Quantitative Trading & Analysis with Python: Group Project

Vinh Phan (vinh.phan@fs-students.de)
Maximilian Pintilie (maximilian.pintilie@fs-students.de)
Tajda Urankar (tajda.urankar@fs-students.de) Group **Beta**
Juan Diego Zucchino (juan_diego.zucchino_aguilar@fs-students.de) March 31, 2023

Table of Contents

1 Objective

The task involves creating a quant strategy that outperforms a value-weighted (market-cap-weighted) portfolio of 50 stocks from the S&P500 universe (BNCH), over the period from 01/2016 to 12/2022. The strategy should adhere to specific constraints:

1. Maximum factor exposure deviation of 0.05 for each considered factor.
2. Maximum weight deviation of 10% of the BNCH weight for each asset
3. Maximum drawdown relative to BNCH of 1% per month.

To achieve this, the strategy should make use of a selection of models including Extreme Gradient Boosting Regressor, Lasso, Ridge, and ElasticNet regression. The performance metrics to evaluate the performance of the model and of the trading strategy are mean absolute error, Sharpe Ratio, Max. Drawdown, Smart Sortino Ratio (also referred to in short as Sortino Ratio or Sortino), Sharpe-, Calmar- and Win Loss -Ratio. Once a model and strategy is selected, trained, and optimized on the in-sample period from 01/2000 to 12/2015, we use this model and strategy and evaluate on the out-of sample period from 01/2016 to 12/2021. During this period the model is not changed or adjusted.

2 Data Preparation & Feature Engineering

With a clear objective defined, multiple features from different sources were retrieved and merged for a total list of 50 permnos. Fama Ken French 4-Factor Model was selected to capture the systematic risk factors and calculate the factor betas. The stock returns, options, and fundamental data was retrieved from respectively CRSP, OptionMetrics, and COMPUSTAT. In addition, advanced statistical techniques were utilized such as Generalized Lower Bounds (GLB) and Risk-Neutral Skewness (MFIS) to analyze the performance of the stocks. Lastly, price pattern for momentum and volatility of the market were calculated to gain insights into short-term price movements. A complete list of the used variables can be seen below which will be used to develop effective investment strategies.

2.1 Overview of Data

1. PERMNO list of 50 stocks
2. Fama Ken French 4-Factor Model
3. WRDS data on stock returns [CRSP], options [OptionMetrics], fundamentals [COMPUSTAT]
4. Factor Betas Calculation
5. Generalized Lower Bounds (GLB)
6. Risk-Neutral Skewness (MFIS)
7. Technical Analysis and Price Patterns

2.2 Feature Engineering Part 1

Moreover, in addition to the previously mentioned variables, an effort was devoted to developing technical labels and price patterns aimed at enhancing the predictive capacity of the model. These newly computed variables encompassed lagged returns per permno for a period of up to 21 days, biweekly rolling mean returns, and biweekly volatility. The inclusion of these variables substantially aim at improving the model's ability to capture the overall momentum of the permno return. A sample snippet for these feature engineered variables can be seen below for one permno.

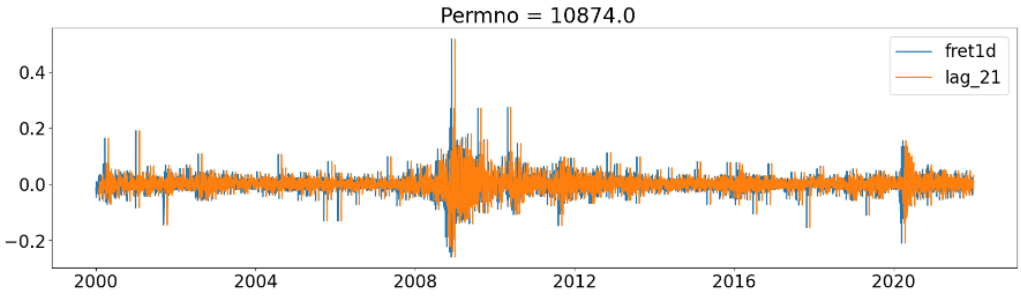

Figure 1: Example of lagged value features

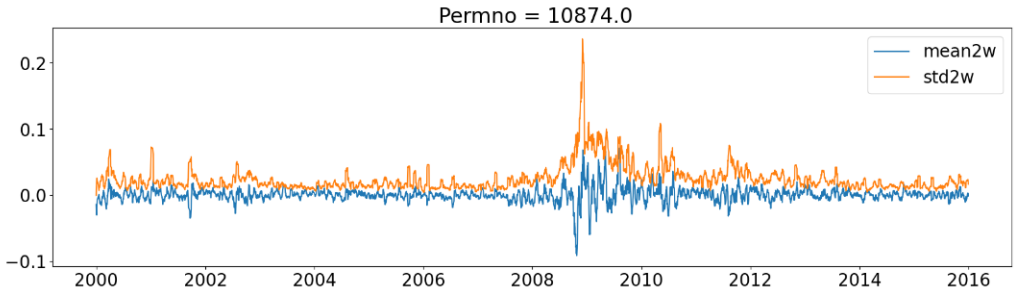

Figure 2: rolling biweekly mean returns, and biweekly volatility

2.3 Preprocess Pipeline

Scikit-learn's Pipeline class was used in the backtester class to preprocess the data before building each model. The pipeline consists of three data transformation steps, performed sequentially on the insample data. The first step applies a winsorization transformation to the in-sample data with a `level_winsorize` level of 0.025, as GLB and MFIS data sets were already winsorized at that level. The second step applies a power transformation to the data using the PowerTransformer function. This transformation is used to adjust the distribution of each feature to be closer to a normal distribution, which can help improve the performance of the regressions. The final step applies a max-abs scaling transformation to the data using the MaxAbsScaler function. This function scales each feature by its maximum absolute value.

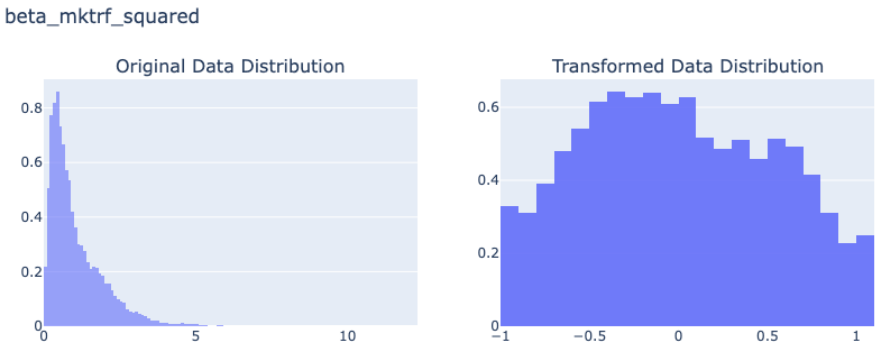

Figure 3: An example of a feature distribution before and after preprocessing. The distribution of the original data is left skewed and after using the preprocess pipeline, the data becomes centered around zero.

Figure 3 shows an example for feature `beta_mktrf_squared` before and after using Pipeline class. Not

all the features were preprocessed using this pipeline. Lagged features, GLB and MFIS were excluded from the preprocessing step (those features were already winsorized at level 0.025 when downloaded), they were only used for model building directly.

3 Feature Selection & Importance

Feature selection and importance was done on in-sample (training) data from 01/2000 until 12/2015. The features selected were then used for XGBoost and Random Forest models and the out-of-sample period from 1/2016 until 12/2021.

3.1 Nonlinear Feature Engineering Part 2

For the purpose of building a model, additional nonlinear features (squared features and feature combinations) were constructed. Squared features or combinations of features were created in order to capture nonlinear relationships between the input features and the target feature. In many cases, the relationship between the input features and the target feature is not linear, and by creating squared features, nonlinearities can be captured better. Similarly, if there is an interaction between two input features, where their combined effect is greater than their individual effects, a product feature (a combination) can be created to capture this interaction.

Creating squared or combined features can increase the expressive power of the model and improve its predictive performance. However, it's important to be careful not to overfit the model by creating too many features that would significantly increase models training time complexity or including irrelevant features.

3.1.1 Squared Features

Selected data for squaring:

- WRDS data on stock returns [CRSP],
- options [OptionMetrics],
- fundamentals [COMPUSTAT],
- factor betas from FF4.

Lagged features, Generalized Lower Bounds and Risk-Neutral Skewness data was not used in the squaring process. As lagged features represent the past values of the feature that the model is trying to predict, they are commonly used in time series analysis and forecasting. When using lagged features in a machine learning model, it is usually not beneficial to square them because they represent a linear relationship with the target feature. Squaring them does not introduce any additional nonlinear behavior and may only introduce noise and overfit the model. Typically, it is more appropriate to use lagged features directly, without any transformations.

3.1.2 Combination of Features

To prevent a significant increase in time complexity when building a model, only combinations of OptionMetrics data were considered. Specifically, the averages of implied volatility for at-the-money calls, out-of-the-money puts, and out-of-the-money calls (`av_atmcall`, `av_otmput`, and `av_otmcall`, respectively) were combined to capture potential nonlinear relationships between implied volatility and stock returns.

Creating combinations of these columns allows for the exploration of more complex relationships between implied volatility and stock returns that may not be easily captured by individual columns.
New generated features:

- `av_atmcall_av_otmput` $\rightarrow$ `av_atmcall` $\times$ `av_otmput`
- `av_atmcall_av_otmcall` $\rightarrow$ `av_atmcall` $\times$ `av_otmcall`
- `av_otmput_av_otmcall` $\rightarrow$ `av_otmput` $\times$ `av_otmcall`
- `av_atmcall_av_otmput_av_otmcall` $\rightarrow$ `av_atmcall` $\times$ `av_otmput` $\times$ `av_otmcall`

3.2 Correlation & Multicollinearity

One way to identify the relevant features is to examine their correlations with the target variable or among each other. Figure 4 shows the correlation matrix between all features.

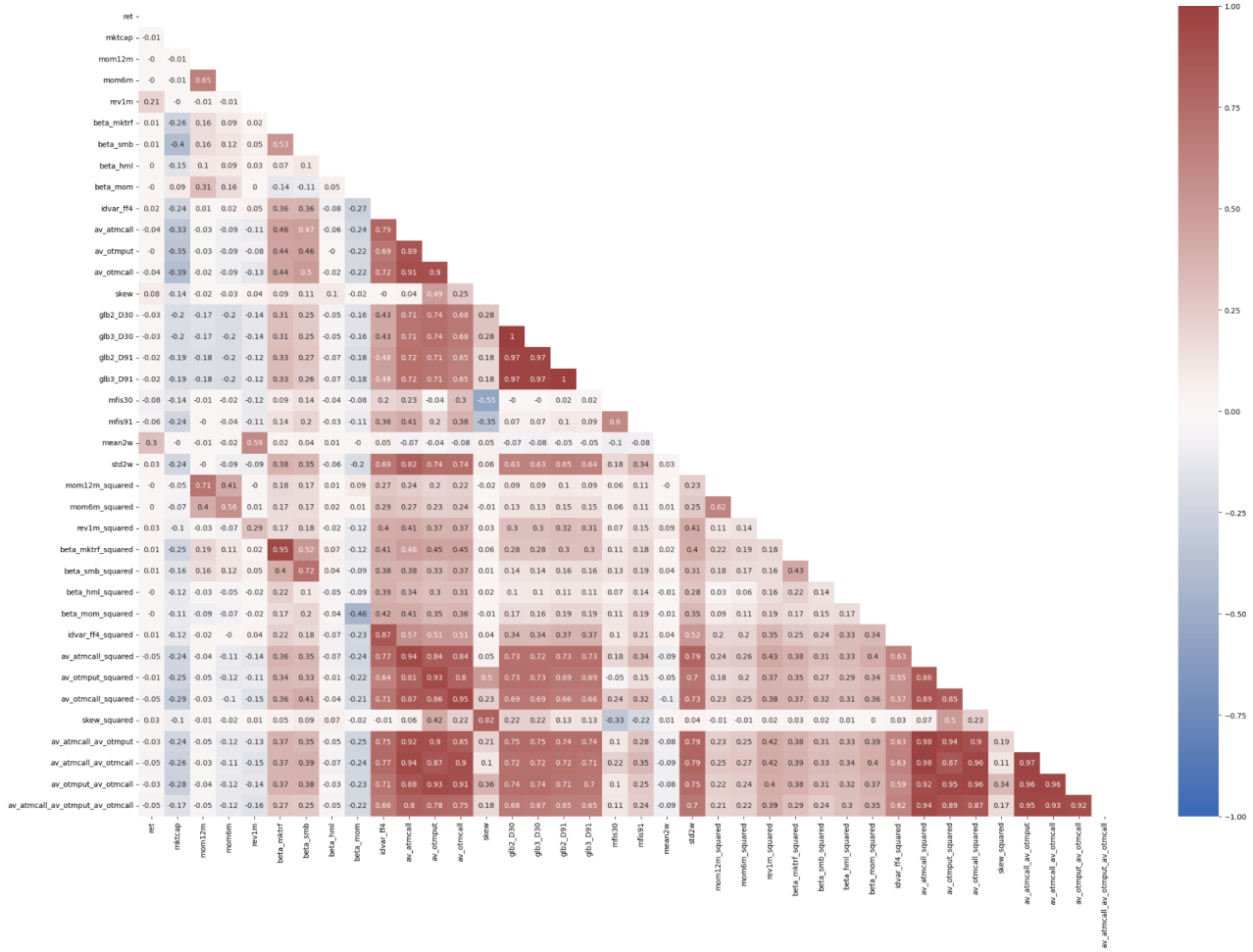

Figure 4: A correlation matrix is a matrix that shows the pairwise correlations between all the features in a dataset. Each entry in the matrix represents the correlation between two features. Diagonal entries were excluded from this matrix, as they represent the correlation of each feature with itself (which is always 1). All the GLB values are highly correlated, which suggests a high degree of linear association between these variables. The high degree of correlation could indicate redundancy or multicollinearity in the data. In this case, using all of the GLBs in a predictive model may lead to unstable or unreliable results due to the high degree of correlation between the variables. In the end, after doing the feature importance over time, only 4 GLB values were kept (`glb3_D30`, `glb3_D91`, `glb2_D30`, `glb2_D91`). Another finding is the correlation between the Option Metrics and GLB data.

To use the correlation matrix for feature selection, one can identify the features with the highest correlation with the target variable and select them for the model. However, it is crucial to remember that correlation does not necessarily imply causation, and there may be other factors that are important for predicting the target variable that are not highly correlated with it.

High correlations between features themselves can also lead to problems such as multicollinearity, which can affect the accuracy and stability of the model. Multicollinearity is a phenomenon that arises when two or more predictor variables in a regression model are significantly linked with one another, causing estimates of their coefficients to be unstable. This may lead to erroneous interpretations of the regression's results and inaccurate forecasts. Therefore, the report examines the correlation between variables in the dataset used to predict next-day returns. Testing for correlation could also be by indicating whether

variables are redundant and can therefore be eliminated or changed prior to running the regression model.

As a result, while choosing features for a machine learning model, it is important to take the correlation matrix into account along with other feature selection techniques and domain knowledge. For the first approach of the feature selection, the most relevant features were selected by examining the pairwise correlations between all features in the dataset. The pairs of most correlated features were identified, and to avoid multicollinearity, only the features that were more highly correlated with the target feature were selected.

3.3 Feature Importance Over Time

Another approach was explored to decide which features have the most impact on the target variable. We ran the Backtesting class firstly once with only XGBoost model on in-sample training data, just to create a daily dictionary of feature importances for every feature (keys of the dictionary are the days, every key stores another dictionary of feature importances, where the key is the name of the feature and the value is the importance of that feature on that specific day).

XGBoost calculates the feature importance using MDI (mean decrease impurity). The MDI method works by computing the total reduction of the impurity measure (such as Gini index or entropy) achieved by a feature in all decision trees of the XGBoost model. The greater the reduction, the more important the feature is considered to be.

Quick description of the method: The reduction of impurity is a measure of the improvement in purity achieved by splitting the data on a particular feature in a decision tree. It quantifies how much the impurity (e.g., Gini impurity or entropy) is reduced as a result of the split and helps to determine the optimal feature to use for the next split. The goal of the splitting process is to maximize the reduction of impurity at each step, ultimately leading to a tree that effectively separates the data into homogeneous subsets based on the target variable.

By construction, MDI has the nice property that feature importances add up to 1, and every feature importance is bounded between 0 and 1. Also it is important to remember that every feature will have some importance, even if they have no predictive power whatsoever [3].

3.4 Feature Selection

Firstly, the features with 0 mean value were excluded immediately. Those features are: `av_atmcall`, `av_otmcall_squared`, `idvar_ff4_squared`, `av_otmput_squared`. Figure 5 shows calculated feature importance for every day in the in-sample training data, excluding the already dropped features as mentioned above.

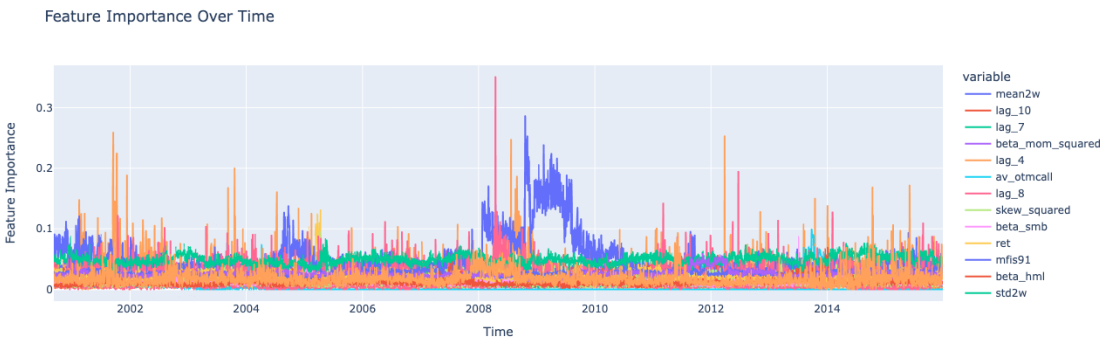

Figure 5: This plot shows feature importance (MDI) on insample data for all the features. There were too many features, so not all of them could be displayed in the legend on the right of the figure.

One of the main reasons for calculating feature importance with XGBoost is to identify the most informative features and reduce the time complexity of non-linear models (Random Forest and XGBoost). By

selecting the most relevant features, the model can focus on the most critical aspects of the data, which can lead to improved performance and faster training times. The idea was to drop the least important features over time. That was done by first calculating mean feature importance for every day (combining all the features) and then calculating the rolling mean (with window 21). Features below that mean value, would be considered irrelevant.

Figure 6 shows how lagged values for 1 and 2 days are above the calculated mean most of the times through the selected in-sample time period, and figure 7 displays some of the features that are around or below the mean value from 2000 to 2016.

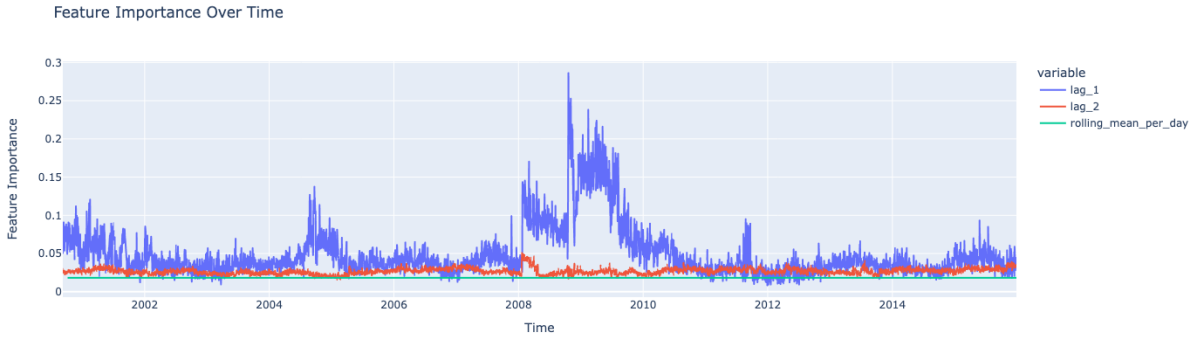

Figure 6: The plot shows the daily feature importance for 1 and 2 days lagged return values. Both lines are above the mean value for the most part of the in-sample period. As autocorrelation is a common phenomenon in financial time series data, where past values of the series are highly correlated with future values, these values had an impact on return predictions. Feature importance of lagged values is mostly above the mean value, so would indicate that the features might contain valuable information for predicting forward returns.

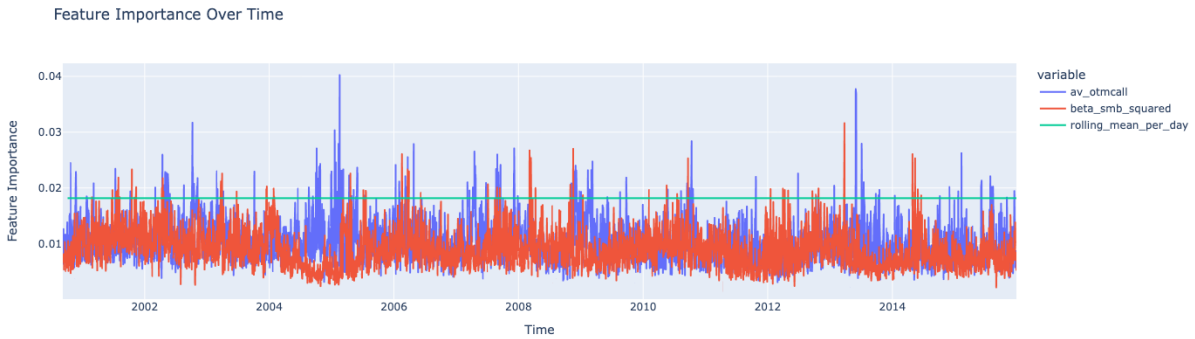

Figure 7: This plot shows an example of two features, which feature importance values were below the mean value more that 90% of the time. Features like that were excluded from nonlinear modeling.

As there were still many features left and the goal was to minimize the time complexity of the nonlinear models, a function was developed to determine whether the feature importance for each feature was above or below a rolling mean value for each day. The resulting dataframe provided the percentage of times each feature was below the mean value. Features that were below the mean vwalue more than 90% of the time were excluded from modeling feature selection for nonlinear models. Those features are shown in table 1.

Variable	Frequency in %
beta_smb_squared_check	98.407311
beta_mktrf_check	97.911227
av_otmcall_check	95.848564
beta_hml_squared_check	95.770235
mktcap_check	95.300261
beta_mom_squared_check	94.725849
mom6m_squared_check	93.994778
beta_hml_check	93.342037
beta_smb_check	93.054830
mom12m_squared_check	92.349869
av_otmput_check	92.036554
av_otmput_av_otmcall_check	91.827676
skew_squared_check	90.939948
mfis30_check	90.626632
beta_mktrf_squared_check	90.313316
av_atmcall_av_otmput_av_otmcall_check	90.130548

Table 1: Frequency represents the percent value of how many times was the importance of the feature below the rolling mean value. Displayed features in the table were below the mean more than 90% of the time, so they were excluded from nonlinear modeling.

After checking the correlation matrix and removing all the features that were above the mean value only 10% of the times, the features that were selected and used for nonlinear models are:

- Lagged features: `lag_1`, `lag_2`, `lag_3`, `lag_4`, `lag_5`, `lag_6`, `lag_7`, `lag_8`, `lag_9`, `lag_10`, `lag_11`, `lag_12`, `lag_13`, `lag_14`, `lag_15`, `lag_16`, `lag_17`, `lag_18`, `lag_19`, `lag_20`, `lag_21`
- extra calculated features: `mean2w`, `std2w`
- GLB data: `glb2_D30`, `glb2_D91`, `glb3_D30`, `glb3_D91`
- MFIS data: `mfis91`
- Option Metrics data: `av_atmcall_av_otmcall`, `av_atmcall_av_otmput`, `av_atmcall_squared`, `skew`
- factor betas: `beta_mom`, `idvar_ff4`
- fundamentals: `mom12m`, `mom6m`, `rev1m`, `rev1m_squared`, `ret`

4 Defining our Model Constraints & Trading Strategies

4.1 Model Constraints

We restrict the Fama French Four Factor betas of the portfolio to a range of 5% above and 5% below the betas of the benchmark portfolio. This is accomplished by passing an upper and a lower inequality constraint per beta as "cons" to the minimize function of the scipy.optimize package.

Using two inequality constraints, we also bound weights to be above 90% and below 110% of the weights in the benchmark portfolio.

Drawdown is limited to 99% of the drawdown of the benchmark per month. As we do daily prediction we take the previous 29 days of realized weights and returns into account meaning that the drawdown actually at all times is made up of historic drawdown so far and incorporates the effect of our predicted weights. The constraint therefore only binds, if the weights would produce a loss in expectation. It actually can be assumed, that this portfolio constraint does not come into effect in the optimization process for us, as we do one day ahead prediction, daily rebalancing and then the weights would not be selected such that there is a loss. We calculate drawdown as the maximum possible loss within the 30 day window which is comprised of 29 days in the past and one day ahead - the day for which we optimize a configuration function (e.g., MVP, with respect to wvec).

4.2 Trading Strategies

Trading Strategies are represented by objective functions we minimize with the minimize function of the scipy.optimize python library under aforementioned constraints.
They can be distinguished via differing models for the returns data and differing objective functions. Each trading strategy's objective function is minimized with respect to portfolio weights under the same constraints described in the previous section, producing our portfolio weights. The Trading Strategies are based on estimates of covariance, variance and expected return ("mu"). These estimates are all based on a cross-sectional regression of the estimates of returns one day ahead, where one day ahead return is regressed on the cross-section of the past 252 days. This is done via different Models described in section Model Explanation & Optimization. Our variance and covariance estimate is based on the current day's realized return. This implies that we know today's open and close price or return when deciding on the weights that are to be applied tomorrow, which is a reasonable assumption. The different optimization functions we use are MVP, MSR and LS.

1. **MVP:** Here, only the covariance estimate which is today's covariance, and portfolio weights constitute the objective function. Thus, for MVP the step of estimating tomorrow's return can be skipped and the models in 5 are irrelevant. Differences between our results and the results that can be obtained when rerunning the strategy on a different system can come only from not setting the same seed for the minimizer.

2. **MSR:** The MSR objective function maximizes the expected one day ahead return over the portfolio covariance estimate, which is today's covariance multiplied by the squared weights. It comes from Markowitz Portfolio Optimization and without the constraints would produce a portfolio optimal under the condition of normally and independently, identically distributed returns [1, pg.5].

3. **LS:** Last, we have a LS objective function (not delta-neutral) that maximizes Sharpe Ratio as objective function. The Long Short Strategy initializes a positive weights vector and a negative weights vector of the length of the number of permnos. Then weights are created such that the long and the short weights sum to one each and from that we create a weights vector that is normalized (=sums to one). It must be stressed that this objective function is constraint by the 3 constraints and therefore will not produce a delta neutral portfolio in the end, but the objective function serves the purpose of exploring a different objective together with our set constraints.

5 Model Explanation & Optimization

5.1 Model Explanation

The purpose of trying out different model architectures was to achieve a robust model and hyperparameter combination that could perform well in predicting cross-sectional stock returns using a 252-day rolling window.

1. **Lasso:** Also known, as L1 regularization, is a linear regression technique that aims to reduce the complexity of models and their features by shrinking the coefficients to zero. In contrast, to regular linear regression, lasso aims to prevent over-fitting by introducing a penalty term that reduces less important feature coefficients to zero, hence effectively eliminating them from the model. We decided to test lasso regression due to its ability to perform better in situations where the number of features is large relative to the number of observations, and because of its ability to perform both variable selection and regularization simultaneously. One possible drawback is the elimination of features completely as the eliminated variables could have captured future outliers better than the remaining features.

2. **Ridge:** A linear regression technique used to mitigate the dilemma of multicollinearity. Like lasso regression, ridge regression adds a penalty to the cost function that shrinks coefficients of less importance towards zero, but it never completely nullifies them. Due to the penalization term, ridge regression is able to handle multicollinearity between variables, leading to more stable and reliable predictions. However, unlike lasso regression it does not perform feature selection, in certain situations, this causes the model to be less interpretable or have reduced predictive accuracy, as well as increased training times.

3. **ElasticNet:** A combination of Lasso and Ridge regression. By adding a penalty term that is a linear combination of the L1 and L2 penalty terms, it achieves a balance between the two systems, meaning it will nullify irrelevant features, and apply regularization to the remaining ones. In theory, it should overcome the limitations of the previous linear models by handling multicollinearity and sparsity of data.One drawback is that, it must necessarily apply regularization, hence it could eliminate features in a situation where all variables are relevant for prediction, thus underfitting the data.

4. **Extreme Gradient Boosting Regressor:** XGboost is an ensemble learning method that offers several advantages when dealing with complex relationships between different stocks and their features that cannot be captured by linear relationships. Here is where XGBoost becomes the superior option as it not only excels at handling nonlinear relationships between variables but can also perform well in the presence of interactions between independent variables. However, as with any model, there are some limitations to this approach, one of them being the lack of visibility of the internal calculations when predicting and its respective computational time. Furthermore, any tree-based model is not well-suited for highly correlated variables since Tree-based models create data divides based on a single variable at a time and do not account for interactions between variables. In highly correlated datasets, multiple variables may be strongly correlated with the target variable, but only one will be chosen as the best predictor using XGBoost, while the others will be ignored. Hence why in the previous sections extensive feature importance selection was conducted.

5.2 Model Optimization

Once the model architectures to try out were selected, the next step was to optimize each of them to improve their predictive accuracy and generalization performance. The metric to minimize was the mean absolute error between the model's prediction and the actual returns for each of the permnos. The assumption was that the investing strategies would profit more from models with better predictive power, hence why reducing mean absolute error was imperative. In addition, optimizing the model for sharpe ratio or any other portfolio performance metrics, would increase the optimisation time from 20 min per per trial to 6 hours as it depends on scipys built in optimization whenever the function "prepare for optimisation" is used. Therefore, an if statement was added to omit the use of "prepare for optimisation" in the backtesting class as we only care about the predictios of the model and not the returns. Finally, it is to highlight that all of the models were optimized using only the in-sample data, hence limiting any bias of the model seeing data from the testing period. For this purpose, we utilized **Optuna**. A python library commonly used for hyper-parameter optimization that automates the process of finding optimal regularization values for the linear models. In short, Optuna evaluates a set of hyper-parameters, discards the worst-performing half, and then re-evaluates the remaining set of hyper-parameters until only one set of hyper-parameters remains. One drawback of this approach is that it requires large amounts of different trials per model to find optimal combinations of hyper-parameters, hence why more computationally expensive models such as ensemble methods and neural networks required considerably more time to optimize than linear models. Overall, 20+ optimization trials were performed with each model, except for the ensemble methods. All of the trials can be seen in the notebook "03 model optimizer", and the best performing hyperparameters/models can be seen in the table 2.

Ranking	Model	Optimized Parameters	Mean Absolute Error
01	Ridge Regression	Alpha = 0.037649	0.007882
02	Lasso Regression	Alpha = 0.000022	0.010906
03	Elastic Net	Alpha = 0.000237 / L1 Ratio = 0.227680	0.013419

Table 2: This table shows the hyper-parameters for Ridge, Lasso and Elastic Net that we got using Optuna.

It is important to mention that a model with a lower MAE does not necessarily lead to higher trading returns. This is because the MAE metric only measures the accuracy of the model's predictions and does not take into account other factors that can affect the profitability of a trading strategy. A model with a higher MAE may perform better if it provides more consistent predictions that are easier to act on in a trading strategy. For instance, compounding a downside deviation should actually be penalized more than an upside deviation from the prediction, as a loss of e.g., 1% will be only compensated by a gain of >1%, whereas a gain of 1% is lost by a subsequent loss of less than 1%. Essentially a risk averse investor prefers a model that predicts lower than realized returns over a model that predicts higher than realized returns, and lastly time dependency of the portfolio value are not reflected in MAE. Hence why the models are tested with its default input and optimized parameters. 15 combinations of models and trading strategies were tested on the in sample data, a sample of them can be seen in table 3.

Model	**Parameters**	**Trading Strategy**
Ridge Regression	Alpha = 0.037649	MSR
Ridge Regression	Default Values	MSR
Ridge Regression	Alpha = 0.037649	Long Short
Lasso Regression	Alpha = 0.000022	MSR
Lasso Regression	Default Values	MSR
Lasso Regression	Alpha = 0.000022	Long Short
Lasso Regression	Default Values	Long Short
Elastic Net	Alpha = 0.000237 / L1 Ratio = 0.227680	MSR
Elastic Net	Default Values	MSR
Random Forest	Default Values	MSR
All Linear Models	Optimised	MVP
All Linear Models	Default Values	MVP
Extreme Gradient Boosting	Default Values	MVP
Extreme Gradient Boosting	Default Values	MSR
Extreme Gradient Boosting	Default Values	Long Short

Table 3: Overview of tested model and trading strategies during in sample time period.

6 In-Sample Model & Trading Performance/Selection

In Section 5, we used Optuna to tune the hyperparameters of our linear models. We will test these models on the in sample period from 01/2000 until 12/2015.

During this section, we show the accomplishments of our best-performing models highlighted in the previous section when utilizing different trading strategies. In the end, the best-performing model and trading strategy will be utilized in the next chapter when evaluating and comparing the performance for the out-of-sample period from 1/2016 until 12/2021. Additionally, it is important to note that all of the linear models unless specified at the end with "default" are using the optimized values of regularization, regardless of whether "opt" is added at the end. This can also be noticed from Table 4 which provides a ranking of the different models tested. Table 4 shows that most of the optimized models do in fact beat their non-optimized counterparts in terms of portfolio performance measures and mean absolute error.

As portfolio performance measures, we use overall Maximum Drawdown, Smart Sortino Ratio, Sharpe Ratio, Calmar Ratio and Win Loss Ratio. All of these measures trade off risk and return which we want as we assume investors are risk averse but prefer higher returns over lower returns. We report and considered also using cumulative returns ("cum_returns"), the mean of negative returns from the strategy or benchmark ("avg_loss"), and annualized standard deviation ("vol_annualized"). However, we find there is too little variation in annualized volatility and average loss. Additionally cumulative return is not adjusted for risk and as we assume investors are risk-averse, the maximum cumulative return strategy is not equal to the one with the best risk-return trade-off.

Maximum Drawdown expresses the maximum possible loss. Nevertheless it doesn't consider the return potential of risky investments, thus e.g., Sharpe is a more adequate risk-return trade-off measure. The Sharpe Ratio is an optimal measure under the condition of normally and independently, identically distributed returns where investors care about deviation of realized return from an expected return both above and below [1, pg.5]. However, these assumptions are clearly violated, e.g., by the volatility clustering around the 2008 crisis in the in-sample period that indicates heteroskedasticity and it can be assumed that investors welcome positive deviations from expected returns. The Calmar Ratio is another Ratio that tries to put drawdown and return into relation to not penalize unforeseen positive returns but also take into account risks associated with strategies that have positive returns in expectation. The quantstats library implementation uses simply the CAGR over the total period maximum drawdown. The CAGR is the cumulative annualized growth rate calculated with discrete compounding. More complex but also to account for downside risk and return is the Smart Sortino Ratio based on the Sortino Ratio. The Sortino Ratio divides the difference of realized return from a target by the root mean square of the returns below the target (in our case target = 0) [4, pg.3]. The quantstats library version is "Smart" as it multiplies the denominator with a penalty for autocorrelation such that also local trends of the return time series are taken into account, unlike with the Calmar Ratio. This measure more than Sharpe represents risk aversion and preference for positive returns, as it doesn't penalize unexpectedly high returns making it a more suitable measure for risk-return trade-off for skewed returns [4, pg.2].

	model_&_strategy	max_drawdown	smart_sortino_ratio	cum_returns	sharpe_ratio	calmar	avg_loss	vol_annualized	win_loss_ratio
01	msr_lasso_opt	-37.225	0.573	121.2	0.44	0.143	-0.006	0.141	0.957
02	ls_xgboost	-37.295	0.571	120.64	0.439	0.142	-0.006	0.141	0.955
03	ls_ridge_opt	-37.493	0.571	120.54	0.439	0.141	-0.006	0.14	0.957
04	mvp	-37.508	0.574	121.23	0.441	0.142	-0.006	0.14	0.956
05	msr_ridge	-37.733	0.557	115.78	0.428	0.137	-0.006	0.141	0.953
06	msr_ElasticNet_default	-37.862	0.573	121.26	0.44	0.141	-0.006	0.14	0.954
07	ls_lasso_opt	-37.93	0.566	118.58	0.435	0.138	-0.006	0.14	0.954
08	msr_lasso_defaul	-37.945	0.561	117.06	0.431	0.137	-0.006	0.141	0.956
09	msr_xgboost	-38.142	0.559	116.55	0.429	0.136	-0.006	0.141	0.961
10	msr_rforest	-38.151	0.56	116.88	0.43	0.136	-0.006	0.141	0.956
11	msr_lasso	-38.425	0.557	116.04	0.429	0.134	-0.006	0.141	0.954
12	msr_ElasticNet_opt	-38.497	0.563	117.87	0.433	0.136	-0.006	0.141	0.956
13	msr_ridge_default	-38.593	0.555	115.03	0.427	0.133	-0.006	0.141	0.956
14	benchmark	-39.222	0.585	132.45	0.448	0.144	-0.007	0.147	0.958

Table 4: In-Sample Results Metrics

1. **MVP**: In-sample MVP performs surprisingly well according to Max. Drawdown, Sharpe Ratio, Calmar and Smart Sortino Ratio. It performs best after benchmark according to Smart Sortino and Sharpe Ratio. This is somewhat surprising. Our estimate of the covariance, which is just current covariance on the day weights are computed for the next day might alone be better more precise or a better suited objective function given the constraints when on its own as opposed

to when used with the forward return estimates. Furthermore, due to a near zero interest rate environment over large parts of our in-sample period, the MVP (without constraints) is close to the Maximum Sharpe Ratio portfolio. Additionally, there is empirical evidence, that the MVP (without constraints) outperforms even mean-varaince optimized portfolios, possibly by selecting low beta stocks over high beta stocks [2, pg. 178]. Therefore it is possible, that this result is generalizable beyond our permno selection.

2. **MSR**: We fit 9 models with objective function MSR.
   The MSR strategy with Lasso Regression tuned by optuna ("=msr_lasso_opt") achieves the highest ratio of average positive return over average negative return ("win_loss_ratio"), but has below average Smart Sortino Ratio and more than average Max. Drawdown.
   The MSR constraint strategy with Lasso Regression achieves the least Max. Drawdown of all strategies with also less drawdown than the benchmark. It's Calmar Ratio is the highest among our strategies which is consistent with the low drawdown and above average Smart Sortino and Sharpe Ratios.

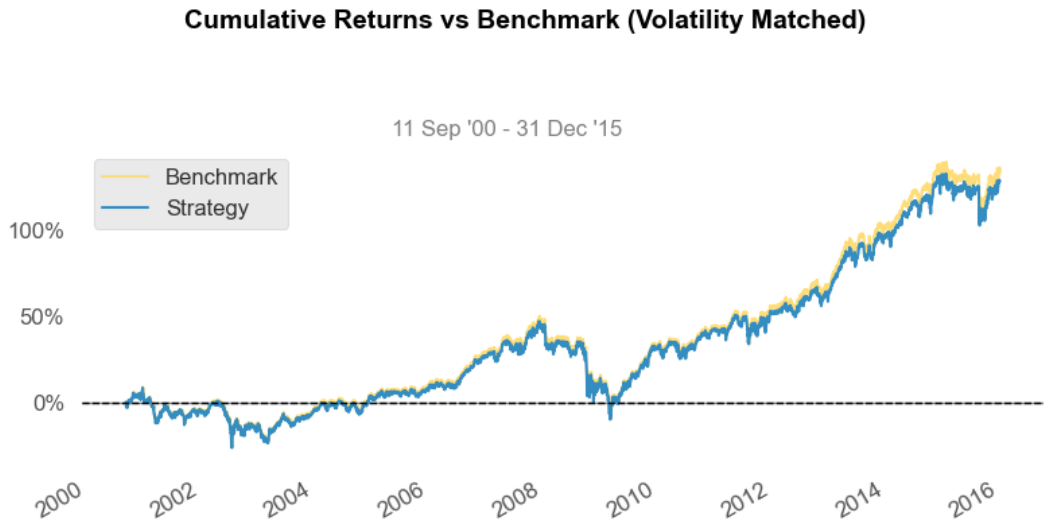

Figure 8: In sample msr_lasso_opt cumulative return volatility-adjusted by normalizing and multiplying by benchmark standard deviation are higher than the benchmark's returns.

3. **LS**: Our long-short strategies performs well in all performance measures, however are outperformed by another strategy or the benchmark everywhere. Additionally 2 out of the 3 long short return prediction model pairs we tried were below average when it come to Win Loss Ratio and the 3rd one did also not convince when it comes to overall metrics.

MVP performs well across most of our performance metrics but is outperformed by the benchmark or another strategy at the top of the ranking per metric consistently. The LS strategies suffer the same issue. Thus, the final strategy considered is MSR based on return estimation from Lasso Regression hyperparameter tuned by Optuna. As can be seen in Figure 9, the benchmark has higher return than our strategies, however Figure 10 shows that this comes at a higher drawdown consistently. We prioritize low drawdown higher than high returns, as due to the path dependency of the portfolio returns an x percentage return can be lost by a subsequent y percentage loss, where y is less than x. Looking at the measures and also from the plots, we decide that msr_lasso_opt performs most consistently well across our performance measures. It has a good risk-return trade-off with good downside protection as can be seen e.g., in its Calmar Ratio. Thus, this is the strategy used in Section 7.

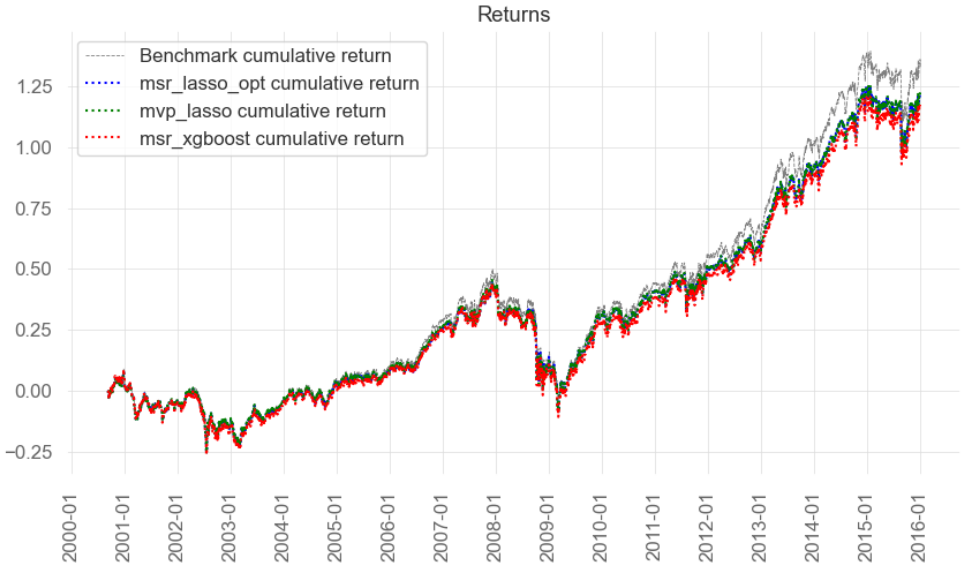

Figure 9: In-sample performance of our 3 best performing trading strategies

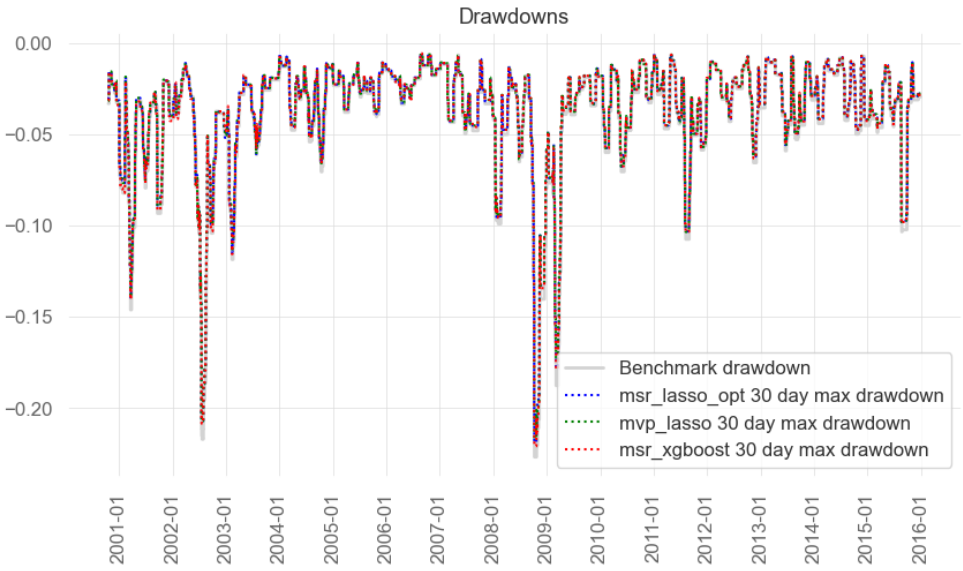

Figure 10: In-sample performance of our 3 best performing trading strategies according to 30 day Max. Drawdown.

7 Out-of-Sample Model & Trading Performance

In Section 6 we concluded that our best performing model on the in-sample data was the Optuna optimized Lasso regression. Furthermore, our best performing trading strategy was optimizing our weights for the maximum Sharpe Ratio (MSR). Now, we apply this prediction model and trading strategy on the out of sample data to obtain returns and PERMNO weights for the portfolio. This data covers January 2016 to December 2021. Our metrics and cumulative returns for the chosen model and trading strategy vs. the benchmark can be seen in table 5.

	model_&_strategy	max_drawdown	smart_sortino_ratio	cum_returns	sharpe_ratio	calmar	avg_loss	vol_annualized	win_loss_ratio
1	msr_lasso_opt	-27.621	1.201	135.43	1.031	0.556	-0.006	0.149	0.938
2	benchmark	-29.385	1.173	140.26	1.011	0.536	-0.006	0.157	0.928

Table 5: Out-Sample Results Metrics

msr_lasso_opt outperforms the benchmark for our core metrics. This is especially apparent when observing the maximum drawdown value. Our created portfolio experiences less severe losses than the benchmark. The portfolio's Smart Sortino Ratio is also higher than the benchmark indicating better risk-adjusted returns. Although the benchmark's cumulative returns is higher than the portfolio, the portfolio may be more appealing to investors who are risk adverse. Since cumulative returns is not risk adjusted, the created portfolio using msr_lasso_opt performs better than the benchmark when factoring in portfolio risk.

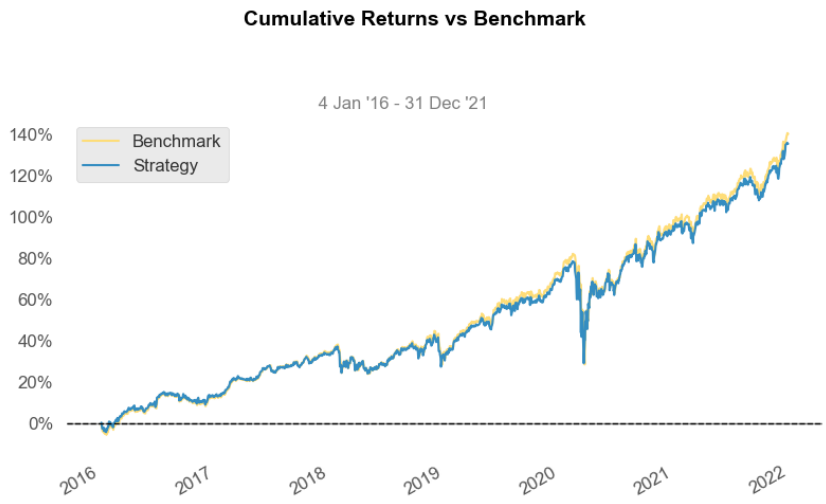

Figure 11: Out-Sample returns of our strategy and benchmark

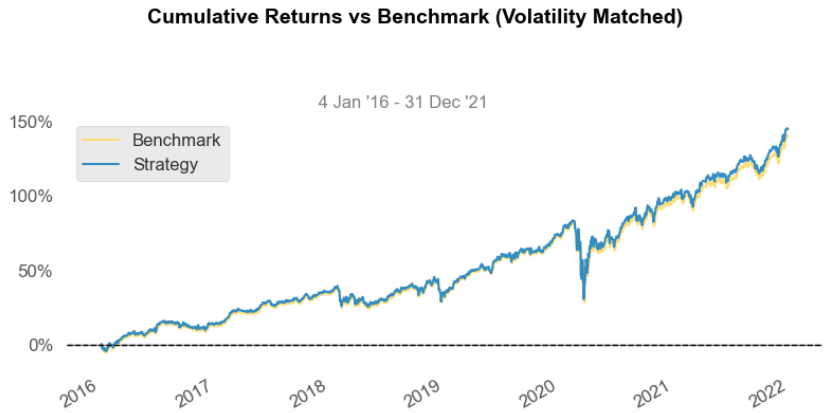

Figure 12: Out-Sample returns adjusted for volatility by normalizing and mulitiplication with benchmark standard deviation.

After adjusting for volatility, our strategy also visibly outperforms the benchmark as seen in Figure 12. Since there is higher volatility in the out of sample data than the in sample data, it becomes more apparent that our chosen strategy beats the benchmark. This can be observed in all of the metrics from Table 5, such as Sharpe Ratio, Win Loss Ratio, Smart Sortino and Calmar. All of these metrics are higher than the benchmark because they are all related to volatility in some way. Thus, we conclude that the created strategy outperforms the benchmark on the out of sample data.

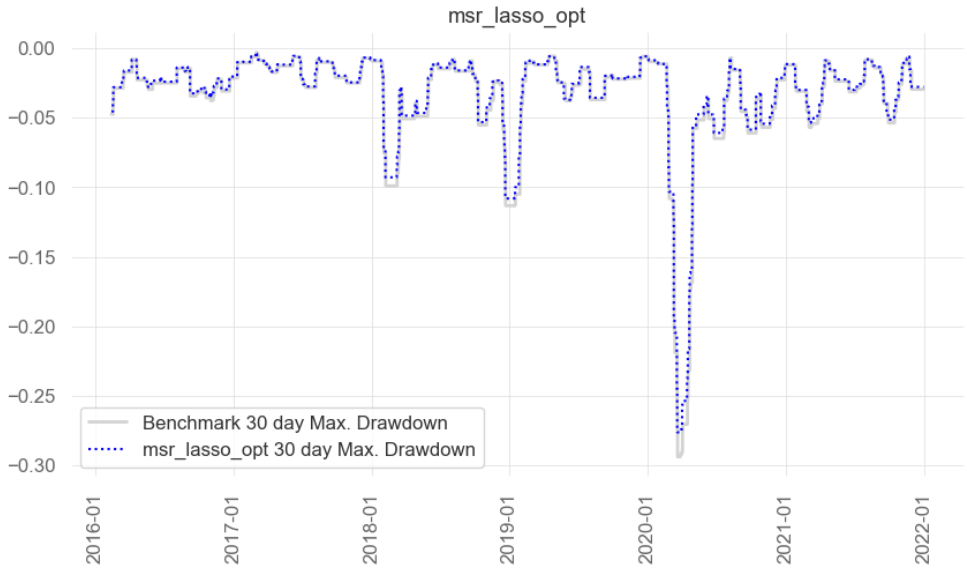

Figure 13: The 30 day drawdown is less severe for our strategy as opposed to the benchmark.

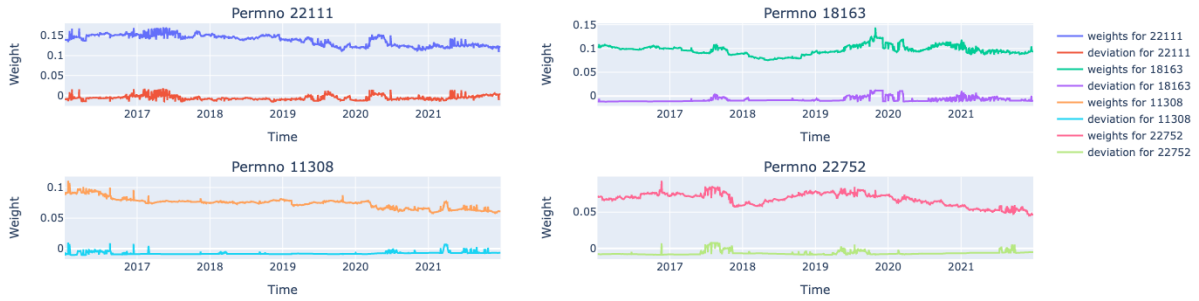

Figure 14: This figure displays a sample of 4 selected PERMNO weights and their deviation over time from the benchmark weights. The full image of weights can bee seen in the appendix. However, it is worth noting that periods of higher deviations in 2017 and 2019 seem to co-occur accross this selection of permnos. In further studies this could be worth investigating.

8 Conclusion & Outlook

We successfully created quant trading strategies that outperform a market-cap-weighted portfolio of 50 stocks from the S&P500 universe, during the out of sample period from 01/2016 to 12/2022. This was achieved by optimizing different models and strategies during the in sample period from 01/2000 to 12/2015 all while adhering to the following constraints:

1. Maximum factor exposure deviation of 0.05 for each considered factor.
2. Maximum weight deviation of 10% of the BNCH weight for each asset
3. Maximum drawdown relative to BNCH of 1% per month.

During our process it has become apparent that replacing the scipy optimizer with something faster might enable us to try even more trading strategies, as most of the compute time was spent on optimizing the objective functions of the trading strategies under the given constraints. Furthermore, as we believe rolling drawdown is a more accurate measure for risk than volatility, trading strategies using an objetive function that minimizes a rolling drawdown-return trade-off could be further explored. Finally, trying out different data sources, rebalancing frequencies, and even more compact constraint notation to assist the optimizer might yield a better trading performance.

In our trials, the best performing Strategies under mentioned constraints used Extreme Gradient Boosting Regression and Lasso for return prediction and were minimum variance portfolio or maximum Sharpe Ratio portfolio strategies. Due to overall performance, espectially Max. Drawdown and Calmar Ratio we chose the Optuna hyperparameter tuned Lasso Regression model with Maximum Sharpe Ratio based on performance from 01/2000 to 12/2015 as our best in sample strategy. This strategy then was evaluated from 01/2016 to 12/2021 where the strategy and model combination outperforms its benchmark even more than in the in-sample period. During this period the model was not changed or adjusted. The increased out-performance is likely due to higher volatility since 2016.

Past performance is not guaranteed to indicate future performance and we investigated this strategy specifically for a subset of constituents of one US market cap weighted equity index and also ignores transaction costs and liquidity constraints. Nevertheless, in this setup our chosen strategy yielded promising results, outperforming the benchmark in- and out of sample.

References

[1] David H Bailey and Marcos Lopez de Prado. The sharpe ratio efficient frontier. *Journal of Risk*, 15(2):13, 2012.

[2] Ziemowit Bednarek and Pratish Patel. Understanding the outperformance of the minimum variance portfolio. *Finance Research Letters*, 24:175–178, 2018.

[3] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley Publishing, 1st edition, 2018.

[4] Thomas N Rollinger and Scott T Hoffman. Sortino: a 'sharper'ratio. *Chicago, Illinois: Red Rock Capital*, 2013.

A Additional Plots

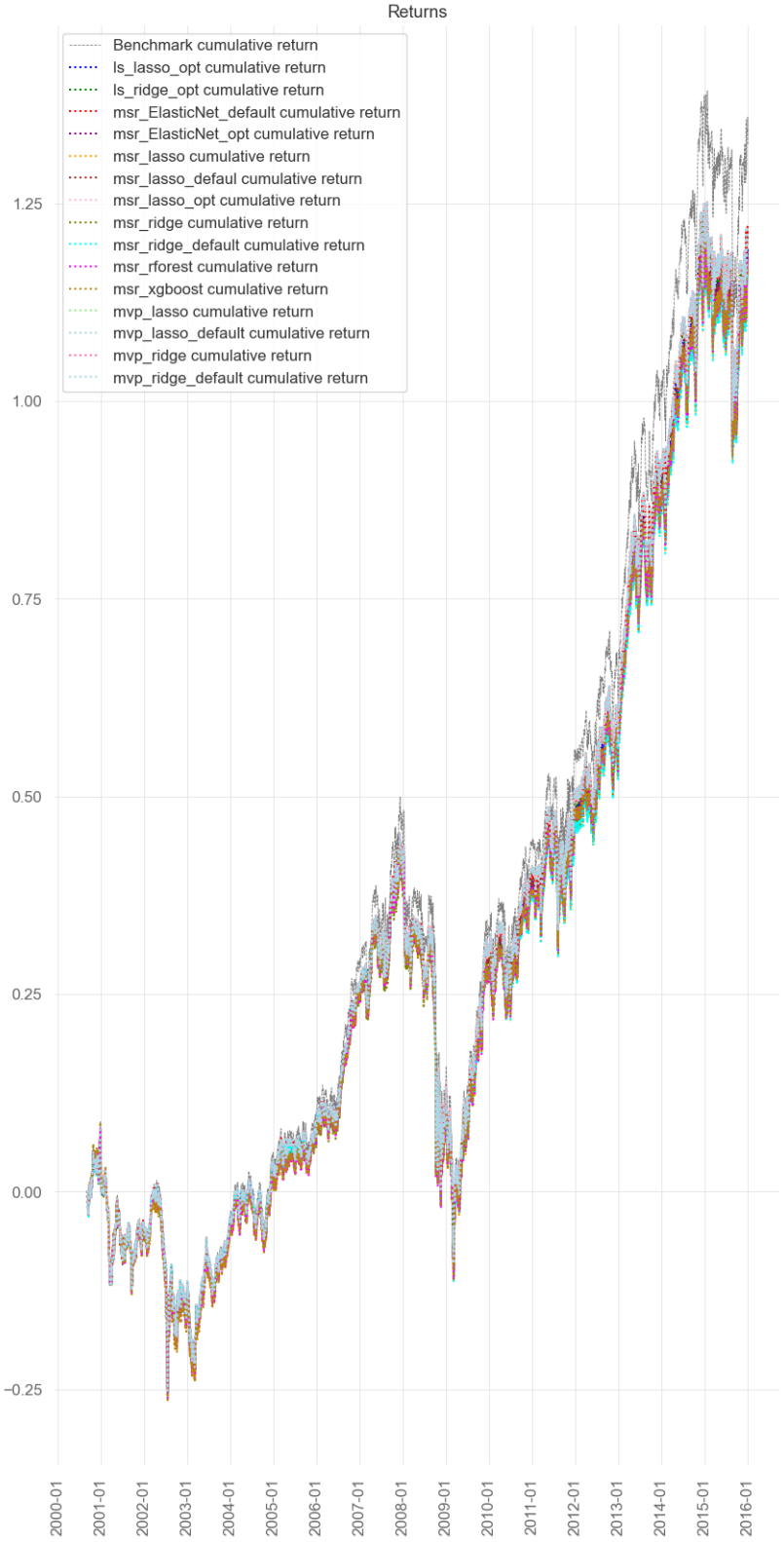

Figure 15: In-sample cumulative return of all considered trading strategies. It can be seen that over time different strategies result in different cumulative return. Although it seems like our strategies are dominated by the benchmark when it comes to performance when looking at return only, Table 4 and Figure 18 illustrate that the benchmark is not the best choice for a risk-averse investor.

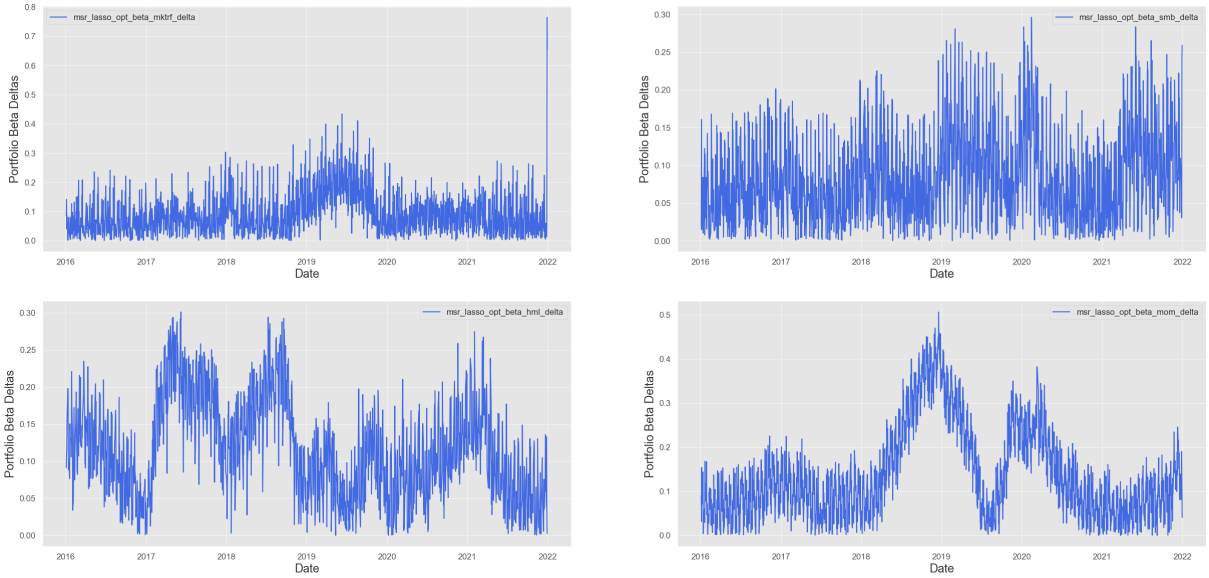

Figure 16: This figure displays the delta between the portfolio and the benchmark betas for the out of sample data. The portfolio uses the msr_lasso_opt strategy.

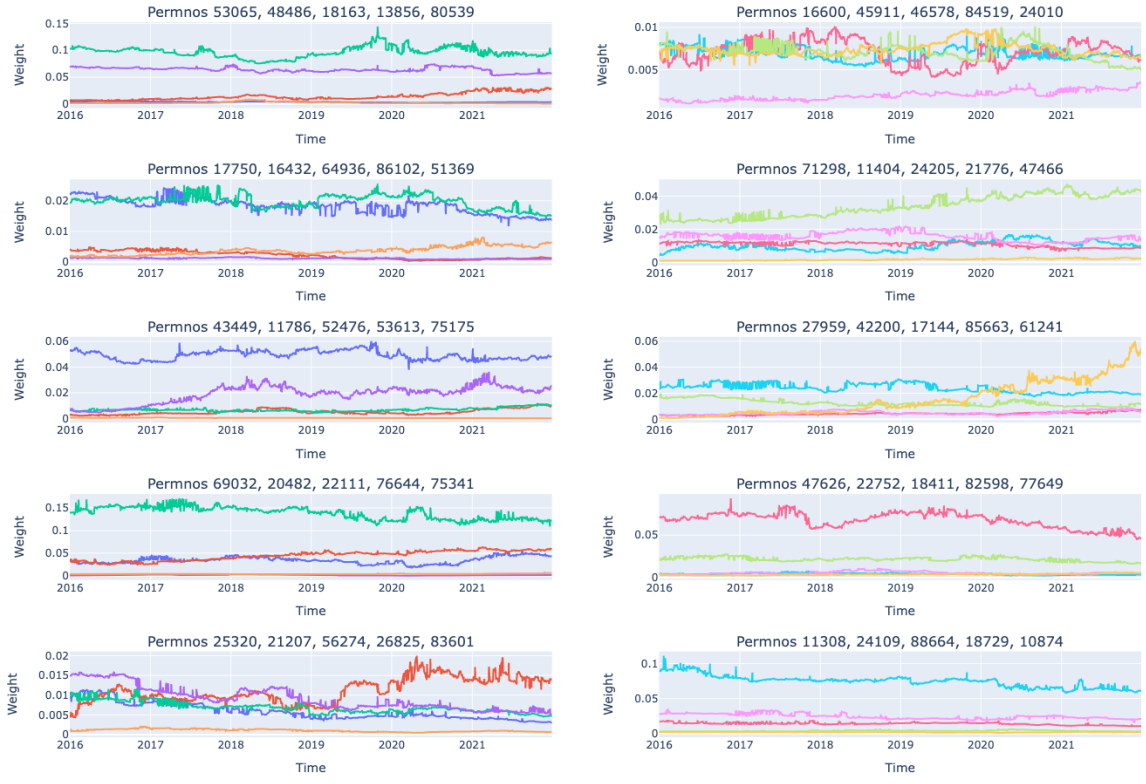

Figure 17: This figure shows weights over time for all the permnos. There are only 5 lines on each plot, otherwise the plot would be unreadable. It can be seen that although we implicitly allow for short selling, due to our constraints we do not end up with large short positions in our portfolio over time.

Figure 18: In-sample drawdown of all considered trading strategies. All of our considered strategies achieve less severe drawdowns than the benchmark.