Market Regime Detection System: Technical Report

1. Introduction

This report documents the implementation of a market regime detection system that uses unsupervised learning techniques to identify distinct market states (regimes) from high-frequency financial data. The system analyzes patterns in order book and trade data to detect changes in market conditions, which can inform trading strategies and risk management.

2. Feature Engineering

The system extracts a rich set of features from raw market data, capturing various market dynamics:

2.1 Custom Features

- **Order Book Imbalance:** Measures the supply/demand asymmetry using the ratio of bid to ask quantities.

```
features['ob_imbalance'] = (raw_data['BidQtyL1'] - raw_data['AskQtyL
1']) / (raw_data['BidQtyL1'] + raw_data['AskQtyL1'] + 1e-10)
```

- **Microprice:** Price weighted by order book quantities, providing a more accurate fair value estimate than mid-price alone.

```
features['microprice'] = (bid_price * ask_qty + ask_price * bid_qty) / (bi
d_qty + ask_qty + 1e-10)
```

- **Relative Spread:** Normalizes spread by price, allowing for comparison across different price levels.

```
features['rel_spread'] = features['spread'] / (features['price'] + 1e-10)
```

- **Depth Imbalance:** Extends beyond level 1 to capture overall order book asymmetry.

```
features['depth_imbalance'] = (features['cum_bid_qty'] - features['cum_ask_qty']) / (features['cum_bid_qty'] + features['cum_ask_qty'] + 1e-10)
```

- **Parkinson Volatility:** Alternative volatility estimator using high-low range, capturing intraperiod price movements.

```
features['parkinson_vol'] = np.log(price_max / price_min) / (4 * np.log(2))
```

- **Directional Volatility:** Separates upside and downside volatility, revealing market sentiment.

```
features['upside_vol'] = returns.rolling(10).apply(lambda x: np.sqrt(np.mean(np.square(x[x > 0])))).fillna(0)
features['downside_vol'] = returns.rolling(10).apply(lambda x: np.sqrt(np.mean(np.square(x[x < 0])))).fillna(0)
```

- **Trend Strength Indicator:** ADX-like measure that quantifies the strength of price trends.

```
features[f'trend_strength_{window}'] = pd.Series(plus_dm).rolling(window).sum().fillna(0) - pd.Series(minus_dm).rolling(window).sum().fillna(0)
```

2.2 Multi-timeframe Analysis

Features are calculated across multiple window sizes (e.g., 5, 15, 30 periods) to capture short, medium, and long-term market dynamics. This multi-timeframe approach enables detection of regime changes at different time scales.

3. Clustering Methodology

3.1 Preprocessing

- **Normalization:** Features are standardized to ensure fair contribution to clustering.
- **Dimensionality Reduction:** Optional PCA to reduce feature dimensionality while preserving variance.

3.2 Clustering Algorithms

The system employs multiple clustering algorithms:

- **K-Means:** Distance-based clustering that partitions data into k clusters.
- **Gaussian Mixture Models (GMM):** Probabilistic model assuming data is generated from a mixture of Gaussian distributions.
- **HDBSCAN:** Density-based clustering that can identify clusters of varying shapes and densities.

3.3 Clustering Metrics

Several metrics evaluate clustering quality:

- **Silhouette Score:** Measures cluster cohesion and separation (higher is better).
- **Davies-Bouldin Index:** Evaluates average similarity between clusters (lower is better).
- **Calinski-Harabasz Index:** Ratio of between-cluster to within-cluster dispersion (higher is better).

3.4 Ensemble Approach

The system can combine results from multiple clustering models to create more robust regime classifications, though the current implementation uses a simplified approach:

```
def _create_ensemble_labels(self):
    # This is a simplified approach to ensemble clustering
    if 'kmeans' in self.labels:
        return self.labels['kmeans']
    else:
        return list(self.labels.values())[0]
```

A more sophisticated ensemble could use cluster alignment techniques or consensus clustering.

4. Regime Analysis

4.1 Regime Characterization

The system analyzes each detected regime to determine its characteristics:

```
def analyze_regimes(self):
    # For each regime...
    for label in unique_labels:
        regime_data = labeled_data[labeled_data['regime'] == label]

        # Calculate statistics
        stats = {}

        # Volatility characteristics
        volatility_cols = [col for col in regime_data.columns if 'volatility' in col]
        if volatility_cols:
            mean_volatility = regime_data[volatility_cols].mean().mean()
            stats['volatility'] = mean_volatility
            stats['volatility_type'] = 'High' if mean_volatility > labeled_data[volati
lity_cols].mean().mean() else 'Low'

        # Additional statistics...
```

4.2 Regime Transition Analysis

The system calculates transition probabilities between regimes:

```
def analyze_regime_transitions(self):
    # Create transition matrix
    transitions = {}
    prev_regime = regime_series.iloc[0]

    for current_regime in regime_series.iloc[1:]:
        transition = (prev_regime, current_regime)
        transitions[transition] = transitions.get(transition, 0) + 1
```

```
        prev_regime = current_regime

    # Convert to probabilities
    # ...
```

This analysis reveals how likely the market is to switch from one regime to another, providing valuable information for strategy adaptation.

5. Visualization Techniques

The system offers multiple visualization methods:

- **2D Clustering Visualization:** Using TSNE or UMAP to project high-dimensional feature space into 2D.
- **Regime Evolution:** Temporal visualization showing how regimes change over time.
- **Cluster Evaluation:** Comparison of clustering models using performance metrics.

6. Results and Conclusions

The market regime detection system successfully identifies distinct market states with clear characteristics. By analyzing these regimes and their transitions, traders can:

1. **Adapt strategies** to current market conditions
2. **Anticipate regime changes** based on transition probabilities
3. **Optimize risk management** by adjusting exposure based on regime volatility
4. **Improve execution** by understanding liquidity conditions in different regimes

Future improvements could include:

- More sophisticated ensemble clustering methods
- Incorporation of external factors (news, macro indicators)
- Real-time regime detection and forecasting
- Strategy optimization based on regime detection

This report provides a technical overview of the market regime detection system. For implementation details and usage instructions, please refer to the project's README and code documentation.