ChatGPT

Monitoring-Only Market Screening & Scoring System for U.S. Stocks/ETFs

Overview

This dossier designs a **monitor-only** framework that screens U.S.-listed stocks/ETFs under a strict price cap and assigns a **monitor priority score** (1–10) and **risk flags**. The system does **not** recommend trades; it produces data-driven insights so a human analyst can decide whether to research further. It is implementable via a local Windows/PowerShell workflow augmented by Python for heavy analytics. The design emphasises data integrity, avoidance of survivorship/ look-ahead bias, regime awareness, and free data sources.

Constraint → Calculation Table

The table below maps each eligibility gate or feature to its definition, formula, required data fields, free data source(s), recommended update frequency and failure modes. **Long sentences are kept outside the table.** For formulas, `t` indexes trading days.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
Price cap (≤ US$10)	Security is eligible only if the **latest traded price** (preferably the latest official close; if real-time quote available) is ≤ US$10. Use Stooq's last close (daily file) as the default; use Finnhub `/quote` for a more recent price if within trading hours. After-hours quotes may be stale; cap should be applied to the most recent official close to avoid mispricing.	`close[t]` from Stooq `*.us` CSV; `c` (current price) and `t` (timestamp) from Finnhub's `/quote` endpoint.	Daily; quotes intraday when updating watchlist.	Missing data from Stooq (e.g., delisted) or API errors. Use last available close or skip security.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
Minimum liquidity (ADV20)	Compute **average daily dollar volume (ADV20)**: `ADV20 = (1/20) * Σ₁²⁰ ( close[t−i] * volume[t−i] )`. Securities must exceed a threshold (e.g., US$500k) to avoid micro-cap illiquidity.	`close` and `volume` from Stooq daily data.	Daily with 20-day lookback.	Stooq volumes occasionally zero for very low-liquidity tickers; treat zero volumes as missing and exclude.
Volatility (ATR14)	**Average True Range (ATR)** measures average intraday range over 14 days: `TrueRange[t] = max( high[t] − low[t], |high[t] − close[t−1]|, | low[t] − close[t−1]| )`; `ATR14 = (1/14) * Σ₁¹⁴ TrueRange[t−i]`. ATR reflects recent volatility and adjusts for gaps [1].	`high`, `low`, `close` from Stooq.	Daily with 14-day lookback.	Missing values at start of series; drop first 14 days.
Historical volatility (σ20)	Standard deviation of log returns over 20 days: `σ20 = sqrt( (1/19) * Σ₁²⁰ ( ln(close[t−i]/ close[t−i−1]) − μ )² )` where `μ` is the average log return.	`close` from Stooq.	Daily with 20-day lookback.	Non-trading days cause irregular spacing; use business-day index.
Momentum (ROC)	**Rate of Change**: `ROC_n = 100 * (close[t] − close[t−n]) / close[t−n]`. Common lookbacks: 20-day (~1 month) and 60-day (~3 month). A positive ROC indicates upward momentum [2].	`close` from Stooq.	Daily; compute for multiple horizons.	Large price jumps after splits can distort ROC; adjust using split-adjusted prices.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
Moving average trend	Compute **simple moving averages (SMA)** over 20, 50 and 200 days. Signals: price above SMA indicates uptrend; crossovers (e.g., 20 > 50) capture momentum. Trend strength can be ratio of price to SMA.	`close` from Stooq.	Daily.	Sensitive to lookback choice; recompute regularly.
Drawdown metrics	**Maximum Drawdown (MDD)** is the largest peak-to-trough decline: `MDD = min_t ( (min_{τ∈[t,T]} close[τ] − close[t]) / close[t] )`; measure over 1-year. The **Ulcer Index** measures depth and duration of drawdowns by computing squared percentage declines from rolling highs, averaging them, and taking the square root [3].	`close` from Stooq.	Monthly; compute on trailing 1-year window.	Start of series lacking full 1-year history; skip until enough data.
Tail risk (VaR/CVaR)	**Value-at-Risk (VaR)** estimates the loss threshold such that the probability of a larger loss over a horizon (e.g., 10 days) is $\leq \alpha$ (e.g., 5%). **Conditional VaR (CVaR)** measures expected loss conditional on exceeding VaR [4]. Use historical simulation from 1-year rolling returns.	`close` from Stooq; choose horizon (e.g., 10-day).	Monthly.	Past returns may not capture future tails; VaR is not subadditive; CVaR more robust.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
Risk-of-ruin proxy	The **Ulcer Index** acts as a risk-of-ruin proxy because it emphasises prolonged drawdowns [3]. Additional proxies include the **probability of ruin** under a simple Kelly-criterion assumption using volatility and drift.	`close` from Stooq.	Monthly.	Requires assumption of geometric Brownian motion; approximate only.
Factor exposures	Estimate exposures (betas) by regressing excess returns on Fama–French factors: Market (Rm−Rf), Size (SMB), Value (HML), Quality/ Profitability (RMW), Investment (CMA), and **Momentum (Mom)**. Fama–French define SMB and HML as returns on small minus big and value minus growth portfolios [5]; momentum factor is ½(Small High + Big High) – ½(Small Low + Big Low) [6]. Use monthly data from French Data Library.	`close` from Stooq; risk-free rate, factor returns from French Library; 36-month rolling regression.	Monthly.	Rolling regression yields noisy estimates for thinly traded stocks; ensure enough observations.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
Macro/ regime indicators	Use **USREC** from FRED (dummy variable for NBER recession: 1 during recessions, 0 otherwise) [7]. Combine with **Fama–French factor returns** to characterise market regimes (e.g., growth vs. value leadership). Additional series: VIX (if accessible), Treasury yield spread (10y–2y), industrial production growth.	FRED macro series; factor returns.	Monthly to quarterly.	FRED updates with lags; treat current month as preliminary.
Clustering features	For regime detection, compute features such as normalized market return, volatility (ATR/ σ20), breadth proxies (advancers/decliners ratio if accessible), macro indicators (USREC), factor returns. Use standardized values for clustering.	Combined dataset from Stooq, FRED, factor library.	Monthly.	Feature scaling and outlier removal required; see PCA warnings below.
Dilution/ Financing risk	From SEC filings: track share-count changes or recent shelf registrations. Parse 10-K/ 10-Q via EDGAR; high new share issuance or at-the-market offerings may flag dilution. SEC requires adherence to fair access guidelines: max 10 requests/second and descriptive user agent [8].	SEC EDGAR (company facts endpoints).	Quarterly.	Parsing filings is complex; optional for free tier; treat as risk flag if data available.

Gate/Feature	Definition & Formula (key variables)	Data fields & sources	Update frequency	Failure modes
ETF classification	Determine whether a symbol is an ETF using `otherlisted.txt` field `ETF` (Y/N) from Nasdaq symbol directory [9]. If ETF, compute additional features like underlying asset class (equity, fixed income) and total expense ratio (from prospectus).	Nasdaq `nasdaqlisted.txt` & `otherlisted.txt`; ETF category from Stooq or free ETF database.	Daily to monthly.	Misclassification if symbol changes; always refresh symbol directory.

Historical Parallels / Regime Module

1. **Objective:** identify historical windows similar to the current market/security context and derive **conditional outcome distributions** for subsequent returns, drawdowns and recovery times. This module informs the monitor priority score without making predictions.

2. **Feature set:** use standardized vectors of market and security features over a lookback window (e.g., 60 days). Components may include: normalized price momentum (ROC), ATR/σ20, drawdown level, factor exposures, macro indicators (USREC, yield curve slope), and market breadth proxies. Each feature is demeaned and scaled to unit variance to ensure comparability.

3. **Similarity metric:** compute distances between current feature vector and historical vectors using **Euclidean distance** or **Mahalanobis distance** (accounts for feature covariance). Alternatively, use **dynamic time warping** if emphasising sequence shape. Pre-filter historical candidates by regime (e.g., recession vs. expansion using USREC) to avoid mixing incomparable macro environments. Distance computation must exclude data beyond the evaluation date to avoid look-ahead bias.

4. **Analog selection:** select the top `k` (e.g., 10–30) most similar historical windows. For each analog, compute **forward returns** over multiple horizons (10, 20, 60 trading days), **maximum drawdown** and **time-to-recovery**. Aggregate analog outcomes by computing medians, percentiles (e.g., 25th, 75th) and counts of positive vs. negative outcomes. Present results as a distribution rather than a point estimate.

5. **Avoiding look-ahead bias:** ensure analog search uses only data available up to the evaluation date. Rolling windows must be anchored strictly in the past; forward returns used for outcome distributions must be from subsequent periods that do not overlap the current window. Exclude any analog occurring after the evaluation date. Use time-series-safe cross validation (see below) when tuning similarity metrics.

6. **Reporting:** produce a summary table for each security showing median forward return, interquartile range, worst drawdown and median time to recovery across analogs. Include the number of analogs

and highlight if sample size is small. Emphasise uncertainty; avoid projecting analog outcomes as predictions.

Scoring Specification

1. Eligibility Gates

1. **Price Gate**: exclude if price > US$10.
2. **Liquidity Gate**: exclude if ADV20 < threshold (e.g., US$500k). Provide an alert if Stooq reports zero volume for >5% of lookback days.
3. **Listing Gate**: include only common stocks and ETFs from Nasdaq symbol directories. Exclude test issues, structured products and securities flagged as financial distress using `Financial Status` codes (D, E, Q in nasdaqlisted.txt [9]).

Only securities passing all gates move to scoring.

2. Sub-Scores

Assign sub-scores (0–100) for the following dimensions. Each sub-score is mapped to a 0–10 scale when combined. Weights may be calibrated using historical performance evaluation (see backtesting). Default weights emphasise balanced contributions.

1. **Liquidity & Size (15%)**: Derived from ADV20 and market capitalisation (if available). Higher liquidity yields higher score; micro-cap discount.
2. **Volatility & Tail Risk (20%)**: Combine ATR14/σ20 (lower volatility favourable) and tail risk metrics (VaR/CVaR). Penalise extreme volatility or fat tails.
3. **Momentum & Trend (25%)**: ROC values (20 & 60 days), moving-average slopes. Positive momentum/trend increases score; negative momentum reduces.
4. **Drawdown & Recovery (15%)**: Evaluate current drawdown and Ulcer Index. Lower current drawdown and faster recovery yield higher score.
5. **Factor Alignment (10%)**: Exposures to momentum, size and value factors relative to prevailing regime. For example, during a small-cap momentum regime, securities with positive SMB and Mom betas receive higher scores.
6. **Regime Fit (15%)**: Based on historical parallels. If analog outcomes show favourable conditional distributions (e.g., positive median forward return, small worst-case drawdown), increase score. If analogs indicate high downside risk, decrease score.

3. Score Calibration

1. **Normalization:** rescale each sub-score between 0 and 1 based on historical 5th–95th percentile values across the universe. Combine using weights to compute a composite **raw score**.
2. **Binning to 1–10 scale:** map raw scores to deciles relative to historical distribution. A score of 10 indicates a historically rare alignment of favourable factors (top decile), while 1 indicates bottom decile. Use rolling history to update decile thresholds. Avoid calibrating on data that includes the evaluation period to prevent look-ahead bias.
3. **Risk Flags:** assign qualitative flags independent of numeric score. Examples:

4. **Red flag:** Stagnant liquidity (ADV20 < threshold), abnormally high ATR relative to peers, extremely negative ROC (e.g., < −20%), regulatory event (e.g., SEC investigation), repeated EDGAR filings of share issuance.
5. **Amber flag:** Near price cap (e.g., $9.50–$10), moderate drawdown (> −30%), high Ulcer Index.
6. **Green flag:** Price well below cap, high liquidity, positive momentum, low volatility.

4. Presentation

The final **monitor priority score** (1–10) and flags should be presented alongside a decomposition showing sub-scores and analog statistics. Emphasise that scores are relative, not predictive. Provide cautionary statements about uncertainty.

Data Engineering Plan

Universe Construction

1. **Symbol directories:** download `nasdaqlisted.txt` and `otherlisted.txt` from Nasdaq Trader daily. These files include symbol, security name, market category, ETF flag, test issue flag, financial status and round lot size [9]. Parse them to create a master universe of U.S. common stocks and ETFs. Exclude test issues and financial distress codes. Record an `is_etf` flag from the ETF column.

2. **Watchlist vs. full universe:** allow user to supply a `watchlist.txt`. If watchlist is empty, scan the full universe; else restrict computations to the watchlist. Provide option to filter by exchange or sector.

Data Retrieval

1. **Finnhub Quotes:** use `/quote` endpoint for latest price and timestamp. Free tier allows **60 calls per minute** [10]; implement rate-limiting (sleep after each batch) and exponential back-off on HTTP 429. Cache quotes locally with timestamp to avoid redundant calls. Document API key storage in a secure file (not hard-coded).

2. **Stooq Historical Data:** Stooq offers free daily OHLCV files accessible via CSV with `.us` suffix for U.S. stocks [11]. Download the CSV once per day. Use the longest available history for computations (30+ years for some tickers). Note: there is no API; rely on HTTP GET of `https://stooq.com/q/d/l/?s=AAPL.us&i=d` pattern. Handle missing values and splits (prices are split-adjusted). Save to local cache and update incrementally.

3. **Factor & Macro Data:** download Fama–French factor returns (five factors + momentum) from Kenneth French's data library (monthly and daily). Retrieve USREC (recession dummy) and macro series from FRED via `fred.stlouisfed.org` API or Python's `pandas_datareader`. Some macro series (VIX, yield spread) are accessible via FRED but may have licensing restrictions; verify before distribution.

4. **Fundamentals from EDGAR:** optional. Use SEC's EDGAR company facts or filings (10-K, 10-Q) to compute fundamental metrics like earnings yield, debt-to-equity, share count changes. Comply with SEC fair access guidelines: maximum **10 requests per second** and include a descriptive user agent

[8]. Implement caching and respect robots.txt. For each company, fetch relevant forms only once per quarter.

Data Handling & Quality Controls

1. **Timezone & calendar:** align all data to **America/New_York** timezone. Use U.S. trading calendar (NYSE) to handle holidays. For cross-source alignment, map Stooq dates to trading days; ensure Finnhub timestamps convert to local date.

2. **Caching & versioning:** store raw downloaded files in a `data/raw/YYYY-MM-DD` folder and processed features in `data/processed/`. Keep versioned snapshots of the output CSV to enable reproducibility.

3. **Deduplication & symbol changes:** handle ticker changes (e.g., due to corporate actions) by cross-referencing CUSIP/CIK from EDGAR or other free mapping service. Maintain mapping tables to avoid mis-attributing historical data.

4. **Missing data:** implement missingness checks in the EDA layer. If more than 20% of data in a window is missing, skip that feature calculation for the security. Provide warnings in the report.

5. **Split & dividend adjustments:** Stooq provides split-adjusted prices but not dividends; for total return metrics, adjust using dividends data if accessible via FRED or other free sources. For simplicity, use price returns and note limitation.

Rate Limit & Concurrency Management

Implement a **pacing controller** in PowerShell that tracks API call timestamps and sleeps when approaching Finnhub's 60/min limit or EDGAR's 10/sec limit. Use asynchronous Python requests within rate limits for factor/macro retrieval. When scanning the full universe (~thousands of tickers), schedule retrieval across multiple minutes to avoid throttling.

Exploratory Data Analysis & Bias Controls

EDA Workflow

1. **Missingness & Outliers:** compute percentage of missing data per feature; identify days with zero volume or extreme returns; winsorize or log-transform heavy-tailed features. Visualise distributions to check for skewness and outliers.

2. **Stationarity & scaling:** for time-series models, check stationarity of returns via Augmented Dickey–Fuller test; difference non-stationary series if needed. Standardise features before applying PCA or clustering.

3. **Correlation & Dimensionality Reduction:** compute correlation matrix among features. Use **Principal Component Analysis (PCA)** for visualization and to identify latent factors. Note that PCA is sensitive to period selection and outliers, and the principal components may change across regimes

[12]. Avoid using PCA components directly for scoring; instead, use them for exploratory insights and ensure recalibration across regimes.

Regime Detection & Clustering

1. **Unsupervised Clustering:** apply **k-means** clustering to standardized regime features (market returns, volatility, breadth, macro indicators). K-means partitions the data into clusters by minimizing within-cluster variance; it requires initialization and may converge to local minima [13]. Determine the optimal number of clusters using the elbow method or silhouette score. Alternatively, use **Gaussian Mixture Models (GMM)** which allow different covariance structures and generalise k-means [14]. Evaluate stability of clusters over time.

2. **Time-Series Regime Models:** implement **Hidden Markov Models (HMM)** or **Markov-switching models** to model latent volatility regimes. HMMs assume hidden states drive observable returns and can detect high-volatility vs. low-volatility regimes [15]. Calibrate HMMs using expectation–maximization on rolling windows; monitor state probabilities.

3. **Labeling:** For supervised tasks, use NBER recession dummy (USREC) as ground truth for macro regimes. For unsupervised tasks, regimes are model-derived; evaluate them qualitatively against known events (e.g., 2008 crisis, COVID-19) and check if clusters correspond to plausible risk regimes.

Time-Series-Safe Validation

1. **Walk-Forward & Rolling Windows:** always split data chronologically. Use scikit-learn's `TimeSeriesSplit` cross-validator which produces train/test splits that respect time order [16]. At each step, train models on data up to time `t` and evaluate on the next period. This avoids training on future data and replicates real-time evaluation.

2. **Nested Validation:** for hyper-parameter tuning (e.g., number of clusters), nest an inner validation within each train window. Avoid using full sample to choose hyper-parameters.

Bias & Pitfalls Mitigation

1. **Look-Ahead Bias:** never use data after the evaluation date when computing features or calibrating models. For example, when computing moving averages, only use past prices. Be careful when shifting indexes; ensure drawdown maxima/minima are lagged by at least one period. See QuantStart's list of look-ahead errors and mitigation [17].

2. **Survivorship Bias:** using only current (surviving) tickers inflates performance. Use Nasdaq symbol directories to include delisted tickers when available; otherwise, restrict backtest to more recent years to reduce the impact [17].

3. **Data Snooping & Overfitting:** avoid trying many features on the same data and selecting the best performing combination. Use out-of-sample evaluation and multiple testing corrections. Publish the full set of tested features rather than only the best.

4. **Transaction Costs & Slippage:** although this system is monitoring-only, any hypothetical backtest used for calibration should include conservative assumptions for bid–ask spreads and slippage. Emphasise that frequent trading of illiquid securities can lead to high costs.

5. **PCA & Clustering Limitations:** PCA results depend on time period and can change when market regimes shift; clusters may be unstable. Use robust PCA or sliding-window recalibration and treat clusters as approximate rather than definitive [12].

Codex Handoff: Implementation Plan

The following tasks should be implemented by Codex in PowerShell and Python. A minimal Python environment (`python>=3.9`) with `pandas`, `numpy`, `scikit-learn`, `statsmodels` and `pandas_datareader` should be installed. Use PowerShell for orchestration and logging; call Python scripts for compute-intensive tasks.

PowerShell Orchestrator

1. **Configuration:** load API keys, rate-limit parameters and user options (watchlist vs. full universe) from a config file.
2. **Universe Build:** download `nasdaqlisted.txt` and `otherlisted.txt` daily; parse and update the universe CSV.
3. **Data Retrieval Loop:** for each ticker in the universe or watchlist:
4. If first time or data is stale, call Finnhub `/quote` (respect 60/min rate limit) and update the latest price/time fields.
5. Download Stooq daily CSV (if not cached) using `Invoke-WebRequest` and save to local folder. Use conditional `-IfModifiedSince` header to avoid unnecessary downloads.
6. **Feature Calculation:** call Python script `compute_features.py` with arguments (ticker, data paths). The script reads Stooq data, computes features (adv20_dollar, ATR, σ20, ROC, moving averages, drawdowns, VaR/CVaR, factor betas, etc.), and returns a JSON or CSV row.
7. **Scoring:** after feature calculation, call Python script `score_security.py` which applies eligibility gates, sub-score calculations, historical parallels module and calibration to produce final score and flags. Append results to `scored.csv` and, if eligible, to `eligible.csv`.
8. **Logging & Error Handling:** log API call responses, errors and missing data. Implement retries with exponential back-off for HTTP errors. Record the number of securities processed and skipped.

Python Feature Computation (compute_features.py)

1. **Load Data:** read daily OHLCV from Stooq. Resample if needed to ensure business-day continuity.
2. **Compute Liquidity & Volatility:** calculate ADV20, ATR14, σ20 as defined. Compute momentum (ROC) for multiple horizons and moving averages.
3. **Drawdowns & Tail Risk:** compute 1-year maximum drawdown, Ulcer Index, VaR (e.g., 95% historical) and CVaR. For VaR/CVaR, use bootstrapping if sample is small.
4. **Factor Betas:** align monthly returns with Fama–French factors; compute rolling 36-month regression using `statsmodels`. Output betas.
5. **Macro Features:** fetch USREC and other macro series via `pandas_datareader` (cached) and align with security dates.
6. **Save Features:** output to CSV/JSON for orchestrator.

Python Scoring & Historical Parallels (score_security.py)

1. **Eligibility Check:** apply price, liquidity and listing gates. If fails, mark as ineligible and return.
2. **Compute Sub-Scores:** normalise each feature using historical percentiles stored in a calibration file. Compute raw scores for each dimension and combine using weights. Generate risk flags based on threshold conditions.
3. **Historical Parallels Module:** load pre-computed feature vectors from historical database; compute distances and select nearest neighbours as described. Compute conditional outcome distributions and incorporate them into the regime fit sub-score.
4. **Calibration & Deciling:** update running percentiles to map raw scores to 1–10. Avoid using current security in calibration to prevent leakage.
5. **Output:** return final score, sub-scores, flags and analog statistics.

Testing & Reporting

1. **Unit Tests:** implement tests verifying formulae: ADV20, ATR14, σ20, ROC, MDD, Ulcer Index, VaR, factor regression. Provide synthetic data to validate output.
2. **Integration Tests:** run orchestrator on a small subset of tickers and verify that data retrieval, feature computation and scoring complete without errors.
3. **Reporting:** generate a `run_report.md` summarising the day's run: number of tickers processed, eligibility pass rate, distribution of scores, flagged securities, missing data warnings and any API errors.

Conclusion

This blueprint offers an end-to-end design for a monitoring-only screening and scoring system for U.S. stocks and ETFs. The framework uses free data sources (Stooq, Finnhub, FRED, Fama–French factors) and emphasises bias mitigation, regime awareness and reproducibility. Implementing the workflow via PowerShell and Python ensures cross-platform compatibility and leverages robust data-science libraries. The scoring system provides relative, interpretable signals and highlights risk through qualitative flags, supporting informed and cautious monitoring rather than actionable investment recommendations.

---

1 Average True Range (ATR) Formula, What It Means, and How to Use It
https://www.investopedia.com/terms/a/atr.asp

2 Rate of Change (ROC) Indicator: Definition and Formula
https://www.investopedia.com/terms/p/pricerateofchange.asp

3 Understanding the Ulcer Index: A Guide to Measuring Downside Risk
https://www.investopedia.com/terms/u/ulcerindex.asp

4 Understanding VaR and Tail Risk: Insights Into Loss Distributions
https://www.investopedia.com/ask/answers/041615/what-does-value-risk-var-say-about-tail-loss-distribution.asp

5 Kenneth R. French - Description of Fama/French Factors
https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_factors.html

6 Kenneth R. French - Detail for Monthly Momentum Factor (Mom)
https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_mom_factor.html

7 NBER based Recession Indicators for the United States from the Period following the Peak through the Trough (USREC) | FRED | St. Louis Fed
https://fred.stlouisfed.org/series/USREC

8 SEC.gov | Accessing EDGAR Data
https://www.sec.gov/search-filings/edgar-search-assistance/accessing-edgar-data

9 Symbol Directory Definitions
https://www.nasdaqtrader.com/trader.aspx

10 Rotational Labs | Finnhub
https://rotational.io/data-playground/finnhub/

11 An Introduction to Stooq Pricing Data | QuantStart
https://www.quantstart.com/articles/an-introduction-to-stooq-pricing-data/

12 Principal Component Analysis (PCA) for Portfolio Risk | QuestDB
https://questdb.com/glossary/principal-component-analysis-pca-for-portfolio-risk/

13 2.3. Clustering — scikit-learn 1.8.0 documentation
https://scikit-learn.org/stable/modules/clustering.html

14 2.1. Gaussian mixture models — scikit-learn 1.8.0 documentation
https://scikit-learn.org/stable/modules/mixture.html

15 Market Regime Detection using Hidden Markov Models in QSTrader | QuantStart
https://www.quantstart.com/articles/market-regime-detection-using-hidden-markov-models-in-qstrader/

16 TimeSeriesSplit — scikit-learn 1.8.0 documentation
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

17 Successful Backtesting of Algorithmic Trading Strategies - Part I | QuantStart
https://www.quantstart.com/articles/Successful-Backtesting-of-Algorithmic-Trading-Strategies-Part-I/