-Based Approach. In 2019 22th International Conference on Information Fusion (FUSION) (pp. 1-8). IEEE.

Yu, J., Chang, K. C. (2020). Neural network predictive modeling on dynamic portfolio management—a simulation-based portfolio optimization approach. Journal of Risk and Financial Management, 13(11), 285.

Verma, R. K., Bansal, R. (2021). Impact of macroeconomic variables on the performance of stock exchange: a systematic review. International Journal of Emerging Markets, 16(7), 1291-1329.

Zagoruyko, S., Komodakis, N. (2016). Paying more attention to attention: Improving

the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

Zhang, J., Ye, L., Lai, Y. (2023). Stock Price Prediction Using CNN-BiLSTM-Attention Model. Mathematics, 11(9), 1985.

Zhao, H., Jia, J., Koltun, V. (2020). Exploring self-attention for image recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10076-10085).

Zhao, Y., Stasinakis, C., Sermpinis, G., Shi, Y. (2018). Neural network copula portfolio optimization for exchange traded funds. Quantitative Finance, 18(5), 761-775.

Zhou, B. (2019). Deep learning and the cross-section of stock returns: Neural networks combining price and fundamental information. Available at SSRN 3179281.

Appendix

Appendix 1 - List of stocks included

The list of stocks used:

- Agilent Technologies, Inc.
- Apple Inc.
- AbbVie Inc.
- Abbott Laboratories
- Accenture plc
- Adobe Inc.
- Analog Devices, Inc.
- Archer-Daniels-Midland Company
- Automatic Data Processing, Inc.
- Autodesk, Inc.
- Ameren Corporation
- American Electric Power Company, Inc.
- The AES Corporation
- Aflac Incorporated
- American International Group, Inc.
- Assurant, Inc.
- Akamai Technologies, Inc.
- The Allstate Corporation
- Allegion plc
- Applied Materials, Inc.
- AMETEK, Inc.
- Amgen Inc.
- Ameriprise Financial, Inc.
- American Tower Corporation
- Amazon.com, Inc.
- Aon plc
- APA Corporation
- Air Products and Chemicals, Inc.
- Amphenol Corporation
- Aptiv PLC
- AvalonBay Communities, Inc.
- Broadcom Inc.
- Avery Dennison Corporation
- American Express Company
- AutoZone, Inc.
- The Boeing Company
- Bank of America Corporation
- Baxter International Inc.
- Best Buy Co., Inc.
- Becton, Dickinson and Company
- Franklin Resources, Inc.
- Brown-Forman Corporation
- Biogen Inc.
- The Bank of New York Mellon Corporation
- Booking Holdings Inc.
- BlackRock, Inc.
- Bristol-Myers Squibb Company
- Berkshire Hathaway Inc.
- Boston Scientific Corporation
- BorgWarner Inc.
- Boston Properties, Inc.
- Citigroup Inc.
- Conagra Brands, Inc.
- Cardinal Health, Inc.
- Caterpillar Inc.
- Chubb Limited
- CBRE Group, Inc.
- Crown Castle Inc.
- Carnival Corporation plc
- CF Industries Holdings, Inc.
- C.H. Robinson Worldwide, Inc.
- The Cigna Group
- Cincinnati Financial Corporation
- Colgate-Palmolive Company

- The Clorox Company
- Comerica Incorporated
- Comcast Corporation
- CME Group Inc.
- Chipotle Mexican Grill, Inc.
- Cummins Inc.
- CMS Energy Corporation
- CenterPoint Energy, Inc.
- Capital One Financial Corporation
- ConocoPhillips
- Costco Wholesale Corporation
- Campbell Soup Company
- Salesforce, Inc.
- Cisco Systems, Inc.
- CSX Corporation
- Cintas Corporation
- Cognizant Technology Solutions Corporation
- CVS Health Corporation
- Chevron Corporation
- Dominion Energy, Inc.
- Delta Air Lines, Inc.
- DuPont de Nemours, Inc.
- Deere Company
- Discover Financial Services
- Dollar General Corporation
- Quest Diagnostics Incorporated
- D.R. Horton, Inc.
- Danaher Corporation
- The Walt Disney Company
- Dollar Tree, Inc.
- Dover Corporation
- Darden Restaurants, Inc.
- DTE Energy Company
- Duke Energy Corporation
- DaVita Inc.
- Devon Energy Corporation
- Electronic Arts Inc.
- eBay Inc.
- Ecolab Inc.
- Consolidated Edison, Inc.
- Equifax Inc.
- Edison International
- The Estée Lauder Companies Inc.
- Eastman Chemical Company
- Emerson Electric Co.
- EOG Resources, Inc.
- Equity Residential
- EQT Corporation
- Eversource Energy
- Essex Property Trust, Inc.
- Eaton Corporation plc
- Entergy Corporation
- Edwards Lifesciences Corporation
- Exelon Corporation
- Expeditors International of Washington, Inc.
- Expedia Group, Inc.
- Ford Motor Company
- Fastenal Company
- Freeport-McMoRan Inc.
- FedEx Corporation
- FirstEnergy Corp.
- F5, Inc.
- Fidelity National Information Services, Inc.
- Fifth Third Bancorp
- FMC Corporation
- First Solar, Inc.
- General Dynamics Corporation
- General Electric Company
- Gilead Sciences, Inc.
- General Mills, Inc.
- Corning Incorporated
- General Motors Company
- Alphabet Inc.
- Alphabet Inc.
- Genuine Parts Company

- Garmin Ltd.
- The Goldman Sachs Group, Inc.
- W.W. Grainger, Inc.
- Halliburton Company
- Hasbro, Inc.
- Huntington Bancshares Incorporated
- The Home Depot, Inc.
- Hess Corporation
- The Hartford Financial Services Group, Inc.
- Honeywell International Inc.
- HP Inc.
- Hormel Foods Corporation
- Host Hotels Resorts, Inc.
- The Hershey Company
- Humana Inc.
- International Business Machines Corporation
- Intercontinental Exchange, Inc.
- International Flavors Fragrances Inc.
- Intel Corporation
- Intuit Inc.
- International Paper Company
- The Interpublic Group of Companies, Inc.
- Iron Mountain Incorporated
- Intuitive Surgical, Inc.
- Illinois Tool Works Inc.
- Invesco Ltd.
- Johnson Controls International plc
- Johnson Johnson
- Juniper Networks, Inc.
- JPMorgan Chase Co.
- Kellanova
- Keurig Dr Pepper Inc.
- KeyCorp
- Kimco Realty Corporation
- KLA Corporation
- Kimberly-Clark Corporation
- Kinder Morgan, Inc.
- CarMax, Inc.
- The Coca-Cola Company
- The Kroger Co.
- Loews Corporation
- Lennar Corporation
- Laboratory Corporation of America Holdings
- Eli Lilly and Company
- Lockheed Martin Corporation
- Lowe's Companies, Inc.
- Lam Research Corporation
- Southwest Airlines Co.
- LyondellBasell Industries N.V.
- Mastercard Incorporated
- Marriott International, Inc.
- Masco Corporation
- McDonald's Corporation
- Microchip Technology Incorporated
- McKesson Corporation
- Moody's Corporation
- Mondelez International, Inc.
- Medtronic plc
- MetLife, Inc.
- Mohawk Industries, Inc.
- McCormick Company, Incorporated
- Martin Marietta Materials, Inc.
- Marsh McLennan Companies, Inc.
- 3M Company
- Monster Beverage Corporation
- Altria Group, Inc.
- The Mosaic Company
- Marathon Petroleum Corporation
- Merck Co., Inc.
- Marathon Oil Corporation
- Morgan Stanley
- Microsoft Corporation
- Motorola Solutions, Inc.
- MT Bank Corporation

- Micron Technology, Inc.
- Nasdaq, Inc.
- NextEra Energy, Inc.
- Newmont Corporation
- Netflix, Inc.
- NiSource Inc.
- NIKE, Inc.
- Northrop Grumman Corporation
- NRG Energy, Inc.
- Norfolk Southern Corporation
- NetApp, Inc.
- Northern Trust Corporation
- Nucor Corporation
- NVIDIA Corporation
- News Corporation
- ONEOK, Inc.
- Omnicom Group Inc.
- Oracle Corporation
- O'Reilly Automotive, Inc.
- Occidental Petroleum Corporation
- Paychex, Inc.
- PACCAR Inc
- PGE Corporation
- Public Service Enterprise Group Incorporated
- PepsiCo, Inc.
- Pfizer Inc.
- Principal Financial Group, Inc.
- The Procter Gamble Company
- The Progressive Corporation
- Parker-Hannifin Corporation
- PulteGroup, Inc.
- Prologis, Inc.
- Philip Morris International Inc.
- The PNC Financial Services Group, Inc.
- Pentair plc
- Pinnacle West Capital Corporation
- PPG Industries, Inc.
- PPL Corporation
- Prudential Financial, Inc.
- Public Storage
- Phillips 66
- Quanta Services, Inc.
- Pioneer Natural Resources Company
- QUALCOMM Incorporated
- Royal Caribbean Cruises Ltd.
- Regeneron Pharmaceuticals, Inc.
- Regions Financial Corporation
- Robert Half Inc.
- Ralph Lauren Corporation
- Rockwell Automation, Inc.
- Roper Technologies, Inc.
- Ross Stores, Inc.
- Republic Services, Inc.
- Starbucks Corporation
- The Charles Schwab Corporation
- The Sherwin-Williams Company
- The J. M. Smucker Company
- Schlumberger Limited
- Snap-on Incorporated
- The Southern Company
- Simon Property Group, Inc.
- SP Global Inc.
- Sempra
- State Street Corporation
- Seagate Technology Holdings plc
- Constellation Brands, Inc.
- Stanley Black Decker, Inc.
- Stryker Corporation
- Sysco Corporation
- ATT Inc.
- Molson Coors Beverage Company
- TE Connectivity Ltd.

- Target Corporation
- The TJX Companies, Inc.
- Thermo Fisher Scientific Inc.
- Tapestry, Inc.
- T. Rowe Price Group, Inc.
- The Travelers Companies, Inc.
- Tractor Supply Company
- Tyson Foods, Inc.
- Texas Instruments Incorporated
- Textron Inc.
- Universal Health Services, Inc.
- UnitedHealth Group Incorporated
- Union Pacific Corporation
- United Parcel Service, Inc.
- United Rentals, Inc.
- U.S. Bancorp
- Visa Inc.
- V.F. Corporation
- Valero Energy Corporation
- Vulcan Materials Company
- VeriSign, Inc.
- Vertex Pharmaceuticals Incorporated
- Ventas, Inc.
- Verizon Communications Inc.
- Waters Corporation
- Walgreens Boots Alliance, Inc.
- Western Digital Corporation
- WEC Energy Group, Inc.
- Welltower Inc.
- Wells Fargo Company
- Whirlpool Corporation
- Waste Management, Inc.
- The Williams Companies, Inc.
- Walmart Inc.
- Weyerhaeuser Company
- Wynn Resorts, Limited
- Xcel Energy Inc.
- Exxon Mobil Corporation
- DENTSPLY SIRONA Inc.
- Xylem Inc.
- Yum! Brands, Inc.
- Zimmer Biomet Holdings, Inc.
- Zions Bancorporation, National Association
- Zoetis Inc.

Appendix 2 - List of Ta-lib technical indicators

List of Ta-lib technical indicators:

- Chaikin A/D Line
- Chaikin A/D Oscillator
- Average Directional Movement Index
- Average Directional Movement Index Rating
- Absolute Price Oscillator
- Aroon
- Aroon Oscillator
- Average True Range
- Average Price
- Bollinger Bands
- Beta
- Balance Of Power
- Commodity Channel Index
- Chande Momentum Oscillator
- Pearson's Correlation Coefficient ®
- Double Exponential Moving Average
- Directional Movement Index
- Exponential Moving Average

- Hilbert Transform - Dominant Cycle Period
- Hilbert Transform - Dominant Cycle Phase
- Hilbert Transform - Phasor Components
- Hilbert Transform - SineWave
- Hilbert Transform - Instantaneous Trendline
- Hilbert Transform - Trend vs Cycle Mode
- Kaufman Adaptive Moving Average
- Linear Regression
- Linear Regression Angle
- Linear Regression Intercept
- Linear Regression Slope
- All Moving Average
- Moving Average Convergence/Divergence
- MACD with controllable MA type
- Moving Average Convergence/Divergence Fix 12/26
- MESA Adaptive Moving Average
- Highest value over a specified period
- Index of highest value over a specified period
- Median Price
- Money Flow Index
- MidPoint over period
- Midpoint Price over period
- Lowest value over a specified period
- Index of lowest value over a specified period
- Minus Directional Indicator
- Minus Directional Movement
- Momentum
- Normalized Average True Range
- On Balance Volume
- Plus Directional Indicator
- Plus Directional Movement
- Percentage Price Oscillator
- Rate of change : ((price/prevPrice)-1)*100
- Rate of change Percentage: (price-prevPrice)/prevPrice
- Rate of change ratio: (price/prevPrice)
- Rate of change ratio 100 scale: (price/prevPrice)*100
- Relative Strength Index
- Parabolic SAR
- Parabolic SAR - Extended
- Simple Moving Average
- Standard Deviation
- Stochastic
- Stochastic Fast
- Stochastic Relative Strength Index
- Summation
- Triple Exponential Moving Average (T3)
- Triple Exponential Moving Average
- True Range
- Triangular Moving Average
- 1-day Rate-Of-Change (ROC) of a Triple Smooth EMA
- Time Series Forecast
- Typical Price
- Ultimate Oscillator
- Variance
- Weighted Close Price
- Williams'
- Weighted Moving Average

Appendix 3 - List of Ta-lib candlestick patterns

List of Ta-lib candlestick patterns:

- To Crows
- Three Black Crows
- Three Inside Up/-Down
- Three Outside Up/Down
- Three Stars In The South
- Three Advancing White Soldiers
- Abandoned Baby
- Advance Block
- Belt-hold
- Breakaway
- Closing Marubozu
- Concealing Baby Swallow
- Counterattack
- Dark Cloud Cover
- Doji
- Doji Star
- Dragonfly Doji
- Engulfing Pattern
- Evening Doji Star
- Evening Star
- Up/Down-gap side-by-side white lines
- Gravestone Doji
- Hammer
- Hanging Man
- Harami Pattern
- Harami Cross Pattern
- High-Wave Candle
- Hikkake Pattern
- Modified Hikkake Pattern
- Homing Pigeon
- Identical Three Crows
- In-Neck Pattern
- Inverted Hammer
- Kicking
- Kicking - bull/bear determined by the longer marubozu
- Ladder Bottom
- Long Legged Doji
- Long Line Candle
- Marubozu
- Matching Low
- Mat Hold
- Morning Doji Star
- Morning Star
- On-Neck Pattern
- Piercing Pattern
- Rickshaw Man
- Rising/Falling Three Methods
- Separating Lines
- Shooting Star
- Short Line Candle
- Spinning Top
- Stalled Pattern
- Stick Sandwich
- Takuri (Dragonfly Doji with very long lower shadow)
- Tasuki Gap
- Thrusting Pattern
- Tristar Pattern
- Unique 3 River
- Upside Gap Two Crows
- Upside/Downside Gap Three Methods

Appendix 4 - Portfolio figures

Portfolio figures:

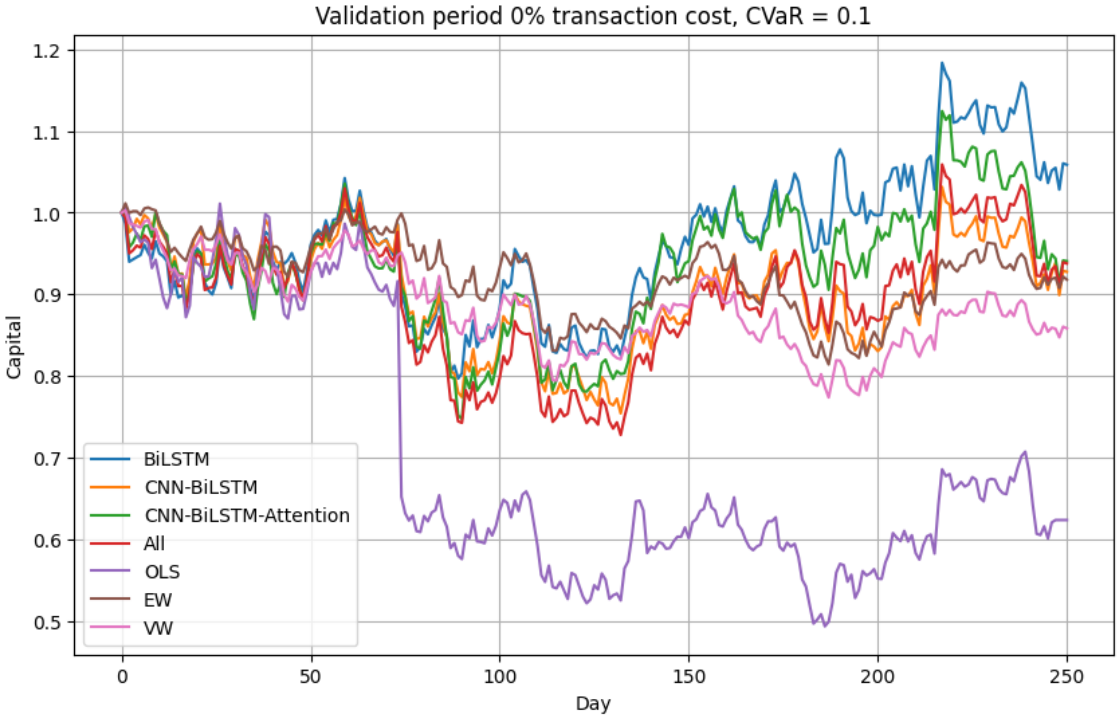

Figure 14: Validation period 0% transaction cost, $\text{CVaR}_{0.05} = 0.1$

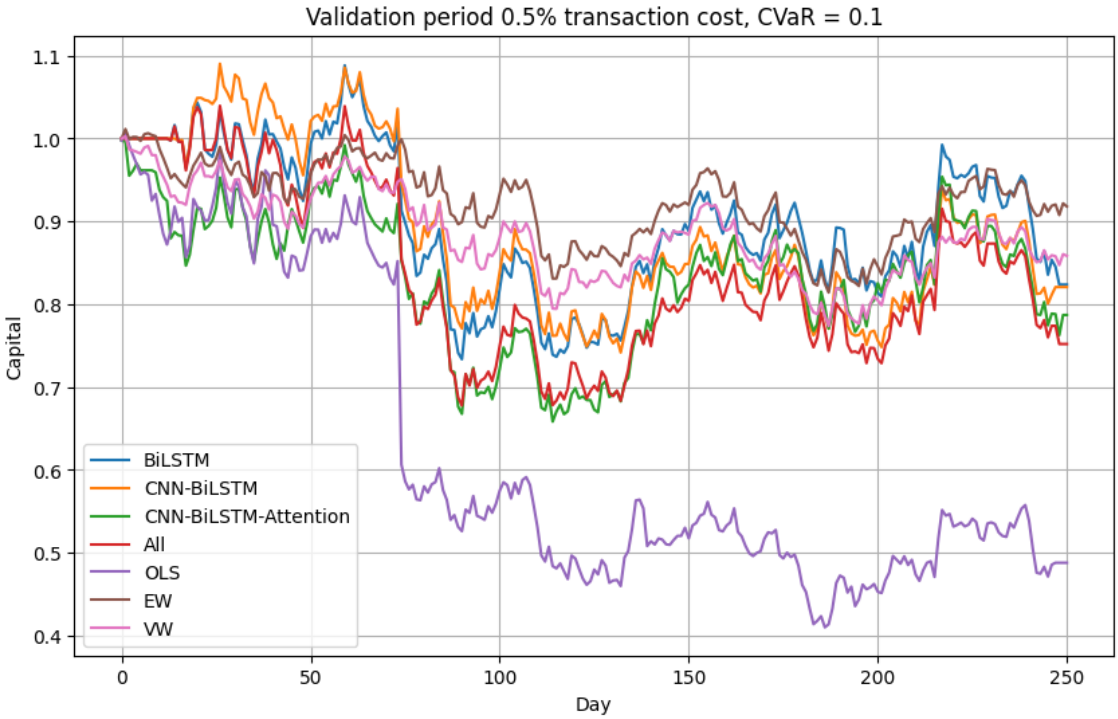

Figure 15: Validation period 0.5% transaction cost, $\text{CVaR}_{0.05} = 0.1$

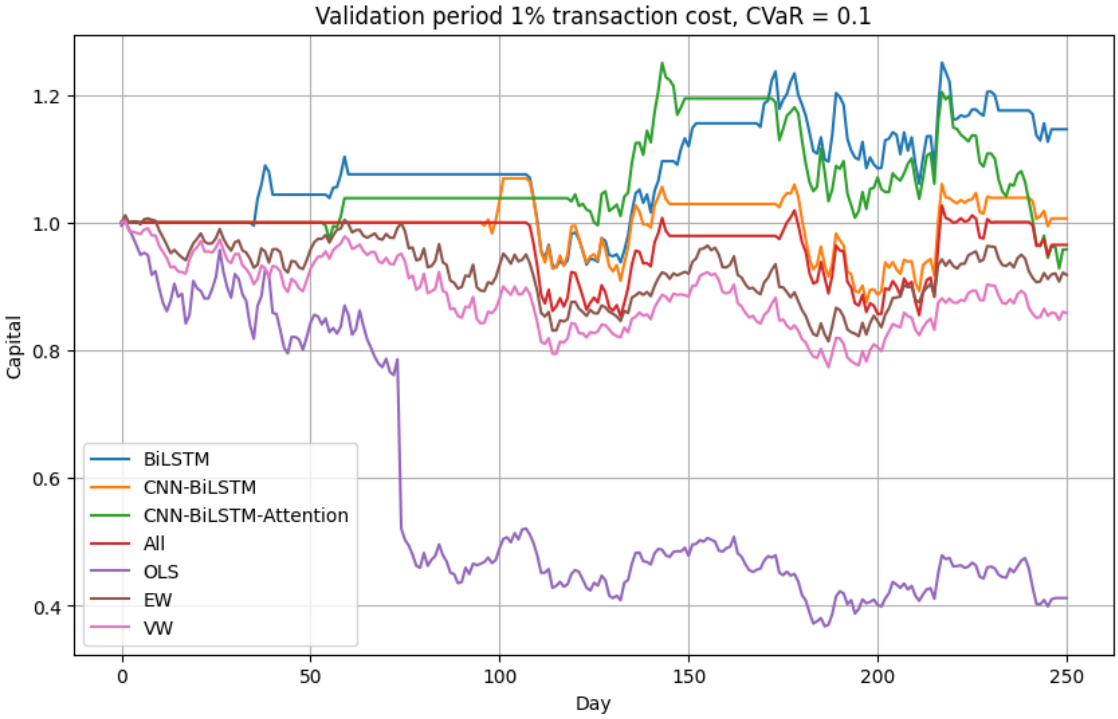

Figure 16: Validation period 1% transaction cost, $\text{CVaR}_{0.05} = 0.1$

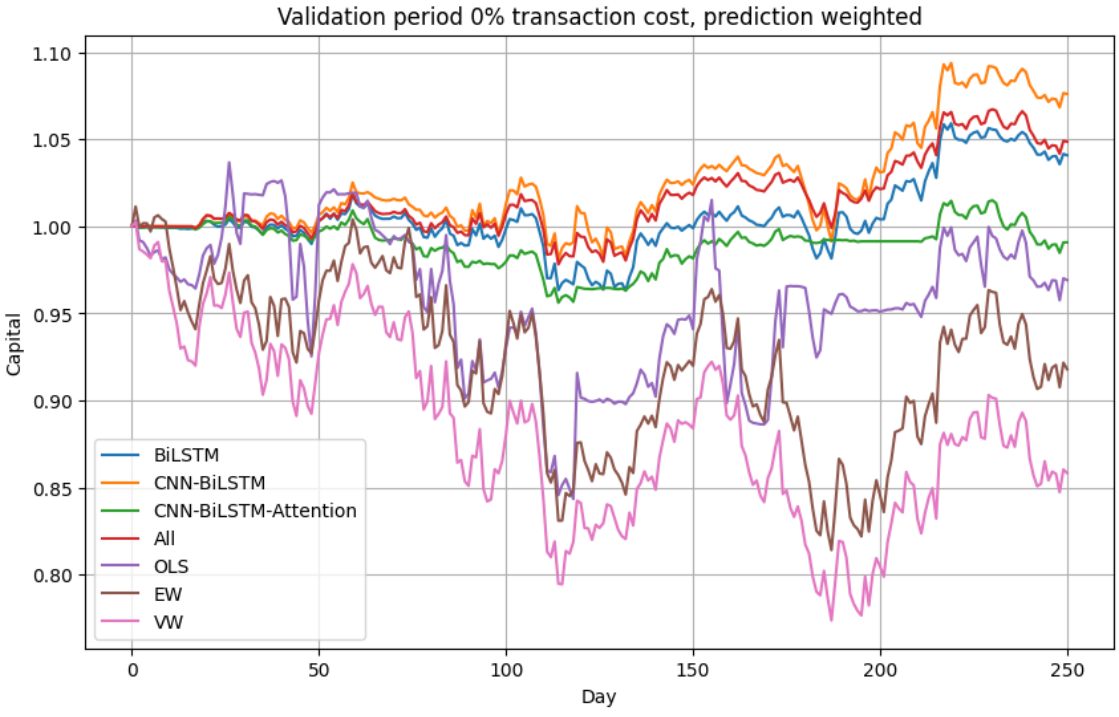

Figure 17: Validation period 0% transaction cost, prediction weighted

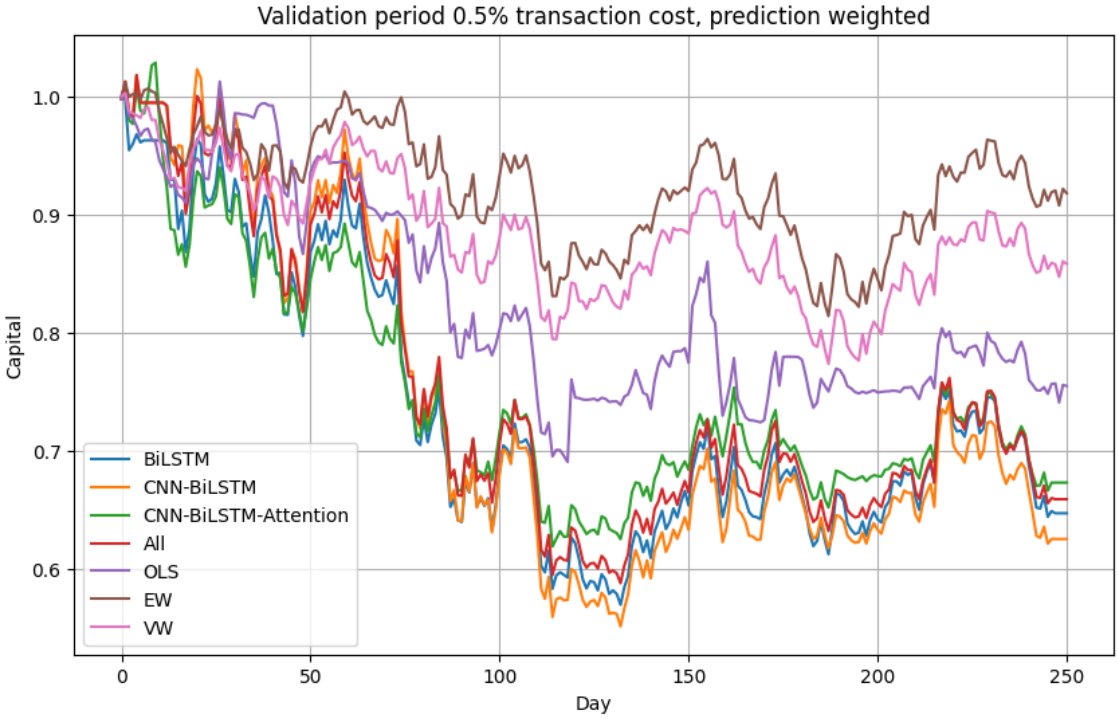

Figure 18: Validation period 0.5% transaction cost, prediction weighted

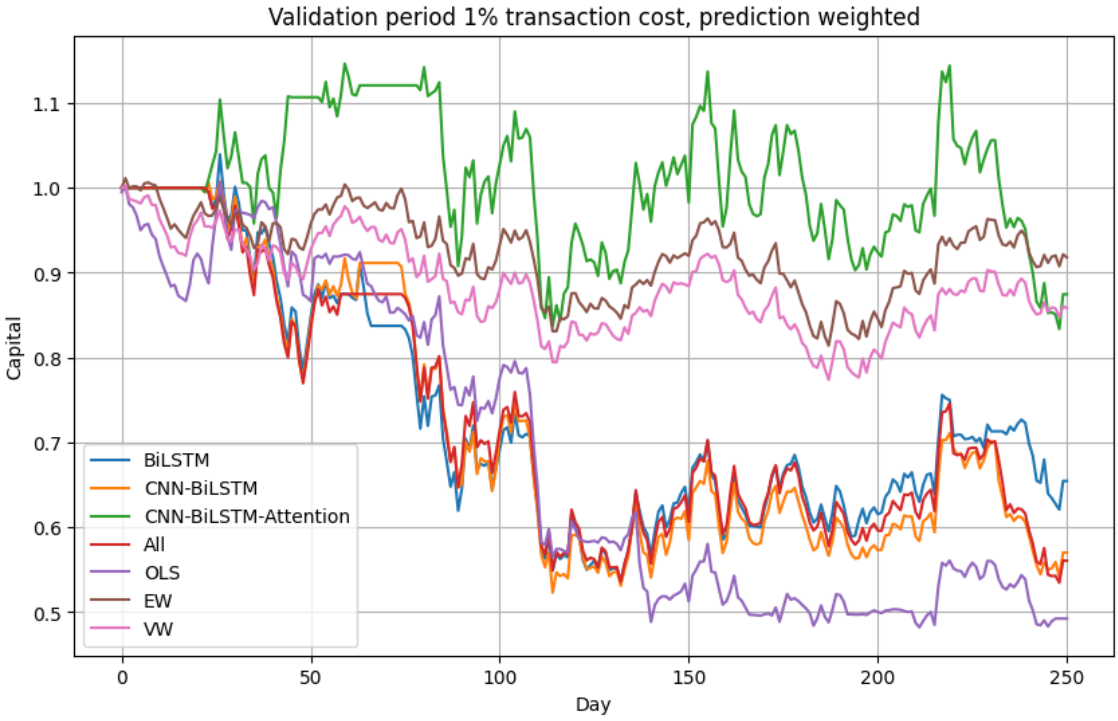

Figure 19: Validation period 1% transaction cost, prediction weighted

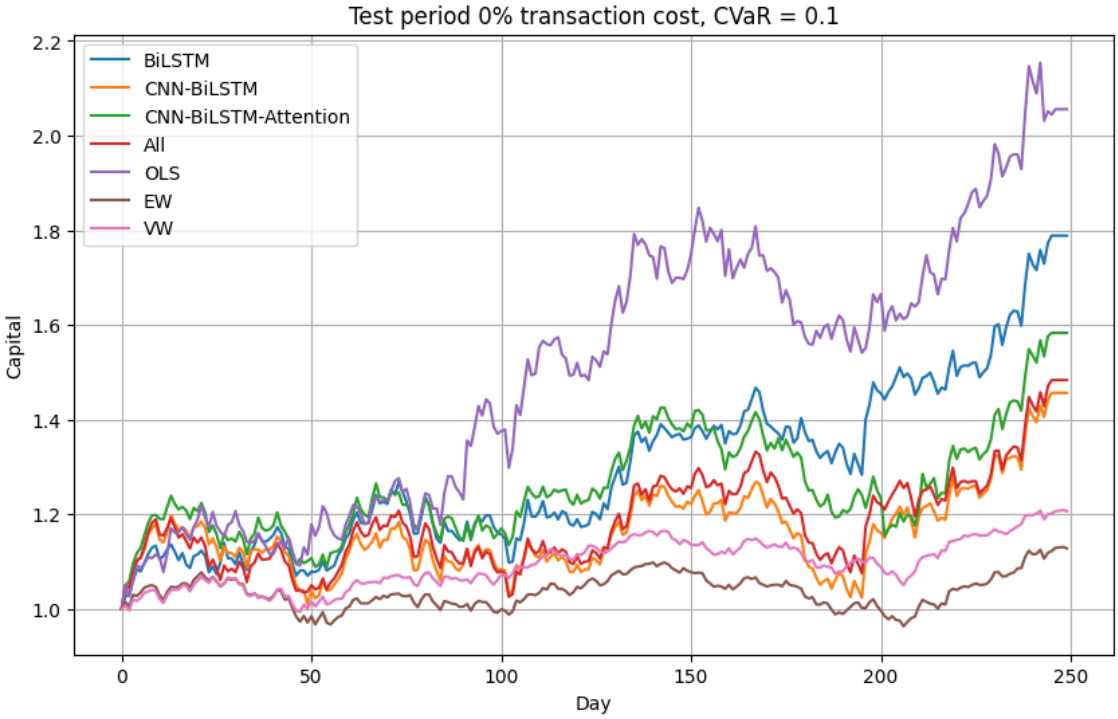

Figure 20: Test period 0% transaction cost, $\text{CVaR}_{0.05} = 0.1$

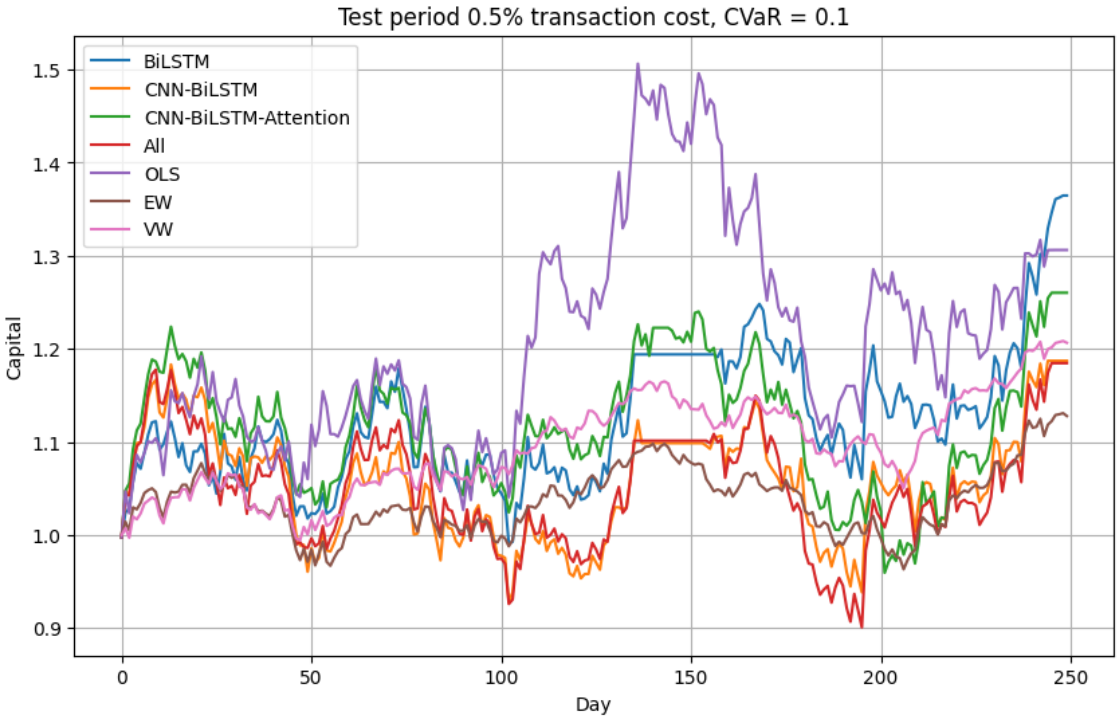

Figure 21: Test period 0.5% transaction cost, $\text{CVaR}_{0.05} = 0.1$

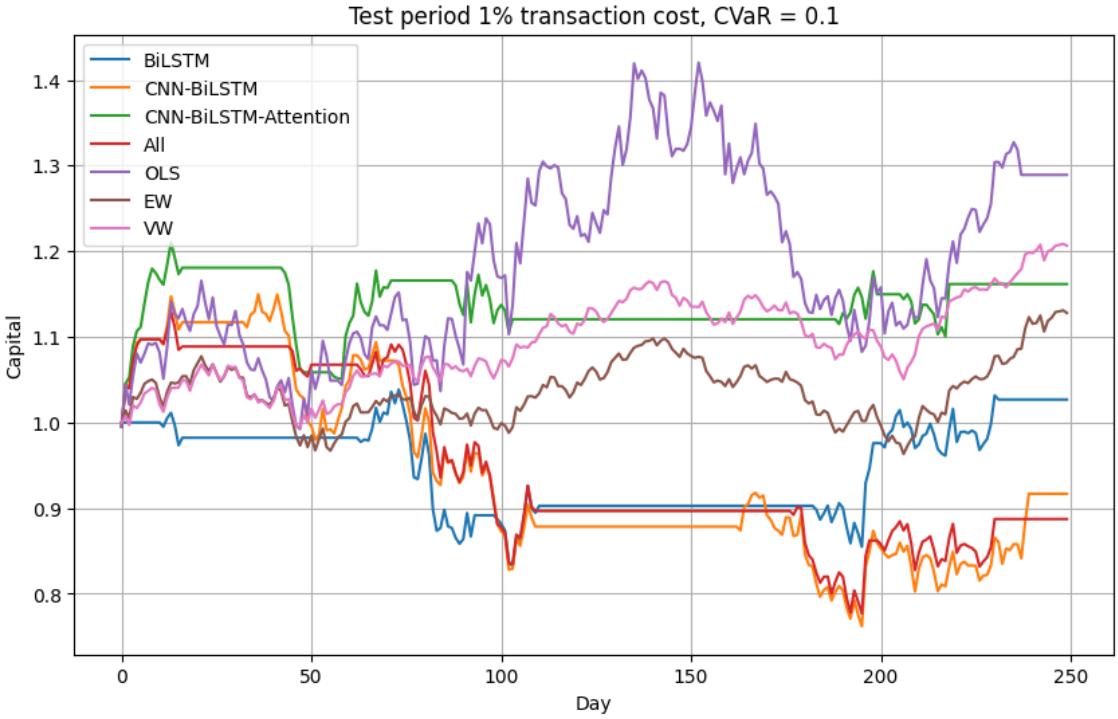

Figure 22: Test period 1% transaction cost, $\text{CVaR}_{0.05} = 0.1$

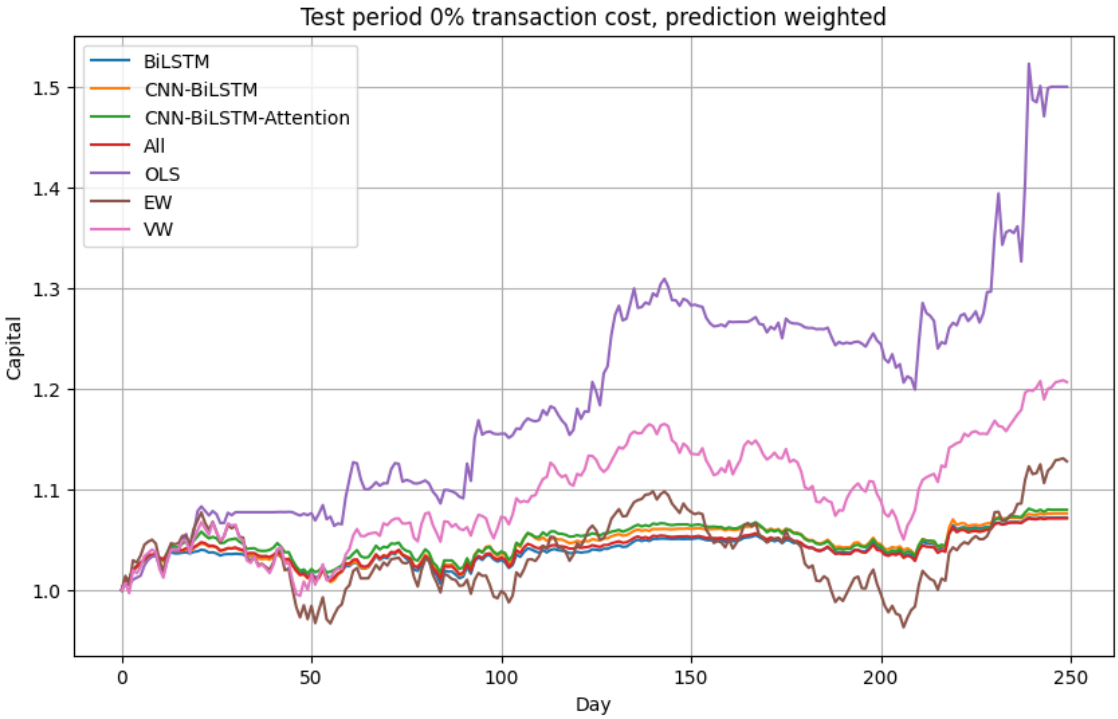

Figure 23: Test period 0% transaction cost, prediction weighted

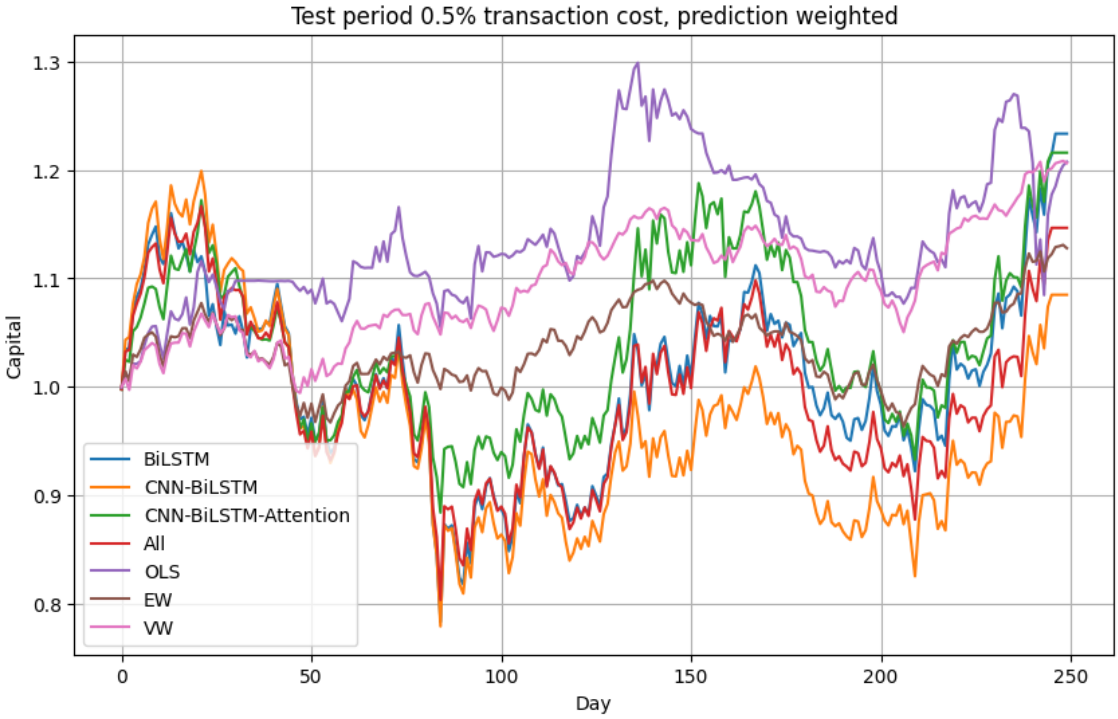

Figure 24: Test period 0.5% transaction cost, prediction weighted

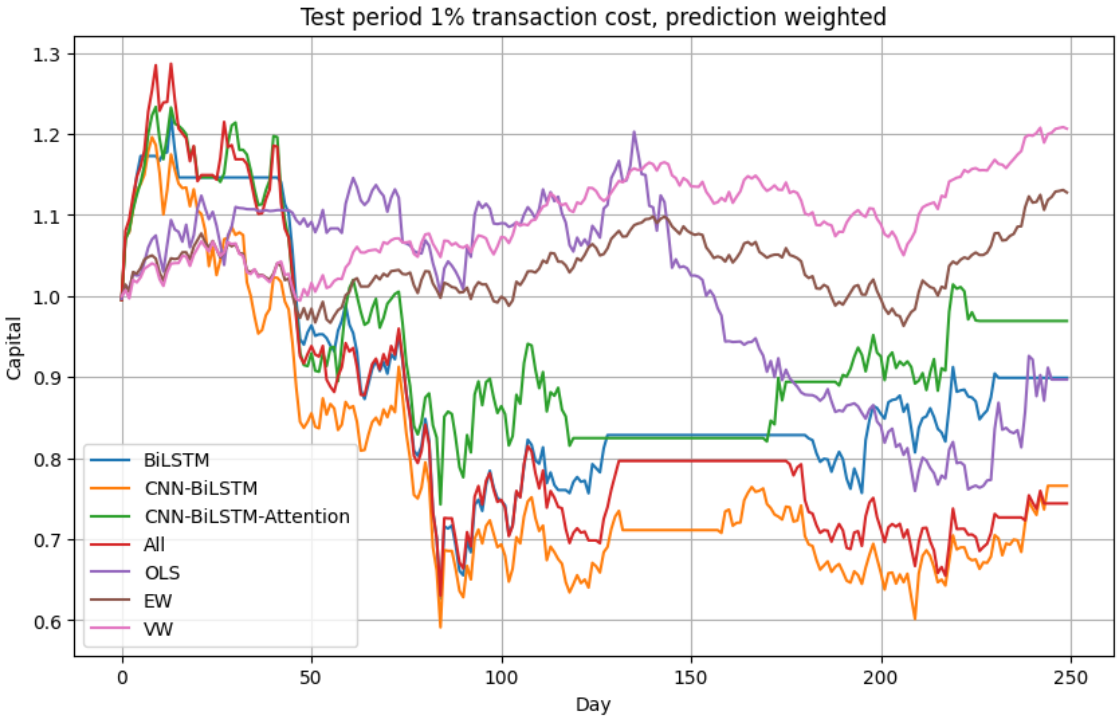

Figure 25: Test period 1% transaction cost, prediction weighted

Appendix 5 - Sample code

```
# Agnostic fundamental analysis

## Packages and data

import numpy as np
import keras
from tensorflow.keras import layers, models

# Here the variables used are shown in 5.1 exclusive of market
    ↪ capitalization
x_pre = np.array((data_point_count,stock_count * variable_count))

# market capitalization
y_pre = np.array((data_point_count,stock_count))

# list of indices of the first market day the month
month_changes_index = []

# list of sectors for each stock
sectors_all = []

# list of unique sectors
sectors = []

## Code

lr = 0.001
drop = 0
hidden_layers = 2
neurons = 100
epochs = 1000
batch_size = 110
decreasing = True
patient = 10
multiplier = 1/100000000000
empty_preds = np.zeros((rows,num_stocks))
dropout_rate = drop

ffnn_model1 = models.Sequential()
ffnn_model1.add(layers.InputLayer(input_shape=(34,)))
if decreasing == False:
  for loops in range(hidden_layers):
    ffnn_model1.add(layers.Dense(neurons, activation="relu"))
    ffnn_model1.add(layers.Dropout(dropout_rate))
else:
  for loops in range(hidden_layers):
    ffnn_model1.add(layers.Dense(int(neurons/(loops+1)), activation="relu")
        ↪ )
```

```
    ffnn_model1.add(layers.Dropout(dropout_rate))
ffnn_model1.add(layers.Dense(1, activation="linear"))

ffnn_model1.compile(
    optimizer=optimizers.Adam(learning_rate=lr),
    loss="mape",
)


for row_index in range(3,len(month_changes_index)-1):
  y_train = np.zeros((stock_count,1))
  x_train = np.zeros((stock_count,34))
  for j in range(stock_count):

    x_row = x_pre[month_changes_index[row_index],(j*12):(j*12+12)].reshape
        ↪ (1,12)
    x_lag = x_pre[month_changes_index[row_index-3],(j*12):(j*12+12)].
        ↪ reshape(1,12)




    index = sectors.index(sectors_all[j])
    sect = np.zeros((1,10))
    if index > 0:
      sect[0,index-1] = 1
    x_row = np.concatenate((x_row,x_lag,sect),axis = 1)

    y_row = y_pre[month_changes_index[row_index],j]

    x_train[j,:] = x_row
    y_train[j,0] = y_row


  x_train[:,:24] *= multiplier
  y_train[:,0] *= multiplier

  callback = keras.callbacks.EarlyStopping(monitor='loss', patience=patient
      ↪ )

  history = ffnn_model1.fit(
      x_train, y_train,
      epochs=epochs,
      batch_size=batch_size,
      verbose=0,
      shuffle=True,
      callbacks = [callback]
  )

  y_pred = ffnn_model1.predict(x_train)
```

```
  y_pred /= multiplier
  for insert_range in range(month_changes_index[row_index],
      ↪ month_changes_index[row_index+1]):
    empty_preds[insert_range,:] = y_pred.reshape(stock_count)


mispricing_signal = (empty_preds - y_pre) / y_pre


# Genetic algorithm for technical indicators

## Packages and data
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from genetic_selection import GeneticSelectionCV

# array of z-score normalized technical indicators during training period
x_train = np.array((data_point_count * stock_count,
    ↪ technical_indicator_count))

# array of z-score normalized five day returns during training period
y_train = np.array((data_point_count * stock_count, 1))

## Code

n_estimators = 5
leaf_perc = 0.1
max_features = 20
pop = 250
cross_over_prob = 0.9
mutation_prob = 0.025


estimator = RandomForestRegressor(n_estimators = n_estimators,
    ↪ min_samples_leaf=int(np.shape(y_train)[0]*leaf_perc), random_state
    ↪ =42)
# Initialize GeneticSelectionCV
selector = GeneticSelectionCV(estimator=estimator,
                              cv=5,
                              verbose=1,
                              scoring="neg_mean_squared_error",
                              max_features=max_features,
                              n_population=pop,
                              n_generations=100,
                              tournament_size = 10,
                              crossover_independent_proba=cross_over_prob,
                              mutation_independent_proba=mutation_prob,
                              n_gen_no_change=2,
                              n_jobs=-1)
```

```
# Fit GeneticSelectionCV
selector.fit(x_train, y_train)

def indexes_of_true(list):
  return [i for i, val in enumerate(list) if val]

indices_chosen = indexes_of_true(selector.support_)

# Neural networks

## Packages and data

import numpy as np
import tensorflow as tf
import keras
import pyvinecopulib as pv
import pypfopt as ppo
import cvxpy as cp
from sklearn.preprocessing import StandardScaler

# array of z-score normalized variables for all stock
x_seqs = np.array((data_point_count * stock_count, timestep_count,
    ↪ variable_count))

# array of z-score normalized five-day ahead returns
ys = np.array((data_point_count * stock_count, variable_count))

## Code

### BiLSTM

n_timesteps = 10
n_features = 54

n_layers = 1
drop = 0.7
lr = 0.001
epoch = 50
mult = 1
batch = int(np.shape(ys)[0] * 0.1)
final_drop = 0.9

for z in range(5):
  model = models.Sequential()
  model.add(keras.layers.Input(shape=(n_timesteps,n_features)))
  model.add(keras.layers.Bidirectional(keras.layers.LSTM(int(n_features/2),
      ↪ return_sequences=False, dropout = drop,recurrent_dropout=drop)))
  model.add(keras.layers.Dense(n_features, activation = "relu"))
  model.add(keras.layers.Dropout(final_drop))
```

```
  model.add(keras.layers.Dense(1, activation = "linear"))
  print(model.summary())

  optimizer = tf.keras.optimizers.Adam(lr)

  model.compile(loss="mse",optimizer=optimizer)

  history = model.fit(x_seqs, ys, epochs=epoch, verbose=0, batch_size=batch
      ↪ , shuffle=True)

  text = f"lstm_model_{z}.h5"
  model.save(text)

### CNN-BiLSTM

n_timesteps = 10
n_features = 54

n_layers = 1
drop = 0.6
lr = 0.001
epoch = 50
mult = 1
batch = int(np.shape(ys)[0] * 0.1)
kernel = 4
conv_drop = 0.1
final_drop = 0.9
leak = 4

for z in range(5):
  model = models.Sequential()
  model.add(keras.layers.Input(shape=(n_timesteps,n_features)))
  model.add(keras.layers.Conv1D(filters=(n_features*mult), kernel_size=
      ↪ kernel, activation=keras.layers.LeakyReLU(alpha=leak), padding = "
      ↪ same"))
  model.add(keras.layers.AveragePooling1D(pool_size=kernel-1,padding="same"
      ↪ ))
  model.add(keras.layers.Dropout(conv_drop))
  model.add(keras.layers.Bidirectional(keras.layers.LSTM(int((n_features*
      ↪ mult)/2),return_sequences=False, dropout = drop,recurrent_dropout=
      ↪ drop)))
  model.add(keras.layers.Dense(n_features, activation = "relu"))
  model.add(keras.layers.Dropout(final_drop))
  model.add(keras.layers.Dense(1, activation = "linear"))

  optimizer = tf.keras.optimizers.Adam(lr)

  model.compile(loss="mse",optimizer=optimizer)
```

```
  history = model.fit(x_seqs, ys, epochs=epoch, verbose=0, batch_size=batch
      ↪ , shuffle=True)

  text = f"cnn_model__{z}.h5"
  model.save(text)

### CNN-BiLSTM-Attention

n_timesteps = 10
n_features = 54
drop = 0.9
lr = 0.001
epoch = 50
mult = 1
batch = num_train*33
final_drop = 0.9
conv_drop = 0.85
kernel = 4
leak = 4
conv_mult = 1

for z in [5]:
  inputs = keras.layers.Input(shape=(n_timesteps,n_features))
  conv1 = (keras.layers.Conv1D(filters=int(n_features*conv_mult),
      ↪ kernel_size=kernel, activation=keras.layers.LeakyReLU(alpha=leak),
      ↪ padding = "same"))(inputs)
  avepool1 = (keras.layers.AveragePooling1D(pool_size=kernel-1,padding="
      ↪ same"))(conv1)
  drop1 = (keras.layers.Dropout(conv_drop))(avepool1)
  bilstm1 = (keras.layers.Bidirectional(keras.layers.LSTM(int((n_features*
      ↪ mult)/2),return_sequences=False, dropout = drop,recurrent_dropout=
      ↪ drop)))(drop1)
  reshaped_output = keras.layers.Reshape((n_features, 1))(bilstm1)
  attn = keras.layers.Attention()([reshaped_output,reshaped_output])
  flattened = keras.layers.Flatten()(attn)
  dense1 = keras.layers.Dense(int(n_features), activation = "relu")(
      ↪ flattened)
  drop3 = (keras.layers.Dropout(final_drop))(dense1)
  final = keras.layers.Dense(1, activation = "linear")(drop3)

  model = keras.Model(inputs=inputs, outputs=final)

  optimizer = tf.keras.optimizers.Adam(lr)

  model.compile(loss="mse",optimizer=optimizer)

  history = model.fit(x_seqs, ys, epochs=epoch, verbose=0, batch_size=batch
      ↪ , shuffle=True)

# GARCH-Copula and portfolio optimization
```

```
## Packages and data

import numpy as np
import tensorflow as tf
import keras
from arch import arch_model
from tensorflow.keras.models import load_model


# The index of the end of the train period
train_end = float()

# The index of the end of the test period
test_end = float()

# days discarded from the beginning of the data
discarded = int()

# close-to-close percentage change
close_change = np.array((data_point_count,stock_count))



## Code

### GARCH

cond_vol_train = np.zeros((train_end-discarded,stock_count))
cond_vol_test = np.zeros((test_end-train_end,stock_count))

for j in range(stock_count):
  am = arch_model(close_change[discarded:test_end,j]*100, dist = "studentst
      ↪ ", vol = "GARCH", p=1,q=1)
  res = am.fit(update_freq=1, last_obs = train_end-1)

  this = res.forecast(horizon = 5, start = 0)

  this = this.variance
  this = this.iloc[:,-1]

  this = this.to_numpy()
  this /= 100

  cond_vol_train[:,j] = this[:train_end-discarded]
  cond_vol_test[:,j] = this[train_end-discarded:test_end-discarded]

### Copula and portfolio optimization

model_num = -1
```

```
# This code runs the BiLSTM model, code can be adjusted with e.g.
# ["cnn"],["attn"],["lstm","cnn","attn"] to compute other models
for model_list in [["lstm"]]:
  model_num += 1
  if model_list != "ols":
    train_list = []
    test_list = []
    for model_text in model_list:
      for i in range(5):
        location = "/content"
        text = f"{location}/{model_text}_model_{i}.h5"
        model = load_model(text)
        # scalery is trained StandardScaler for the train period five-day
            ↪ returns
        model_train = scalery.inverse_transform(model.predict(x_seqs))
        model_test = scalery.inverse_transform(model.predict(x_seqs_test))

        # Concatenate to the corresponding lists
        train_list.append(model_train)
        test_list.append(model_test)

        # Delete arrays to save memory
        del model_train
        del model_test

    # Concatenate arrays in the lists
    train = np.concatenate(train_list, axis=1)
    test = np.concatenate(test_list, axis=1)

    # Delete the lists to save memory
    del train_list
    del test_list

    train_return_regression_pred = np.mean(train, axis=1)
    test_return_regression_pred = np.mean(test, axis=1)
  else:
    train_return_regression_pred = scalery.inverse_transform((ols_model.
        ↪ predict(x_train)).reshape(-1,1))
    test_return_regression_pred = scalery.inverse_transform((ols_model.
        ↪ predict(x_test)).reshape(-1,1))

  train_return_regression_real = scalery.inverse_transform(ys)
  test_return_regression_real = scalery.inverse_transform(ys_test)



  y_preds_sidebyside = np.zeros((num_test,num_stocks))
  for j in range(num_stocks):
    y_preds_sidebyside[:,j] = (test_return_regression_pred[j*num_test:(j+1)
        ↪ *num_test]).reshape(num_test,)
```

```
train_return_regression_real = train_return_regression_real.astype(np.
    ↪ float64)
train_return_regression_pred = train_return_regression_pred.astype(np.
    ↪ float64)
train_return_regression_pred = train_return_regression_pred.reshape(
    ↪ train_return_regression_pred.shape[0],1)



train_errors = train_return_regression_pred -
    ↪ train_return_regression_real
#train_errors = train_return_regression_real

y_errors_sidebyside = np.zeros((num_train,num_stocks))
for j in range(num_stocks):
  y_errors_sidebyside[:,j] = (train_errors[j*num_train:(j+1)*num_train
      ↪ ,0]).reshape(num_train,)




y_preds_sidebyside_train_vol = cond_vol_train

y_errors_normalized_train = y_errors_sidebyside /
    ↪ y_preds_sidebyside_train_vol

y_preds_sidebyside_test_vol = cond_vol_test



u_train = empirical_pit_transform(y_errors_normalized_train)

controls = pv.FitControlsVinecop(family_set = pv.one_par, num_threads =
    ↪ 9999, select_trunc_lvl = True, nonparametric_method = "quadratic")
cop = pv.Vinecop(data = u_train, controls=controls)
print(cop)

sims = 1000


sims = cop.simulate(n=sims)
```

```
sims_untransformed = inverse_empirical_pit_transform(sims, u_train,
    ↪ y_errors_normalized_train)


for tc in [0,0.005,0.01]:
  for target in [0.075]:
    for beta in [0.95]:
      weights_during_test = np.zeros((num_test,329))
      incur_tc = []



      i = 0
      while i < (num_test-5):
        return_preds = y_preds_sidebyside[i,:].copy()

        sim_errors = return_preds + sims_untransformed*cond_vol_test[i]
        return_with_errors = return_preds + sim_errors

        return_preds -= tc

        return_with_errors = np.nan_to_num(return_with_errors, nan = 0.0)
        return_preds = np.nan_to_num(return_preds, nan = 0.0)


        this = ppo.EfficientCVaR(returns = return_with_errors,
            ↪ expected_returns=return_preds, verbose = False,
            ↪ weight_bounds=(0, 1), beta = beta)


        try:
          weights = this.efficient_risk(target)
          if this.portfolio_performance()[0] < 0:
            weights_array = np.zeros((num_stocks))
          else:
            weights_array = np.array(list(weights.values()))
            weights_array = np.round(weights_array, decimals=2)
        except:
          weights_array = np.zeros((num_stocks))




        if np.sum(weights_array) == 0:
          i +=1
          continue


        weights_during_test[i+1:i+5+1,:] = weights_array
        incur_tc.append(i)
```

```
        incur_tc.append(i+4)
        i += 5
```