Data Science and Advanced Programming 2025

Detecting Risk-On and Risk-Off Regimes in Crypto Markets: A Comparative Modeling Approach

Final Project Report

Elvis Hoxha
elvis.hoxha@unil.ch
Student ID: 22404503

January 8, 2026

Abstract

This project addresses the problem of identifying market regimes in cryptocurrency markets, with a focus on Bitcoin, in the absence of explicit regime labels. We investigate whether unsupervised learning can recover a small number of latent states that summarize changing market conditions and can be interpreted through a Risk-On/Risk-Off framework. Using daily data for BTC and ETH together with risk proxies (S&P 500 and VIX) sourced from Yahoo Finance, we engineer features capturing returns, rolling volatility, volume dynamics, and cross-asset correlations. We then compare three unsupervised methodologies: a Gaussian Hidden Markov Model (HMM) to capture temporal dependence, a Gaussian Mixture Model (GMM) for static clustering, and an autoencoder followed by KMeans to learn a non-linear latent representation before clustering. Results show that all models separate the sample into two regimes with clearly different levels of realized return dispersion, while the HMM produces more persistent regime segments and a more stable transition structure than clustering-based approaches. The main contributions are an end-to-end Python pipeline for data acquisition, feature engineering, regime inference, and evaluation, along with a comparative analysis using regime-conditional statistics, regime timelines, and transition-count matrices.

Keywords: data science, Python, machine learning, unsupervised learning, market regime detection, cryptocurrency, Hidden Markov Model, Gaussian Mixture Model, autoencoder, Risk-On/Risk-Off

Contents

1 Introduction

In financial markets, the terms Risk-On and Risk-Off refer to broad market regimes characterized by changes in investors' risk appetite. A Risk-On environment denotes periods in which investors are willing to take risk and allocate more capital to assets perceived as riskier (e.g., equities or cryptoassets), often in search of higher returns. Conversely, a Risk-Off environment corresponds to periods of heightened risk aversion and market stress, during which investors tend to reduce exposure to risky assets, prefer liquidity, and reallocate toward assets perceived as safer. While Risk-On/Risk-Off does not represent a formal binary state and may vary across asset classes, it is commonly associated with systematic differences in market behavior, most notably higher realized volatility and more unstable return dynamics during Risk-Off episodes, versus more stable conditions during Risk-On periods. In this project, these notions are used as an interpretative framework for the regimes inferred from market data using unsupervised models.

Cryptocurrency markets, and Bitcoin in particular, are known for pronounced volatility, sudden drawdowns, and changing correlations with traditional assets. For investors and risk managers, these shifts matter because they influence the risk profile of portfolios and can affect the performance of allocation or hedging decisions. This motivates the use of data-driven, unsupervised regime detection methods to summarize market conditions into a small number of recurring latent states.

1.1 Problem statement

Despite the intuitive appeal of regime narratives, market regimes are not directly observed and there is no universally accepted ground-truth label for Risk-On/Risk-Off in crypto. The problem addressed in this project is therefore to determine whether unsupervised learning methods can infer a small number of latent regimes from historical market data, and whether these regimes exhibit economically meaningful differences in return and risk characteristics.

1.2 Objectives and goals

The objectives of this project are: (i) to build a reproducible pipeline that downloads daily market data (BTC, ETH, and risk proxies such as SPX and VIX) and constructs engineered features capturing returns, volatility, and cross-asset co-movement; (ii) to apply and compare three unsupervised regime-detection approaches—Gaussian HMM, Gaussian mixture clustering, and an autoencoder followed by KMeans; and (iii) to evaluate the inferred regimes using diagnostic tools appropriate for an unsupervised setting, including regime-conditional return statistics, regime timelines, and transition-count matrices. Throughout, Risk-On/Risk-Off is used as an interpretative framework rather than a claimed ground truth.

1.3 Report organization

The remainder of the report is structured as follows. Section 2 reviews related work on market regime modelling and unsupervised methods. Section 3 presents the dataset and feature engineering choices and describes the three modelling approaches. Section 4 reports the main empirical results and visual diagnostics. Section 5 discusses what worked well, key challenges, limitations, and future directions, and Section 6 concludes.

2 Literature Review / Related Work

2.1 Market regimes and the Risk-On/Risk-Off concept

Financial markets are often described as switching between a small number of latent "regimes", such as calm growth phases and stress or crisis phases. In practitioner language, these broad conditions are frequently referred to as *Risk-On* (higher risk appetite) and *Risk-Off* (heightened risk aversion), although the boundaries between regimes are not objectively observed and may differ across asset classes. A common empirical signature of regime changes is a shift in volatility, correlation structure, and tail risk. To capture such regime dynamics, prior studies have typically relied on unsupervised or probabilistic models, such as regime-switching models or clustering-based approaches, to infer latent market states from observed financial data.

2.2 Methodological approaches to regime detection

A large body of work models market regimes as latent states that generate observable returns and risk indicators. A classical probabilistic approach is the Hidden Markov Model (HMM)[1], where the latent regime follows a Markov chain and observations are emitted conditionally on the regime. HMMs are attractive for financial regimes because they explicitly model persistence and transitions between states, providing an interpretable transition structure[2].
An alternative family of approaches treats regime detection as a clustering problem. Gaussian Mixture Models (GMMs)[3]. represent the distribution of observations as a mixture of Gaussian components and assign each observation to a component based on likelihood. Compared to HMMs, GMMs are static (independent across time) and can react more quickly to local changes, but they may also produce less temporally coherent regime sequences when the underlying process is strongly autocorrelated.
More recent work explores representation learning for financial time series, using neural networks to construct low-dimensional embeddings that capture non-linear structure. Autoencoders learn to compress input features into a latent space that preserves information needed for reconstruction. Clustering[4]. can then be performed in this latent space (e.g., with KMeans), potentially capturing non-linear separations not accessible to linear methods. This pipeline typically trades interpretability for flexibility: the latent dimensions are not directly tied to economic variables, so regimes often require post-hoc interpretation[5].

2.3 Regimes in cryptocurrency markets and datasets

Cryptocurrency markets exhibit properties that can make regime detection both relevant and challenging: high and time-varying volatility, heavy-tailed returns, and frequent structural breaks. As a result, regime boundaries may be less stable than in traditional assets, and models with strong distributional assumptions (e.g., Gaussian emissions) can be misspecified. As a consequence, most empirical studies rely on relatively simple but robust datasets, typically based on daily price and volume information. Related studies commonly rely on daily OHLCV data and augment it with risk proxies or cross-asset information (e.g., equity indices, volatility indices, or macro indicators) to capture broader market sentiment. In this project, we follow this line by using BTC and ETH returns and volatility features, together with correlation features involving SPX and VIX[8], as inputs for unsupervised regime discovery.

3 Methodology

3.1 Data Description

The data used in this project originate from publicly available financial time series obtained from Yahoo Finance through the `yfinance`[10]. Python library. The data collection process is

implemented in a dedicated data loader module, which retrieves historical OHLCV data for both cryptocurrency assets and market-related indicators. Examples include major crypto assets such as Bitcoin and Ethereum, as well as traditional market proxies like the S&P 500 index and the VIX.
For practical and reproducibility reasons, the downloaded data are saved locally as CSV files and subsequently used throughout the project. An offline-first data loading strategy is adopted: when a local CSV file already exists for a given symbol, it is reused directly and no external request to Yahoo Finance is made. Online data retrieval is only triggered if the corresponding local file is missing. This design choice ensures that the project can be executed reliably on the Nuvolos platform at any time, without depending on external data availability or API stability. As a result, all experiments presented in this report are based on fixed, versioned datasets stored locally.
Raw time series data are stored in the `data/raw/` directory, with each file corresponding to a specific asset or indicator. Feature engineering is performed in a separate processing step, where derived variables such as returns and volatility measures are computed and saved in a consolidated feature dataset. All series are aligned on a common time axis, with each observation representing a single time period. Minor data quality issues, primarily related to missing values at the beginning of certain series, are handled during the preprocessing stage.

3.2 Approach

This project detects and compares Risk-On/Risk-Off regimes in crypto markets using an unsupervised, model-comparison framework. The goal is not to predict labeled regimes, but to infer latent market states from engineered financial features and assess their consistency.
Three complementary algorithms are used. A two-state Hidden Markov Model (Gaussian emissions) is applied to capture regime persistence and time-dependent transitions. A two-component Gaussian Mixture Model provides a static clustering baseline where observations are assigned independently to mixture components. Finally, a non-linear autoencoder learns low-dimensional embeddings of the standardized feature space, which are then clustered into two regimes using KMeans.
Raw OHLCV data (crypto assets and macro-financial indicators such as the S&P 500 and the VIX) are transformed into model-ready inputs through feature engineering. The resulting feature set includes log returns, rolling volatility (21-day window, annualized), rolling correlations with benchmark return series, and a rolling z-score of trading volume computed over a 60-day window. Rolling windows introduce initial missing values, which are removed during preprocessing. Prior to training, features are standardized (zero mean, unit variance) using a dedicated scaler fitted on the training sample and reused for inference.
Evaluation focuses on interpretability and robustness rather than predictive accuracy. Regimes are assessed via regime-conditional return statistics (mean, standard deviation, and sample count of BTC returns), transition-count matrices to characterize regime persistence and switching behavior, and cross-model agreement to identify periods of strong consensus versus ambiguity. For the GMM, information criteria (AIC/BIC) are additionally reported at fitting time to support model inspection and comparison.

3.3 Implementation

The project is implemented in **Python 3.11**. Data manipulation and feature engineering rely on `pandas` and `numpy`. Model training uses three main libraries: `hmmlearn` for the Gaussian Hidden Markov Model, `scikit-learn` for the Gaussian Mixture Model, KMeans clustering, and standardization (`StandardScaler`), and `PyTorch` for the autoencoder neural network. Visualizations and reporting are produced with `matplotlib` and `seaborn`. Configuration is handled through YAML files (`pyyaml`), and intermediate outputs are stored as CSV files[11].

The implementation follows a modular pipeline structure: (i) raw data ingestion, (ii) feature engineering, (iii) model training/inference, and (iv) evaluation and plotting. Each stage writes its outputs to disk so that the full workflow can be executed reliably in offline environments. Raw time series are stored in `data/raw/`, engineered features are stored in `data/processed/features.csv`, model regime assignments are exported to `results/*.csv`, and figures are generated in `results/figures/`. The pipeline can be orchestrated through the main entry point (`main.py`) or by running each stage independently via scripts in `scripts/`.

Core components are organized as follows:

- `src/data/yfinance_loader.py`: data loader responsible for downloading OHLCV time series (when online) and saving them as CSV files.
- `src/features/build_features.py`: feature engineering module computing log returns, rolling volatility (21-day, annualized), rolling correlations with benchmark return series, and a rolling volume z-score (60-day window). The final feature matrix is saved to `data/processed/features.csv`.
- `src/models/hmm.py`: Gaussian HMM training (`fit_hmm`) and decoding (`decode_hmm`) using standardized features.
- `src/models/gmm.py`: GMM fitting (`fit_gmm`) and regime prediction (`predict_gmm`) using standardized features.
- `src/models/autoencoder.py`: PyTorch autoencoder training and embedding extraction, followed by KMeans clustering for regime assignment.
- `src/evaluation/metrics.py` and `src/evaluation/plots.py`: regime statistics, transition-count matrices, and visualization utilities.

Listing 1 illustrates the standardized preprocessing shared by the three models: missing values are removed, then a `StandardScaler` is fitted on the training sample and reused for inference.

```
def prepare_features(df, feature_cols=None):
    if feature_cols is None:
        feature_cols = [c for c in df.columns if c != "Date"]

    X_df = df.loc[:, feature_cols].dropna()
    idx = X_df.index

    scaler = StandardScaler()
    X = scaler.fit_transform(X_df.values)

    return X, idx, scaler, tuple(feature_cols)
```

Listing 1: Shared preprocessing: drop missing values and standardize features

4 Results

This section reports the main empirical findings obtained from the engineered feature set and the three unsupervised regime detection models (HMM, GMM, and Autoencoder+KMeans). Results are presented both qualitatively (timeline plots) and quantitatively (regime-conditional return statistics and transition patterns).

4.1 Experimental Setup

Hardware. CPU-only execution; no GPU acceleration; standard consumer hardware; offline-compatible pipeline.

Software. Python 3.11; pandas; numpy; scikit-learn; hmmlearn; PyTorch (CPU); matplotlib; seaborn; fixed dependency environment.

Hyperparameters and training details. Two regimes; full covariance matrices; maximum 500 EM iterations (HMM, GMM); autoencoder latent dimension 4; hidden dimension [8]; 100 epochs; batch size 32; learning rate 10^{-3}; fixed random seed.

4.2 Performance Evaluation

Since regime detection is unsupervised and no ground-truth regime labels are available, evaluation focuses on economic interpretability and robustness rather than predictive accuracy. We report regime-conditional BTC return statistics (mean, standard deviation, and sample count) and assess regime persistence using transition-count matrices. Additionally, for the GMM we report information criteria (AIC/BIC), which are derived from the model log-likelihood and penalize model complexity, with lower values indicating a better fit–complexity trade-off. AIC/BIC are reported only for the GMM, as these likelihood-based criteria are readily available for Gaussian mixture models and are not directly applicable to the HMM and autoencoder components in our implementation.

4.3 Visualizations

For each model, regime labels (0/1) are arbitrary and are interpreted a posteriori using regime-conditional BTC return statistics. In the following, the regime with higher return volatility (standard deviation of daily returns) is interpreted as Risk-Off, while the lower-volatility regime is interpreted as Risk-On.
In risk-off periods, market uncertainty and risk aversion typically increase, leading to larger price swings and thus higher realized volatility. This volatility-based interpretation provides a consistent economic meaning to the unsupervised regime labels across models.

Table 1: Regime-conditional BTC return statistics (mean/std/count).

Model	**Regime**	**Mean return**	**Std. return**	**Count**
HMM	0	0.002256	0.031929	1218
HMM	1	-0.002238	0.045104	768
GMM	0	0.000658	0.025078	1571
GMM	1	-0.000010	0.066358	415
Autoencoder	0	-0.006321	0.034353	1578
Autoencoder	1	0.026972	0.038078	408

Across all models, regimes exhibit clear economic differences. In particular, regimes associated with lower mean returns tend to display higher volatility and lower sample counts, consistent with stress or risk-off market conditions

4.3.1 Regime timelines

Figure 1 displays the inferred regimes over time, overlaid with BTC daily log returns. The HMM produces the smoothest regime sequence, reflecting its temporal persistence assumption. In contrast, the GMM exhibits more frequent switches because each day is clustered independently. The Autoencoder-based regimes typically lie between these two extremes, producing fewer one-day flips than the GMM while remaining more reactive than the HMM.

(a) HMM (b) GMM (c) Autoencoder+KMeans

Figure 1: Inferred regimes over time (background shading) with BTC returns overlay.

4.3.2 Regime-conditional return statistics

To quantify how regimes differ economically, we compute regime-conditional statistics of BTC returns (mean, standard deviation, and sample count). Figure 2 summarizes the conditional mean returns by regime and model. A stronger separation between regimes (indicating distinct return behavior) suggests that the model captures meaningful latent market states states with distinct economic characteristics rather than noise.

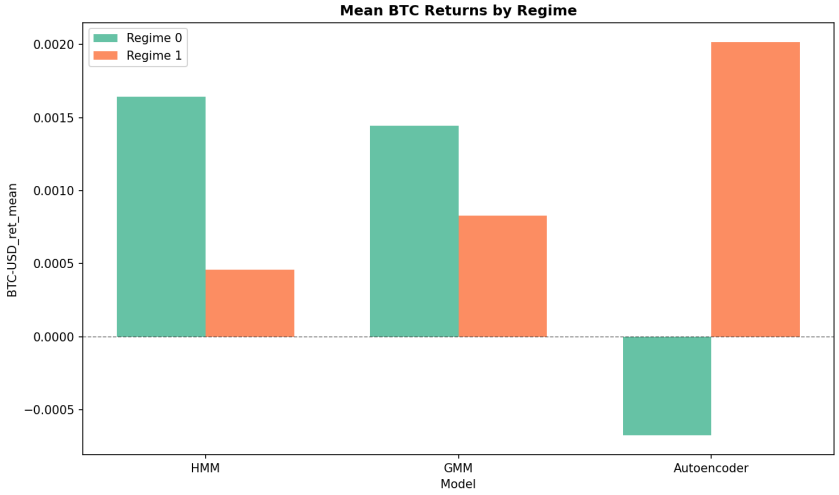
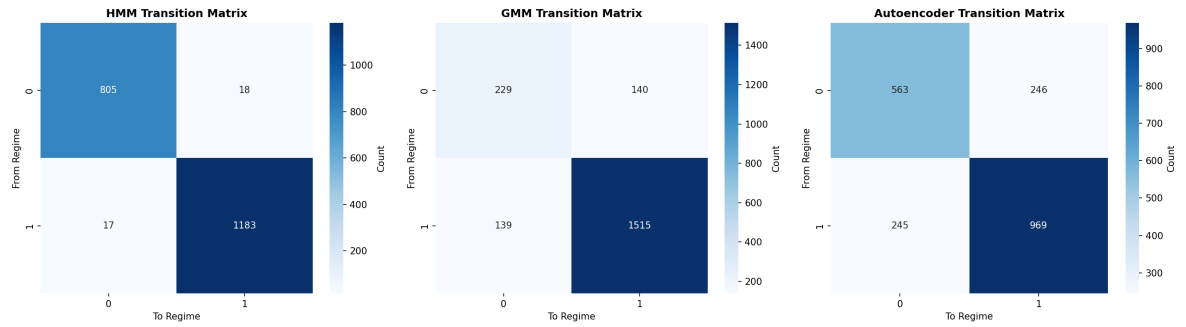


Figure 2: Mean BTC returns conditioned on inferred regimes for each model.

4.3.3 Regime persistence and transition behaviour

We further analyze regime persistence through transition-count matrices computed from each regime sequence. The HMM typically shows stronger diagonal dominance (fewer switches), whereas the GMM and Autoencoder transitions reflect more frequent changes in cluster assignment. These differences reflect the underlying modeling assumptions: explicit temporal dependence in the HMM versus independent or representation-based clustering in the GMM and autoencoder



(a) HMM. (b) GMM. (c) Autoencoder.

Figure 3: Transition-count matrices for the three models.

4.4 Clustering visualization in PCA space

Figure 4 visualizes the two inferred regimes in a 2D PCA projection of the engineered feature space. Across models, the two labels show partial separation but substantial overlap, which is expected given the noisy nature of daily crypto returns and the information loss from projecting a high-dimensional space into two components. Importantly, PCA plots are used here as a qualitative diagnostic of clustering structure rather than as a definitive measure of regime quality

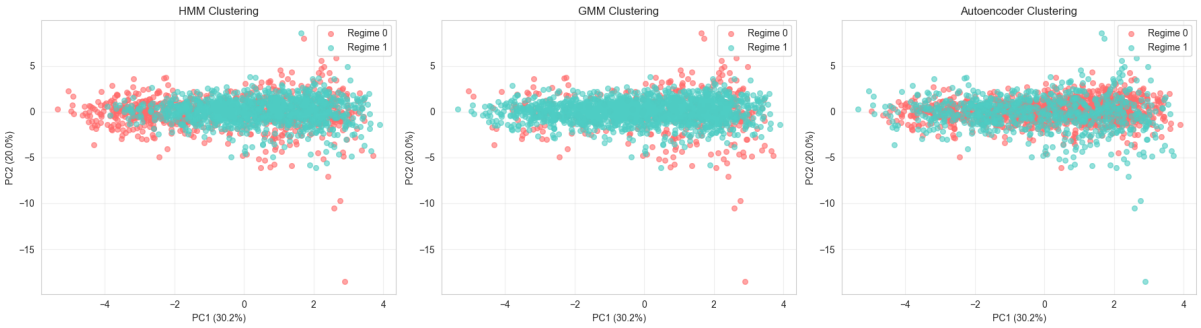


Figure 4: PCA projection (PC1–PC2) of the engineered feature space, colored by inferred regimes for each model (HMM, GMM, Autoencoder+KMeans).

5 Discussion

5.1 What worked well

Several elements worked well and produced interpretable results. First, the regime-conditional BTC return statistics provide a simple and transparent way to interpret otherwise arbitrary unsupervised labels. Across models, the inferred regimes consistently separate into two distinct market states characterized by different levels of return dispersion, supporting the idea that the methods capture meaningful market conditions rather than random partitions.

Second, the HMM is well matched to regime analysis because it explicitly models temporal dependence. This is reflected in the regime timeline, which appears more coherent, with longer contiguous segments, than the clustering-based approaches Figure 1. The transition-count matrices further support this interpretation: the HMM exhibits the strongest diagonal dominance (highest persistence), whereas the GMM and autoencoder display more frequent switching, particularly for one of the states Figure 2.

Finally, the visual diagnostics—regime timelines overlaid with returns and transition-count matrices—proved effective for validating and communicating the results. Together, these figures

highlight both economic interpretability and temporal structure, providing complementary evidence for the presence of distinct market regimes in the BTC market.

5.2 What were the challenges?

The main challenges stem from the unsupervised nature of regime detection and from the application of the Risk-On/Risk-Off framework to crypto markets. First, regime labels are inherently arbitrary (label switching), meaning that numerical state identifiers (e.g., 0/1) do not carry a fixed meaning and may permute across models or across different runs. As a result, regimes must be interpreted *a posteriori* using economic diagnostics such as return dispersion and volatility.
Second, the absence of ground-truth regime annotations makes model evaluation non-trivial. Validation therefore relies on indirect evidence—such as regime-conditional return statistics, persistence patterns, and transition behaviour—rather than standard supervised performance metrics.
Third, differences across models reflect both methodological assumptions and sensitivity to modelling choices, particularly for clustering-based approaches (GMM and autoencoder + KMeans), which tend to produce more frequent regime switches. Finally, the identified regimes depend on the chosen feature set and time horizon, which may emphasize specific market properties (e.g., volatility) while under-representing others. These challenges highlight the need to interpret the results comparatively rather than relying on a single model output.

5.3 How do the results compare to expectations?

The results are broadly consistent with initial expectations, but they remain limited in scope and should be interpreted cautiously. Given the engineered feature set especially rolling volatility measures—it is unsurprising that the inferred regimes are primarily differentiated by realized return dispersion (Table 1), which supports a Risk-On/Risk-Off style interpretation. However, with a relatively limited set of assets and indicators, the detected structure may reflect volatility clustering more than a comprehensive market regime definition.
Regarding model behaviour, the HMM produces more persistent segments in the regime timeline (Figure 5) and stronger diagonal dominance in the transition-count matrix (Figure 3), which is consistent with its Markov assumption. In contrast, the GMM and autoencoder switch more frequently (Figure 6, Figure 8), as expected for clustering-based methods, but this also suggests higher sensitivity to short-lived fluctuations and modelling choices (e.g., scaling, window lengths, and hyperparameters).

5.4 Limitations of the approach

Finally, the imperfect agreement across models indicates that the inferred regimes are not unique: each method emphasizes different aspects of the same data, and none can be treated as a definitive "ground truth" detector. Taken together, these results provide suggestive evidence of regime-like behaviour in BTC returns, but the conclusions remain model-dependent and constrained by the asset coverage and feature design of this study.

6 Conclusion and Future Work

6.1 Summary

This project investigated whether unsupervised learning methods can recover meaningful market regimes in the BTC market using a small set of engineered risk and return features. We implemented three complementary approaches—Gaussian HMM, Gaussian mixture clustering,

and an autoencoder with KMeans—and evaluated them using regime-conditional return statistics, regime timelines, and transition-count matrices.
The models identify two regimes that differ markedly in realized return dispersion, supporting an interpretable separation between lower-volatility and higher-volatility market conditions. In addition, the HMM tends to produce more persistent regime segments and a more stable transition structure, whereas clustering-based approaches exhibit more frequent switching, highlighting the importance of explicitly modelling temporal dependence for regime detection.
At the same time, the findings should be interpreted as exploratory. In the absence of ground-truth regime labels, evaluation remains indirect, and the detected regimes are strongly influenced by the chosen feature set—particularly volatility-related features. Moreover, we did not conduct a rigorous out-of-sample or walk-forward validation, nor a trading or risk-management backtest, so the practical usefulness of the regimes cannot be concluded from this study alone.

6.2 Future Directions

Methodologically, the approach could be improved by adopting a walk-forward (out-of-sample) evaluation framework, performing systematic model selection over the number of regimes, and reporting regime stability across random seeds and hyperparameters. Additional experiments could include feature ablation studies, testing richer crypto-specific inputs (e.g., funding rates, liquidity measures, on-chain metrics), and comparing against alternative regime models (e.g., heavy-tailed emissions or regime-switching volatility models). From an application perspective, the inferred regimes could be integrated into real-world workflows such as dynamic risk limits, volatility targeting, hedging overlays, or exposure scaling rules, and evaluated through transaction-cost-aware backtests.

References

1. Hamilton, J. D. (1989). *A New Approach to the Economic Analysis of Nonstationary Time Series and the Business Cycle.* Econometrica, 57(2), 357–384.

2. Rabiner, L. R. (1989). *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, 77(2), 257–286.

3. McLachlan, G., & Peel, D. (2000). *Finite Mixture Models.* Wiley.

4. Engle, R. F. (1982). *Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation.* Econometrica, 50(4), 987–1007.

5. Bollerslev, T. (1986). *Generalized Autoregressive Conditional Heteroskedasticity.* Journal of Econometrics, 31(3), 307–327.

6. Hinton, G. E., & Salakhutdinov, R. R. (2006). *Reducing the Dimensionality of Data with Neural Networks.* Science, 313(5786), 504–507.

7. Katsiampa, P. (2017). *Volatility estimation for Bitcoin: A comparison of GARCH models.* Economics Letters, 158, 3–6.

8. Corbet, S., Meegan, A., Larkin, C., Lucey, B., & Yarovaya, L. (2018). *Exploring the dynamic relationships between cryptocurrencies and other financial assets.* Economics Letters, 165, 28–34.

9. Aroussi, R. (2019). *yfinance: Yahoo! Finance market data downloader.* GitHub repository. Available at: `https://github.com/ranaroussi/yfinance`.

10. Yahoo Finance. (n.d.). *Historical market data (BTC-USD, ETH-USD, ĜSPC, V̂IX).* Available at: `https://finance.yahoo.com/`.

11. Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825–2830.

AI Tool Usage Disclosure

ChatGPT

ChatGPT was used as a supportive tool to clarify technical concepts related to unsupervised learning (e.g., Hidden Markov Models, Gaussian Mixture Models, and autoencoders), assist with debugging, and help interpret error messages during development.

Claude Sonnet 4.5

Claude Sonnet 4.5 was used to support documentation writing, improve clarity and structure of explanations, and suggest refinements to project organization and presentation.
All modelling decisions, implementations, experiments, and interpretations were performed and validated by the author. AI tools were used as learning and productivity aids and did not replace independent reasoning or original work.

A Additional Figures

Include supplementary figures or tables that support but aren't essential to the main narrative.

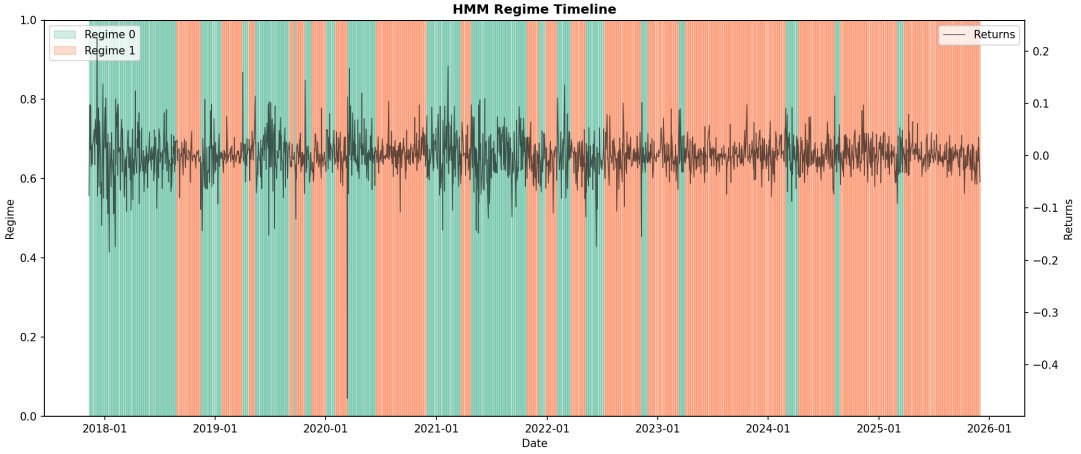


Figure 5: HMM regime timeline (additional result).

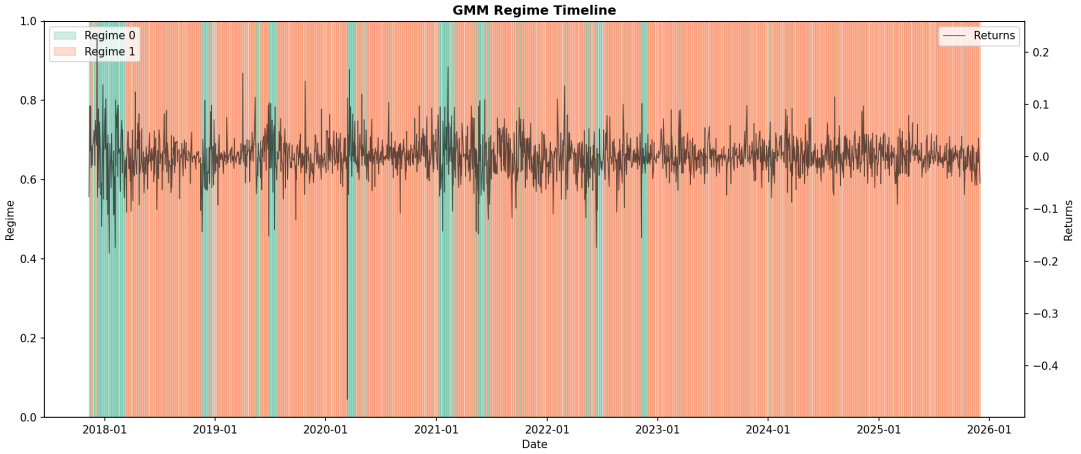


Figure 6: GMM regime timeline (additional result).

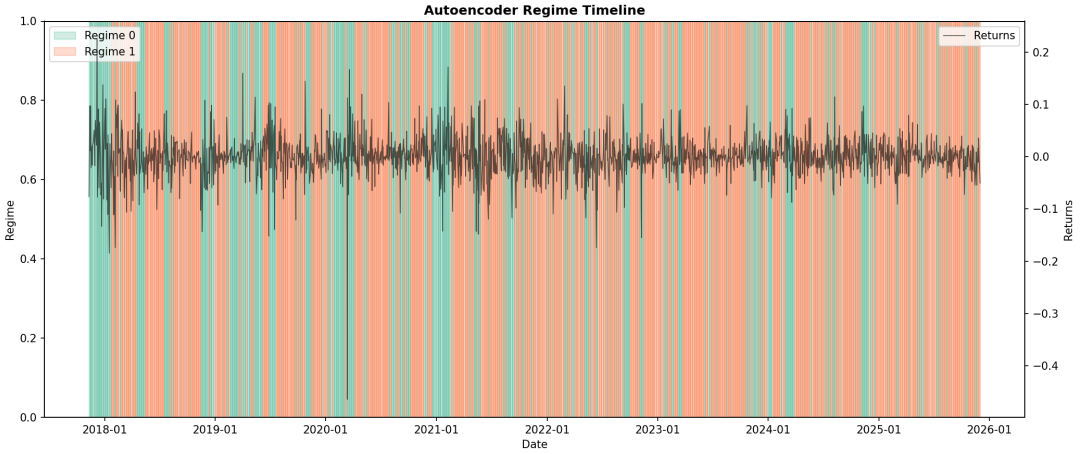


Figure 7: Autoencoder regime timeline (additional result).

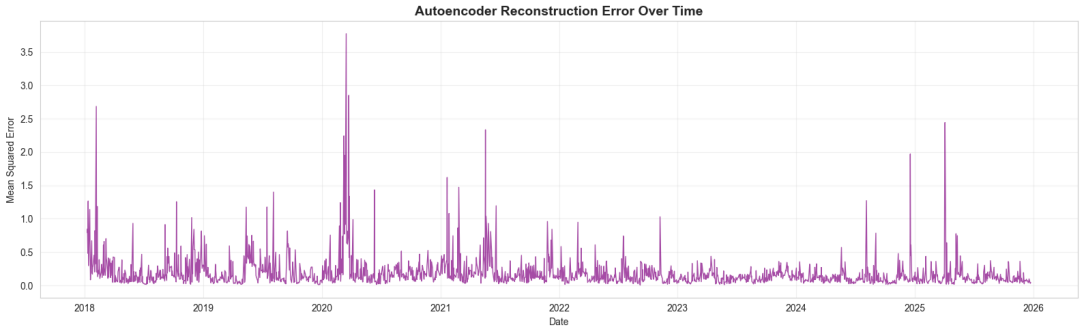


Figure 8: reconstruction error autoencoder (additional result).

B Code Repository

B.1 Repository Information

The complete source code for this project is available on GitHub:

`https://github.com/elvisbinks/risk-on-risk-off-crypto`

B.2 Repository Structure

The project follows a modular architecture with clear separation of concerns:

```
risk-on-risk-off-crypto/
 main.py                    # Main pipeline entry point
 src/
    data/
       yfinance_loader.py # Data download from Yahoo Finance
    features/
       build_features.py  # Feature engineering pipeline
    models/
       hmm.py             # Hidden Markov Model
       gmm.py             # Gaussian Mixture Model
       autoencoder.py     # Autoencoder + KMeans
    evaluation/
        metrics.py         # Evaluation metrics
        plots.py           # Visualization functions
 scripts/                   # Individual pipeline steps
    fetch_data.py
    build_features.py
    run_hmm.py
    run_gmm.py
    run_autoencoder.py
    evaluate_models.py
 configs/                   # YAML configuration files
    default.yaml           # Data configuration
    hmm.yaml               # HMM hyperparameters
    gmm.yaml               # GMM hyperparameters
    autoencoder.yaml       # Autoencoder hyperparameters
 data/                      # Data directory (gitignored)
    raw/                   # Downloaded OHLCV data
```

```
    processed/              # Engineered features
 results/                    # Model outputs (gitignored)
    figures/                # Plots and visualizations
    hmm_regimes.csv
    gmm_regimes.csv
    autoencoder_regimes.csv
 notebooks/                  # Jupyter notebooks
 tests/                      # Unit and integration tests
 environment.yml             # Conda dependencies
 requirements.txt            # pip dependencies
 README.md                   # Documentation
```

B.3 Installation Instructions

B.3.1 Prerequisites

- Python 3.11 or higher
- pip (Python package manager) or Conda/Miniconda
- Internet connection (for data download)

B.3.2 Setup

Option 1: Using Conda (Recommended)

```bash
# Clone repository
git clone https://github.com/yourusername/risk-on-risk-off-crypto
cd risk-on-risk-off-crypto

# Create and activate environment
conda env create -f environment.yml
conda activate risk-on-risk-off-crypto
# IMPORTANT: PyTorch must be installed manually via pip (see step below)
# Install PyTorch via pip
pip install torch==2.2.2 --index-url https://download.pytorch.org/whl/cpu
```

Option 2: Using venv + pip

```bash
# Clone repository
git clone https://github.com/yourusername/risk-on-risk-off-crypto
cd risk-on-risk-off-crypto

# Create virtual environment
python3 -m venv .venv
source .venv/bin/activate  # On Windows: .venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

B.4 How to Reproduce Results

B.4.1 Full Pipeline Execution

To reproduce all results presented in this report, run the main pipeline:

```bash
python main.py
```

This single command executes the complete workflow:

1. **Data Download**: Fetches historical OHLCV data for BTC-USD, ETH-USD, ^GSPC, and ^VIX from Yahoo Finance (2016-01-01 to present).
2. **Feature Engineering**: Computes log returns, rolling volatility (21-day window), rolling correlations, and volume z-scores.
3. **Model Training**: Trains three models (HMM, GMM, Autoencoder+KMeans) using configurations in `configs/`.
4. **Evaluation**: Computes regime statistics, transition matrices, and model agreement metrics.
5. **Visualization**: Generates all figures saved to `results/figures/`.

B.4.2 Step-by-Step Execution

For more control, individual pipeline steps can be run separately:

```
# 1. Download data
python -m scripts.fetch_data --config configs/default.yaml

# 2. Engineer features
python -m scripts.build_features --config configs/default.yaml

# 3. Train models
python -m scripts.run_hmm --config configs/hmm.yaml
python -m scripts.run_gmm --config configs/gmm.yaml
python -m scripts.run_autoencoder --config configs/autoencoder.yaml

# 4. Evaluate and visualize
python -m scripts.evaluate_models
```

B.4.3 Reproducibility Notes

Reproducibility is supported through fixed random seeds in all models (`random_state=42` in configuration files). However, minor numerical differences may still occur across computing environments due to:

- Library versions (scikit-learn, hmmlearn, PyTorch)
- Numerical backends (MKL, OpenBLAS)
- Data source updates (Yahoo Finance may revise historical data)

For strict reproducibility, use the exact software environment specified in `requirements.txt` or `environment.yml`, and consider freezing a dataset snapshot with a documented download date.