Financial Time Series Analysis Using Gaussian Hidden Markov Models

F A N U M

1 Introduction

In this report, we explore the application of **Gaussian Hidden Markov Models (HMMs)** to financial time series data, such as stock returns. The goal is to identify *hidden market regimes* (e.g., periods of high and low volatility) by modeling observable financial data, such as daily stock returns.

We use data from the *S&P 500 ETF (SPY)* over a period of 10+ years to train the HMM and infer hidden market states (bull and bear markets). This report outlines the steps taken, including data preprocessing, model fitting, and interpretation of the results, with visualizations to support the analysis.

2 Data Collection and Preprocessing

2.1 Data Acquisition

We collected historical stock price data using the **Yahoo Finance API** for the S&P 500 ETF (SPY). The data spans from **January 1, 2010** to **January 1, 2023**, covering key periods of market activity, including periods of growth and recession.

- **Ticker**: SPY (S&P 500 ETF)
- **Data**: Adjusted Close prices

2.2 Preprocessing

The raw data was preprocessed to compute daily **returns**, defined as the percentage change in adjusted closing prices between consecutive days. Returns were calculated using the formula:

$$\text{Returns} = \frac{\text{Adj Close}_t - \text{Adj Close}_{t-1}}{\text{Adj Close}_{t-1}} \tag{1}$$

Any missing data points were handled appropriately to ensure a clean dataset for modeling.

3 Gaussian Hidden Markov Model (HMM)

3.1 Model Fitting

We employed a **Gaussian Hidden Markov Model (HMM)** to model the daily returns data. This model assumes that stock returns are generated by a process with hidden market regimes, each represented by a Gaussian distribution.

We used a two-state HMM to capture distinct market regimes, such as:

- **State 0**: Bull market (low volatility, positive returns)
- **State 1**: Bear market (high volatility, negative returns)

The model was trained using the `hmmlearn` library, and the hidden states were inferred using the Viterbi algorithm.

3.2 Transition Matrix

The transition matrix of the HMM provides insight into the likelihood of transitioning between market regimes. This matrix captures the probabilities of switching from a bull market to a bear market and vice versa.

4 Interpretation and Inference

4.1 Inferred Hidden States

The hidden states inferred by the HMM represent different market regimes over time. Each day in the dataset is assigned a hidden state (either 0 or 1). To visualize the results, we plotted stock prices and color-coded the periods based on the inferred hidden states.

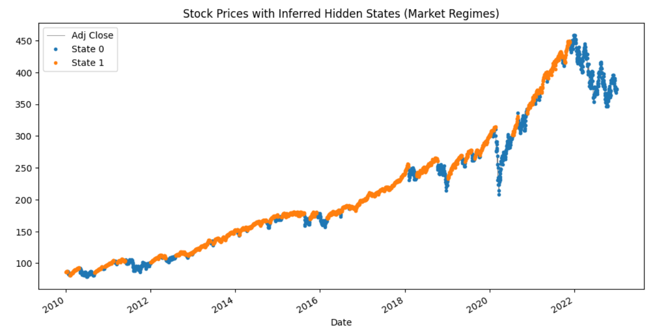


Figure 1: Stock Prices with Inferred Hidden States

4.2 Mean and Variance of Hidden States

We analyzed the mean and variance of the returns for each hidden state:

- **Bull Market (State 0)**: Characterized by positive returns and lower volatility.
- **Bear Market (State 1)**: Characterized by negative or low returns and higher volatility.

These characteristics are consistent with traditional market behavior during periods of growth and decline.

5 Evaluation and Visualization

5.1 Visualization of Returns and States

We also visualized the daily returns, color-coded based on the hidden states. This helps identify periods of high and low volatility.

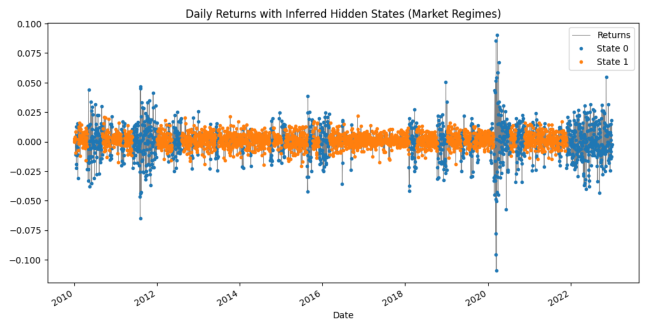


Figure 2: Daily Returns with Inferred Hidden States

5.2 Model Evaluation

The HMM successfully identified two distinct market regimes, providing meaningful insights into market behavior. The model's effectiveness was demonstrated by its ability to differentiate between bull and bear markets, highlighting periods of high and low volatility.

6 Conclusions and Insights

6.1 Market Regime Interpretation

The Gaussian Hidden Markov Model identified two key market regimes:

- **Bull Market (State 0)**: Low volatility, characterized by positive returns.

- **Bear Market (State 1)**: High volatility, characterized by negative or low returns.

These hidden market regimes provide valuable insights for **risk management** and **portfolio adjustment**. For instance, during bear markets (State 1), investors might consider shifting their portfolios to more defensive positions.

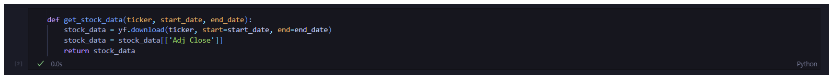
6.2 Future State Prediction

The transition matrix allows us to predict the likelihood of transitioning to a different market regime in the future. Based on the last known hidden state, we can forecast the probability of entering a bull or bear market in the short term.

The code snippets for the Model in python are provided in the next page

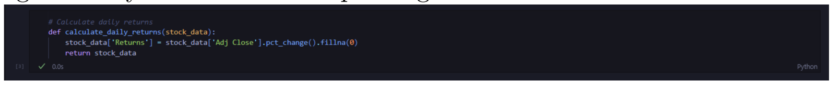
7 Python Code

The Python code used to implement the analysis of financial time series using a Gaussian Hidden Markov Model is shown in the image below:



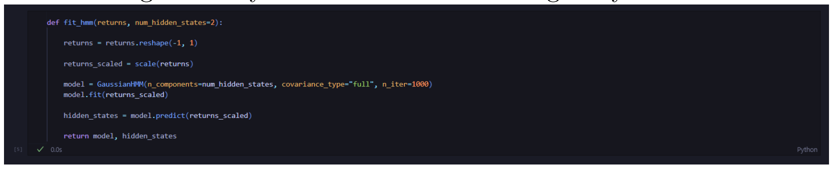
```python
def get_stock_data(ticker, start_date, end_date):
    stock_data = yf.download(ticker, start=start_date, end=end_date)
    stock_data = stock_data[['Adj Close']]
    return stock_data
```

Figure 3: Python Code for requesting stock data via Yahoo Finance API



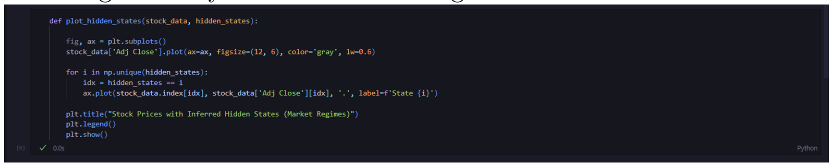
```python
# Calculate daily returns
def calculate_daily_returns(stock_data):
    stock_data['Returns'] = stock_data['Adj Close'].pct_change().fillna(0)
    return stock_data
```

Figure 4: Python Code for calculating daily returns



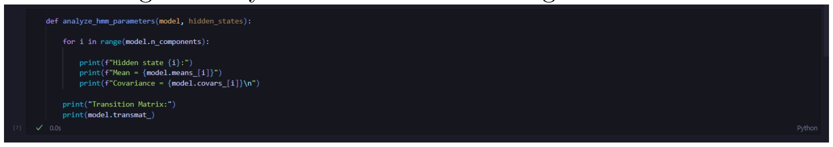
```python
def fit_hmm(returns, num_hidden_states=2):

    returns = returns.reshape(-1, 1)

    returns_scaled = scale(returns)

    model = GaussianHMM(n_components=num_hidden_states, covariance_type="full", n_iter=1000)
    model.fit(returns_scaled)

    hidden_states = model.predict(returns_scaled)

    return model, hidden_states
```

Figure 5: Python Code for fitting the dataset in the model



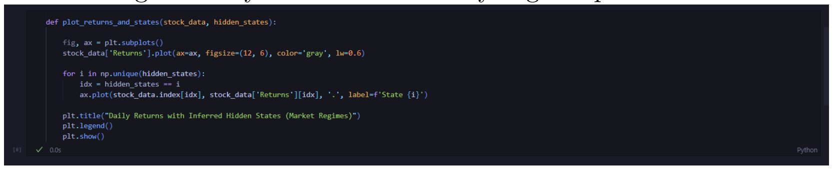
```python
def plot_hidden_states(stock_data, hidden_states):

    fig, ax = plt.subplots()
    stock_data['Adj Close'].plot(ax=ax, figsize=(12, 6), color='gray', lw=0.6)

    for i in np.unique(hidden_states):
        idx = hidden_states == i
        ax.plot(stock_data.index[idx], stock_data['Adj Close'][idx], '.', label=f'State {i}')

    plt.title("Stock Prices with Inferred Hidden States (Market Regimes)")
    plt.legend()
    plt.show()
```

Figure 6: Python Code for visualizing hidden states



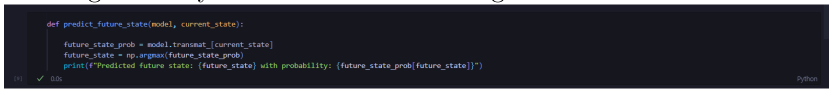
```python
def analyze_hmm_parameters(model, hidden_states):

    for i in range(model.n_components):

        print(f"Hidden state {i}:")
        print(f"Mean = {model.means_[i]}")
        print(f"Covariance = {model.covars_[i]}\n")

    print("Transition Matrix:")
    print(model.transmat_)
```

Figure 7: Python Code for analyzing the parameters



```python
def plot_returns_and_states(stock_data, hidden_states):

    fig, ax = plt.subplots()
    stock_data['Returns'].plot(ax=ax, figsize=(12, 6), color='gray', lw=0.6)

    for i in np.unique(hidden_states):
        idx = hidden_states == i
        ax.plot(stock_data.index[idx], stock_data['Returns'][idx], '.', label=f'State {i}')

    plt.title("Daily Returns with Inferred Hidden States (Market Regimes)")
    plt.legend()
    plt.show()
```

Figure 8: Python Code for visualizing the returns and states



```python
def predict_future_state(model, current_state):

    future_state_prob = model.transmat_[current_state]
    future_state = np.argmax(future_state_prob)
    print(f"Predicted future state: {future_state} with probability: {future_state_prob[future_state]}")
```

Figure 9: Python Code for predicting future state