F540 Project 4423N

2024-03-30

Contents

1 Diversification & Selecting Investible Universe 1

1.1 Cross-sectional Diversification 1

1.2 Diversification over Time 4

1.3 Transaction Costs 5

2 Traditional Approaches 7

2.1 Unconditional Mean-Variance 7

2.2 Conditional Mean-Variance 7

2.3 Penalised MV-Efficient Portfolios 8

3 Prediction & Signal Generation 11

4 Portfolio Construction 14

5 Conclusion 15

1 Diversification & Selecting Investible Universe

1.1 Cross-sectional Diversification

We begin with an investigation into the correlation structure of our assets. There is no single unambiguous measure of dependence to use for our analysis, so we employ a range of measures and compare the results. Specifically, we compute the Pearson, Kendall, Spearman, Gerber, Tail Area, and Quadrant dependence for our assets. This set allows us to see linear dependence, as in Pearson's correlation, as well as non-linear dependence captured by the copula-based measures of Kendall's Tau and Spearman's Rho. The Gerber statistic, for which we use a threshold $t_i = \frac{1}{2}\sigma_i$, is insensitive to outliers and thus gives a robust estimate of the dependence structure.

The tail area dependence, particularly that of the lower tail, is of great importance for portfolio construction. In practice, one wants to avoid large drawdowns at all costs, and if the portfolio constituents have significant lower-tail dependence this increases the probability of a catastrophic loss. For this reason we focus on the joint probabilities in the lower tail with a threshold of $\alpha = 0.95$. While it is not conventional, since the function of a heatmap is to analyse the dependence *between* assets we set the diagonal of each matrix to the mean of its elements - this markedly increases the fidelity of the plots, and the intra-asset correlations are of no importance (and trivially 1) in all of the above cases.

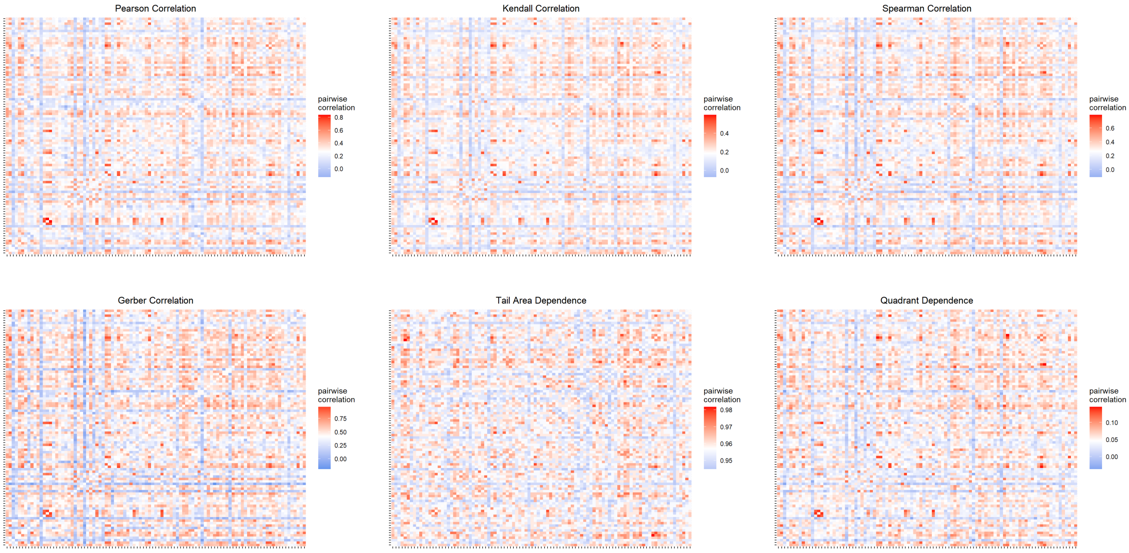

Figure 1: Dependence heatmaps for allocated permnos

As we can see from Figure 1, all of our broad measures are in general agreement on the dependence structure of the assets, with only tail area dependence differing significantly. This is not unexpected, since it is estimating 'asymptotic dependence' while the others are concerned with dependence across the entire distribution. The Gerber statistic, as a robust estimator, illustrates that a non-trivial portion of asset 'correlation' is simply down to noise; compared to the Pearson, Kendall and Spearman heatmaps there are significantly more assets found to have low dependence. This is driven by the significant estimation error associated with historical sample moments, and we see here the Gerber statistic can be one way to try and mitigate this.

Consistent across all of the general dependence measures is a cluster of three highly correlated assets in the lower left corner, with a Pearson correlation coefficient of $\rho \approx 0.8$. To get an idea of how co-dependent their returns are, we select the two with the highest correlation, model their joint distribution, and simulate a large number of returns.

To model the joint distribution we use a bivariate t-copula $C_{\nu,\rho}(u,v)$:

$$C_{\nu,\rho}(u,v) = T_{\nu,\rho}\left(t_\nu^{-1}(u), t_\nu^{-1}(v)\right)$$

- ν: degrees of freedom,
- ρ: correlation coefficient ,
- $x = t_\nu^{-1}(u)$ and $y = t_\nu^{-1}(v)$ where $t_\nu^{-1}(\cdot)$ is the inverse CDF of the t-distribution.
- $T_{\nu,\rho}(x,y)$ is the CDF evaluated at $x = t_\nu^{-1}(u)$ and $y = t_\nu^{-1}(v)$ from the corresponding PDF of the bivariate t-distribution, $f(x,y;\nu,\rho)$, given by:

$$f(x,y;\nu,\rho) = \frac{\Gamma\left(\frac{\nu+2}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\nu\pi(1-\rho^2)}}\left(1+\frac{1}{\nu}\left(\frac{x^2-2\rho xy+y^2}{1-\rho^2}\right)\right)^{-\frac{\nu+2}{2}}$$

The reason we choose the t-copula is driven by the fact that there is leptokurticity in the return series; the t-distribution is able to accommodate this, and therefore produces a better fit than a Gaussian distribution. It is important to note that the standard t-distribution function in R does not support shifting or scaling, and leads to a very poor fit of our return data. To overcome this we make use of the metRology package's *t.scaled* distribution which, as seen in Figure 2, gives a very good fit for our return series.

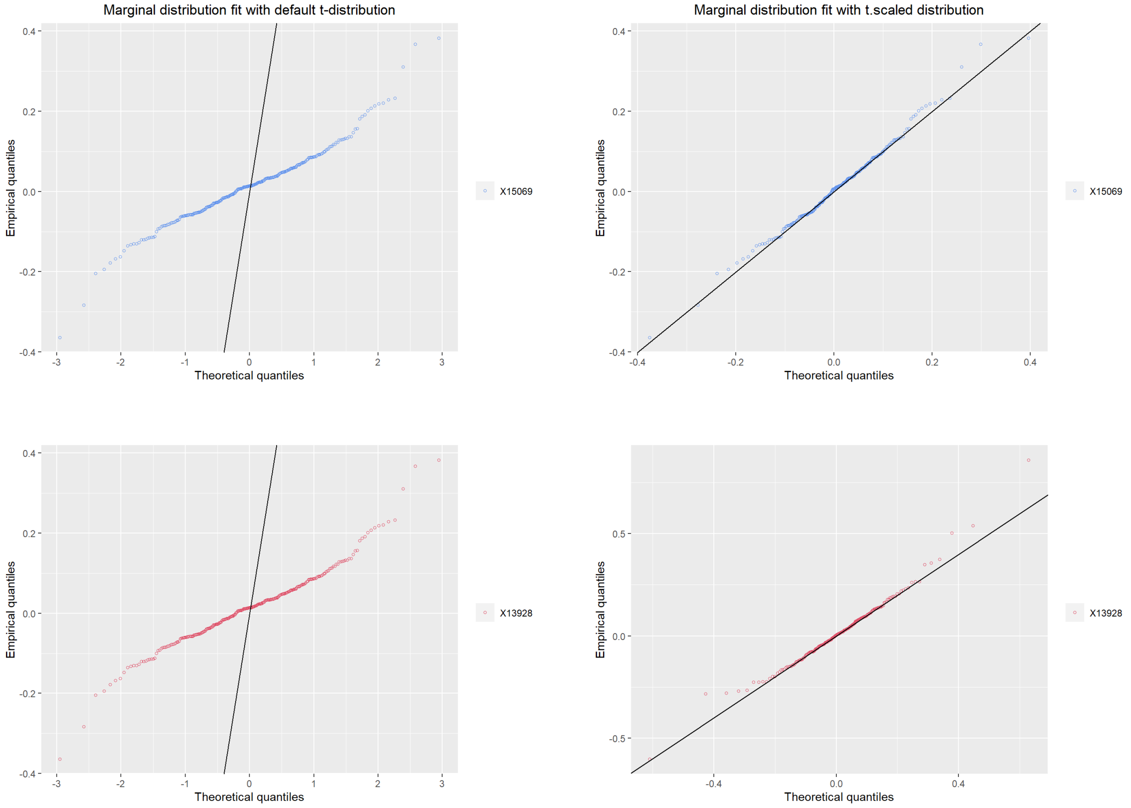

Figure 2: Q-Q plots for standard and scaled t-distributions

Using the fitted t-distributions, we transform the returns to their uniform marginals and fit an unstructured t-copula by maximum likelihood. From this we simulate 1000 pairs of returns, and then transform the data back to the original asset scale using the inverse t-distribution function. For comparison, we include an equivalent simulation for two other assets selected randomly from our universe. The results are presented in Figure 3, and we can see that the returns are extremely highly correlated, and one should definitely consider whether they want to hold both of these assets in their portfolio. That said, the approach of asset selection by filtering on pairwise correlation is not very theoretically satisfying, since it overlooks both the returns and the interaction of the assets with all of the others, but it is one way in which we could consider refining our initial set.

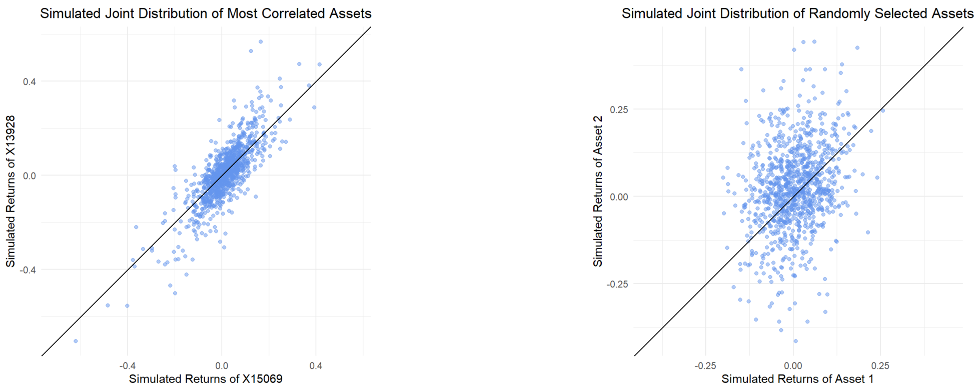

Figure 3: Simulated Joint Distribution of Asset Returns

1.2 Diversification over Time

We next analyse how the correlation and diversification of our assets evolves over time and with market conditions, calculating our measures of dependence in the temporal dimension; of particular focus is the empirical regularity that asset correlations tend to increase during market downturns, reducing portfolio diversification when it is needed most. Additionally, we use two different measures of portfolio-level diversification:

1. The inverse of the weight-based metric of Goetzmann, Li and Rouwenhorst, the ratio of the portfolio variance to the weighted average of the constituent assets variances:

$$GLR(w)^* = GLR(w)^{-1} = \left(\frac{w' \Sigma w}{\sum_{k=1}^{N} w_k \sigma_k^2} \right)^{-1}$$

   We invert the standard GLR statistic to make it consistent with other measures of diversification, for which a higher value corresponds to greater diversification, for more intuitive comparison.

2. The effective number of minimum torsion bets, MTB, defined as in Meucci (2015) by the entropy of the portfolio variance factors after PCA:

$$MTB(w) = exp\left(- \sum_{k=1}^{N} p_k^* \; ln(p_k^*) \right)$$

   where the rotation of the principal axes is chosen such that it minimises the sum of squared errors between the factor returns and those of the original assets.

Using a 12-month rolling window, we plot the average Pearson, Kendall, Spearman and Gerber correlations alongside the GLR* and MTB values that are achieved by an equal-weighted portfolio of our assets, as a proxy for 'aggregate market diversification'. The results, which can be found in Figure 4, emphasise that the asset dependencies and portfolio diversification do indeed vary significantly over time and with market conditions.

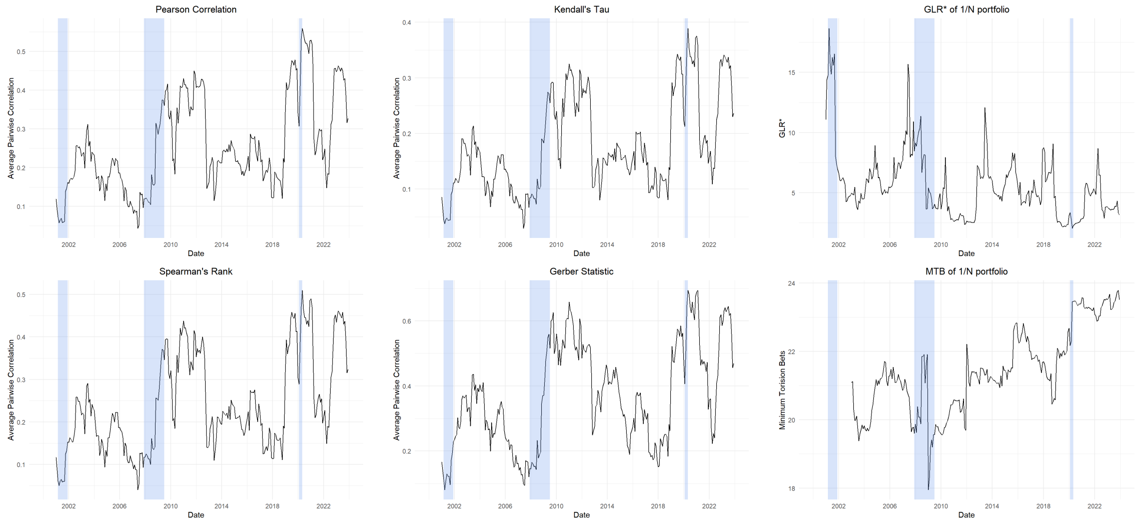

Figure 4: Correlation and Diversification measures over time

Highlighted in blue are the periods corresponding to NBER recessions, and we can see that aggregate inter-asset correlation tends to drastically increase in these periods. The diversification measures are strongly affected by market conditions, markedly decreasing in the shaded periods, particularly the 2007Q4 - 2009Q2 recession. In other words, during poor market conditions correlation-adjusted concentration increases and the effective number of minimum torsion bets decreases, reducing portfolio diversification and increasing correlation risk exposure just when negative movements are most likely.

1.3 Transaction Costs

The importance of transaction costs has historically been overlooked in the literature, despite being critical to whether a potential strategy is actually *implementable*; notwithstanding the great decrease in transaction costs over the last two decades, a significant difference remains between trading the largest and smallest stocks on the market. The cost of trading also varies across factors, and is why many strategies incorporating short-term reversal and momentum, which inherently include significant turnover, find their gross alpha eroded by the cost of implementing them. In this paper we endeavor wherever possible to incorporate transaction costs directly into our portfolio weight selection, as well as correcting our results *ex-post* for their impact.

It is here that we begin refining our selection of assets. Three are no longer traded, two having undergone mergers and one a voluntary delisting in 2023. We use daily OHLC data from CRSP to calculate our transaction costs, and some of our assets have significant missing data. Due to the sensitivity of the OHLC volatility estimation, we remove assets with more than 1% missing observations and impute using cubic spline interpolation within each asset any missing values. This is an attempt to strike a balance between having accurate, reliable data without dropping too many assets, and reduces the set of assets considered to 88.

We compute our OHLC volatilities using the Yang-Zhang estimator, and pass this to the Ardia, Guidotti & Kroencke (2021) estimator for the bid-ask spread, which is halved to estimate the transaction costs. We analyse the cost of trading in both dimensions, taking the mean across assets and plotting it against time, as well as calculating the mean for each asset and plotting it against market capitalisation. The results, seen in Figure 5, show that volatility and transaction costs vary significantly both cross-sectionally and with market conditions, again notably rising in the periods corresponding to NBER recessions. Only one microcap stock has made it this far into the analysis, effectively answering the question of whether to drop them for us. Given our results, *if* machine learning strategies can learn to divest leading up to recessions, they will not only be side-stepping downwards market movements but also avoiding the increased transaction costs associated with them.

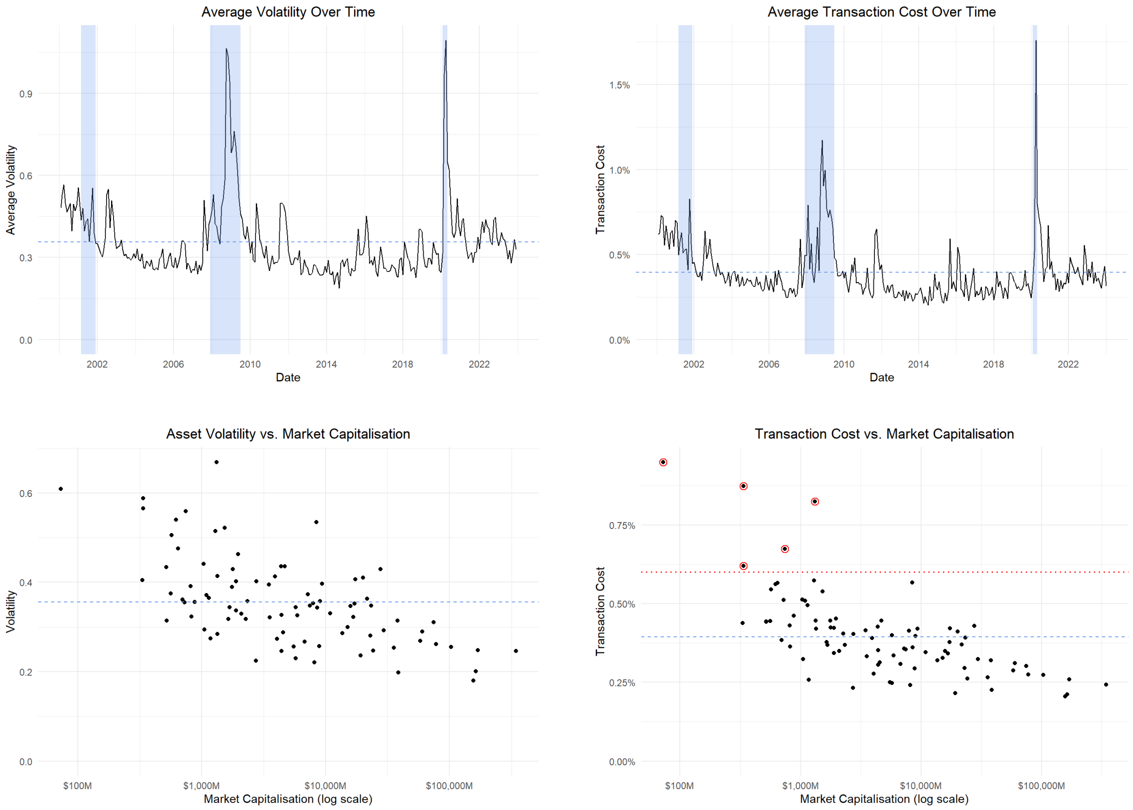

Figure 5: Volatility and Transaction Costs

We conclude this section with our asset selection, filtering out those that have historical mean transaction costs of greater than 60bps, shown by the red dotted line in Figure 5. This leaves us an investible set of 83 assets, with mean transaction costs of 37bps. Furthermore, a Gibbons-Ross-Shanken test against the complete set of assets does not reject the null of mean-variance efficiency at the 5% level ($F = 1.51,\ p = 0.11$). That our subset of assets is mean-variance efficient with respect to the original set is good, and we have significantly reduced our exposure to systematic/macro risk and cost of trading with our selection.

2 Traditional Approaches

2.1 Unconditional Mean-Variance

In this section we undertake a review of strategies derived from mean-variance analysis, with focus on their performance across different points in the market cycle. Due to word-count constraints this section is forced to be brief, attempting to summarise the main approaches. There are simply too many strategies to present clearly at once, so we attempt to chunk them into intuitive groups with comparison to a consistent benchmark.

For each strategy we conduct rolling analysis at a monthly frequency with a 60 month window, incorporating within the optimisation proportional transaction costs of 37bps, and using asset-specific transaction costs when calculating net returns.[1]

We begin by considering, in part to illustrate the shortcomings of unconstrained mean-variance optimisation, the following portfolio specifications:

1. Unconstrained Mean-Variance (MV-U)
2. Constrained Mean-Variance (long only) (MV-C)
3. Minimum Variance (MinVar)
4. 1/N

As can be seen from Figure 6, the 1/N portfolio comprehensively outperforms the unconditional mean-variance portfolios both un-adjusted for and net of transaction costs. The inclusion of transaction costs *ex-ante* into our optimisation tempers the worst eccentricities of true unconstrained mean-variance optimisation, and leads to very similar (but poor) performance across the specifications.

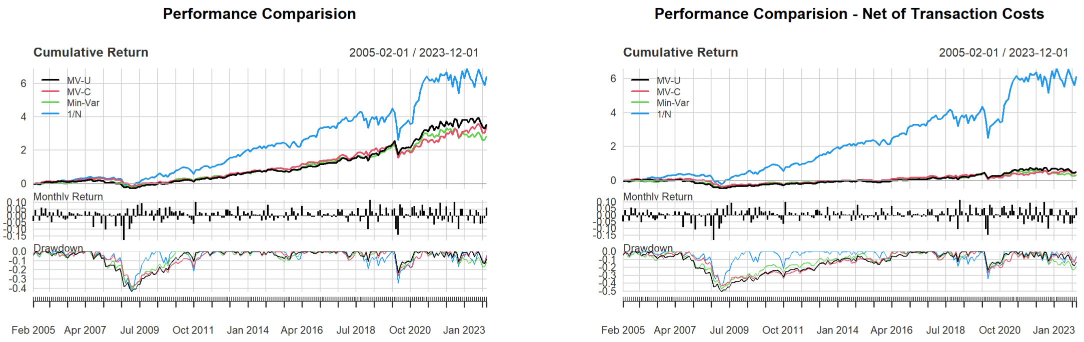

Figure 6: Caption for MVU plots

2.2 Conditional Mean-Variance

We next consider conditional mean-variance strategies, including for comparison the MV-U and 1/N portfolios, conducting the same analysis for each of the following specifications:

1. Linear Shrinkage (MVC-LS)

[1] A flexible way to do this is to take the absolute value of the difference of the $T \times N$ portfolio weights matrix, and multiply it by the $N \times 1$ vector of asset transaction costs. This gives a $T \times 1$ vector corresponding to the transaction costs in each period, which can then simply be subtracted from the gross return. As with everything else I cover briefly, full details/implementation can be found in the RMarkdown document.

2. Non-linear Shrinkage (MVC-NLS)
3. Gerber Covariance (MVC-Gerber)
4. Volatility Timing (MVC-VT)

As can be seen from Figure 7, the conditional mean-variance strategies perform significantly better than their unconditional counterparts. In particular, the linear shrinkage approach is able to match the 1/N portfolio before transaction costs, but its higher turnover means that it falls short in a net sense. We should note that each of the portfolios experience huge drawdowns of up to 50% and 20% in the two recession periods respectively; unsurprisingly the mean-variance strategies have not learned to divest leading up to recessions!

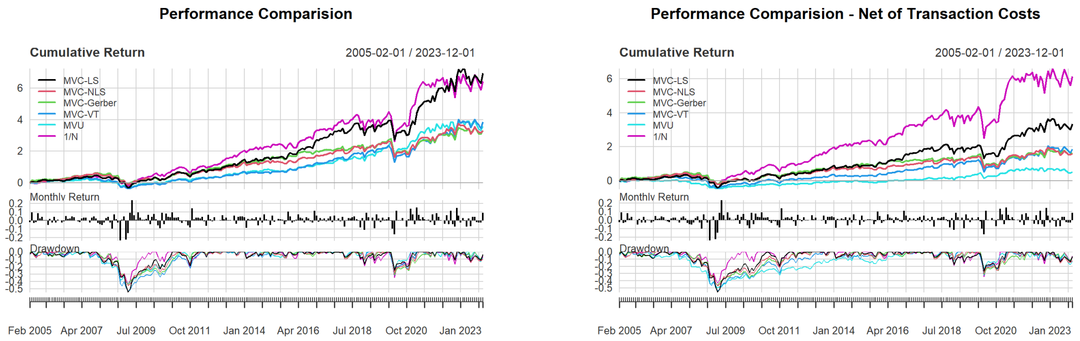

Figure 7: Conditional Mean-Variance Performance

2.3 Penalised MV-Efficient Portfolios

Given the noise inherent in estimating the means and covariance matrix, and the marked improvement seen from incorporating shrinkage into our weight estimation, it makes sense to try to estimate and shrink the weights directly. To do this, we use the Britten-Jones (1999) result that mean-variance efficient weights can be obtained from the regression $\iota = R\beta + \varepsilon$ and estimate this using the following approaches, including for comparison the 1/N portfolio:

1. OLS
2. Lasso
3. Ridge
4. Elastic Net

We continue to use the framework described above, but we are forced to increase the length of our rolling window since, with 83 assets, a width of 60 gives an underdetermined system. To balance the trade-off between having reasonable degrees of freedom and analysis period we set the width to 156, giving 10 years of out-of-sample results. The results, found in Figure 8, show that there is a significant improvement from estimating the weights directly, but weight-regularisation is essential to produce sparser portfolios with reduced turnover.

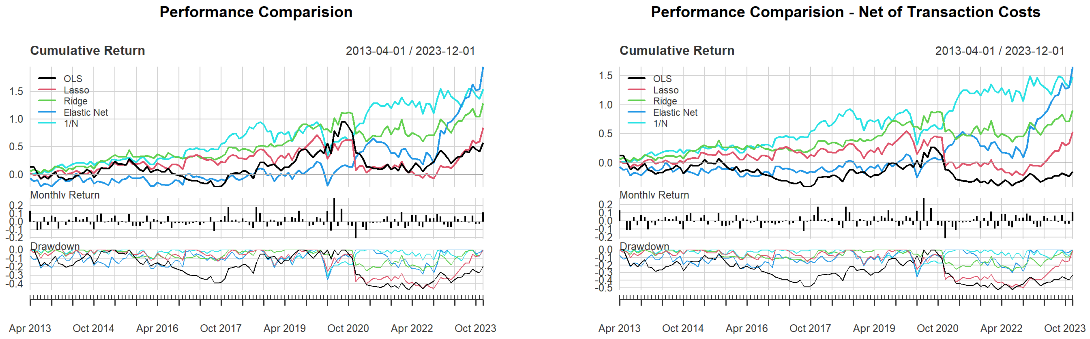

Figure 8: Penalised Regression Performance

All penalised regressions markedly outperform OLS, which takes such aggressive positions that it makes a loss net of transaction costs, and the approaches which incorporate an L_2 norm do better than those that don't due to the quadratic norm smoothing the weights, resulting in less turnover between periods. However, the sparsity induced by the L_1 penalty is complementary to this, and the ENET specification produces the best result.

We include in Table 3.1 the net results from these strategies, foregoing an extended discussion for brevity. In short, all traditional approaches are comprehensively outperformed by a 1/N portfolio for our set of assets in a mean-variance sense, and they suffer huge drawdowns in market downturns. Unsurprisingly, there is no identification or exploitation of market cycles visible in their performance.

Table 1: Results: Net of Transaction Costs

Metric	MV-U	MV-C	Min-Var	MVC-LS	MVC-NLS	MVC-Gerber	MVC-VT	OLS	Lasso	Ridge	ENET	1/N
Mean	0.034	0.033	0.025	0.097	0.063	0.065	0.073	0.015	0.067	0.074	0.126	0.116
StdDev	0.148	0.145	0.140	0.192	0.157	0.160	0.181	0.249	0.227	0.166	0.266	0.186
SharpeRatio	0.156	0.155	0.110	0.420	0.333	0.330	0.317	-0.060	0.180	0.373	0.358	0.557
ES	-0.114	-0.115	-0.111	-0.154	-0.114	-0.159	-0.160	-0.129	-0.223	-0.104	-0.134	-0.127
Turnover	322.328	318.080	326.476	163.894	143.108	149.192	147.084	173.372	47.674	51.730	24.502	3.593
MaxDD	0.503	0.435	0.471	0.557	0.473	0.463	0.543	0.525	0.494	0.278	0.355	0.455

3 Prediction & Signal Generation

We now significantly improve upon the methods covered in the previous section by forming conditional forecasts of the expected returns using modern statistical tools and a wide range of potential features. We construct our baseline feature dataset from the following sources, in each case analysing the number of missing values for each characteristic and retaining only those that are missing fewer than 1% of their values for our selected assets:

1. CRSP: all 71 asset-specific fundamentals → 45 monthly-frequency characteristics retained
2. Compustat: all 22 asset-specific 'informative fundamentals' → 14 quarterly-frequency characteristics retained

We impute our feature data using cubic spline interpolation (within-assets) to account for the remaining missing values. This approach is markedly more efficient than only preserving complete cases à la na.omit() - despite restricting our attention to characteristics with <1% missing values, taking the union of non-missing observations results in dropping a significant fraction of the data (not to mention losing its coherence as a time series). The way in which we accommodate the merging of the monthly and quarterly series depends on the frequency at which we intend to implement the forecasting/portfolio constriction: we form a monthly dataset using LOCF on the quarterly data and a quarterly equivalent by simply using the last monthly observation in each quarter.

Finally, we standardise our data in two dimensions. The first, following Brandt & Santa-Clara (2009), is a cross-sectional standardisation across all assets at each date. The most important consequence of this is that our standardised features are stationary, while the originals are almost certainly not, which is important when using any linear regression-based models. The second is a longitudinal/temporal standardisation for each feature, to ensure proper functioning and improved convergence speed of the ML optimisation algorithms we use later. Most obviously this applies to the SGD-type optimisers we use for our neural networks, but also has implications for the PCR/PLS-based strategies we employ. Obviously, we lag the feature data by one period before joining it to the excess return data for our assets.

We summarise the prediction methods we use and their hyperparameter selection below. As before, we are unfortunately forced to be brief, full detail can be found in lines 1650-1950 of the RMd.

1. OLS
2. Lasso: tuned using cv.glmnet to select λ_{min} with the lowest cross-validation MSE
3. Elastic Net: tuned using cv.glmnet to select λ_{min} with the lowest cross-validation MSE
4. PCR: tuned using pcr's validation argument to select *ncomp* with the lowest cross-validation MSE
5. PLS: tuned using the plsr's validation argument to select *ncomp* with lowest cross-validation MSE
6. RF: tuned using a parameter grid with caret since ranger does not have inbuilt hyperparameter tuning, but randomForest is hopelessly slow. Lowest cross-validation MSE given by: *num.trees = 1000, mtry = 9, min.node.size = 10, max.depth = NULL.*

7. GBRT: tuned using a parameter grid in combination with xgb.cv for its early-stopping and k-fold cv capabilities. Lowest cross-validation MSE given by: *eta = 0.01, max_depth = 8, subsample = 1, min_child_weight = 10, nrounds = 532.*

8. NN[1:5]: optimised using 'adam' with *learning_rate = 0.001*, ReLU activations for all hidden layers and linear activation for the output layer. While the specific regularisation depends on the network the general approach is as follows: mild l_1 and l_2 penalisation in combination with dropout (rate = 0.3) on the upper layer; l_2 penalisation in combination with dropout on the middle layers; l_2 penalisation on the deeper layers. This produces well-behaved networks reluctant to overfit, with consistent OOS prediction across a wide range of complexities. (*Full specifications can be found on lines 1830-1960.*)

We present in Table 2 statistical tests of relative forecasting ability for each of our models. The first row of the corresponds to the out-of-sample R^2 for each model, while the upper-triangular matrix below corresponds to one-sided Diebold-Mariano tests of equal predictive accuracy for the column-model against the row-model, with the DM statistics that imply a rejection of the null at the 5% marked with an asterisk.

Table 2: Monthly out-of-sample Prediction Performance

	OLS	Lasso	ENet	PCR	PLS	RF	GBRT	NN1	NN2	NN3	NN4	NN5
R^2_{OOS}	0.709	0.864	0.82	0.818	0.721	-0.839	0.415	0.879	1.023	1.123	1.081	1.108
OLS	-	2.384*	1.727*	0.929	1.348	-5.882	-1.525	2.546*	2.081*	2.853*	2.533*	2.537
Lasso		-	-5.092	-0.656	-2.215	-6.494	-2.242	1.89*	1.58*	2.541*	2.062*	1.823*
ENet			-	-0.028	-1.555	-6.292	-1.99	2.384*	1.945*	2.814*	1.74*	2.038*
PCR				-	-0.83	-5.92	-1.773	2.905*	3.189*	3.291*	2.878*	2.439*
PLS					-	-5.95	-1.594	2.451*	2.952*	2.849*	2.467*	2.464*
RF						-	5.835*	6.688*	6.717*	7.025*	6.329*	6.887*
GBRT							-	2.721*	2.825*	3.544*	3.229*	3.744*
NN1								-	0.605	2.131*	1.251	1.275
NN2									-	2.093*	0.939	0.825
NN3										-	-0.426	-0.219
NN4											-	0.605
NN5												-

The first thing we see is that the R^2_{OOS} is generally increasing in the 'sophistication' of the models. Furthermore, in a Diebold-Mariano sense, we are able to significantly improve on the OLS forecast using the more advanced statistical methods, with OLS statistically outperformed by Lasso, ENet and NN1-5. The R^2_{OOS} of the RF and GBRT models are strangely low, but as we will see in the portfolio construction they perform well - I believe they are getting a few outlier-type observations hugely wrong due to the minimum node size/child weight restrictions, and this is negatively skewing their performance, but proves not to be an issue in portfolio construction. The best performing models are the neural networks, and we can see that there is a return to additional complexity, but only up to a point; neither the NN4 or NN5 models are able to outperform the NN3 despite their additional hidden layers. In light of this, as we move on to forming our return-timing portfolios, we drop the NN4 and NN5 models since repeatedly re-training them is time-consuming and they do not appear to offer any additional performance over the more parsimonious NN3 model.

4 Portfolio Construction

We proceed to form monthly return-timing portfolios using the predictions of the models above. Once again, *full details of the approach can be found on lines (2070-2180)* but a summary is as follows: for each model, we train it on the data from 2000-01-01 to 2010-01-01 and use 2010-01-01 to 2012-01-01 for cross-validation. We then roll through the test data, predicting the returns from the features, and (using the fact that we previously sorted our data by permno and date) are able to transform the unnamed, ~ 15000 long predictions vector into an xts object of predictions for each asset at each date by considering the outputted predictions mod 83. The merits to this approach are twofold - from here it is straightforward to form arbitrary n-tile return-timing portfolios by simple operations on the rows of the predictions matrix, as well as having effectively plugged ourselves back into the Portfolio/PerformanceAnalytics world and all of its features (and we can continue to use our matrix multiplication approach on the weights matrix to calculate net returns).

In order to analyse the question of whether the use of machine-learning models leads to portfolios that outperform traditional approaches *specifically* through the identification and exploitation of market cycles we construct two different sets of portfolios: the first implements return-timing with a long-only decile portfolio with predictions using only the features discussed above; the second augments the features data with a range of macroeconomic variables and sentiment indices from FRED.[2] The idea is to try and separate the essence of the question, which is whether the outperformance is driven *specifically* by the identification of market cycles, from the more general observation of outperformance.[3]

We present in Figure 9 the results for the long-only decile portfolios without the macroeconomic and sentiment features. The neural networks, random forest and GBRT perform by far the best out of the strategies, both gross and net of transaction costs. With comparison to our earlier figures, one can also see that our return-timing ML portfolios utterly dominate the 1/N portfolio in gross returns, and beat it modestly in net returns; this advantage is only extended by mitigating

[2] Macroeconomic Variables: GDP (percent change), Unemployment Rate, CPI (percent change), Federal Funds Rate, 1Y, 5Y, 10Y Treasury Bill yield; Sentiment Indices: VIX, UoM Consumer Sentiment Index, UoM Inflation Expectation Index, CB Leading Economic Index, Global Economic Policy Uncertainty Index

[3] *We also have fully working and estimated long/short decile portfolios (as well as PPP which delivers an extra 4% annualized return over 1/N) along with their associated results tables in the RMd, but sadly not the word count to discuss them. I focus on the long-only portfolios here because the exposition of market-cycle identification is more direct.*

transaction costs, either simply rebalancing quarterly or conditioning on the previous periods weights *(as in Di Miguel et. al, 2015)*, but is not the focus of this paper.

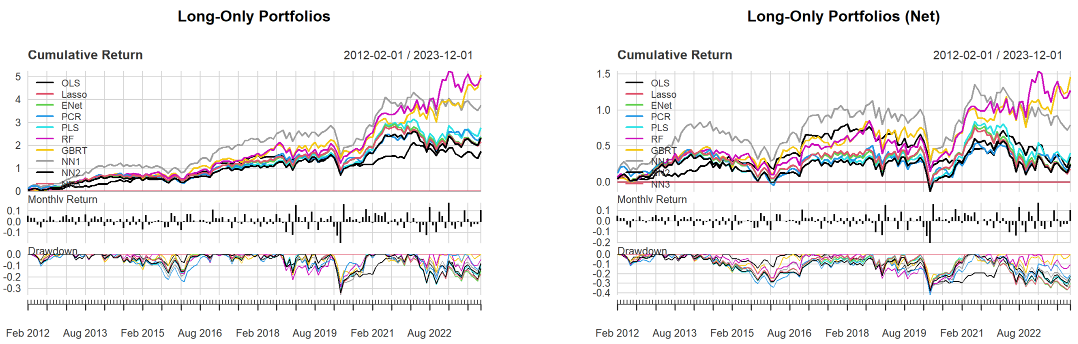

Figure 9: ML Strategy Performance without Macro & Sentiment Indices

In Figure 10 we can see that including the Macroeconomic variable and Sentiment Indices has a positive outcome, particularly for the neural networks and particularly for net returns. The other models are also generally improved by the addition, but nowhere near to the extent that the neural networks are. It is not entirely one-way, however; the GBRT model is significantly worsened by the addition. Upon investigation, it does not seem to be an issue of overfitting *per se* (and we did not change the parameters), but rather that the GBRT algorithm struggles to cope with the addition of characteristics which are the same across all stocks in each period when predicting individual stock returns.

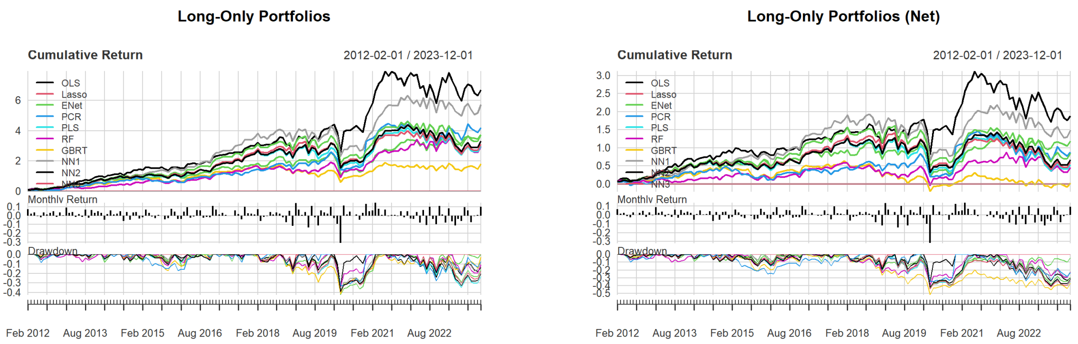

Figure 10: ML Strategy Performance with Macro & Sentiment Indices

5 Conclusion

So, we have demonstrated that machine-learning methods can be used to form portfolios that outperform traditional methods (that much is very clear) and that their performance improves when adding macroeconomic and sentiment indicators, but is this *specifically* because they can identify market cycles?

This is not straight-forward to answer, particularly given the standard issues with interpretability of machine-learning models, but we attempt to get something tractable with the following approach:

1. We use the estimated models to form two ensemble forecasts, the first is an ML ensemble consisting of the return predictions for the NN, RF and GBRT models, and the other consisting of the standard SM models.
2. For each ensemble, we standardise each of the model predictions to make them comparable, and then take compute the ensemble return prediction as the mean of the individual predictions.
3. We then take rowsums of the resulting ensemble prediction matrices to get the average return prediction for the ensemble at each point in time. The idea is that this proxies for the ensemble's prediction for the aggregate market return, and we compare this over time for the two different ensembles.

The results in Figure 11 are extremely interesting; the ML ensemble does seem to loosely lead the SM ensemble, but more importantly it predicts very consistently low returns in the 2017Q4-2019Q1 period in which the Fed raised interest rates several times, the US/China Trade War began to significantly impact the stock market, and there was a general slowdown in global growth. The SM ensemble does clock on to this too, but it is slower, later and less convicted in its view that returns will be low through this macroeconomic cycle.

Clearly, significantly more research needs to be put into analysing whether machine-learning models are capable of identifying market cycles, as Kelly and other suggest, in order to state it as fact. We do, however, find unequivocal evidence that machine-learning models are able to form portfolios which are economically profitable beyond that of traditional methods, and tentative evidence that part of the way they are doing this is through identification of market cycles, justified by the observations that the NN models benefit by far the most from the addition of the macroeconomic & sentiment indices, and the conviction with which the ML ensemble appears to forecast low aggregate returns during the period of fundamental-driven economic slowdown.

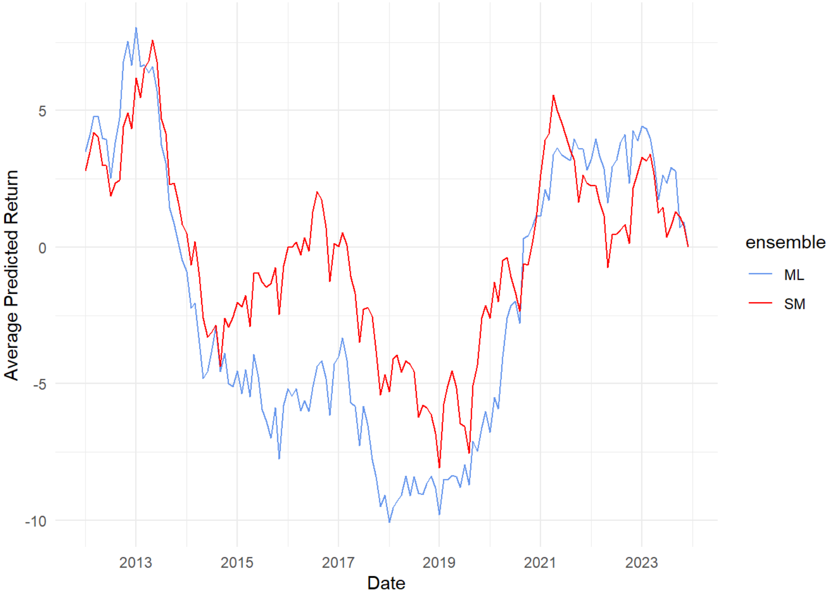

Figure 11: Return Prediction Comparison between Ensemble Forecasts