UC Berkeley, Haas School of Business, Master of Financial Engineering, (2023)

META ANALYSIS ON REGIME DETECTION AND DYNAMIC PORTFOLIO CONSTRUCTION

YVES D'HONDT, MATTEO DI VENTI, ROHAN RISHI and JACKSON WALKER

Abstract. This paper presents an in-depth exploration of regime-dependent portfolio optimization and introduces the Python package MarketMoodRing as a tool to implement different regime detection models and portfolio optimizers. Hidden Markov Models (HMM) and Wasserstein K-Means are investigated for regime detection, with corresponding one-period and stochastic programming models for portfolio optimization. This study finds that regime-based portfolio optimization outperforms traditional mean-variance optimization and equal risk contribution between 2004 and 2023. Concretely, Stochastic programming, optimizing for maximum expected Sharpe ratio, in combination with Gaussian Mixture Model HMM performs the best out of all strategies under investigation. This paper contributes to a better understanding of the potential of and challenges with regime-dependent portfolio optimization.

§1. Introduction

The mean-variance model introduced by Markowitz [20] is the basis of modern portfolio construction. It is relatively unrestricted in the types of assets it can handle as it only relies on the expected returns and variance-covariance matrix of the assets. However, in practice the model often fails to realize its theoretical properties. Firstly, it assumes a constant mean and variance-covariance while empirical evidence suggests these parameters are non-constant [21]. Secondly, it is a one-period model that does not incorporate any stochastic features. Finally, standard mean-variance optimization (MVO) often produces non-robust portfolio weights due to parameter uncertainty [25]. As a result, various researchers and practitioners have extended Markowitz's model. One such extension incorporates the idea that the market is characterized by different regimes across time with distinct return distributions.

This paper aims to provide an overview on the current state of regime detection and dynamic portfolio construction, while also evaluating the strengths and weaknesses of different techniques. Furthermore, this paper expands on the field by investigating the performance of different regime-dependent portfolio optimization techniques using Wasserstein K-Means regime detection, as first proposed by Horvath, et. al. [17].

§2. Literature Review

Tu provides evidence that there are losses associated with ignoring regime switching and that accounting for regime switching is substantially independent from incorporating model and parameter uncertainty in portfolio decisions [23]. Therefore, Tu argues that "the more realistic regime switching model is fundamentally different from the commonly used

Keywords: market regimes, portfolio optimization, optimal transport, HMM, quantitative finance

single-state model, and hence should be employed instead in portfolio decisions irrespective of concerns about model or parameter uncertainty" [23]. In line with Tu's observations, there have since been numerous papers that argue for incorporating regime dependency in investment decisions.

2.1. Regime Detection

A key assumption behind regime detection is that markets are characterized by a finite number of latent, or hidden, regimes, each with their own return generating process, and at any point in time returns are generated from one of these regimes. Regime detection methods can be grouped into two categories: parametric and non-parametric methods.

The Hidden Markov Model (HMM) has been a popular parametric model to detect latent regimes and has been studied in numerous financial applications [13][14][15][24]. Within the context of dynamic portfolio construction, Bae, et al. [1] and Costa, et al. [8][9] provide evidence in favor of using HMMs for regime detection.

Advances in machine learning have also led researchers to start investigating non-parametric models in latent regime detection. Bilokon, et al. [4] suggest using path signatures on returns time series in combination with a modified K-means algorithm to detect regimes. In similar fashion, Horvath, et al. [17] suggest a modified K-means algorithm using Wasserstein distance and Wasserstein Barycenters to detect regimes. This is based on the field of optimal control and helps to define distances between and aggregations of probability distributions.

2.2. Dynamic Portfolio Construction

An important feature of HMMs is that they provide transition probabilities between the different states over time, given their interpretation as a state-transition model. This allows for stochastic portfolio optimization as regime sequences can be simulated from the fitted Markov chain. Bae, et al. use stochastic programming to construct optimal portfolios. They conclude that "the regime information helps portfolios avoid risk during left-tail events" [1]. One major drawback of stochastic programming is that it suffers from the *curse of dimensionality* and is therefore time and resource intensive when applied to a broad universe of assets or many states.

To overcome this drawback, Costa, et al. [8] suggest to use a simpler one-period model. They propose a regime-switching factor model, based on the Fama-French 3 Factor Model, that fully characterizes the systematic portion of the expected returns and covariance matrix of an asset universe at each point in time. They provide evidence that a risk-parity strategy based on this model offers higher returns at a similar ex-post level of risk compared to its nominal counterpart [8].

In a follow-up paper, Costa, et al. [9] propose a regime-switching factor model that allows for both systematic and idiosyncratic regime-dependency. They show that MVO using this novel framework consistently displays higher returns and similar to lower volatility than its nominal counterpart [9].

§3. Methods Under Investigation

3.1. Regime Detection

3.1.1. Hidden Markov Models

Introduced by Baum in 1966 [3], Hidden Markov Models (HMM) are a framework to model and forecast time-series data generated from a number of undetectable *latent* states which each generate data from a state-dependent distribution. Moreover, the model incorporates a static *transition matrix* which describes the probabilities of a state change from one period to the next. A data generation process for each of the latent states together with a transition matrix defines an entire HMM model.

Following [18], a general Hidden Markov Model is defined in detail by:

1. A sequence of **observations** from a (multivariate) time series $\{r_t\}_{t=1}^{T}$ (returns in the context of this paper)
2. A set of K **latent states** $Q_1, Q_2, ..., Q_K$ (market regimes in the context of this paper)
3. A transition probability matrix $A_{ij} = P(S_t = Q_j | S_{t-1} = Q_i)$
4. A sequence of **observational likelihoods**, called emission probabilities $B = b_i(r_t)$, that each express the probability of observation r_t being generated from state i
5. An **initial probability distribution** over the states $\pi \in \mathbb{R}_+^K$ such that $\pi^T \mathbf{1} = 1$

The first order Hidden Markov Model used in this paper makes use of two additional assumptions:

1. **Markov Assumption**: $P(s_t | s_1, \ldots, s_{t-1}) = P(s_t | s_{t-1})$
2. **Output Independence**: $P(r_t | s_1, \ldots, s_T, r_1, \ldots, r_T) = P(r_t, s_t)$

Rabiner [22] characterises HMM by three fundamental problems:

1. **Likelihood estimation**: Given an HMM, (A, B), and an observation sequence $\{r_t\}_{t=1}^{T}$, determine the likelihood $P(O|A, B)$
2. **Decoding**: Given an HMM, (A, B), and observation sequence $\{r_t\}_{t=1}^{T}$, determine the best hidden state state sequence $\{s_t\}_{t=1}^{T}$
3. **Learning**: Given observation sequence $\{r_t\}_{t=1}^{T}$, and the set of states in the HMM, determine the best parameters $\theta = (A, B)$

The learning and decoding problems are of primary interest for latent regime detection. By prescribing the number of latent states and the emission distribution, the set of parameters, $\theta = (A, B)$, can be learned through the forward-backward algorithm, also known as the Baum-Welch algorithm [2], a special case of the Expectation-Maximization algorithm.

The Python package *hmmlearn* was used to fit HMMs. This package allows for control over two crucial hyperparameters: the *covariance type* of each regime's return distribution and the *emission model*. For the covariance type, this paper considers both *diagonal* matrices, in which every state's covariance matrix is diagonal, and *full* matrices, in which every state's covariance matrix is unrestricted. For the emission model, this paper considers both *Gaussian* emissions, in which each state's returns are generated from Gaussian distributions, and *Gaussian Mixture* emissions, in which each state's returns are generated from Gaussian Mixture Models.

3.1.2. Wasserstein K-Means

Wasserstein distance and barycenters are closely related to the *optimal transport* problem. Intuitively, one can think of the problem of moving a mountain of mass m into a hole that exactly fits a mass m. In a 1-dimensional setting, the 1-Wasserstein distance answers the question what the minimal cost is to complete this problem when the cost of moving a mass $\gamma(x,y)dxdy$ is defined as $c(x,y)\gamma(x,y)dxdy$, where $c(x,y)$ is the distance between point x and y. Similarly, the 2-Wasserstein distance answers the same question in a 2-dimensional setting.

Rigorously, the Wasserstein p-distance [19] is defined between two probability measures $\mu_1(x)$ and $\mu_2(y)$ on the same metric space. A necessary condition for the p-distance is that both distributions have finite p-moments. The distance is then defined as :

$$W_p(\mu_1,\mu_2) = \Big(\inf_{\gamma\in\Gamma(\mu_1,\mu_2)} \mathbf{E}_{(x,y)\sim\gamma} d(x,y)^p \Big)^{1/p}$$

where $\Gamma(\mu_1,\mu_2)$ is the set of all joint probability functions whose marginal according to x, y integrate respectively to $\mu_1(x)$ and $\mu_2(y)$ and $d(x,y)$ is the underlying distance metric of the metric space. From the previous context, one can identify the optimal transport plan in the minimizing joint probability distribution and the cost function for transporting from point x to point y in metric distance.

In the context of distributions, one can think of a distribution as being represented by its probability density function or histogram, in approximation. Now, the Wasserstein distance can be used to define the distance between two distributions as the cost it takes to move one distribution into the other distribution. In this case, the *p-Wasserstein* distance will primarily be driven by differences in the first p moments of the two distributions. For example for distributions centered around 0, the 2-Wasserstein distance would primarily account costs for differences in the mean and standard deviation of two distributions, while the 4-Wasserstein distance would also account costs for differences in the skewness and kurtosis of two distributions. Finally, Wasserstein barycenters provide a way to aggregate a collection of distributions by constructing a single distribution which aims to minimize the p-Wasserstein distance to all the distributions in the collection.

Using the ideas outlined above, Horvath, et al. [17] defined Wasserstein K-means. This technique is defined as standard K-means, but the p-Wasserstein distance is used as the distance metric between two points, and cluster centroids are defined as the p-Wasserstein barycenter of all distributions in a cluster. Due to the atypical distance metric and centroid construction, pre-existing K-means packages could not be used to implement this novel algorithm. Instead, a Wasserstein K-means implementation was built from scratch, relying on the *POT* package in Python to calculate Wasserstein distances and barycenters.

The Wasserstein K-means algorithm has a number of important hyperparameters: p, *n_bins*, *window_size* and *kde_smoothing*. p controls which p-Wasserstein distance is to be used. Standard values range from 2 to 4 for this parameter. *n_bins* controls the number of histogram bins that are to be used to represent the distributions. A higher number of bins leads to a higher granularity of the histogram, but has the drawback that it requires more data to create "well-behaved" histograms. *window_size* controls the lookback window of the time series to sample distributions. A larger lookback window incorporates more historical data and leads to smoother histograms but could incorporate too much stale data. Finally, *kde_smoothing* controls how much, if any, smoothing should be applied to the histograms. Smoothing is applied by overlaying a Gaussian KDE on top of the sampled histogram. Smoothing has the advantage that a small amount of data can still give "well-behaved" histograms, but on the flip side smoothing can lead to information loss on higher moments of the distribution. Although Wasserstein K-means was initially developed to work on intraday data, it will be applied on daily data in the remainder of this paper due to data access limitations. Histogram smoothing was added to the implementation to partially account for the lower data frequency that arises from using daily over intraday data.

Finally, unlike HMMs, non-parametric models, such as K-means, do not offer transition probabilities out of the box as they are built in static environments and perform hard clustering. Instead, transition probabilities are backed out empirically after detecting the regimes.

3.2. Dynamic Portfolio Construction

3.2.1. Stochastic Programming

Bae, et al. outline an example of stochastic programming for regime dependent portfolio optimization [1]. This paper presents a simplified, but generalized way to perform stochastic programming for regime dependent portfolio optimization. The proposed stochastic programming algorithm is described in Listing 1. First, a Monte Carlo simulation is run to generate N state sequences of length L. Second, it calculates the expected distribution of returns over each state sequence. Finally, an optimization step can solve for a global objective over all sampled distributions.

In this paper's implementation, the first step uses a transition probability matrix generated by one of the aforementioned regime detection methods to sample state sequences from a Markov chain. The length of the sampled sequences can be set equal to the rebalancing frequency of the underlying strategy. For instance five trading days for a strategy with weekly rebalancing. Next, each state's return distribution is assumed to be multivariate normal. As a result, the expected distribution of returns over each state sequence is simplified to the average of the return distributions of the different states, weighted by the frequency of each state in the sampled sequence. Finally, a numerical optimizer is used to solve for the weights that give either the maximum average Sharpe Ratio or the minimum average Value at Risk over the entire sample.

The stochastic programming implementation described above has two main advantages. First, the normality assumption of each state's underlying distribution allows for the entire algorithm to be vectorized and hence makes the algorithm computationally efficient. Secondly, the global optimization step allows to optimize for various objectives, such as expected Sharpe Ratio, expected VaR, maximum-minmum Sortino Ratio, etc. This leads to a more general algorithm as opposed to Bae, et al. [1] as no assumptions on the underlying utility function of the investor are required.

```
input: int N, int L, float[:,:] trans_mat,
       float[:,:] mu, float[:,:,:] sigma
output: float[:] weights

define sample_distribution(L, trans_mat, mu, sigma):
  # Generate random state sequence of length L
  state_sequence ← generate_sequence(length=L, trans_mat)

  # Initialize simulated mu and sigma
  new_mu ← 0
  new_sigma ← 0

  # Calculate the average mu and sigma weighted by state frequency
  for s in state_sequence:
    new_mu ← new_mu + mu[s]
    new_sigma ← new_sigma + sigma[s]
  new_mu ← new_mu / L
  new_sigma ← new_sigma / L

  # Return simulated mu and sigma
  return (new_mu, new_sigma)

define stochastic_programming(N, L, trans_mat, mu, sigma)
  # Simulate N regime−dependent distributions
  distributions ← []
  for _ in range(N):
    distribution.append(sample_distribution(L, trans_mat, mu, sigma))

  # Optimize portfolio weights
  return optimize_weights(distributions)
```

Listing 1: Pseudo-algorithm to perform regime dependent portfolio optimization through stochastic programming. *generate_sequence* generates a state sequence of length L given a transition probability matrix. *optimize_weights* finds the optimal weights for an objective defined over the entire sample of simulated distributions.

3.2.2. Factor Based Optimization

As a computationally efficient alternative to stochastic programming, Costa, et al. propose a one-period model for regime dependent portfolio optimization [8]. First, they assume that asset returns are generated by a factor model, such as the Fama-French 3 Factor model. Second, they assume that the relationship between factor returns and asset returns is different between and constant within regimes. This leads to a regime-dependent factor model of stock returns. Let r_t be the returns of an asset at time t, V_i be the factor loadings in state i, f_{it} the factor returns in state i at time t, and I_i an indicator function which is 1 if the current state is i and 0 otherwise. Then the regime-dependent factor model can be described by the following equation [8]:

$$r_t = \alpha + I_1(V_1^T f_{1t}) + I_2(V_2^T f_{2t}) + \epsilon_t$$

Note that this equation assumes that both the constant α and model error ϵ_t are regime-independent. This factor model can now be estimated through standard OLS after transforming the data to account for the state indicator function. Using this model, in combination with a regime detection model and the corresponding transition probability matrix, the regime-dependent expected returns and covariances of each asset under consideration can be calculated in closed-form [8]. These expected returns and covariances can then be used as inputs for classical portfolio optimization, such as mean-variance optimization (MVO) or equal risk contribution (ERC).

While this leads to a computationally efficient model, a drawback is that this closed-form solution only gives the expectation for the next period. In practice, this means that there can be a mismatch between the rebalancing frequency of a trading strategy and the expected parameters over which the portfolio weights are optimized. Nonetheless, this simple one-period model still promises to offer improved returns over classical portfolio optimization.

3.2.3. Factor Based Optimization with Regime-Dependent Idiosyncracity

In a follow-up paper, Costa, et al. propose an improved regime-dependent factor model of asset returns [9]. Unlike the model described above, both the constant α and the model error ϵ are made regime-dependent. Through this modification, they aim to incorporate not only the regime-dependency of systematic risk, described by the factors, but also the regime-dependency of idiosyncratic risk, captured by the model's α and ϵ. This leads to the following factor model [9]:

$$r_t = I_1(\alpha_1 + V_1^T f_{1t} + \epsilon_{1t}) + I_2(\alpha_2 + V_2^T f_{2t} + \epsilon_{2t})$$

This regime-dependent factor model can then be used in a similar fashion to the model described above to calculate expected asset returns and covariances as inputs for classical portfolio optimization techniques.

While this model offers improved results over the previous model [9], it still suffers from the same drawbacks, most notably the potential mismatch between this one-period model and a strategy's rebalancing frequency.

§4. Data Description

4.1. Market Data

The backtests and analysis focus on index returns rather than individual stock returns. To that extent, daily index data on the S&P 500 (SPX), investment grade bonds (LBUSTRUU), commodities (BCOMTR), and value equities (RU10VATR) was collected through Bloomberg from 1992-02 until 2023-03 [5]. Furthermore, to replicate the techniques from Costa, et al. [8][9], daily Fama French 3 Factor data was collected through Kenneth R. French's data library [11]. All collected data was pre-processed and transformed into log-returns by taking the first difference between the log-closing prices of any two consecutive trading days in the dataset. This resulted in 7849 trading days of historical returns data, corresponding to a backtest period of 231 months, or 19.25 years, after accounting for model burn-in.

4.2. Synthetic Data

As market regimes are latent variables, real data does not allow an analysis on the classification performance and parameter sensitivity of regime detection methods. Therefore, synthetic price paths are generated over a time period of 3000 days, roughly 12 years. Asset prices are assumed to follow a Geometrical Brownian Motion (GBM), where the drift and volatility term are regime dependent. This price generating process is discretized to daily data. Let there be N regimes each with their own parameters μ_i and σ_i, then for every time t the process finds itself in a single state i_t and log-returns are sampled from a normal distribution: $r_t \sim N(\mu_{i_t}, \sigma_{i_t})$. The analysis in the remainder of this report focuses on models with two distinct regimes, shown in Table 1, although the analysis can be extended for more regimes.

	μ	σ
regime 1 (good)	7%	10%
regime 2 (bad)	-4%	30%

Table 1.: Simulation parameter choices for average annualized log-return and volatility.

Next to sampling log-returns from regimes, a process is required to generate sequences of regimes over time. Three distinct methods are set up:

1. Bernoulli Blocks, 2 regimes: time is split up into sequential blocks of 30 days. For each block, a regime is selected from i.i.d. Bernoulli distributions with probability p of ending up in the *good* state.
2. Uniform Blocks, 2 regimes: 10 dates are chosen at random over the entire sample period. At every selected date, a *bad* state starts where the length of this state is sampled from i.i.d. half-normal distributions with parameters μ and σ. All other dates are labeled as the *good* state.
3. Markov Chain, 2 regimes: regime sequences are simulated through a Markov chain represented by its transition matrix:

$$P = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}$$

   where $0 \leq p, q \leq 1$.

All regime generating processes were calibrated to display the *good* regime at at least 80% of the time to match with the intuition of being the base state. The inclusion of non-Markovian regime generating processes ensures that there is no explicit bias towards Hidden Markov Models which assume a Markov structure for the latent states.

§5. Codebase

For the purpose of this of this paper, a Python package, *MarketMoodRing* [10], was developed to handle the regime detection and regime-dependent portfolio optimization and allow for modular trading strategies. This package contains two abstract classes to perform regime detection and portfolio optimization: *RegimeDetectionBase* and *PortfolioOptimizationBase*, respectively. Each of these two classes contains a set of subclasses to implement all the methods under investigation listed in Section 3. The main goal of MarketMoodRing is to provide a uniform API to different regime detection models and regime-dependent portfolio optimizers. As a result, backtests can be set up in a modular fashion by swapping in different models without the need to update the core code of the actual backtest. UML diagrams for the most important classes can be found in Figures 12 and 13 in Appendix A.

A simple investment flow using MarketMoodRing is shown in Figure 1. First, price data has to be downloaded and transformed into (daily) log returns. Second, the chosen regime detection model has to be fit on the cleaned data. Third, the outputs from the regime detection model, together with the log returns of the investable assets have to be passed to the portfolio optimizer to construct weights. Finally, the weights can be used in a backtest or directly as investment inputs. Note that the data on which regimes are fitted and the data on which portfolio weights are calculated do not necessarily have to match. For instance, regimes can be detected on the S&P 500, while portfolio weights are calculated for a selection of S&P 500 constituents.

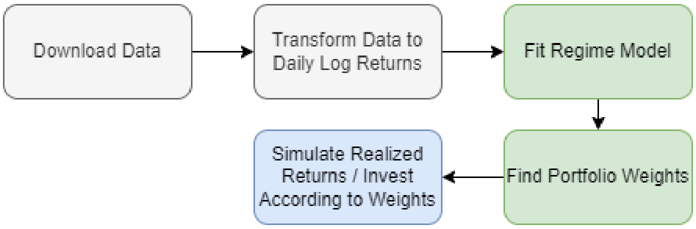
Download Data
Transform Data to Daily Log Returns
Fit Regime Model
Find Portfolio Weights
Simulate Realized Returns / Invest According to Weights

Figure 1. : Steps required to calculate regime-dependent portfolio weights. The steps highlighted in green use the classes implemented in MarketMoodRing.

§6. Empirical Framework and Results

6.1. Regime Detection Accuracy and Model Sensitivity

6.1.1. Framework

Given the large hyperparameter space, four HMM and four Wasserstein K-means (WKM) models were selected. Each of these models was selected because they displayed relatively robust results during testing and implementation and showcase important differences in hyperparameters. A full list of the hyperparameters for each model can be seen in Table 2 and Table 3. Furthermore as discussed in Section 4.2, synthetic data was generated to evaluate model sensitivity and accuracy. To gain an intuitive understanding of how this process works, figure 2 shows a simulated regime sequence and price path, alongside the model predictions. For this example, the regime sequence was generated through a Markov chain with 2 regimes. As can be seen on this figure, each of the models has slightly different opinions on the regime predictions.

#	Model Name	hmm_type	covariance_type	n_components
1.	hmm_gaus_2	GaussianHMM	full	2
2.	hmm_gmm_2	GMMHMM	diag	2

Table 2.: Hyperparameters for the HMM models under consideration. Additionally each model sets n_iter=100 and tol=0.01.

#	Name	n_regimes	window_size
3.	wkm_a_2	2	20
4.	wkm_b_2	2	10

Table 3.: Hyperparameters for the Wasserstein K-means models under considerations. Additionaly each model sets n_bins=21, min_range=-0.045, max_range=0.045, p=3, reg=1e-3, and max_iter=200.

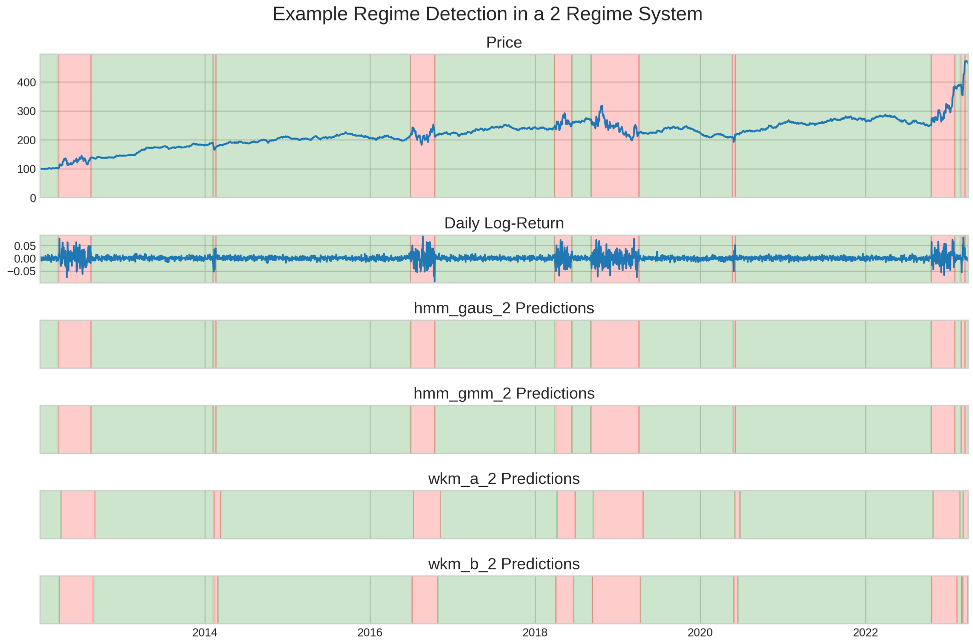

Figure 2. : Simulated regime sequence and related price path (top) from a Markov chain with 2 regimes. The green and red regions represent the good and bad states, respectively.

6.1.2. Classification Performance

Four metrics are employed to evaluate the performance of each regime detection model: accuracy, F1 score, precision, and recall. The discussion below focuses on accuracy, while the other metrics can be found in Appendix B as similar conclusions are drawn from all four metrics.

300 independent regime sequences are generated using Uniform Blocks with 2 regimes, each with a simulated price path. The accuracy for each model on each price path is calculated, giving rise to a distribution of accuracies. The accuracy distributions for each model across these paths are shown in figure 3. Based on this simulation, HMM seems to be more robust than WKM. Furthermore, WKM with a lookback window of 20 days seems to consistently offer better accuracies than WKM with a 10 day lookback window.

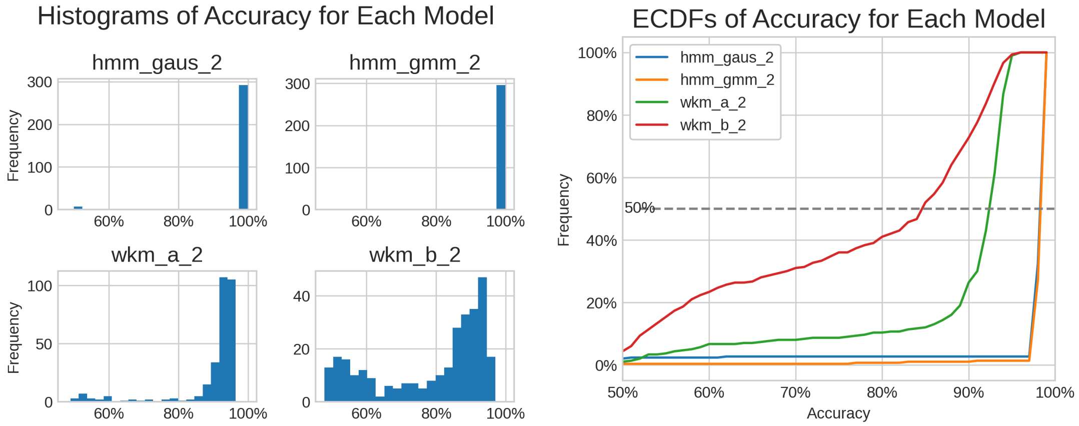

Figure 3. : The PDF (left) and ECDF (right) of the accuracy over 300 simulated price paths for each model.

6.1.3. Sensitivity to Regime Parameters

Beyond from classification performance, the sensitivity of each regime detection model against the return distributions of the underlying regimes can be checked[1]. Both the sensitivity to differences in mean returns and volatilities are investigated independently in a 2 regime system. To check the sensitivity to mean differences, daily volatilities are fixed, while the absolute difference between the mean returns of each regime increases. Next, for a given difference in mean returns, the average accuracy over 30 independent price paths is calculated for each regime detection model. For this simulation, Uniform Blocks are used as the underlying regime generating process. The results are shown in Figures 4 and 5. Analogously, the sensitivity to differences in volatility is shown in Figure 6.

As can be seen from Figure 4, neither HMM nor WKM display a higher accuracy for regimes with more spread out mean returns. This is due to the relatively small size of the mean returns versus volatility on daily data. Daily mean returns are 0.04% and 0.035%, while daily volatilities are 1% and 2% for the two regimes, respectively. If lower volatilities are used, there is a more outspoken sensitivity to mean differences, as seen in Figure 5. However, on real market data, where daily asset returns are small compared to daily volatility, latent regimes are therefore unlikely to be picked up by differences in their mean returns.

In contrast to mean returns, volatility differences clearly have an impact on model accuracy as can be seen in Figure 6. The graph for each of the models show an S-shape with model accuracy increasing steeply after a certain threshold and then plateauing near 100%. This suggests that on real market data, volatility will be an important driver to distinguish between latent regimes.

[1] The hyperparameters of the WKM model need to be updated as the range of simulated returns changes throughout the simulation. Concretely, min_range and max_range are changed to the 2nd and 98th percentile of each simulated return sequence. This ensures that the underlying algorithm captures the entire return distribution.

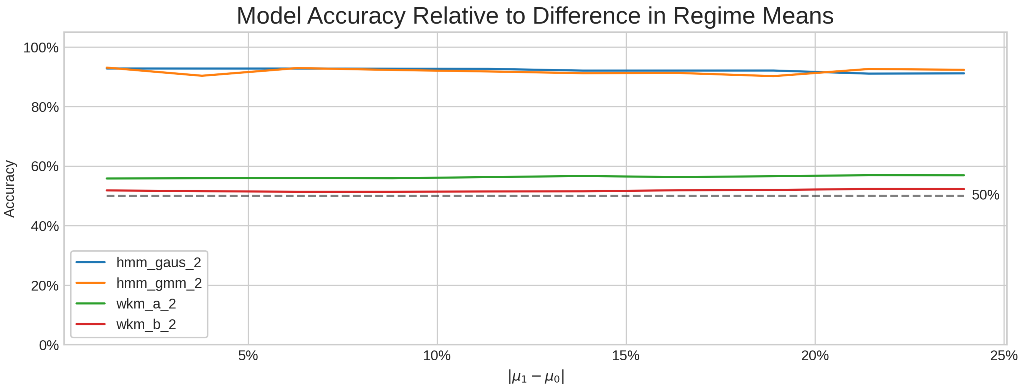

Figure 4. : Regime detection accuracy as a function of difference in annualized average return. Annualized volatilities are fixed at 16% and 32% for the two regimes, while annualized mean returns start at 10% and 9%, respectively. Mean returns drift apart by equal amounts in the simulation.

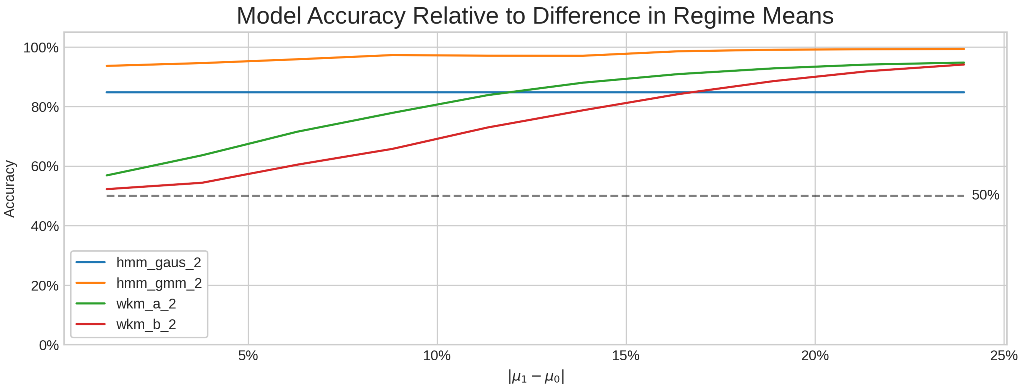

Figure 5. : Regime detection accuracy as a function of difference in annualized average return. Similar setup to Figure 4, but annualized volatilities are fixed at 0.8% and 1.6% instead.

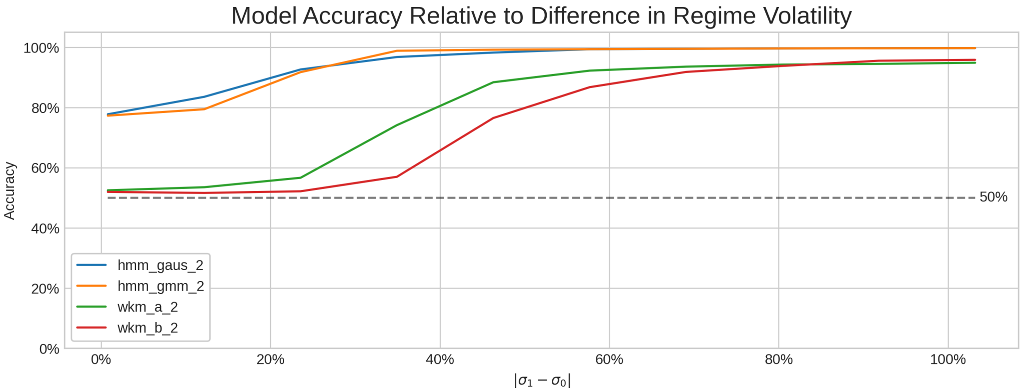

Figure 6. : Regime detection accuracy as a function of difference in annualized volatility. Annualized mean returns are fixed at 10% and -10% for the two regimes, while annualized volatility is fixed at 24% for the first regime and grows until 120% for the second regime.

6.1.4. Sensitivity to Regime Generating Process

Beyond from sensitivity to the characteristics of the underlying regimes, regime detection models can also be sensitive to the regime generating process itself. HMM makes the assumption that latent regimes follow a Markov chain, whereas WKM does not make any assumptions. A natural question is how this impacts their sensitivity to the regime generating process. Figure 7 shows the performance of different regime detection models over 100 independent price paths for each regime generating process. Surprisingly, WKM appears to be more sensitive to the regime generating process than HMM, especially for WKM with a small window.

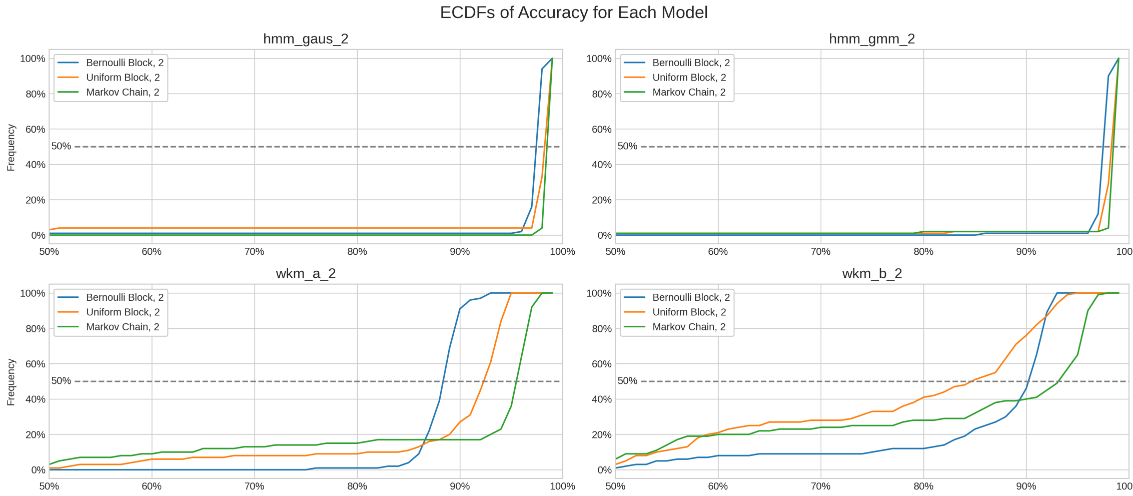

Figure 7. : ECDF of accuracies for the different regime detection models for different regime generating processes.

6.2. Regime Detection on Market Data

In addition to Section 6.1, the performance and robustness of regime detection was evaluated on real market data.

6.2.1. Performance

As regime detection is an unsupervised problem, with unknown true regimes, different techniques are required compared to the previous section. However, the clustering performance of the algorithms can still be evaluated. Ideally, the regime detection algorithms would create clusters that have a high degree of similarity within and a low degree of similarity between clusters.

To evaluate the return distribution clustering, the Davies-Bouldin, Dunn and Silhouette Coefficient of WKM and HMM are reported in Appendix C. Based on these metrics, WKM seems to perform a better clustering than HMM although it has to be noted that these metrics are sensitive to the metric space and might be biased in favor of WKM.

As these previous metrics are sensitive to the metric space, an integrable probability metric like Maximum Mean Discrepancy is a more suitable choice to investigate the goodness of the clustering. Maximum Mean Discrepancy (MMD) is statistical test employed to determine whether two distributions are the same. In the context of machine learning, MMD is also used as a loss metric. Formally, the MMD is defined as follows [12]: Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ and let μ, ν be distributions on $\mathcal{X}$. Then, the maximum mean discrepancy (MMD) between μ and ν is defined as

$$\mathrm{MMD}[\mathcal{F}, \mu, \nu] := \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{\mu}[f(x)] - \mathbb{E}_{\nu}[f(y)] \right)$$

As a class of functions $\mathcal{F}$, the set of Gaussian kernels is chosen to employ the kernel trick [12]. The empirical score calculation is based on the biased MMD estimate [17]. Given samples $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$, a biased empirical estimate is given by

$$\mathrm{MMD}_b[\mathcal{F}, x, y] = \left[\frac{1}{n^2} \sum_{i,j=1}^{n} k\left(x_i, x_j\right) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k\left(x_i, y_j\right) + \frac{1}{m^2} \sum_{i,j=1}^{m} k\left(y_i, y_j\right) \right]^{\frac{1}{2}}$$

As WKM works by clustering return distributions, this score can be used to evaluate its clustering performance [17]. Median Biased MMD^2 as a score for between-cluster and intra-cluster similarity, is shown for WK-means in Appendix C. The score is calculated by sampling $n = 10000$ from the target clusters and Gaussian kernel with parameter $\sigma = 0.1$. As can be seen, the Median Biased MMD^2 is lower for the most common regime, identified as 1 in both plots, but significantly higher for the infrequent regime. The Median Biased MMD^2 for regime 0 is also greater than the value for between-regimes. A potential explanation for the results is the classification of all outlier distributions in regime 0 implying higher heterogeneity between the distributions belonging to the cluster and higher MMD.

6.2.2. Robustness

The robustness of regime detection over time is also investigated from two perspectives: through retracting windows and through expanding windows. In the retracting window analysis, the end date for the regime detection is fixed while the start date moves from past to present. This answers the question how model parameters and predictions change when there is access to more or less historical data.

In the expanding window analysis, the start date for the regime detection is fixed while the end date moves from past to present. This answers the question how model parameters and predictions change as one moves throughout time and encounters new information, while also retaining all past information. This is especially important for the portfolio backtests in Section 6.3 as they use an expanding window for the regime detection.

A critical issue when analysing regime detection on different, but overlapping windows is matching the regimes. As all regime detection models are unsupervised, the actual label that they assign to a regime has no inherent meaning. Furthermore, regime predictions and parameters can slightly change between two overlapping windows. A number of different metrics were investigated to yield a consistent identification of regimes between windows. An example analysis was conducted on expanding windows in a 2-regime system to identify the label of the most stable regime in each window. As can be seen from the example in Table 4, stability of the transition matrix, frequency of the predicted regime, and low volatility and excess kurtosis are especially consistent to find the stable regime.

	Stability	Frequency	Calmness	Low Absolute Mean	Central	Platykurtic	High Sharpe
1996	1	1	1	1	1	1	1
1998	1	1	1	1	0	1	1
2000	1	1	1	0	0	1	1
2002	0	0	0	1	1	0	0
2004	1	1	1	0	1	1	1
2006	0	0	0	1	1	0	0
2008	1	1	1	1	0	1	1
2010	0	0	0	1	1	0	0
2012	0	0	0	0	1	0	0
2014	1	1	1	1	0	1	1
2016	1	1	1	0	0	1	1
2018	1	1	1	0	1	1	1
2020	0	0	0	0	0	0	0
2022	1	1	1	1	1	1	1
2024	1	1	1	1	1	1	1

Table 4.: Identification table of the "stable" regime for GMM HMM in a 2-regime system. Every row represents one window starting in 1992 and ending in the year indicated in the first column. The value in a row indicates which label was identified as the stable regime. Different columns indicate different metrics to identify the stable regime.

In line with this example analysis, Figure 8 shows the evolution of mean returns, volatility, and correlations for the different regimes with expanding and retracting windows. For both types of windows, model parameters seem to converge to a stable value once the window is large enough. In the case of retracting windows, this means that someone with little historical data will get biased results compared to someone with large historical data. Similarly, in the case of expanding windows, this means that a certain burn-in period is required for these models to converge into stable parameters. Similar conclusions are drawn for other regime characteristics such as excess kurtosis and skewness as shown in Appendix D.

Finally, by running retracting windows with starting dates between 1992-2016 and an end date of 2023, regime detection is compared over the period 2018-2023 by calculating correlations between the identified regime sequences. This is shown in Figure 9. A clear result is that models with access to historically significant market periods, e.g. the Great Financial Crisis of 2008 or the dot.com bubble, tend to predict differently from models that do not have access to these periods. Furthermore, the size of the windows in Wasserstein K-means and the type of emission distribution in HMM significantly influence the detected regimes.

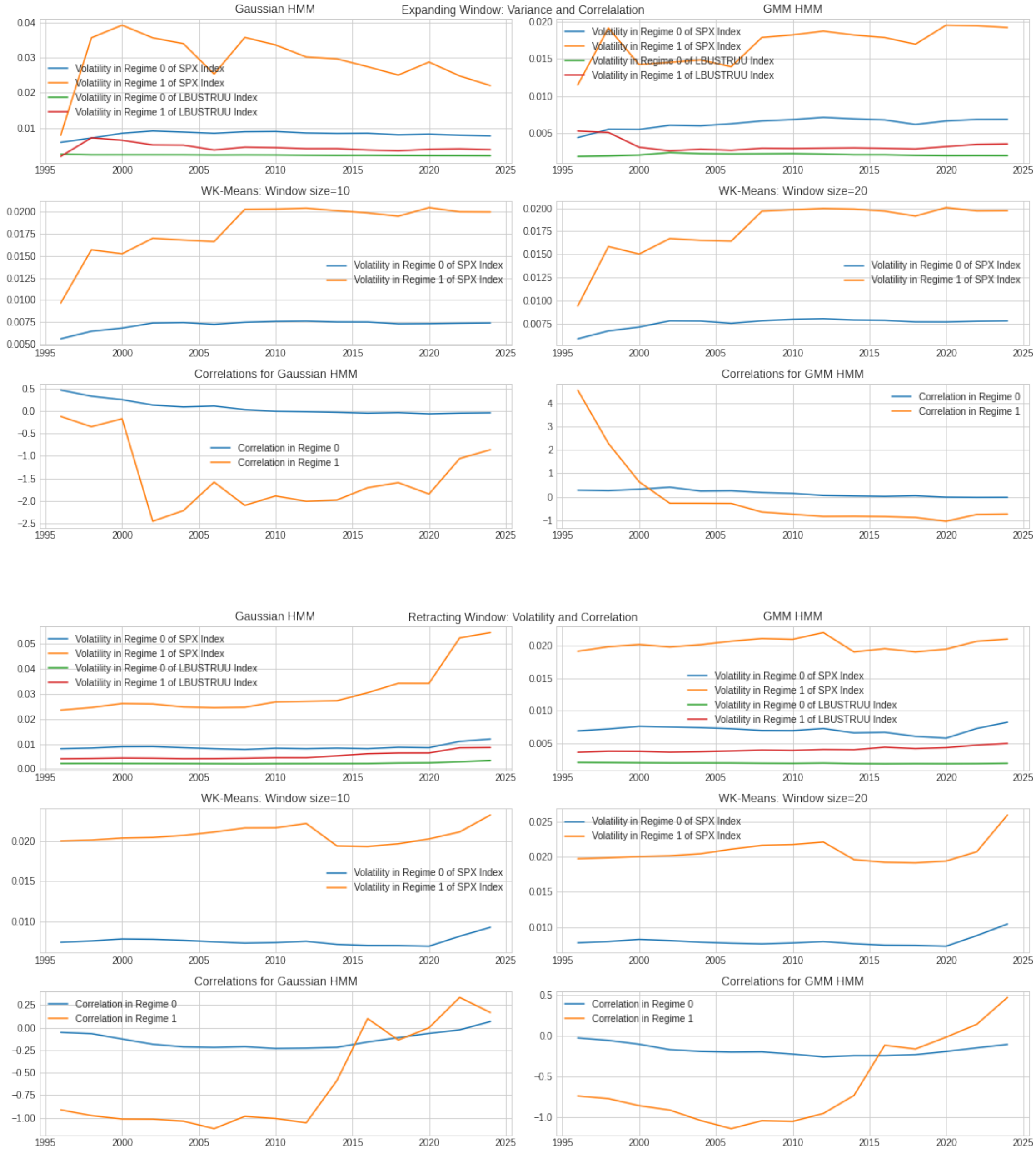

Figure 8. : Left: Expanding window analysis of the volatility and correlation parameters for selected regime detection models, Right: the equivalent analysis for retracting windows

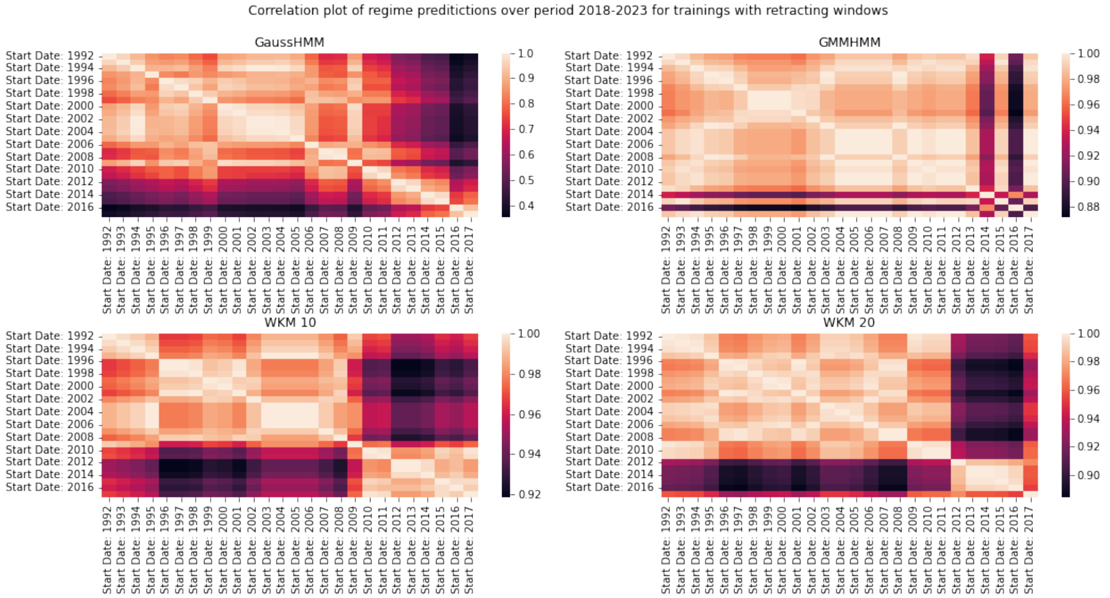

Figure 9. : Correlation analysis for detected regimes over 2018-2023 for models trained with retracting windows with starting date between 1992 and 2016. Model predictions are drastically impacted by access to historical market breakdowns, as indicated by a drop in correlation between predictions prior and post to such events.

6.3. Regime-Dependent Portfolio Optimization

6.3.1. Framework

To limit the amount of strategies under consideration, a selection of regime models, optimization models, and hyper-parameters was chosen which displayed robustness during testing and were economically sensible. For reference to the hardware for the backtest, all tests were run on a machine with 16GB RAM and an 11th Gen Intel(R) Core(TM) i7-11390H @ 3.40GHz, 3418 Mhz, 4 Core(s), 8 Logical Processors.

On a high level, the backtest is set up similar to the process shown in Figure 1. The key difference is that for the backtest, regimes are fitted upfront and their inputs are cached. This avoids having to re-fit the same regime models for different optimization models and ensures a fair playing field wherein all optimizers use the same regime detection outputs as inputs.

Concretely, the backtest set-up is in line with the backtests from Costa, et al. [8], with an ever expanding window for the regime detection, and a fixed-size rolling window for the portfolio optimization. This allows for a large training sample on the regime detection, but a more contemporary view on asset characteristics for the optimization step [8]. A trading strategy consists of a combination of a Regime Detection model and a Portfolio Optimization model with monthly rebalancing and accumulating returns. There is a burn-in period from 1992-02 until 2004-01, 12 years, to allow the regime detection to have ample training data. Starting at 2004-01, each trading strategy undertakes the following step on the first trading day of the month:

1. Fit the regime model from 1992-02 up until and including the rebalance date
2. Find the optimal portfolio weights by applying the optimization model on a 12 year lookback window, including the rebalance date
3. Calculate the realized returns from this strategy from EOD of the rebalance date until EOD of the next rebalance date

Four regime detection and five portfolio optimization methods were selected and each combination was included in the backtest. For the regime detection, two Hidden Markov Models were included, one with Gaussian emissions and one with Gaussian Mixture Model emissions. Next, two Wasserstein K-means models were included, one with a 20 day window size and one with a 10 day window size. Furthermore, both Wasserstein K-means models have a $p = 3$, meaning that the distance metric considers the first three moments of each return distribution, roughly the mean, variance, and skewness. Finally, for simplicity each regime model fits two regimes. A full overview of each regime detection model and its hyperparameters can be seen in Listing 2.

Fitting all four models for the entire backtest took 8 hours on the machine mentioned above. 90% of this time was spent on fitting the Wasserstein K-means models due to the computational intensity of K-means and the Wasserstein-distance calculations.

For the portfolio optimization, three methods were based on Costa, et al. [8][9] and two methods were based on Bae, et al. [1]. The first three models are each factor based, using the Fama-French 3 Factor model. In line with Costa, et al. [8], the first model optimizes for equal risk contribution (ERC), considering idiosyncratic risk to be regime-independent. Similarly, the second and third model respectively optimize for ERC and MVO, considering idiosyncratic risk to be regime-dependent. This is in line with the follow-up paper of Costa, et al. [9]. The MVO optimizer is subject to no-leverage and no-short-selling constraints. Finally, in line with Bae, et al. [1], the last two models perform stochastic programming by simulating many possible return sequences and the corresponding return distributions. One model optimizes for the maximum average Sharpe Ratio across all simulations, while the other model optimizes for the minimum average Value at Risk (VaR) across all simulations. A full overview of each portfolio optimization model and its hyperparameters can be seen in Listing 2.

Fitting all five models for the entire backtest, given the fitted regimes, took 20 minutes on the machine mentioned above. Although stochastic programming suffers from the curse of dimensionality, this implementation relies heavily on NumPy vectorization as well as Monte Carlo simulations which limits the computational complexity.

To benchmark the trading strategies outlined above, three benchmark strategies were set up, each fitted similar to the trading strategies, but without incorporating regime dependency. The first benchmark is a simple ERC strategy, the second benchmark is a simple equal-weighted (EW) strategy, and the third benchmark is a simple MVO strategy without leverage or short-selling.

```python
# Regime Models
regime_models = [
    # HMM with Gaussian Distributions - full covariance matrix
    HiddenMarkovRegimeDetection(n_regimes, hmm_type="GaussianHMM",
        covar_type="full", n_iter=100),
    # HMM with Gaussian Mixture Model
    HiddenMarkovRegimeDetection(n_regimes, hmm_type="GMMHMM",
        covar_type="diag", n_iter=100),
    # Large Window WKM
    WassersteinKMeansRegimeDetection(n_regimes, window_size=20, n_bins=21,
        min_range=-0.045, max_range=0.045, p=3, reg=1e-3, max_iter=200,
        kde_smoothing=0.5),
    # Medium Window WKM
    WassersteinKMeansRegimeDetection(n_regimes, window_size=10, n_bins=21,
        min_range=-0.045, max_range=0.045, p=3, reg=1e-3, max_iter=200,
        kde_smoothing=0.5),
]

# Optimization Models
opt_models = [
    # Costa & Kwon (2019) - ERC
    SimpleFactorPortfolioOptimization(n_regimes=n_regimes, optimizer="ERC"),
    # Costa & Kwon (2020) - ERC
    IdiosyncraticFactorPortfolioOptimization(n_regimes=n_regimes, optimizer="ERC"),
    # Costa & Kwon (2020) - MVO
    IdiosyncraticFactorPortfolioOptimization(n_regimes=n_regimes, optimizer="MVO"),
    # Stochastic Programming - Maximize Average Sharpe Ratio
    JointStochasticProgOptimization(n_regimes=n_regimes, objective="max_avg_sharpe"),
    # Stochastic Programming - Minimize Average Value at Risk
    JointStochasticProgOptimization(n_regimes=n_regimes, objective="min_avg_VaR")
]
```

Listing 2: Regime detection and regime-dependent portfolio optimization models used within the backtest with their corresponding hyper-parameters.

6.3.2. Results

The results of this backtest allow to compare the regime-dependent optimization utilized by Costa, et al. [8][9] and Bae, et al. [1] as well as expand on the research of Horvath, et al. [17] by applying regime-dependent optimization in tandem with Wasserstein K-means. The investment universe for these strategies consisted of two indices: the S&P 500 (SPX) and the Bloomberg US Agg Total Return Value Unhedged USD (LBUSTRUU). Hidden Markov Models were fitted on both indices, while Wasserstein K-means models were fitted only on the SPX.

The results of this backtest are shown in Appendix E. Table 5 gives an overview of the distributional characteristics of each trading strategy and the benchmarks, while Table 6 gives an overview of the risk-return characteristics and win-rate of each trading strategy and the benchmarks. Based on these results, stochastic programming which optimizes for maximum expected Sharpe Ratio tends to dominate the other strategies. In combination with Gaussian Mixture Model HMM, this leads to the best strategy across the board.

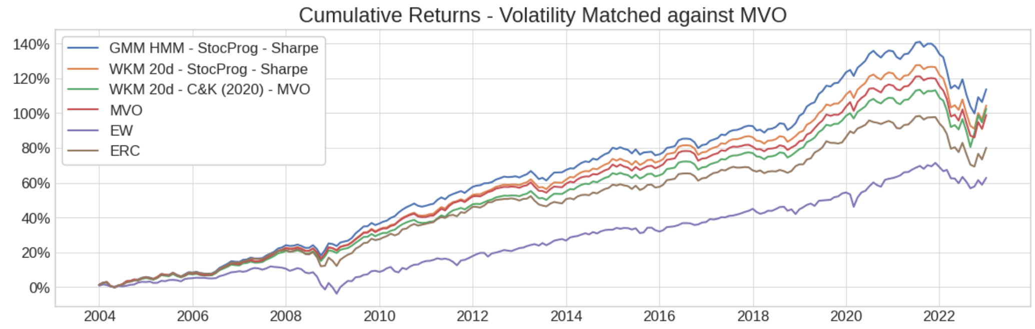

Figure 10. : Cumulative returns of the three best trading strategies against the benchmarks. Monthly returns are volatility matched against the volatility of the MVO benchmark to enhance comparability.

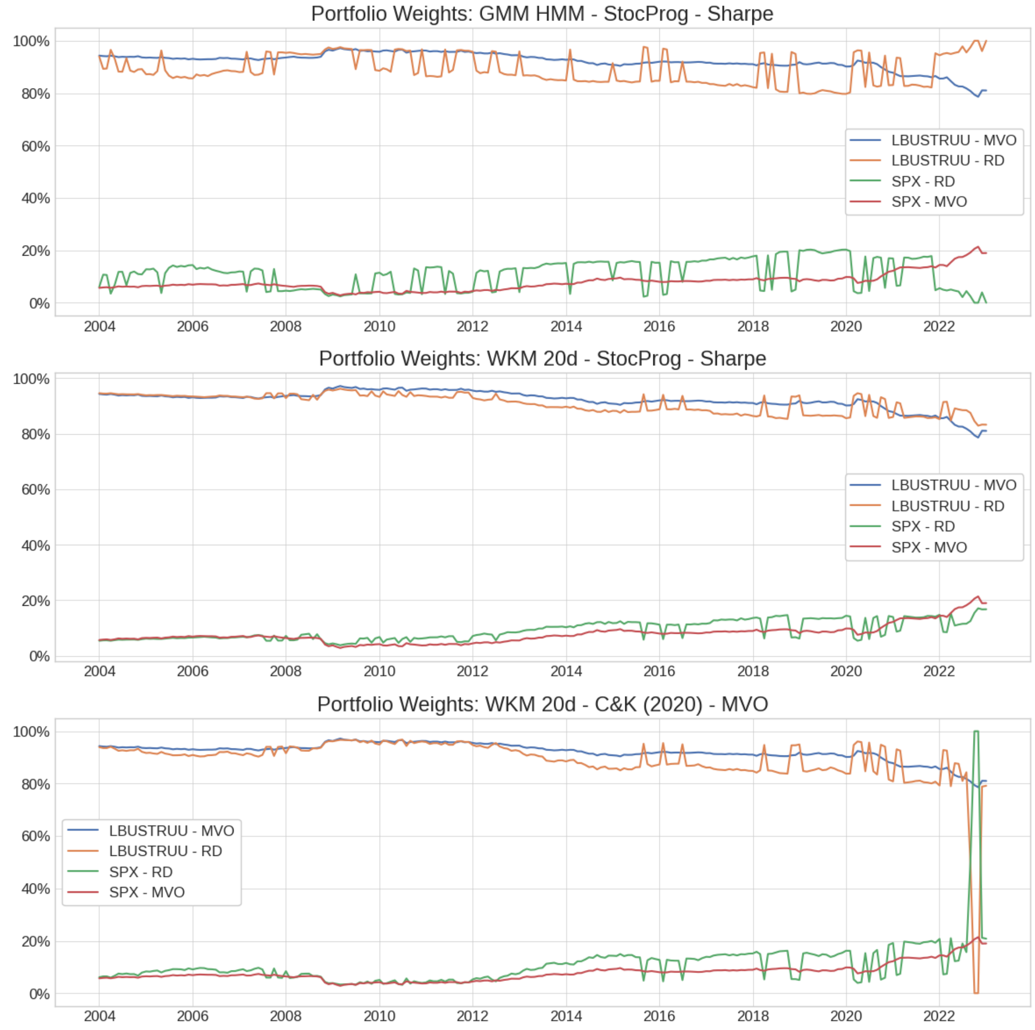

Figure 11. : Monthly portfolio weights of the three best trading strategies against the MVO benchmark. Weights are shown on the first business day of the month, after rebalancing each strategy. Weights of the regime dependent (RD) strategies hover around the MVO benchmark in a risk-on, risk-off fashion.

Regarding the regime detection models, strategies using Wasserstein K-means with a 20 day lookback window and Gaussian Mixture Model HMM tend to dominate the other strategies. This is in line with the analysis above which suggested that these were the best WKM and HMM models for regime detection.

As can be seen in Table 6, the three strategies with the highest Sharpe Ratio outperform all other strategies and each of the benchmarks on each risk-return metric as well as on win-rate. However, as can be seen in Table 5, only one of these three strategies has a lower Maximum Drawdown (MDD) than the ERC and MVO benchmarks. Similarly, the three strategies do not show a significant improvement on standard deviation and value at risk (VaR). The results of this backtest provide evidence that regime-dependent portfolio optimization leads to higher returns at a similar ex-post level of risk compared to its nominal counterparts, in line with Costa, et al. [8][9]. This can be seen by the higher return and win-rate of the three best strategies compared to the benchmarks. However, there is no strong evidence that incorporating regime-dependency leads to a smaller exposure to left-tailed events. This is in contradiction to Bae, et al. [1] who showed proof for improved downside risk protection through stochastic programming. However, these results could be different if the portfolio was rebalanced whenever a regime switch was detected instead of on a monthly basis.

As shown in Tables 7 and 8 of Appendix E, these results are robust to expanding the investable universe to include commodities and value equity. The same three strategies outperform on this expanded universe and display higher returns at a similar ex-post level of risk compared to their nominal counterparts.

Focusing on the 2 asset environment consisting of SPX and LBUSTRUU, the three best performing strategies are investigated in more detail. Figure 10 shows the (volatility matched) cumulative return of each strategy and the benchmarks. Both in the financial crisis of 2008-2009 and the Covid crisis of 2020, the regime dependent strategies do not display large drawdowns. However, each strategy encounters a large drawdown in 2022. A potential explanation for this is that 2022 was characterized by rapidly increasing inflation and federal rate hikes after a long period of near-zero interest rates and low inflation. As this is the first time that these economic conditions occur in the data set, this could indicate a new, unseen regime in the market. This outlines a flaw of the regime detection models under investigation, namely that they can only detect regimes that they have seen before and need a burn-in period to learn about new regimes.

MVO is the nominal counterpart of each of the three best strategies. As discussed above, the performance of the regime dependent strategies mainly stems from higher returns at a similar ex-post level of volatility. Figure 11 helps to understand where these higher returns stem from. While each of the three strategies provide differing weights, there is a common pattern among each of them. Concretely, each strategy could be interpreted as a risk-on, risk-off version of MVO. Each strategy's weights hover around the MVO weights and switch between a higher and lower allocation to SPX depending on the detected regime. Interestingly, the best performing strategy, shown in the top chart of Figure 11, has the least correspondence to the MVO benchmark, while the other two strategies only start diverging from MVO post-2014. This suggests a failure of Wasserstein K-means to pick up on the correct regimes prior to 2014. Finally, the weights of the strategy based on Costa, et al. [9], shown on the bottom chart of Figure 11, display an aggressive change near the end of the backtesting period, indicating a potential over-sensitivity to certain changes in the data.

§7. Conclusion

In this paper, we have undertaken a comprehensive exploration of regime-dependent portfolio optimization, presenting an evaluation of various regime detection and portfolio optimization techniques. Our empirical findings underscore the outperformance of regime-based stochastic programming for portfolio optimization, particularly when optimizing for maximum expected Sharpe Ratio, compared to standard benchmarks and other regime-dependent optimization models. Regarding regime detection, we find that non-Gaussian models are superior over models that assume Gaussian distributions in the underlying regimes. Gaussian Mixture Model HMM performs particularly well, followed by Wasserstein K-means with a 20 day lookback window.

Our investigation into regime detection methods spanned both parametric and non-parametric approaches. The Hidden Markov Model (HMM), a widely utilized parametric model, has been thoroughly examined in the context of financial applications. Concurrently, we have delved into non-parametric models, focusing on Wasserstein K-Means, a product of recent advancements in machine learning. These explorations have yielded valuable insights into the respective strengths and limitations of these methods, and have highlighted the potential of machine learning in detecting latent regimes. There are three key insights that apply to both types of models. First of all, a burn-in period is required for a regime detection model to converge to stable parameters. Secondly, different regimes are detected primarily through differences in volatility and higher moments of asset returns rather than through differences in mean returns. Thirdly, incorporating data on significant market breakdowns, e.g. the Great Financial Crisis of 2008, drastically changes regime predictions and parameters.

In terms of dynamic portfolio construction, our research has scrutinized stochastic programming and factor-based optimization. These methods have demonstrated their potential in enhancing portfolio performance, while also revealing the computational challenges inherent in their implementation. Although one-period factor models are much more computationally efficient than stochastic programming, we still find that stochastic programming offers a significant outperformance.

This paper also introduced a new Python package, MarketMoodRing [10]. This package represents a significant contribution to the field and is expected to be a valuable resource for both researchers and practitioners. At the time of writing, this tool offers out-of-the-box implementations for HMM and WKM regime detection as well as stochastic programming and factor-based regime-dependent portfolio optimization.

While our research has made significant strides in the understanding of regime-dependent portfolio optimization, it is evident that this is a complex and rapidly evolving field. Future research should continue to refine these methods, with a particular focus on improving computational efficiency, robustness, and parameter uncertainty. Moreover, the potential of machine learning and artificial intelligence in this field is vast and largely untapped, offering a fertile ground for future exploration.

§8. Discussion and Future Work

As discussed above, we find that incorporating regime information improves portfolio optimization, in line with the conclusions of Tu [23]. However, to further validate this hypothesis, specific econometric specification tests would be required. Hamilton [16] already provides these specification tests for regime-switching regressions. However, these econometric tests are valid only in the univariate case and a multivariate version, as required for market regime detection, is still lacking in the literature.

Regarding parametric regime detection models, it is worth considering the potential of Hidden Semi-Markov Models (HSMM) as a promising tool for future research. HSMMs, an extension of Hidden Markov Models, offer an intriguing feature in that they allow for variable state durations. This feature could be particularly beneficial in capturing the temporal dynamics of financial markets, as regime durations can vary significantly and influence investment outcomes. Looking at the past few years, HSMMs could be well suited to capture relatively short market breakdowns, such as the Covid Crisis of 2020.

Moreover, the forward–backward algorithms of HSMMs, as discussed by Yu [26], could be leveraged to estimate model parameters, evaluate the goodness of an observation sequence fitting to the model, and find the best state sequence of the underlying stochastic process. This could potentially lead to more accurate regime detection and thus, more effective dynamic portfolio construction. However, the implementation of HSMMs in this context would require careful consideration of the specific assumptions made on state transitions, duration distributions, and observation distributions. Future research could explore these aspects in detail, investigating the practicality and efficacy of HSMMs in the realm of regime-dependent portfolio optimization.

There is also potential for the further development of non-parametric regime detection models. Although these methods are data hungry, often preferring intraday data, our findings show their potential within regime-dependent portfolio optimization.

A first extension is a multivariate Wasserstein K-means algorithm. Wasserstein K-means is already computationally complex, and this complexity increases exponentially when moving from the univariate to the multivariate case. However, recent advancement in Wasserstein distance proxies offer the potential to overcome this complexity. Specifically, sliced Wasserstein distance [7] and spherical sliced Wasserstein distance [6] aim to reduce this complexity by averaging the Wasserstein distance of many different, random, projections of multivariate distributions on a low-dimensional subspace. Correspondingly, sliced Wasserstein barycenters [7] were also developed. These algorithms are already developed in the Python package POT and could therefore be incorporated into the WKM implementation of MarketMoodRing.

A second extension is to incorporate regime detection based on signature paths into MarketMoodRing, as proposed by Bilokon et al. [4]. Through modification of the open-source code from Bilokon et al. his signature paths K-means algorithm could be incorporated into MarketMoodRing, offering additional flexibility on the non-parametric regime detection models.

Finally, this paper only touched the surface of portfolio optimization. On a basic level, different rebalancing schemes could be investigated, for instance rebalancing whenever a regime change is detected, to find the optimal balance between relevant weights and portfolio turnover. On a more advanced level, regime detection outputs could serve as an input into statistical and machine learning models to predict, relative, asset returns. This could give rise to complex trading strategies compared to simple portfolio optimization, although this would be more relevant for hedge funds compared to wealth and investment management.

To conclude, our research has made a substantial contribution to the understanding of the potential and challenges of regime-dependent portfolio optimization. The methodologies and tools presented in this paper are expected to be of significant value to the academic and professional community in the field of regime-dependent portfolio optimization. There are still many untouched opportunities in this field and we hope that our research provides a comprehensive starting point for future research and development.

§9. Bibliography

References

[1] G. I. Bae, W. C. Kim, and J. M. Mulvey. Dynamic asset allocation for varied financial markets under regime switching framework. *European Journal of Operational Research*, 234(2):450–458, 2014.

[2] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In O. Shisha, editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, University of California, Los Angeles, 1972. Academic Press.

[3] L. E. Baum and T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554 – 1563, 1966.

[4] P. Bilokon, A. Jacquier, and C. McIndoe. Market regime classification with signatures. *arXiv preprint arXiv:2107.00066*, 2021.

[5] Bloomberg L.P. SPX, LBUSTRUU, BCOMTR, RU10VATR Index closing prices from 1992-02 until 2023-04, 2023. Retrieved from Bloomberg database.

[6] C. Bonet, P. Berg, N. Courty, F. Septier, L. Drumetz, and M.-T. Pham. Spherical sliced-wasserstein. *arXiv preprint arXiv:2206.08780*, 2022.

[7] N. Bonneel, J. Rabin, G. Peyré, and H. Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51:22–45, 2015.

[8] G. Costa and R. H. Kwon. Risk parity portfolio optimization under a markov regime-switching framework. *Quantitative Finance*, 19(3):453–471, 2019.

[9] G. Costa and R. H. Kwon. A regime-switching factor model for mean-variance optimization. *Journal of Risk*, 2020.

[10] Y. A. D'hondt, M. M. Di Venti, R. Rishi, and J. Walker. MarketMoodRing, July 2023. URL https://github.com/yvesdhondt/MarketMoodRing.

[11] K. R. French. Fama/french 3 factors [daily], 2023. URL https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html. Retrieved from Kenneth R. French Data Library.

[12] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola. A kernel method for the two-sample problem, 2008.

[13] M. Guidolin and A. Timmermann. Asset allocation under multivariate regime switching. *Journal of Economic Dynamics and Control*, 31(11):3503–3544, 2007.

[14] M. Guidolin and A. Timmermann. International asset allocation under regime switching, skew, and kurtosis preferences. *The Review of Financial Studies*, 21(2):889–935, 02 2008.

[15] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57(2):357–384, 1989. ISSN 00129682, 14680262.

[16] J. D. Hamilton. Specification testing in markov-switching time-series models. *Journal of Econometrics*, 70(1):127–157, 1996. ISSN 0304-4076.

[17] B. Horvath, Z. Issa, and A. Muguruza. Clustering market regimes using the wasserstein distance. *arXiv preprint arXiv:2110.11848*, 2021.

[18] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. 02 2008.

[19] S. Kolouri, S. Park, M. Thorpe, D. Slepcev, and G. Rohde. Optimal mass transport: Signal processing and machine-learning applications. *IEEE Signal Processing Magazine*, 34:43–59, 07 2017.

[20] H. Markowitz. Portofolio selection. *Journal of finance*, 7:77–91, 1952.

[21] B. H. Putnam and J. M. Quintana. Mean-variance optimal portfolio models and the inappropriateness of the assumption of a time-stable variance-covariance matrix. *Review of Financial Economics*, 1(1):1, 1991.

[22] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[23] J. Tu. Is regime switching in stock returns important in portfolio decisions? *Management Science*, 56(7):1198–1215, 2010.

[24] C. M. Turner, R. Startz, and C. R. Nelson. A markov model of heteroskedasticity, risk, and learning in the stock market. *Journal of Financial Economics*, 25(1):3–22, 1989.

[25] R. H. Tütüncü and M. Koenig. Robust asset allocation. *Annals of Operations Research*, 132:157–187, 2004.

[26] S.-Z. Yu. Hidden semi-markov models. *Artificial Intelligence*, 174(2):215–243, 2010. ISSN 0004-3702. Special Review Issue.

§A. UML Diagrams

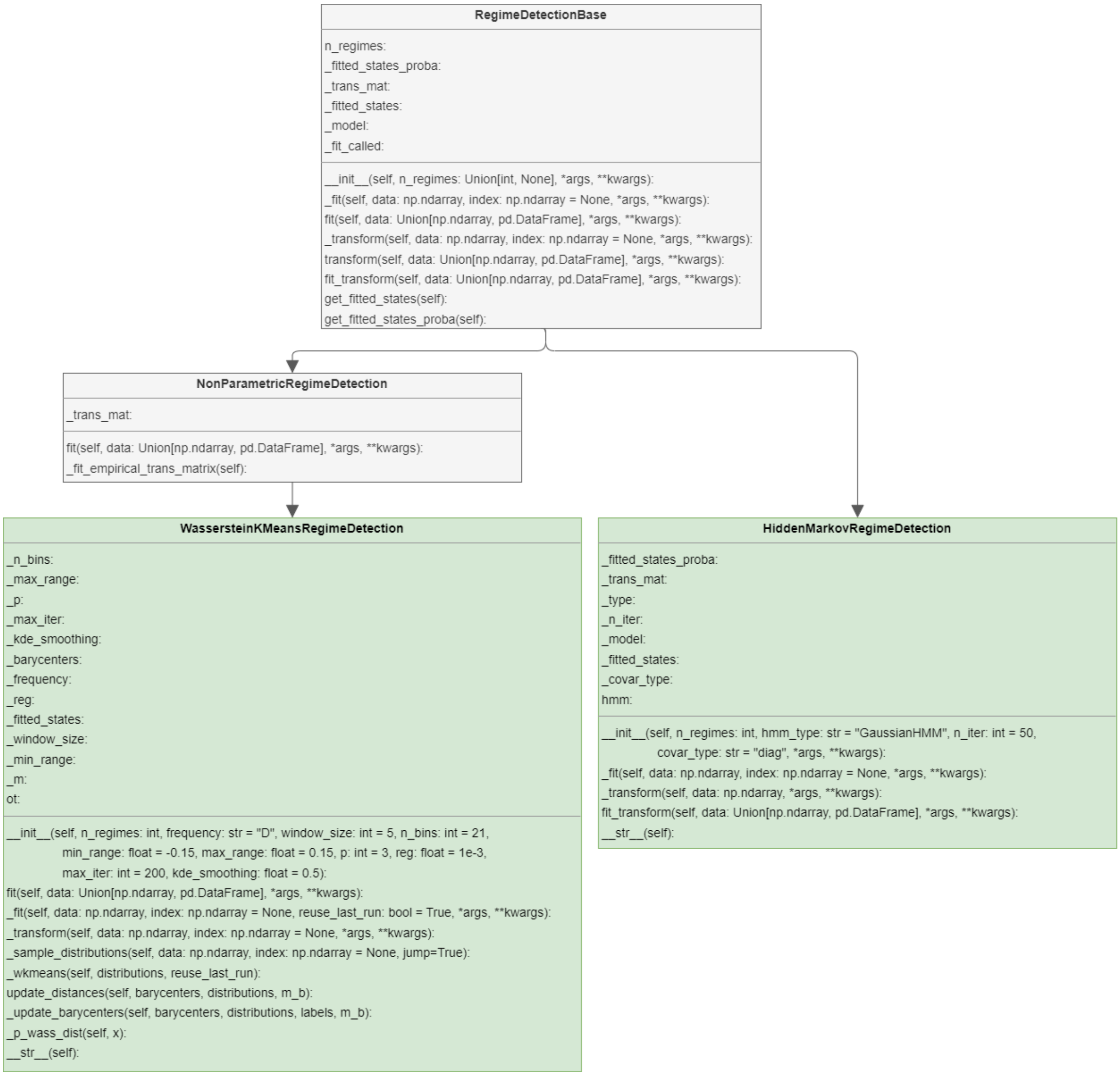
RegimeDetectionBase
n_regimes:
_fitted_states_proba:
_trans_mat:
_fitted_states:
_model:
_fit_called:
__init__(self, n_regimes: Union[int, None], *args, **kwargs):
_fit(self, data: np.ndarray, index: np.ndarray = None, *args, **kwargs):
fit(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
_transform(self, data: np.ndarray, index: np.ndarray = None, *args, **kwargs):
transform(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
fit_transform(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
get_fitted_states(self):
get_fitted_states_proba(self):
NonParametricRegimeDetection
_trans_mat:
fit(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
_fit_empirical_trans_matrix(self):
WassersteinKMeansRegimeDetection
_n_bins:
_max_range:
_p:
_max_iter:
_kde_smoothing:
_barycenters:
_frequency:
_reg:
_fitted_states:
_window_size:
_min_range:
_m:
ot:
__init__(self, n_regimes: int, frequency: str = "D", window_size: int = 5, n_bins: int = 21,
min_range: float = -0.15, max_range: float = 0.15, p: int = 3, reg: float = 1e-3,
max_iter: int = 200, kde_smoothing: float = 0.5):
fit(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
_fit(self, data: np.ndarray, index: np.ndarray = None, reuse_last_run: bool = True, *args, **kwargs):
_transform(self, data: np.ndarray, index: np.ndarray = None, *args, **kwargs):
_sample_distributions(self, data: np.ndarray, index: np.ndarray = None, jump=True):
_wkmeans(self, distributions, reuse_last_run):
update_distances(self, barycenters, distributions, m_b):
_update_barycenters(self, barycenters, distributions, labels, m_b):
_p_wass_dist(self, x):
__str__(self):
HiddenMarkovRegimeDetection
_fitted_states_proba:
_trans_mat:
_type:
_n_iter:
_model:
_fitted_states:
_covar_type:
hmm:
__init__(self, n_regimes: int, hmm_type: str = "GaussianHMM", n_iter: int = 50,
covar_type: str = "diag", *args, **kwargs):
_fit(self, data: np.ndarray, index: np.ndarray = None, *args, **kwargs):
_transform(self, data: np.ndarray, *args, **kwargs):
fit_transform(self, data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
__str__(self):

Figure 12. : Class hierarchy of the regime detection models. Abstract classes are indicated in grey while implemented classes are indicated in green. There is one class to fit HMM models and one class to fit Wasserstein K-means models.

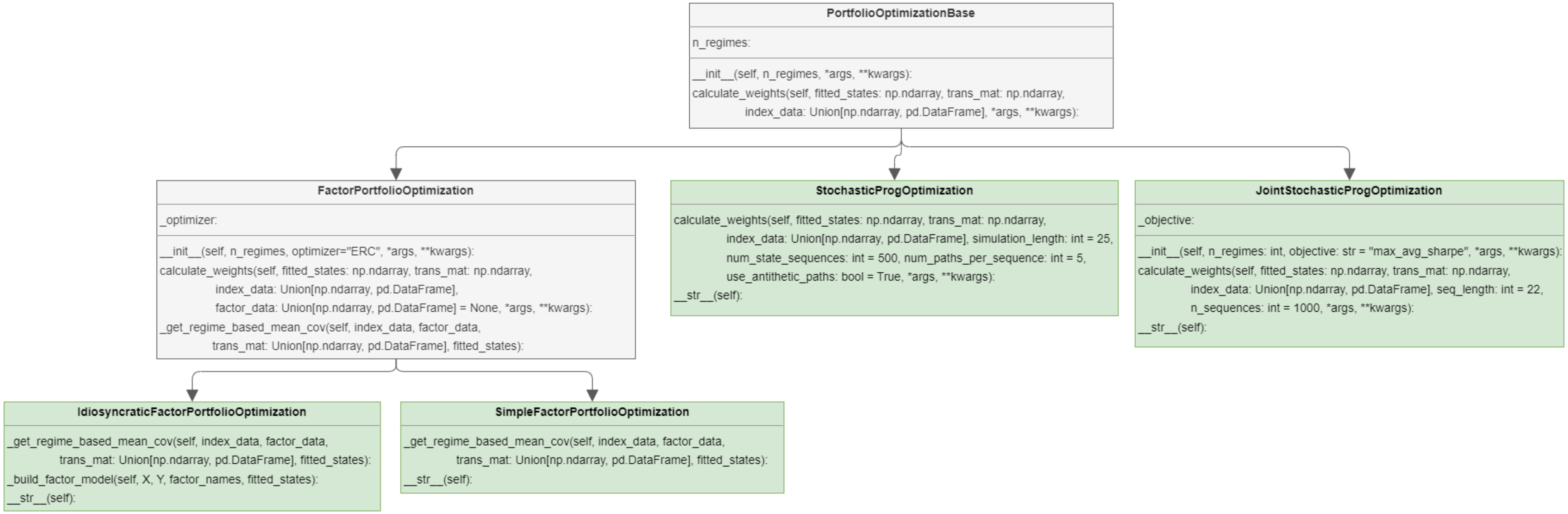
PortfolioOptimizationBase
n_regimes:
__init__(self, n_regimes, *args, **kwargs):
calculate_weights(self, fitted_states: np.ndarray, trans_mat: np.ndarray,
index_data: Union[np.ndarray, pd.DataFrame], *args, **kwargs):
FactorPortfolioOptimization
_optimizer:
__init__(self, n_regimes, optimizer="ERC", *args, **kwargs):
calculate_weights(self, fitted_states: np.ndarray, trans_mat: np.ndarray,
index_data: Union[np.ndarray, pd.DataFrame],
factor_data: Union[np.ndarray, pd.DataFrame] = None, *args, **kwargs):
_get_regime_based_mean_cov(self, index_data, factor_data,
trans_mat: Union[np.ndarray, pd.DataFrame], fitted_states):
StochasticProgOptimization
calculate_weights(self, fitted_states: np.ndarray, trans_mat: np.ndarray,
index_data: Union[np.ndarray, pd.DataFrame], simulation_length: int = 25,
num_state_sequences: int = 500, num_paths_per_sequence: int = 5,
use_antithetic_paths: bool = True, *args, **kwargs):
__str__(self):
JointStochasticProgOptimization
_objective:
__init__(self, n_regimes: int, objective: str = "max_avg_sharpe", *args, **kwargs):
calculate_weights(self, fitted_states: np.ndarray, trans_mat: np.ndarray,
index_data: Union[np.ndarray, pd.DataFrame], seq_length: int = 22,
n_sequences: int = 1000, *args, **kwargs):
__str__(self):
IdiosyncraticFactorPortfolioOptimization
_get_regime_based_mean_cov(self, index_data, factor_data,
trans_mat: Union[np.ndarray, pd.DataFrame], fitted_states):
_build_factor_model(self, X, Y, factor_names, fitted_states):
__str__(self):
SimpleFactorPortfolioOptimization
_get_regime_based_mean_cov(self, index_data, factor_data,
trans_mat: Union[np.ndarray, pd.DataFrame], fitted_states):
__str__(self):

Figure 13. : Class hierarchy of the portfolio optimization models. Abstract classes are indicated in grey while implemented classes are indicated in green. There are two factor-based models, in line with Costa, et al. [8][9], and two stochastic programming models, in line with Bae, et al. [1].

§B. Regime Detection Classification Metrics

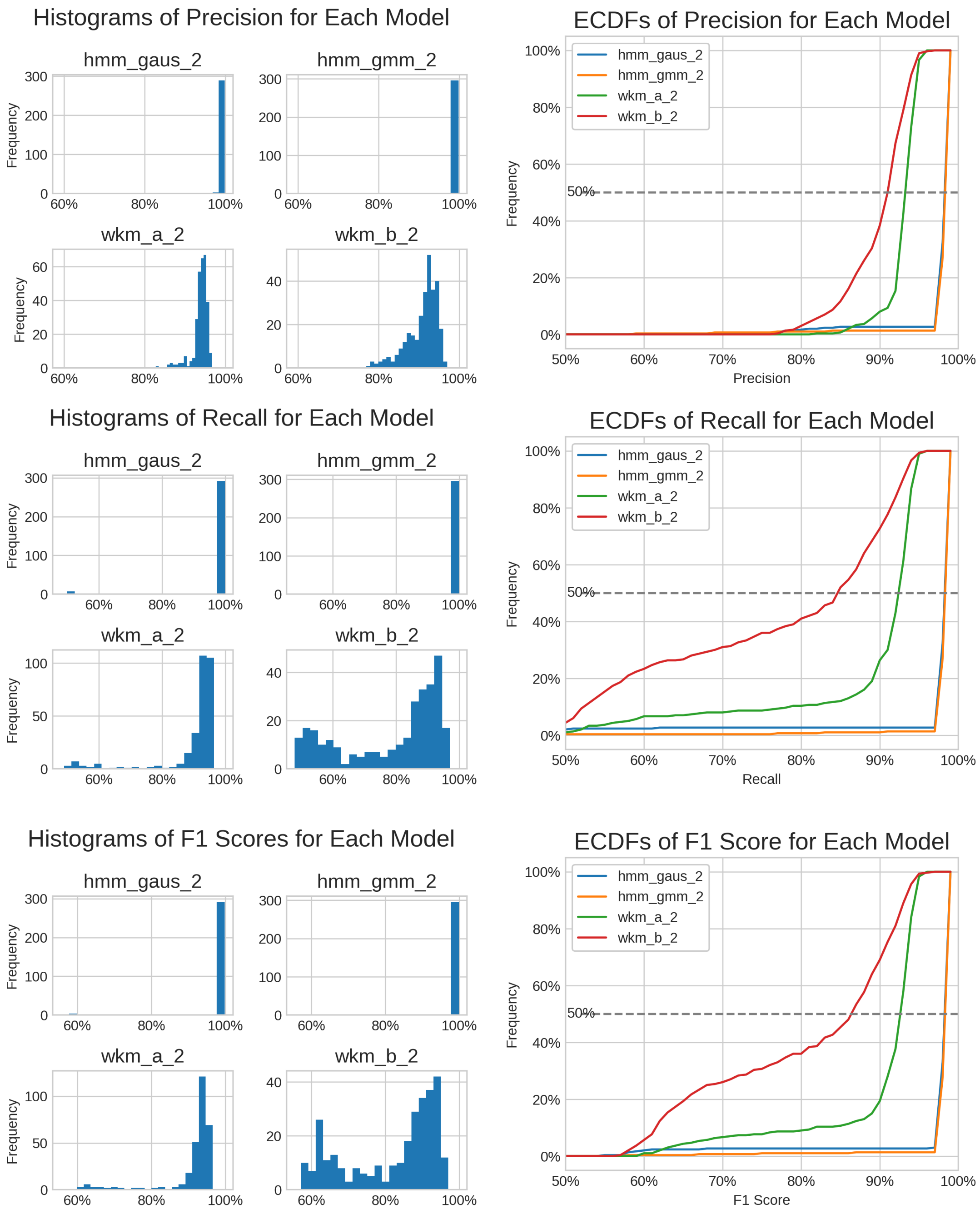

Figure 14. : The PDF (left) and ECDF (right) of the precision, recall, and F1 score over 300 simulated price paths for each model.

§C. Regime Detection Clustering Metrics

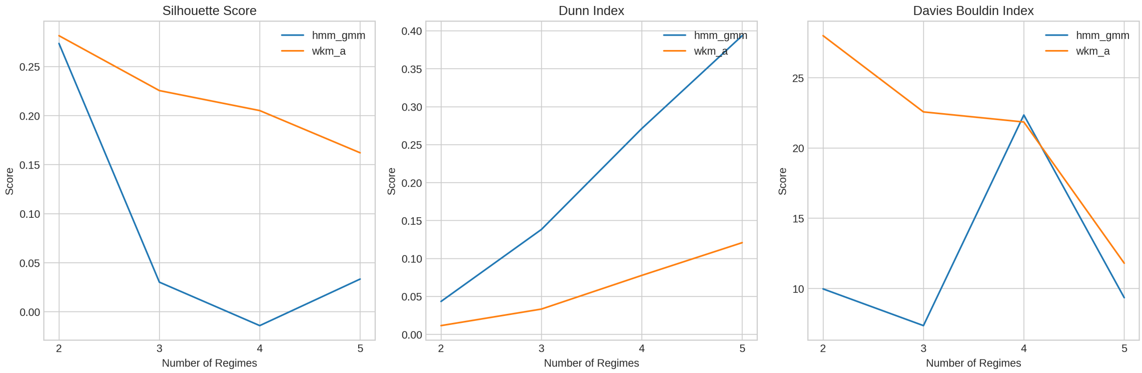

Figure 15. : Silhouette Score, Dunn Index, Davies-Bouldin Index for different models for varying number of regimes over the entire data period. For Silhouette score, the higher the number, the better clusters can be distinguished from each other. Dunn Index is defined as the smallest distance of two points between two clusters divided by the largest distance of two points within either cluster. Therefore, lower scores are better. The Davies-Bouldin index is the average similarity of each cluster with its most similar cluster. Therefore, lower scores are also better.

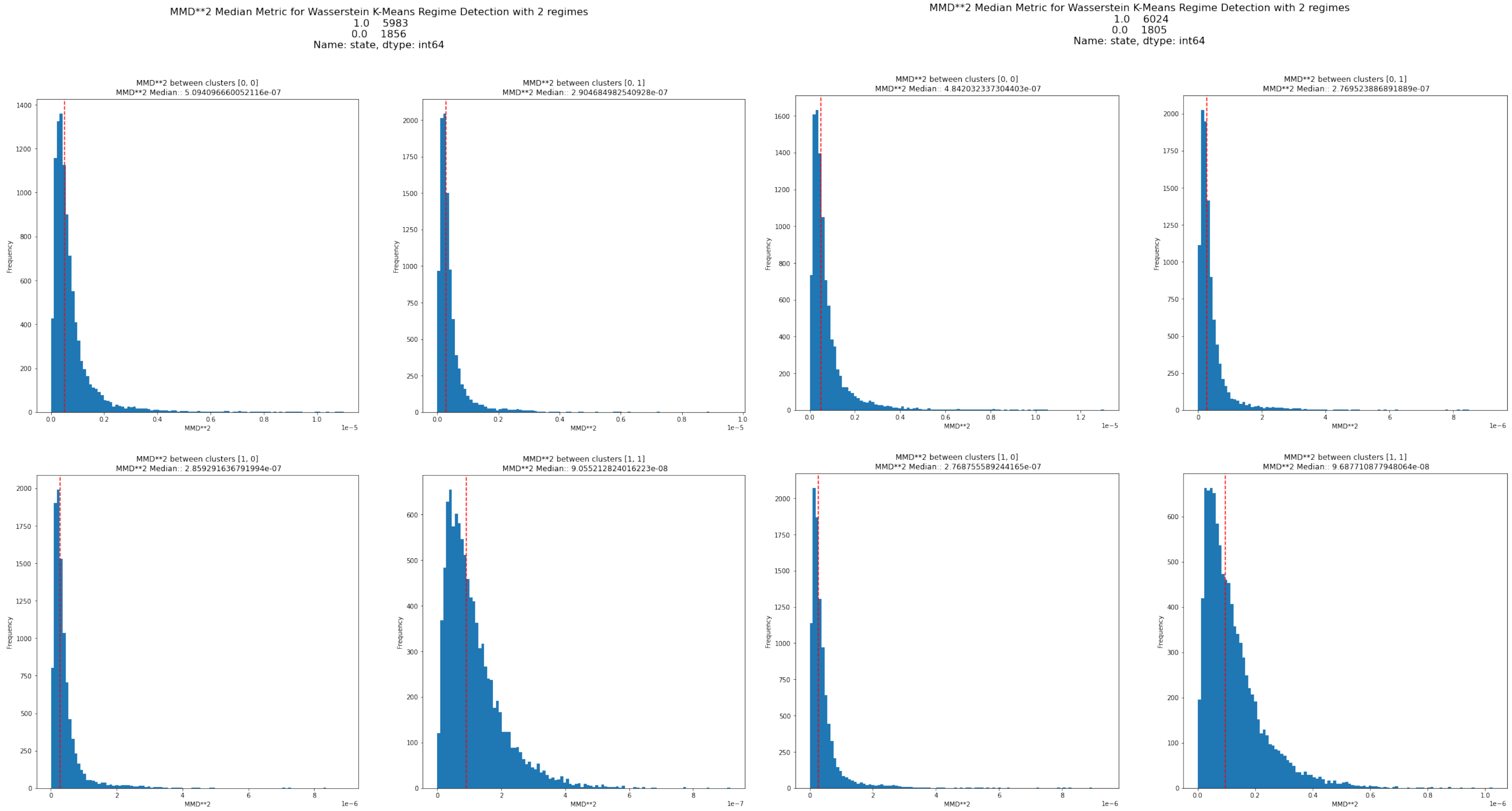

Figure 16. : (Left) Biased Squared MMD bootstrapped histogram for WK-Means with window size 10, (Right) equivalent analysis for a window size of 20. 2 regimes are detected for both statistics.

§D. Regime Detection Robustness Charts

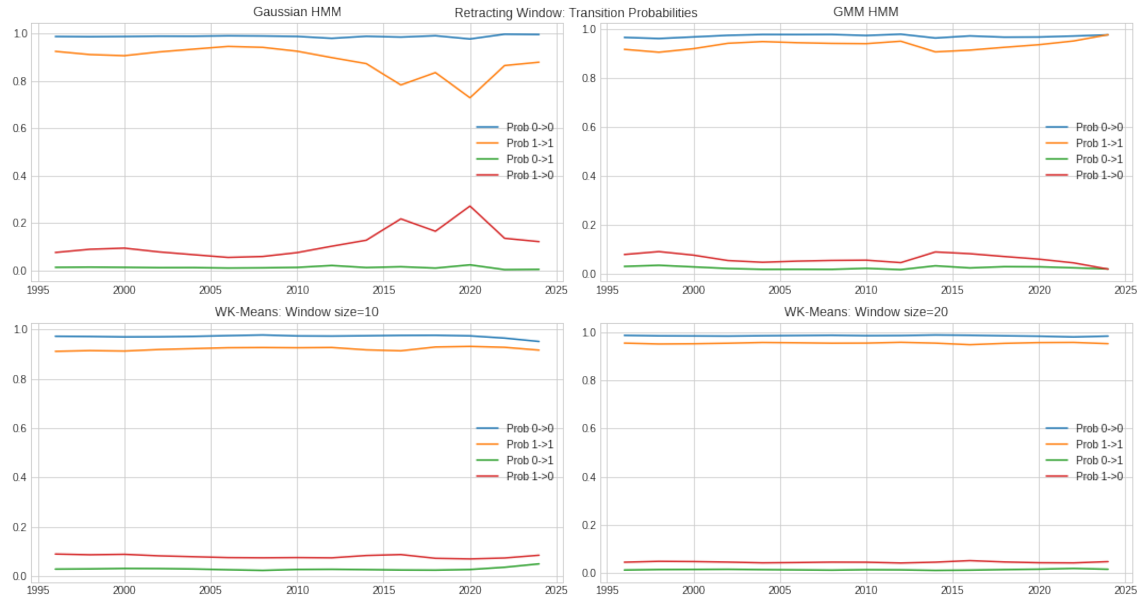

Figure 17. : Retracting windows analysis: Transition Matrix

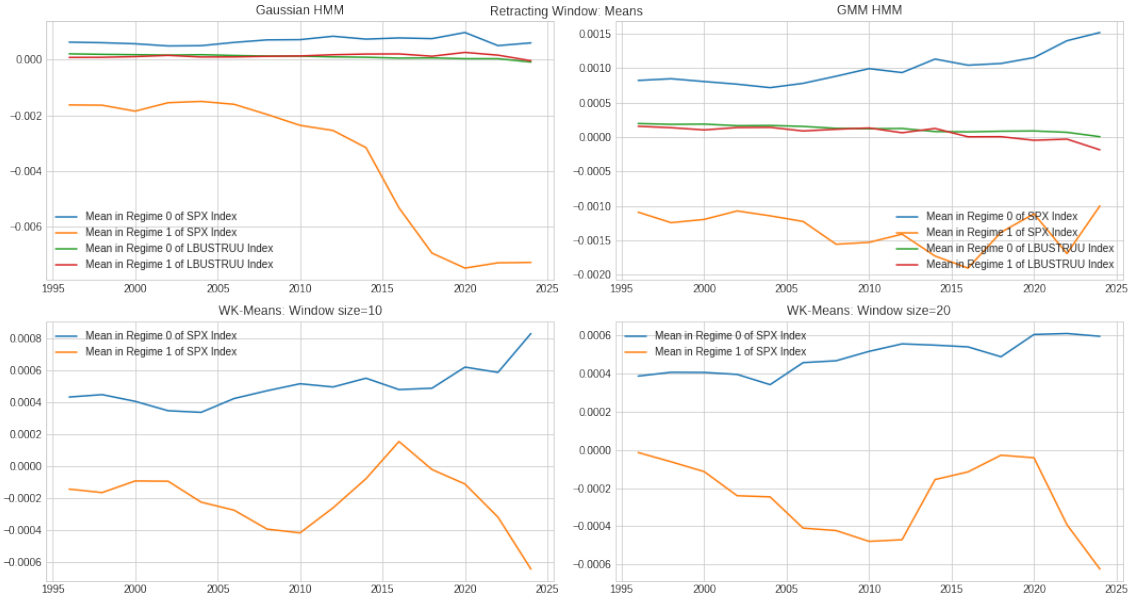

Figure 18. : Retracting windows analysis: Means

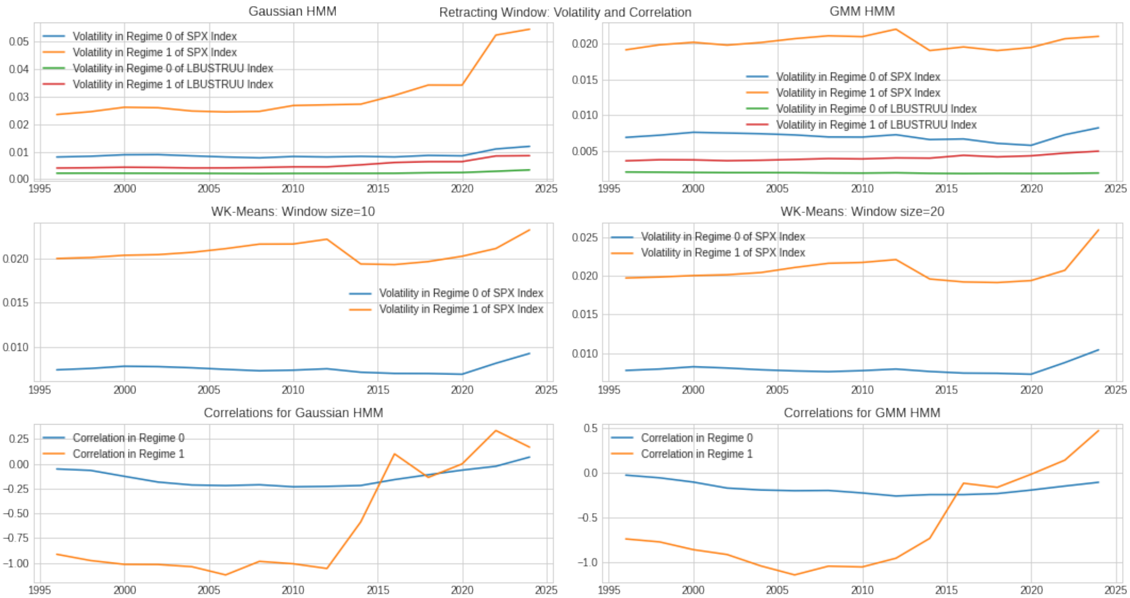

Figure 19. : Retracting windows analysis: Volatility and Correlation

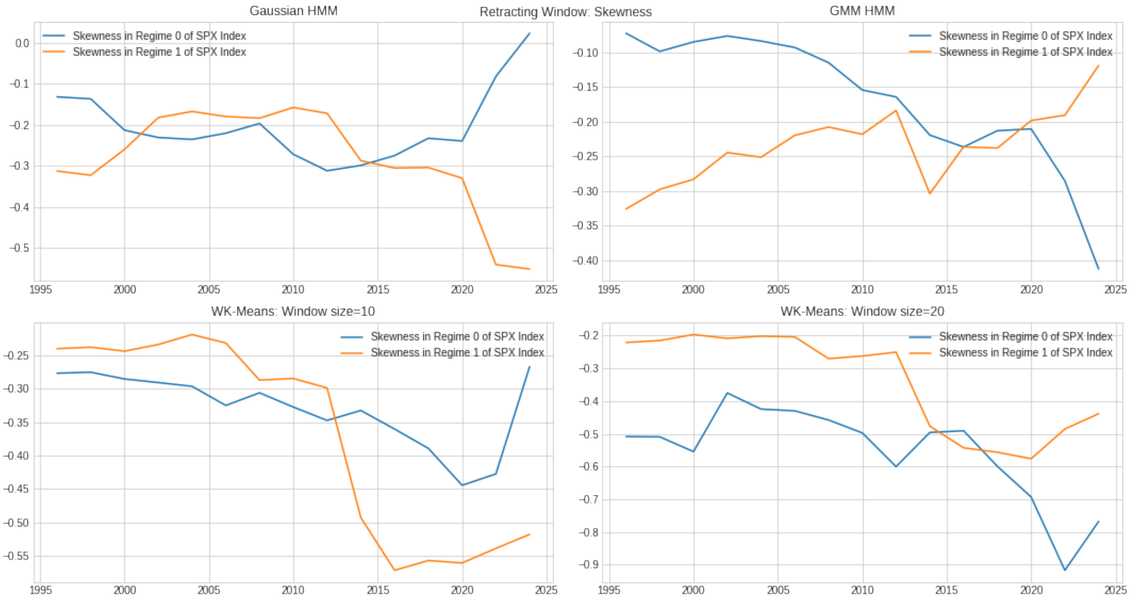

Figure 20. : Retracting windows analysis: Skewness

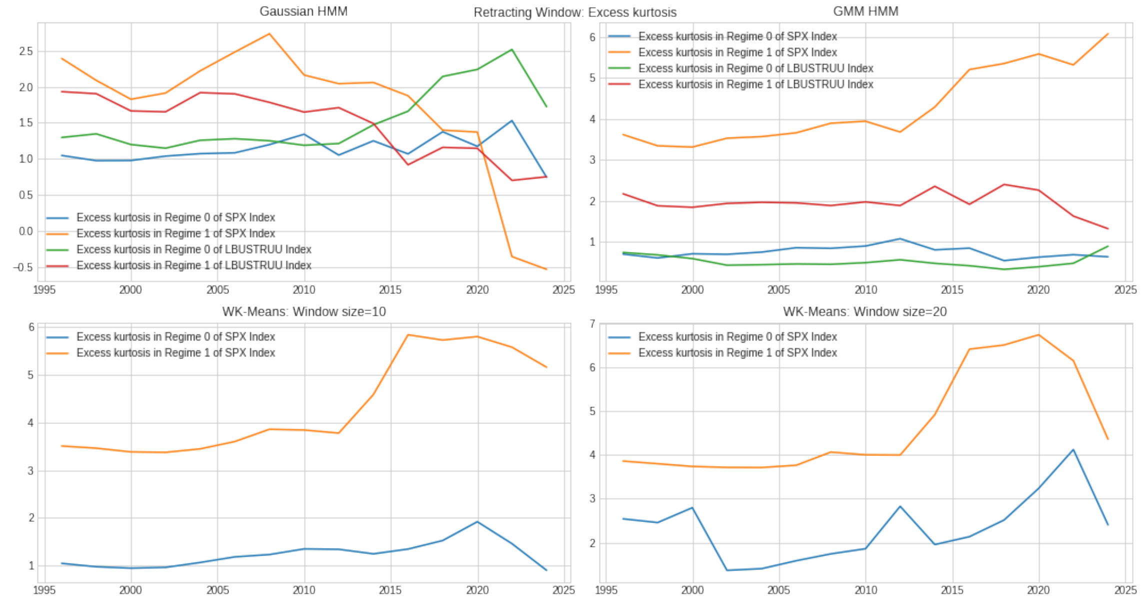

Figure 21. : Retracting window analysis: Excess Kurtosis

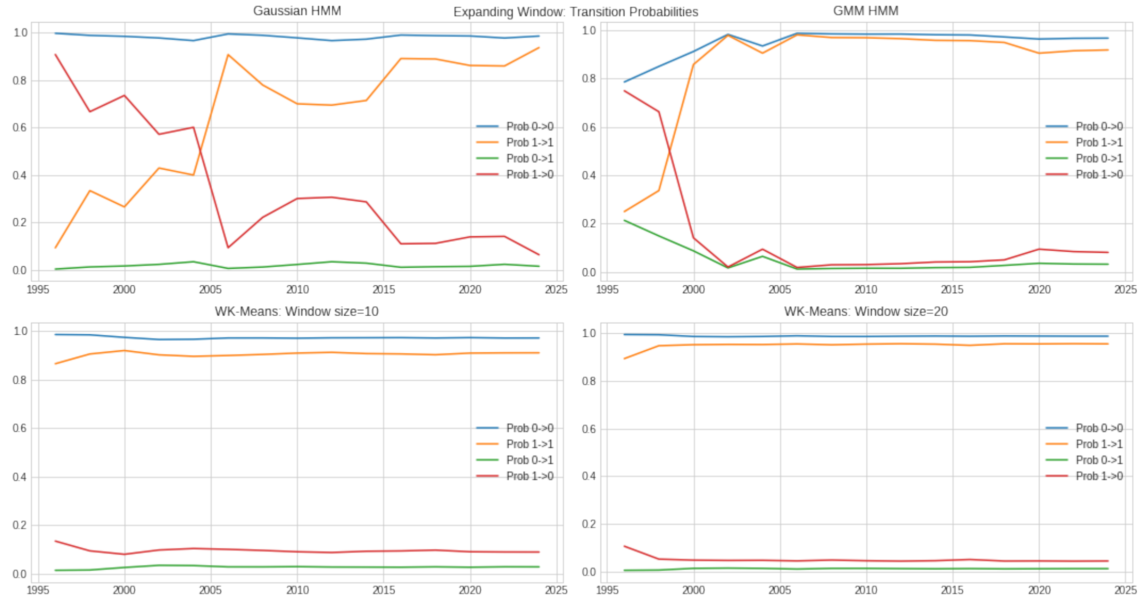

Figure 22. : Expanding windows analysis: Transition Matrix

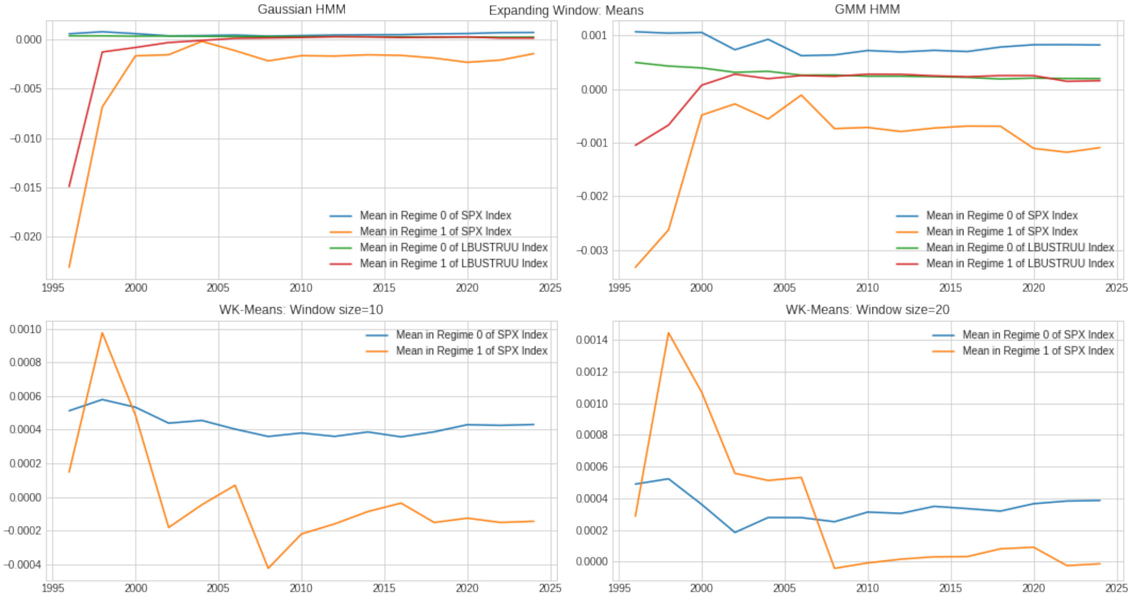

Figure 23. : Expanding windowd analysis: Means

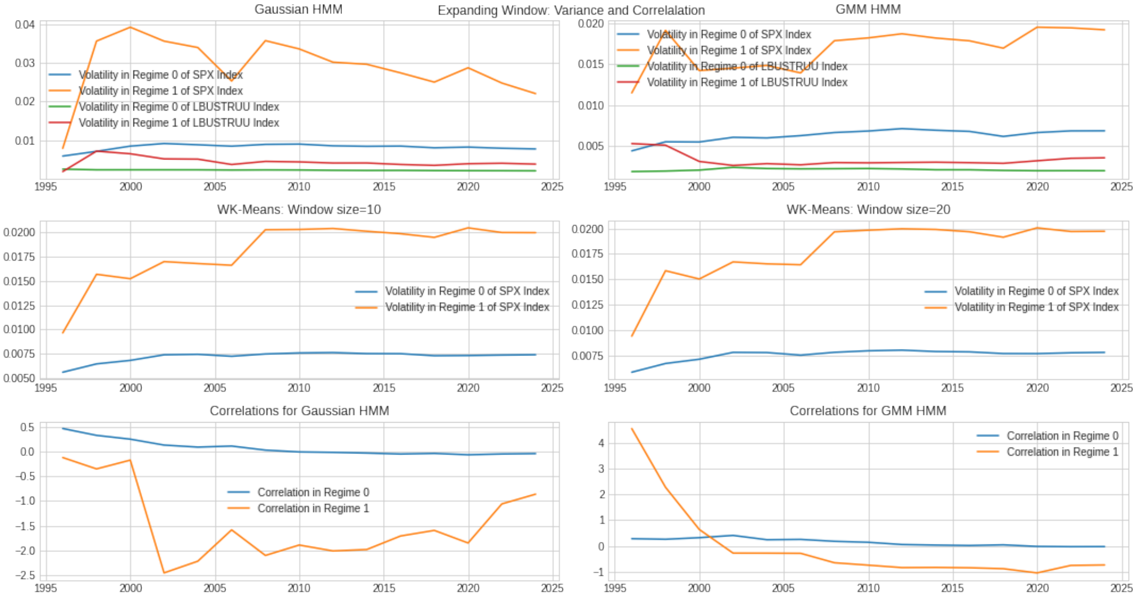

Figure 24. : Expanding windows analysis: Volatility and Correlation

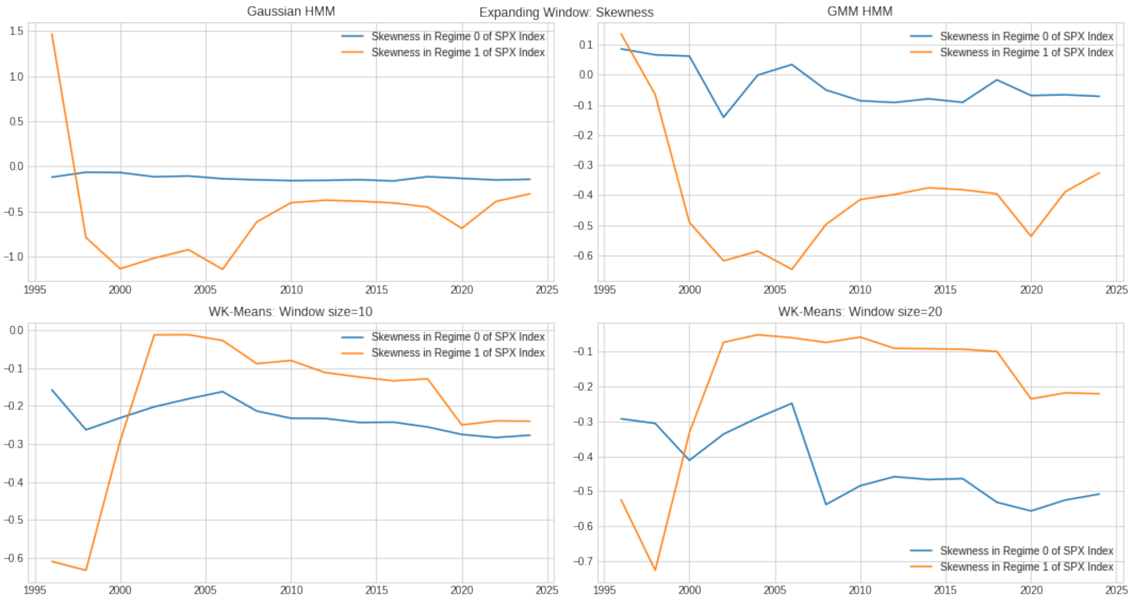

Figure 25. : Expanding windows analysis: Skewness

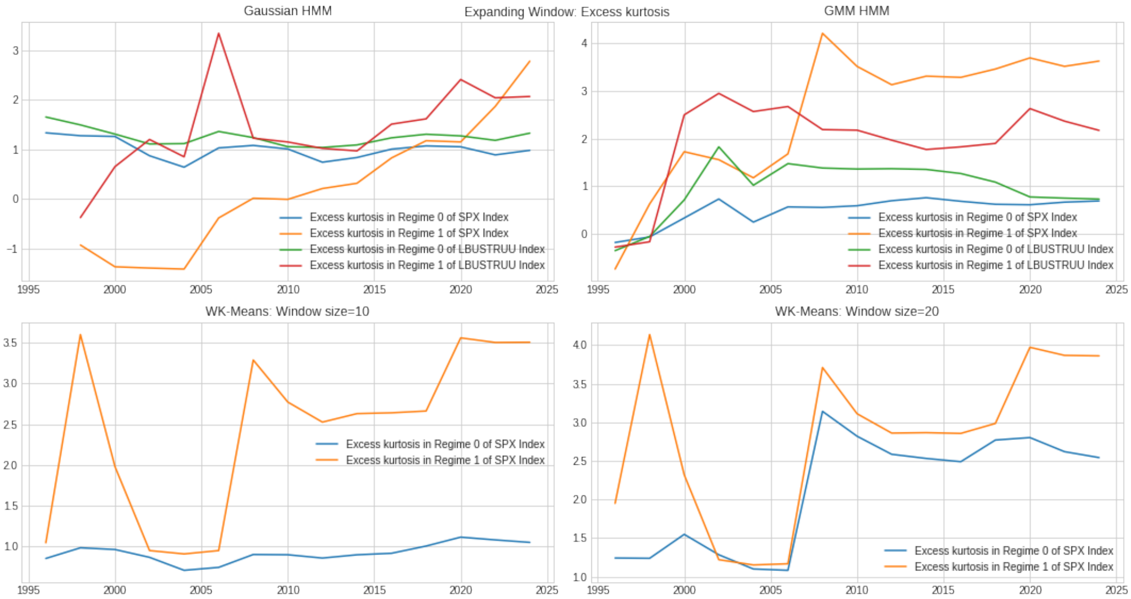

Figure 26. : Expanding windows analysis: Excess Kurtosis

§E. Trading Strategy Backtests

		Avg. Return	Std.	Skew	Kurtosis	VaR	MDD
C&K (2019) - ERC	GMMHMM	7.87%	11.91%	-0.35	9.44	-5.00%	-24.22%
	GaussianHMM	6.52%	11.42%	-0.39	13.52	-4.88%	-25.55%
	WKM_10d	5.51%	13.25%	-0.97	9.89	-5.83%	-42.56%
	WKM_20d	5.38%	13.27%	-0.94	9.86	-5.85%	-46.97%
C&K (2020) - ERC	GMMHMM	7.19%	9.64%	-0.98	18.07	-3.98%	-20.05%
	GaussianHMM	6.90%	11.07%	-0.40	15.39	-4.68%	-25.50%
	WKM_10d	6.63%	12.40%	-0.82	11.85	-5.33%	-29.77%
	WKM_20d	7.13%	10.97%	-0.67	15.90	-4.62%	-20.50%
C&K (2020) - MVO	GMMHMM	3.82%	4.98%	-0.46	9.18	-2.05%	-19.50%
	GaussianHMM	3.42%	5.13%	-0.87	9.60	-2.15%	-21.62%
	WKM_10d	4.05%	4.30%	-0.45	5.21	-1.70%	-16.46%
	WKM_20d	3.97%	4.02%	-0.40	4.29	-1.58%	-15.32%
StocProg - Sharpe	GMMHMM	3.89%	3.84%	-0.68	2.99	-1.50%	-16.37%
	GaussianHMM	4.00%	4.66%	-0.35	9.35	-1.88%	-15.27%
	WKM_10d	3.78%	3.96%	-0.63	3.56	-1.57%	-15.90%
	WKM_20d	3.78%	3.92%	-0.60	3.35	-1.55%	-16.23%
StocProg - VaR	GMMHMM	6.09%	7.21%	-0.06	1.73	-2.92%	-16.55%
	GaussianHMM	5.37%	6.40%	-0.86	5.78	-2.59%	-21.63%
	WKM_10d	4.96%	5.98%	-0.72	2.40	-2.42%	-19.76%
	WKM_20d	5.47%	6.29%	-0.09	2.27	-2.53%	-23.46%
benchmark	ERC	3.42%	4.33%	-0.49	3.52	-1.77%	-15.76%
	EW	5.82%	8.86%	-0.65	3.09	-3.72%	-29.05%
	MVO	3.69%	4.01%	-0.58	3.59	-1.60%	-15.88%

Table 5.: Distributional return metrics across different trading strategies, all with monthly rebalancing. All numbers are annualized. The three highlighted strategies display the best risk-return characteristics as shown in Table 6.

		Sharpe Ratio	Smart Sharpe Ratio	Sortino Ratio	Smart Sortino Ratio	Win Rate
C&K (2019) - ERC	GMMHMM	0.65	0.61	1.02	0.96	69.74%
	GaussianHMM	0.56	0.53	0.86	0.81	68.42%
	WKM_10d	0.41	0.38	0.57	0.54	69.30%
	WKM_20d	0.40	0.37	0.56	0.52	68.86%
C&K (2020) - ERC	GMMHMM	0.74	0.69	1.13	1.05	71.05%
	GaussianHMM	0.62	0.58	0.96	0.89	68.42%
	WKM_10d	0.53	0.49	0.77	0.72	69.30%
	WKM_20d	0.64	0.60	0.99	0.93	69.30%
C&K (2020) - MVO	GMMHMM	0.75	0.71	1.14	1.07	66.67%
	GaussianHMM	0.65	0.61	0.94	0.88	67.11%
	WKM_10d	0.93	0.87	1.46	1.36	67.54%
	WKM_20d	0.97	0.91	1.53	1.44	68.42%
StocProg - Sharpe	GMMHMM	1.00	0.93	1.56	1.46	70.18%
	GaussianHMM	0.84	0.79	1.32	1.23	67.98%
	WKM_10d	0.94	0.88	1.46	1.37	67.98%
	WKM_20d	0.95	0.89	1.48	1.38	68.42%
StocProg - VaR	GMMHMM	0.83	0.78	1.38	1.29	65.35%
	GaussianHMM	0.83	0.77	1.27	1.18	66.23%
	WKM_10d	0.81	0.76	1.22	1.15	67.54%
	WKM_20d	0.86	0.80	1.40	1.31	66.96%
benchmark	ERC	0.77	0.72	1.19	1.12	64.04%
	EW	0.65	0.61	0.95	0.89	66.67%
	MVO	0.90	0.85	1.41	1.32	66.23%

Table 6.: Summarizing risk-return metrics across different trading strategies, all with monthly rebalancing. All numbers are annualized. The three highlighted strategies display the best risk-return characteristics and outperform all benchmark strategies on each risk-return metric.

		Avg. Return	Std.	Skew	Kurtosis	VaR	MDD
C&K1 - ERC	GMMHMM	5.02%	13.23%	-1.61	7.29	-5.87%	-51.02%
	GaussianHMM	7.02%	11.22%	-1.41	7.33	-4.74%	-36.96%
	WKM_10d	5.96%	12.83%	-1.72	8.20	-5.59%	-47.01%
	WKM_20d	5.49%	13.10%	-1.66	7.55	-5.76%	-52.06%
C&K2 - ERC	GMMHMM	6.11%	13.32%	-1.60	7.40	-5.81%	-46.73%
	GaussianHMM	6.33%	11.65%	-1.34	6.80	-5.01%	-37.06%
	WKM_10d	5.67%	13.02%	-1.64	7.81	-5.71%	-47.14%
	WKM_20d	5.47%	13.14%	-1.64	7.66	-5.79%	-52.21%
C&K2 - MVO	GMMHMM	4.19%	5.00%	-0.33	6.99	-2.02%	-18.09%
	GaussianHMM	3.67%	4.99%	-0.81	8.37	-2.07%	-19.88%
	WKM_10d	4.20%	4.45%	-0.37	6.84	-1.76%	-16.89%
	WKM_20d	4.09%	4.18%	-0.41	4.59	-1.64%	-15.09%
StocProg - Sharpe	GMMHMM	4.87%	4.59%	0.99	10.33	-1.77%	-14.78%
	GaussianHMM	3.86%	8.32%	-3.92	51.65	-3.63%	-24.13%
	WKM_10d	3.77%	3.95%	-0.76	3.93	-1.56%	-15.49%
	WKM_20d	3.75%	3.92%	-0.77	4.04	-1.55%	-15.44%
StocProg - VaR	GMMHMM	6.62%	9.20%	-0.63	1.71	-3.81%	-22.53%
	GaussianHMM	4.54%	9.04%	-1.07	3.50	-3.91%	-24.73%
	WKM_10d	4.80%	8.16%	-0.73	1.59	-3.48%	-23.88%
	WKM_20d	4.70%	8.63%	-0.81	2.58	-3.71%	-27.49%
benchmark	ERC	6.50%	12.17%	-1.23	5.26	-5.24%	-26.31%
	EW	5.70%	11.48%	-1.14	4.35	-4.97%	-40.01%
	MVO	3.76%	4.05%	-0.80	4.11	-1.61%	-15.57%

Table 7.: Distributional return metrics across different trading strategies, all with monthly rebalancing. All numbers are annualized. Each trading strategy has an investable universe consisting of the S&P 500 (SPX), investment grade bonds (LBUSTRUU), commodities (BCOMTR), and value equities (RU10VATR). The three highlighted strategies display the best risk-return characteristics as shown in Table 8.

		Sharpe Ratio	Smart Sharpe Ratio	Sortino Ratio	Smart Sortino Ratio	Win Rate
C&K1 - ERC	GMMHMM	0.37	0.36	0.49	0.48	67.11%
	GaussianHMM	0.62	0.60	0.87	0.85	67.54%
	WKM_10d	0.46	0.45	0.61	0.59	67.98%
	WKM_20d	0.41	0.40	0.54	0.53	68.42%
C&K2 - ERC	GMMHMM	0.45	0.44	0.60	0.59	67.54%
	GaussianHMM	0.54	0.52	0.74	0.73	68.42%
	WKM_10d	0.43	0.42	0.57	0.56	67.98%
	WKM_20d	0.41	0.40	0.54	0.53	68.42%
C&K2 - MVO	GMMHMM	0.82	0.81	1.27	1.25	71.93%
	GaussianHMM	0.72	0.70	1.05	1.03	67.98%
	WKM_10d	0.93	0.91	1.45	1.42	69.74%
	WKM_20d	0.96	0.94	1.52	1.48	71.05%
StocProg - Sharpe	GMMHMM	1.05	1.02	1.91	1.86	69.30%
	GaussianHMM	0.45	0.44	0.58	0.57	69.30%
	WKM_10d	0.94	0.92	1.43	1.40	68.42%
	WKM_20d	0.94	0.92	1.43	1.40	69.74%
StocProg - VaR	GMMHMM	0.71	0.69	1.06	1.03	67.54%
	GaussianHMM	0.49	0.48	0.68	0.66	64.04%
	WKM_10d	0.58	0.56	0.83	0.81	65.79%
	WKM_20d	0.53	0.52	0.76	0.74	64.47%
benchmark	ERC	0.53	0.51	0.73	0.72	65.35%
	EW	0.49	0.48	0.67	0.66	63.16%
	MVO	0.91	0.89	1.38	1.35	68.42%

Table 8.: Summarizing risk-return metrics across different trading strategies, all with monthly rebalancing. All numbers are annualized. Each trading strategy has an investable universe consisting of the S&P 500 (SPX), investment grade bonds (LBUSTRUU), commodities (BCOMTR), and value equities (RU10VATR). The three highlighted strategies display the best risk-return characteristics and outperform all benchmark strategies on each risk-return metric.