Unsupervised Clustering on Market Data

Mohd Nasar Siddiqui (220661)

April 20, 2025

1 Introduction

This project explores unsupervised machine learning techniques on high-frequency financial market data. Specifically, we aim to uncover latent structures and behavioral patterns by applying clustering algorithms to trading data. The data is obtained from Binance and includes trade and order book snapshots. By preprocessing and combining raw data into structured formats, we enable further analysis and feature engineering necessary for clustering.

2 Data Extraction

The raw market data was provided in multiple `.txt` files, separated by type and symbol. The following preprocessing steps were carried out to consolidate the data:

- All `aggTrade .txt` files were combined into a single CSV file named `aggTrade.csv`. These files contain aggregated trade data for different market pairs.
- All four `depth20_1000ms_symbol.txt` files were merged into a single CSV file named `adepth20_1000ms.csv`. These files provide snapshots of the top 20 levels of the order book at 1-second intervals.

This consolidation step was essential for synchronizing timestamps and aligning the features before performing clustering and further analysis.

3 Feature Engineering

- **spread** : Measures the bid-ask spread at Level 1, indicating market liquidity and transaction cost.
- **microprice** : Reflects the fair value by weighting best bid and ask prices by their respective quantities.
- **imbalance_lvl1** : Captures the Level 1 order book imbalance, a proxy for short-term price pressure.

- **cum_bid_qty** : Total bid quantity across top 20 levels, indicating aggregate buying interest.
- **cum_ask_qty** : Total ask quantity across top 20 levels, indicating aggregate selling interest.
- **mid_price** : Average of best bid and ask prices; used as a reference price for return calculations.
- **log_return** : Captures the logarithmic return of mid-price between successive time intervals.
- **vol_10s** : Measures short-term volatility using standard deviation of log returns over 10 seconds.
- **vol_30s** : Measures medium-term volatility using standard deviation of log returns over 30 seconds.
- **cum_vol_10s** : Cumulative trading volume over a 10-second window, indicates recent trading activity.
- **cum_vol_30s** : Cumulative trading volume over a 30-second window, for broader activity view.
- **vwap_shift** : First-order difference of the Volume Weighted Average Price, shows directional movement.
- **bid_slope_top5** : Slope of bid price curve over top 5 levels, quantifying depth steepness on the bid side.
- **ask_slope_top5** : Slope of ask price curve over top 5 levels, quantifying depth steepness on the ask side.
- **vol_ratio_10s_30s** : Ratio of 10s to 30s volatility, identifies rapid shifts in price variability.
- **price_jump** : Normalized log return by volatility, helps detect abnormal price movements.
- **hour_of_day** : Encodes the hour extracted from the timestamp, capturing intra-day market behavior.
- **is_ny_open** : Binary feature indicating whether the New York market is open at that timestamp.

4 Dimensionality Reduction and Clustering

4.1 Autoencoder for Feature Extraction

To reduce the dimensionality of the engineered features and capture the underlying structure of the data, an autoencoder neural network was employed. The autoencoder consists of an encoder-decoder architecture trained to minimize the reconstruction error. The encoder compresses the input features into a lower-dimensional latent space, which is then used as input for clustering.

- Input: Standardized features from the previous section.
- Architecture: A symmetrical neural network with dense layers, where the bottleneck layer represents the latent representation.
- Loss Function: Mean Squared Error (MSE) between the input and the reconstructed output.
- Optimizer: Adam optimizer for efficient convergence.
- Output: Latent space representation of the input data (encoded features).

This compressed representation captures non-linear relationships and denoises the features, improving clustering quality.

4.2 HDBSCAN Clustering

To explore density-based clustering, the HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm was also applied. Unlike GMM, HDBSCAN does not require the number of clusters to be specified in advance and is robust to noise.

- Input: Same encoded features from the autoencoder.
- Model: HDBSCAN with a minimum cluster size and minimum samples parameters tuned for market data.
- Output: Discrete cluster labels, with noise points labeled as `-1`.

5 Result

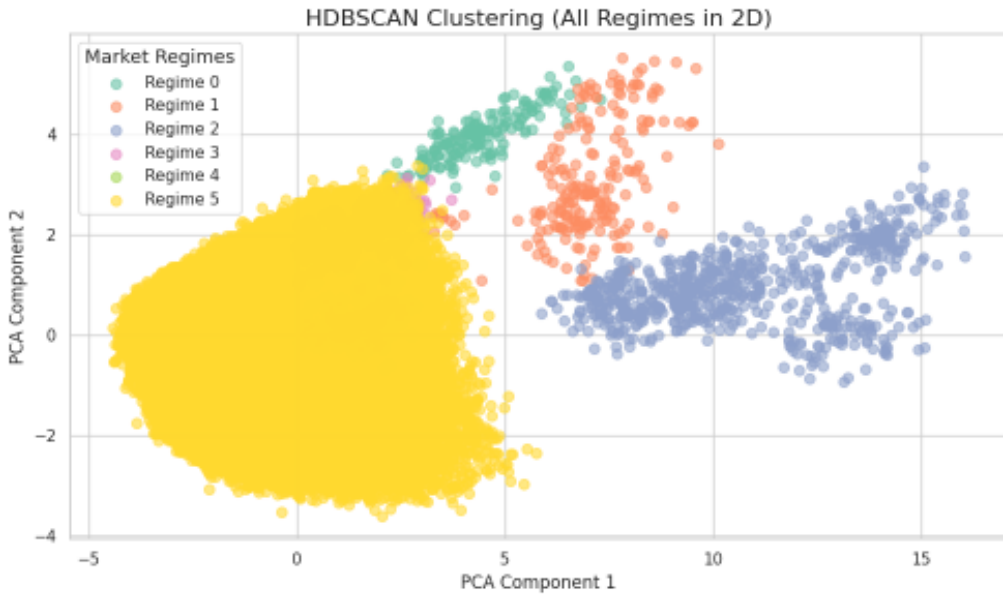

Figure 1: Visualization of clustering results on autoencoder-encoded market data using GMM and HDBSCAN.