Market Regime Detection Using Gaussian Hidden Markov Models

Tumelo Tshwanelo Mkwambe
School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
2446873@students.wits.ac.za

***Abstract*—A market regime refers to a period during which financial markets exhibit consistent and persistent behaviors or patterns. The most common regimes are directional (trend-based), such as bull markets, characterized by investor optimism and rising index levels, and bear markets, marked by investor pessimism and declining prices. Another key feature that defines market regimes is volatility—a statistical measure of the degree of variation in the price of a financial instrument over time, often used as a proxy for market uncertainty or risk. In this study, we apply Gaussian Hidden Markov Models (HMMs) to identify and characterize distinct market regimes in the South African Top 40 Index.**

I. Introduction

The South African Top 40 (JTOPI) index is a capitalization-weighted index that represents the performance of the top 40 companies listed in the Johannesburg Stock Exchange (JSE). The companies are listed by their free-float market capitalization, which reflects shares that are publicly available for trading. The composition of the South African Top 40 is reviewed periodically to ensure it accurately reflects the largest companies on the JSE. The index is considered a market proxy for the South African stock market.

Hidden Markov Models (HMMs) are a class of dynamic Bayesian networks designed to model sequential data. An HMM is composed of two sets of variables or states, hidden/latent and observed. The hidden states represent the underlying conditions of a system that evolves over time. The observed variables are the data we directly observe; their distribution depends only on the current hidden state.

Hidden Markov Models (HMMs) view the market as switching between hidden states, each exhibiting unique patterns of returns, volatility, and other characteristics. By analyzing market data, HMMs try to uncover these hidden states and understand how the market transitions between them. This makes HMMs very competent in detecting market regimes compared to other statistical methods.

II. Problem Formulation

Prior to constructing and evaluating investment strategies and assessing risk, it is required for quantitative analysts to be aware of the current market conditions. A substantial part of this consists of knowing investors' attitudes towards a certain market, and the current underlying sentiment can be analyzed from the trading price. Market sentiment is shaped by numerous factors, including political events, company performance, and environmental occurrences, as exemplified by the Covid-19 outbreak triggering a significant bear trend with high volatility across many markets. However, market sentiment itself represents a collective quantification of investors' expectations regarding the impact of these factors. Therefore, the problem is to develop a robust and timely methodology to infer prevailing market sentiment from historical and real-time trading data.

III. Data And Analysis

A. Data

The historical time series data for the South African top 40 index was downloaded from Investing.com in a csv format. Investing.com is a financial markets platform and news website. The downloaded csv file contains daily close price, open price and volume data of the index from 2015-01-01 to 2025-01-01. The entries in the csv file are in string format and contained 13 null entries for the Volume column. Post float-casting and data cleaning, the data was well prepared for analysis.

B. Analysis

Raw prices, while representing the direct value of an asset over time, suffer from several limitations that make them less suitable for many financial analyses. The traditional approach is to use log returns instead, which are mathematically convenient. Log returns are additive over time and exhibit stationarity (a property that raw prices lack) and approximate the Gaussian probability distribution (which complements the Gaussian emission assumption for Gaussian hidden Markov models).

The Stationarity / Time Invariance property for hidden Markov models states that the probability of transitioning between states remains constant.

$$P(X(t+1) = \xi | X(t) = \xi) = P(X = \xi | X = \xi)$$

The stationarity assumption in HMMs assumes the process is stationary. This justifies the use of log returns in constructing the model. Do note that time series data is stationary if its statistical properties (mean, variance, etc.) remain constant.

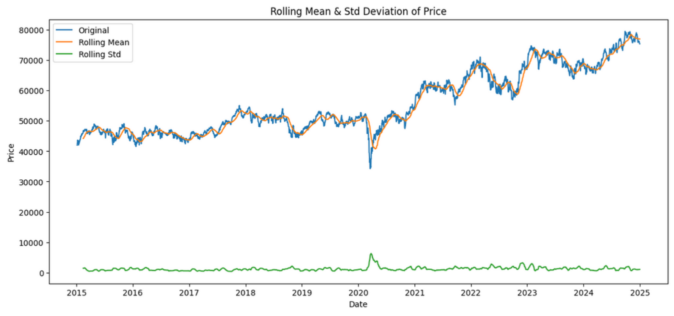


Fig. 1. Rolling mean and standard deviation for closing price.

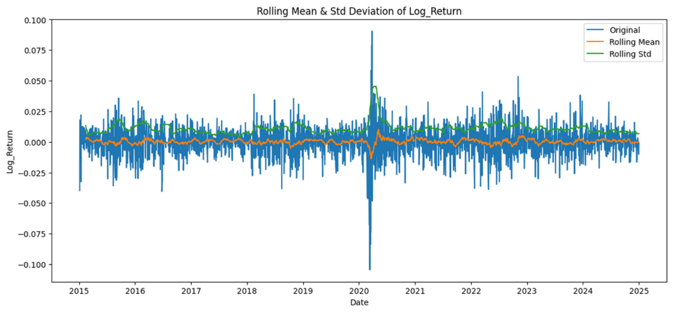


Fig. 2. Rolling mean and standard deviation for log returns.

Fig. 1 illustrates the non-stationarity of the index's closing price. In contrast, Fig. 2 demonstrates that the log returns exhibit stationarity, with a notable spike in the middle due to the COVID-19 outbreak. Furthermore, the suitability of log returns for training Gaussian hidden Markov models is supported by their approximate normal distribution, as shown in Fig. 3. For Gaussian hidden Markov models, the emission probability associated with each hidden state is modeled as a normal distribution.

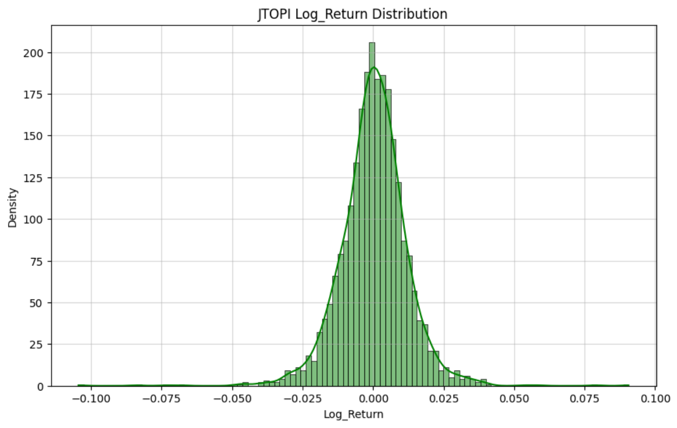


Fig. 3. Log Return Distribution.

The summary statistics for the log returns of the index are displayed in table I.

TABLE I
DESCRIPTIVE STATISTICS OF LOG RETURNS

Statistic	Value
Minimum	-0.10450424958302106
Maximum	0.09056979463126316
Mean	0.00021828897779282446
Variance	0.0001415182722127598
Skewness	-0.30760510824628073
Kurtosis	7.134709836054412

IV. HIDDEN MARKOV MODELS

A Markov process is a stochastic process whereby the next state only depends on the current state and is independent of all other previous states.
Let $X_0, X_1, ..., X_{t-1}, X_t$ be a sequence of random variables or states of a stochastic process, then the Markov assumption states that:

$$P(X_t = x|X_0, ..., X_{t-1}) = P(X_t = x|X_{t-1}). \quad (1)$$

A hidden Markov model is a Markov process specified by the following components:

- **States:** $Q = \{q_0, q_1, ..., q_{N-1}\}$
- **Transition Probability Matrix:** $A \in \mathbb{R}^{N\times N}$, whereby $a_{i,j}$ is the probability of transitioning from state i into state j.
- **Emission Probability Matrix**: $B \in \mathbb{R}^{N\times M}$ for an observation alphabet $O \in \mathbb{R}^M$, whereby $b_{ij} = b_i(O_t = O_j) = b_i(O_j)$ is the probability of observation symbol O_j at time t given state q_j at time t.
- **Initial Probabilities:** $\pi \in \mathbb{R}^N$ where π_i is the probability of being in state i at time t_0,

where N is the number of states.

Hidden Markov models can be used to address three core problems. In this section, we provide a brief overview of these problems, followed by efficient algorithms for solving them.

A. *The Likelihood Problem and the Forward Algorithm*

The likelihood problem is stated as follows: Given a hidden Markov model $\lambda = (A, B, \pi)$ and a sequence of observations $O = \{O_0, O_1, \ldots, O_{T-1}\}$, what is the probability of the observation sequence given the model, $P(O|\lambda)$?

To compute $P(O|\lambda)$, we use the *forward algorithm*, which recursively calculates the probability of observing the sequence up to time t and being in state q_i at time t. Let

$$\alpha_t(i) = P(O_0, O_1, \ldots, O_t, q_t = q_i \mid \lambda),$$

then the algorithm proceeds as follows:

$$\alpha_0(i) = \pi_i b_i(O_0), \qquad \text{for } i = 0, 1, \ldots, N-1, \tag{2}$$

$$\alpha_t(i) = b_i(O_t) \sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji}, \quad \text{for } t = 1, \ldots, T-1, \tag{3}$$

$$P(O|\lambda) = \sum_{i=0}^{N-1} \alpha_{T-1}(i). \tag{4}$$

B. Posterior State Estimation and the Backward Algorithm

Given a model $\lambda = (A, B, \pi)$ and an observation sequence $O = \{O_0, O_1, \ldots, O_{T-1}\}$, we wish to compute the posterior probability of being in state q_i at time t, given the full observation sequence:

$$P(q_t = q_i \mid O, \lambda).$$

This problem is commonly referred to as *soft-decoding*, in contrast to *hard-decoding*, which aims to find the most likely sequence of states. While hard-decoding is solved via the Viterbi algorithm, soft-decoding makes use of the *backward algorithm.* Define:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_{T-1} \mid q_t = q_i, \lambda),$$

then the backward algorithm computes:

$$\beta_{T-1}(i) = 1, \tag{5}$$

$$\beta_t(i) = \sum_{j=0}^{N-1} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad \text{for } t = T-2, \ldots, 0. \tag{6}$$

The posterior probability of being in state q_i at time t is:

$$\gamma_t(i) = P(q_t = q_i \mid O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}. \tag{7}$$

The most probable state at time t is the one maximizing $\gamma_t(i)$ over all i.

C. Parameter Estimation and the Baum-Welch Algorithm

The Baum-Welch algorithm is a variant of the Expectation-Maximization (EM) algorithm that estimates the parameters of a hidden Markov model from data. It proceeds in two main steps:

E-step: Compute the expected values of hidden variables using the current parameter estimates: - Forward probabilities $\alpha_t(i)$ - Backward probabilities $\beta_t(i)$ - Posterior state probabilities $\gamma_t(i)$ - Posterior transition probabilities $\xi_t(i,j)$, defined as:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \alpha_t(k) a_{kl} b_l(O_{t+1}) \beta_{t+1}(l)}. \tag{8}$$

M-step: Update the model parameters using the computed expectations:

- **Initial state distribution:**

$$\pi_i = \gamma_0(i)$$

- **Transition probabilities:**

$$A_{ij} = \frac{\sum_{t=0}^{T-2} \xi_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$$

- **Emission probabilities (discrete case):**

$$b_i(k) = \frac{\sum_{t=0}^{T-1} \mathbb{I}[O_t = O_k] \gamma_t(i)}{\sum_{t=0}^{T-1} \gamma_t(i)},$$

  where $\mathbb{I}[\cdot]$ is the indicator function.

D. Gaussian Emission Probabilities

In a Gaussian hidden Markov model, emissions are continuous and modeled by Gaussian distributions. Each hidden state q_i emits observations $O_t \in \mathbb{R}$ according to:

$$P(O_t = o \mid q_t = q_i) = \mathcal{N}(o; \mu_i, \sigma_i^2).$$

The parameters of these distributions are updated during the M-step as follows:

$$\mu_i = \frac{\sum_{t=0}^{T-1} \gamma_t(i) O_t}{\sum_{t=0}^{T-1} \gamma_t(i)}, \tag{9}$$

$$\sigma_i^2 = \frac{\sum_{t=0}^{T-1} \gamma_t(i)(O_t - \mu_i)^2}{\sum_{t=0}^{T-1} \gamma_t(i)}. \tag{10}$$

E. Gaussian Emission Probabilities

When using a Gaussian hidden Markov model, the emission probabilities are not discrete, but instead modeled using Gaussian distributions. Each hidden state emits an observation $O_j \in \mathbb{R}$ in accordance with its distribution: $P(O_t = O_j | q_t = q_i) = \mathcal{N}(O_j; \mu, \sigma^2)$. Consequently, the parameters of the distribution in each state are updated during the M-step as:

$$\mu_i = \frac{\sum_{t=0}^{T-1} \gamma_t(i) O_t}{\sum_{t=0}^{T-1} \gamma_t(i)}, \tag{11}$$

$$\sigma_i^2 = \frac{\sum_{t=0}^{T-1} \gamma_t(i)(O_t - \mu_i)^2}{\sum_{t=0}^{T-1} \gamma_t(i)}. \tag{12}$$

F. Hard-decoding And The Viterbi Algorithm

Hard-decoding refers to the process of finding the single most probable sequence of hidden states given a sequence of observations. Let $\delta_t(i)$ denote the highest probability of any state sequence that ends in state q_i at time t, given the observation sequence up to time t. The Viterbi algorithm computes $\delta_t(i)$ recursively for each state i at each time step t, using the previous values $\delta_t(j)$. At each step, it selects the transition from state q_j that maximizes the probability of reaching q_i, and it stores a pointer $\psi_t(i)$ to the state q_j that yields this maximum. This allows the most probable state sequence to be recovered through backtracking after the final time step:

Initialization:

$$\delta_0(i) = \pi_i b_i(O_0) \tag{13}$$

$$\psi_0(i) = 0 \tag{14}$$

Iteration:

$$\delta_t(i) = b_i(O_t) \max_k [\delta_{t-1}(k) a_{ki}] \tag{15}$$

$$\psi_t(i) = \max_k [\delta_{t-1}(k) a_{ki}] \tag{16}$$

Termination:

$$P^* = \max_i \delta_{T-1}(i) \tag{17}$$

$$Q^* = \arg\max_i \delta_t(i) \tag{18}$$

Backtracking:

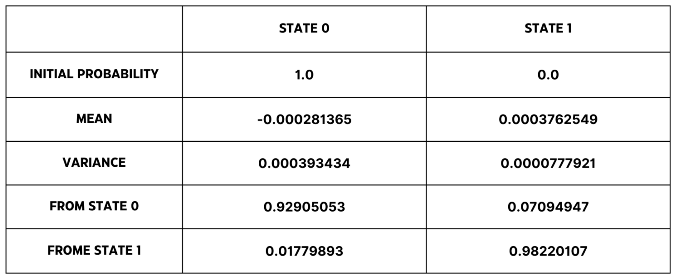
$$Q_t^* = \psi_{t+1}(Q_{t+1}^*), \text{ for } t = T-2, T-3, ..., 0 \tag{19}$$

V. Model Implementation And Results

We use the Baum-Welch algorithm to train two different models on the same South African top 40 index dataset. The first model has two distinct states and the other three. The initial parameters for both models are randomized prior to training. Each model is allowed to execute the Baum-Welch algorithm for a maximum of 100 iterations given that the change in log-likelihood is no greater than a threshold of 0.00001 for convergence. We make use of log probabilities and appropriate scaling to avoid underflow during the Viterbi, forward and backward algorithms.

A. Two-State Model Specifications And Results

The estimated parameters post-training for the two-state model are presented in Fig. 4.



	STATE 0	STATE 1
INITIAL PROBABILITY	1.0	0.0
MEAN	-0.000281365	0.0003762549
VARIANCE	0.000393434	0.0000777921
FROM STATE 0	0.92905053	0.07094947
FROME STATE 1	0.01779893	0.98220107

Fig. 4. Parameters for two-state model.

State 0 (blue) characterized by a negative mean and high variance, this state corresponds to a bear market with high volatility. State 1 (orange) shows a positive mean near the average log return and low variance, representing stable or sideways-moving markets with a slight bullish trend. This regime reflects periods of optimism or consolidation with low risk.

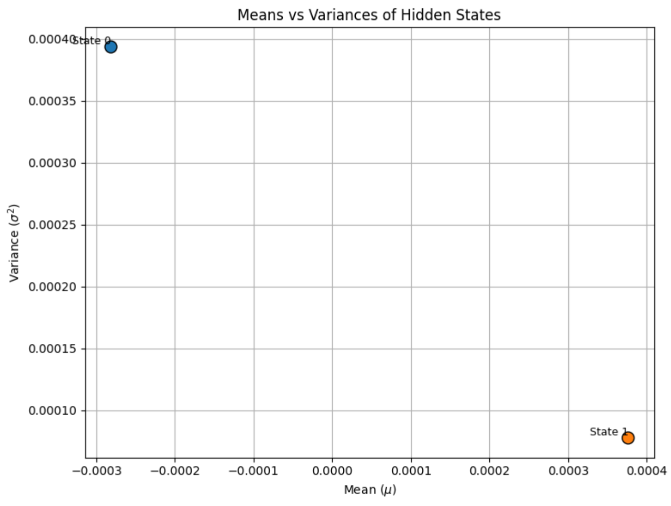


Fig. 5. Mean-variance plot for the two-state model.

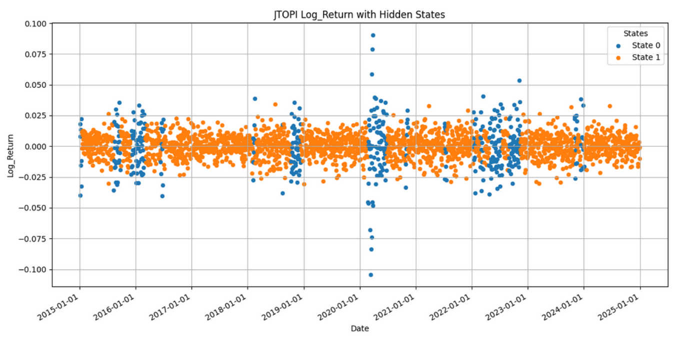
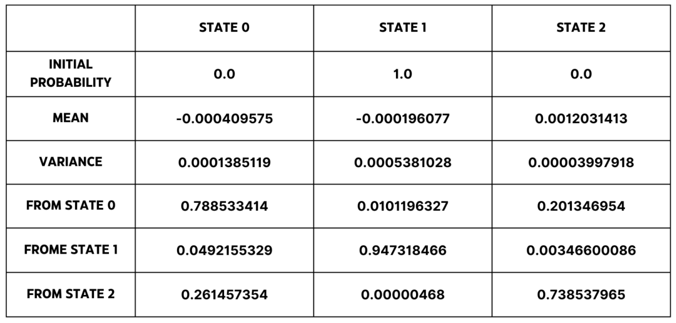


Fig. 6. Two-state model market regime detection.

B. Three-State Model Specifications And Results

The estimated parameters for the three-state model are presented in Fig. 7.



	STATE 0	STATE 1	STATE 2
INITIAL PROBABILITY	0.0	1.0	0.0
MEAN	-0.000409575	-0.000196077	0.0012031413
VARIANCE	0.0001385119	0.0005381028	0.00003997918
FROM STATE 0	0.788533414	0.0101196327	0.201346954
FROME STATE 1	0.0492155329	0.947318466	0.00346600086
FROM STATE 2	0.261457354	0.00000468	0.738537965

Fig. 7. Parameters for three-state model.

State 2 (green) displays a positive mean and low variance, indicative of a steady bull market with low volatility. This regime likely captures periods of gradual growth and investor optimism. State 1 (orange) characterized by a negative mean and high variance, signifying a highly volatile bear market. It captures periods of steep declines and increased uncertainty, typically associated with crises or downturns. This state aligns with events like the COVID-19 crash in early 2020 (Fig. 9), where returns were both negative and unstable. State 0 (blue) exhibits a negative mean and low variance, representing a steady bear trend. This state captures persistent downturns with relatively stable declines.

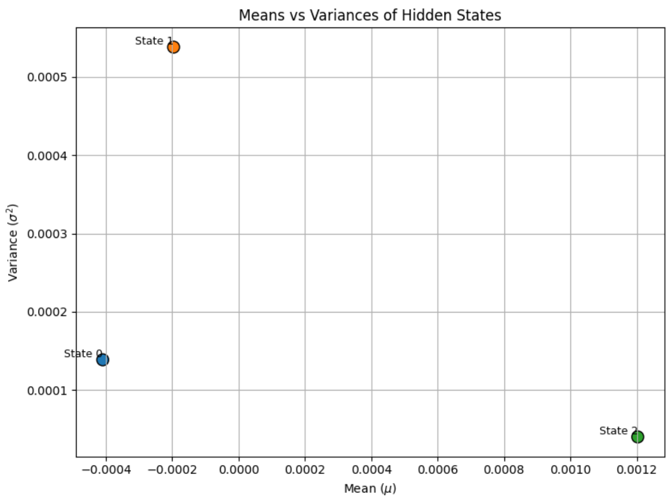


Fig. 8. Mean-variance plot for the three-state model.

Such a model could be leveraged to detect crisis-inspired regimes.

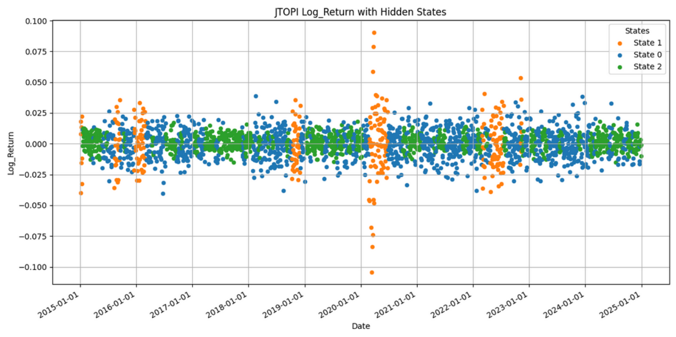


Fig. 9. Three-state model market regime detection.

VI. Benchmarking

For benchmarking and model comparison, we utilize the Akaike Information Criterion (AIC) and Bayesian Information Criterion. Both methods favor models that fit the data well but are not overly complex, with BIC penalizing complexity more harshly. The model with the lower score for AIC and BIC is preferred. Let k denote the number of parameters, n the size of the dataset and L the maximum likelihood, then the AIC and BIC scores of a model are computed as:

$$AIC = 2k - 2\log(L) \tag{20}$$
$$BIC = k\log(n) - 2\log(L) \tag{21}$$

The Three-State Model has a lower AIC score (Fig. 10), suggesting a better fit to the data while considering model complexity, according to this criterion. The Two-State Model has a lower BIC score, which, with equal dataset sizes, indicates that this model achieves a better trade-off between fit and complexity by penalizing the larger number of parameters in the three-state model more heavily.

	TWO-STATE MODEL	**THREE-STATE MODEL**
AIC	**-15248.790200474596**	**-15280.82230824407**
BIC	**-15208.129095327386**	**-15199.500097949649**

Fig. 10. AIC and BIC scores for two-state and three-state models.

Since the BIC tends to favor more parsimonious models, especially with consistent dataset sizes, the Two-State Model would likely be considered more prevalent in this scenario. The BIC suggests that the added complexity of the three-state model does not provide a sufficiently better fit to warrant the increased number of parameters.

VII. Conclusion

This study highlights the effectiveness of Gaussian Hidden Markov Models in identifying market regimes in the South African Top 40 Index. By modeling log returns as emissions from hidden states, the models capture distinct patterns such as bull, bear, and crisis periods. The two-state HMM differentiates between high- and low-volatility regimes, while the three-state model offers a more detailed segmentation, including crisis-specific behavior. Overall, the study confirms that Gaussian HMMs are a valuable tool for market regime detection, offering both interpretability and analytical rigor. These models can support better-informed investment strategies, improved risk management, and more timely recognition of market transitions.

VIII. Repository And Code

The complete codebase is available in the following GitHub repository: `Gaussian-Hidden-Markov-Model`.

The repository includes the following files:

- `2015_2025_Historical_Data.csv` – Historical market data retrieved from Investing.com.
- `data.py` – Defines a class containing methods for data preprocessing.
- `analysis.py` – Provides a class for time series analysis applicable beyond the JTOPI index.
- `hidden_markov_model.py` – Implements the Hidden Markov Model, including the Viterbi, forward, backward, and Baum-Welch algorithms.
- `main.ipynb` – Jupyter notebook demonstrating data loading, preprocessing, analysis, and HMM training on JTOPI data.
- `two_state_model.pkl` – Serialized parameters for the trained two-state HMM.
- `three_state_model.pkl` – Serialized parameters for the trained three-state HMM.