GROUP WORK PROJECT # 2 MScFE 660: RISK MANAGEMENT
Group Number: 8183

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Ebenezer Yeboah	Ghana	ebenezeryeboah46@gmail.com	
Christson Hartono	Indonesia	hartono.christson@gmail.com	
Eric Walter Pefura-Yone	Cameroon	pefurayone@gmail.com	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).	
Team member 1	**Ebenezer Yeboah**
Team member 2	**Christson Hartono**
Team member 3	**Eric Walter Pefura-Yone**

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed. **Note:** You may be required to provide proof of your outreach to non-contributing members upon request.

Step 1: Pseudocode for Hidden Markov Model (HMM)(Sean R. Eddy; Sean R Eddy; Rabiner and Juang)

a. Forward algorithm pseudocode and toy example

a1. Forward algorithm pseudocode

Goal: Compute $P(O|\lambda)$, the probability of observing O given the model $\lambda = (\pi, A, B)$.

Input: Observation sequence O = [o_1, o_2, ..., o_T]
States Q = [q_1, q_2, ..., q_N] (N possible hidden states)
Initial probabilities π = [π_1, π_2, ..., π_N]
Transition matrix A = [a_{ij}] (size N × N)
Emission matrix B = [$b_i(o_t)$] (size N × M)

Output: Probability P(O | λ)

// Step 1: Initialization (t = 1)
for i = 1 to N:
  α[1, i] = π_i * $b_i(o_1)$
  // α(1, i) is the probability of being in state q_i at time t = 1

// Step 2: Recursion (t = 2 to T)
for t = 2 to T:
  for j = 1 to N:
    α[t, j] = (Σ from i = 1 to N of α[t - 1, i] * a_{ij}) * $b_j(o_t)$
    // Sum over previous states q_i, multiply by transition a_{ij} and emission $b_j(o_t)$

// Step 3: Termination
P(O | λ) = Σ from i = 1 to N of α[T, i]
// Sum over all states at final time T

return P(O | λ)
End Forward Algorithm

a2. Toy example for forward algorithm

Our model λ is described as follow:

- For N = 2 states (example: "Rain" and "Sunshine");
- T = 2 observations (example: "Umbrella" or "Sunglasses");
- and initial probabilities π = [0.6, 0.4].

We Suppose that:

Transition Matrix (A):

$$A = \begin{bmatrix} P(\text{Rain} \mid \text{Rain}) & P(\text{Sunshine} \mid \text{Rain}) \\ P(\text{Rain} \mid \text{Sunshine}) & P(\text{Sunshine} \mid \text{Sunshine}) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

Emission Matrix (B):

$$B = \begin{bmatrix} P(\text{Umbrella} \mid \text{Rain}) & P(\text{Umbrella} \mid \text{Sunshine}) \\ P(\text{Sunglasses} \mid \text{Rain}) & P(\text{Sunglasses} \mid \text{Sunshine}) \end{bmatrix} = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}$$

We can apply, forward algorithm as below:

Step 1: Initialization $(t = 1)$ For the first observation o_1:

$\alpha(1,1) = \pi_1 \cdot b_1(o_1) = 0.6 \cdot 0.9 = 0.54$

$\alpha(1,2) = \pi_2 \cdot b_2(o_1) = 0.4 \cdot 0.2 = 0.08$

Step 2: Recursion $(t = 2)$ For the second observation o_2 :

1. **State 1**:
   $\alpha(2,1) = (\alpha(1,1) \cdot a_{11} + \alpha(1,2) \cdot a_{21}) \cdot b_1(o_2)\$\$ = (0.54 \cdot 0.7 + 0.08 \cdot 0.4) \cdot 0.1 = (0.378 + 0.032) \cdot 0.1 = 0.041$
2. **State 2**:
   $ \alpha(2, 2) = \left(\alpha(1, 1) \cdot a_{12} + \alpha(1, 2) \cdot a_{22} \right) \cdot b_2(o_2)
   $= (0.54 \cdot 0.3 + 0.08 \cdot 0.6) \cdot 0.8 = (0.162 + 0.048) \cdot 0.8 = 0.168$

Step 3: Termination Sum the probabilities at $(t = 2)$:

$$P(O|\lambda) = \alpha(2,1) + \alpha(2,2) = 0.041 + 0.168 = \mathbf{0.209}$$

Interpretation

There is a 20.9% chance of observing the sequence O = [o_1, o_2] giving our model λ

b. Backward algorithm pseudocode and toy example

b1. Backward algorithm pseudocode

Goal: Compute the backward probabilities $\beta(t, i)$, which represent the likelihood of future observations given the current state.

Input: Observation sequence O = [o_1, o_2, ..., o□]

States Q = [q_1, q_2, ..., q□] (N possible hidden states)

Transition matrix A = [a_i□] (size N × N)

Emission matrix B = [b_i(o□)] (size N × M)

Output: Probability P(O | λ)

// Step 1: Initialization (t = T)
for i = 1 to N:
  β[T, i] = 1
  // β(T, i) is the probability of the "empty" future given state q_i at time T

// Step 2: Recursion (t = T - 1 to 1)
for t = T - 1 down to 1:
  for i = 1 to N:
    β[t, i] = Σ from j = 1 to N of a_{i} * $b_{}(o_{+1})$ * β[t + 1, j]
    // Sum over next states $q_{}$, multiply by transition a_{i}, emission $b_{}(o_{+1})$, and β[t + 1, j]

// Step 3: Termination
P(O | λ) = Σ from i = 1 to N of π_i * $b_i(o_1)$ * β[1, i]
// Sum over all states at time t = 1, weighted by initial probability π_i and emission $b_i(o_1)$

return P(O | λ)
End Backward Algorithm

b2. Toy example for backward algorithm

We used the same parameters for model λ.

Step 1: Initialization $(t = T = 2)$

At the final time step (t = 2), initialize backward probabilities to **1** for all states:

$$\beta(2,1) = 1 \quad (\text{Rain}), \quad \beta(2,2) = 1 \quad (\text{Sunshine})$$

Step 2: Recursion $(t = 1)$

Compute backward probabilities for $t = 1$ using the recursion formula:

$$\beta(t,i) = \sum_{j=1}^{2} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta(t+1,j)$$

1. **For state** $q_1 = \text{Rain}$: $\beta(1,1) = (a_{11} \cdot b_1(o_2) \cdot \beta(2,1)) + (a_{12} \cdot b_2(o_2) \cdot \beta(2,2))$
   $= (0.7 \cdot 0.1 \cdot 1) + (0.3 \cdot 0.8 \cdot 1) = 0.07 + 0.24 = \mathbf{0.31}$
2. **For state** $q_2 = \text{Sunshine}$: $\beta(1,2) = (a_{21} \cdot b_1(o_2) \cdot \beta(2,1)) + (a_{22} \cdot b_2(o_2) \cdot \beta(2,2))$
   $= (0.4 \cdot 0.1 \cdot 1) + (0.6 \cdot 0.8 \cdot 1) = 0.04 + 0.48 = \mathbf{0.52}$

Step 3: Termination

Compute the total likelihood $P(O \mid \lambda)$:

$$P(O \mid \lambda) = \sum_{i=1}^{2} \pi_i \cdot b_i(o_1) \cdot \beta(1,i) = (0.6 \cdot 0.9 \cdot 0.31) + (0.4 \cdot 0.2 \cdot 0.52)$$
$$= (0.6 \cdot 0.279) + (0.4 \cdot 0.104) = 0.1674 + 0.0416 = \mathbf{0.209}$$

Interpretation

- The backward algorithm confirms lambda= 0.209 , matching the forward algorithm result.

- This consistency validates that **both algorithms compute the same likelihood** for the observation sequence ($O = [Umbrella, Sunglasses]$.

c. Backward Viterbi algorithm

c1. Backward Viterbi pseudocode

Goal: Reconstruct the most likely sequence of hidden states $ q^* = (q_1^, *q_2^*, ..., q_T^*)$ given observations $ O = (o_1, o_2, ..., o_T)$ and model $ \lambda $.

Input: Observation sequence O = [o_1, o_2, ..., o_T]
States Q = [q_1, q_2, ..., q_N] (N possible hidden states)
Initial probabilities π = [π_1, π_2, ..., π_N]
Transition matrix A = [a_{ij}] (size N × N)
Emission matrix B = [$b_i(o_t)$] (size N × M)

Output: Most likely hidden state sequence q* = [q_1*, q_2*, ..., q_T*]

// Step 1: Forward Pass (Compute path probabilities)
// Initialization (t = 1)
for i = 1 to N:
  δ[1, i] = π_i * $b_i(o_1)$
  ψ[1, i] = 0
  // δ(1, i) = probability of starting in state q_i and observing o_1

// Recursion (t = 2 to T)
for t = 2 to T:
  for j = 1 to N:
    δ[t, j] = max$_k$ (δ[t - 1, k] * a_{kj}) * $b_j(o_t)$
    ψ[t, j] = argmax$_k$ (δ[t - 1, k] * a_{kj})
    // Track predecessor state for traceback

// Step 2: Backward Pass (Trace optimal path)
// Termination (t = T)
q_T* = argmax$_k$ (δ[T, j])

// Traceback (t = T - 1 to 1)
for t = T - 1 down to 1:
  q_t* = ψ[t + 1, q_{t+1}*]

// Reconstruct path using ψ pointers

return q*
End Viterbi Algorithm

c2. Toy example for Viterbi algorithm

Model λ:

- **States**: {Rain (1), Sunshine (2)}.
- **Observations**: $O = [\text{Umbrella } (o_1), \text{Sunglasses } (o_2)]$.
- **Matrices**:

$$\pi = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \quad A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}, \quad B = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}$$

Step 1: Forward Pass
At $t = 1$ (observation $o_1 = \text{Umbrella}$):
$\delta_1(1) = \pi_1 \cdot b_1(o_1) = 0.6 \cdot 0.9 = \mathbf{0.54}, \quad \psi_1(1) = 0$

$\delta_1(2) = \pi_2 \cdot b_2(o_1) = 0.4 \cdot 0.2 = \mathbf{0.08}, \quad \psi_1(2) = 0$

At (t = 2) (observation (o_2 = \text{Sunglasses})):

- **State 1 (Rain)**:
  $\delta_2(1) = \max\left[\delta_1(1) \cdot a_{11},\ \delta_1(2) \cdot a_{21}\right] \cdot b_1(o_2)$

$= \max\left[0.54 \cdot 0.7,\ 0.08 \cdot 0.4\right] \cdot 0.1 = \mathbf{0.0378}$

$\psi_2(1) = \arg\max\left[1, 2\right] = \mathbf{1}$

- **State 2 (Sunshine)**:
  $\delta_2(2) = \max\left[\delta_1(1) \cdot a_{12},\ \delta_1(2) \cdot a_{22}\right] \cdot b_2(o_2)$

$= \max\left[0.54 \cdot 0.3,\ 0.08 \cdot 0.6\right] \cdot 0.8 = \mathbf{0.1296}$

$\psi_2(2) = \arg\max\left[1, 2\right] = \mathbf{1}$

Step 2: Backward Pass (Traceback)

1. **Final state at** $t = 2$: $q_2^* = \arg\max\left[\delta_2(1) = 0.0378,\ \delta_2(2) = 0.1296\right] = \mathbf{2\ (Sunshine)}$
2. **Traceback to** $t = 1$: $q_1^* = \psi_2(q_2^*) = \psi_2(2) = \mathbf{1\ (Rain)}$

Result
Most Likely Hidden state sequence:
$q^* = (\text{Rain}, \text{Sunshine})$

Probability of the optimal path:
$P(O, q^* \mid \lambda) = \delta_2(2) = 0.1296$

Interpretation:
The sequence **Rain → Sunshine** is the most probable path for the observations $O = [\text{Umbrella}, \text{Sunglasses}]$ according to the Viterbi algorithm.

d. Baum-Welch algorithm

d1. Backward algorithm pseudocode

Goal: Learns the model parameters λ* from observations
Input: Observation sequence O = [o_1, o_2, ..., o□]
Number of hidden states N

Number of possible observations M
Initial guess for parameters λ = (π, A, B)
Maximum iterations or convergence threshold

Output: Learned parameters λ* = (π*, A*, B*)

// Step 1: Initialize
Initialize π, A, B with random or guessed values
Set iteration = 0

Repeat until convergence or max iterations:
// Step 2: Expectation (E-step)
Compute forward probabilities α(t, i) using Forward Algorithm
Compute backward probabilities β(t, i) using Backward Algorithm
Compute total likelihood $P(O \mid \lambda) = \Sigma_i\, \alpha(T, i)$

// Step 3: Compute γ and ξ
for t = 1 to T-1:
for i = 1 to N:
// Gamma: Probability of being in state i at time t
γ(t, i) = α(t, i) * β(t, i) / P(O | λ)
for j = 1 to N:
// Xi: Probability of transitioning i→j at time t
ξ(t, i, j) = α(t, i) * $a_{i\square}$ * $b_{\square}(o_{\square+1})$ * β(t+1, j) / P(O | λ)

// Special case for γ(T, i) (no transitions after T)
for i = 1 to N:
γ(T, i) = α(T, i) * β(T, i) / P(O | λ)

// Step 4: Maximization (M-step)
// Update transition matrix A
for i = 1 to N:
for j = 1 to N:
$a_{i\square}$* = $\Sigma_{\square=1}$^{T-1} ξ(t, i, j) / $\Sigma_{\square=1}$^{T-1} γ(t, i)

// Update emission matrix B
for j = 1 to N:
for k = 1 to M:
$b_{\square}$*(k) = $\Sigma_{\square=1}$^T γ(t, j) * δ($o_{\square}$ = $v_{\square}$) / $\Sigma_{\square=1}$^T γ(t, j)
// δ($o_{\square}$ = $v_{\square}$) = 1 if $o_{\square}$ = $v_{\square}$, else 0

// Update initial probabilities π

for i = 1 to N:
$\pi_i^* = \gamma(1, i)$

// Check convergence (e.g., small change in A, B, π)
iteration += 1

Return $\lambda^* = (\pi^*, A^*, B^*)$
End Baum-Welch Algorithm

d2. Toy example for Baum-Welch algorithm

Model λ:

- **States**:{Rain (1), Sunshine (2)}.
- **Observations**: $O = [\text{Umbrella } (o_1), \text{Sunglasses } (o_2)]$.
- **Initial Parameters**:

$$\pi = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \quad A = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}, \quad B = \begin{bmatrix} 0.9 & 0.2 \\ 0.1 & 0.8 \end{bmatrix}$$

Step 1: Forward-Backward Procedure

Forward Probabilities α:

- **At** $t = 1$ (observation $o_1 = \text{Umbrella}$):
  $\alpha_1(1) = 0.54, \quad \alpha_1(2) = 0.08$
- **At (t = 2)** (observation (o_2 = \text{Sunglasses})):
  $\alpha_2(1) = 0.041, \quad \alpha_2(2) = 0.168$
- **Total Likelihood**:
  $P(O \mid \lambda) = 0.209$

Backward Probabilities β:

- **At** $t = 2$: $\beta_2(1) = 1, \quad \beta_2(2) = 1$
- **At (t = 1)**: $\beta_1(1) = 0.31, \quad \beta_1(2) = 0.52$

Step 2: Compute Gamma γ

$\gamma_t(i) = \frac{\alpha(t,i)\cdot\beta(t,i)}{P(O|\lambda)}$

- **At** $t = 1$: $\gamma_1(1) = \frac{0.54\cdot 0.31}{0.209} \approx 0.8, \quad \gamma_1(2) = \frac{0.08\cdot 0.52}{0.209} \approx 0.2$
- **At (t = 2)**: $\gamma_2(1) = \frac{0.041\cdot 1}{0.209} \approx 0.196, \quad \gamma_2(2) = \frac{0.168\cdot 1}{0.209} \approx 0.804$

Step 3: Compute ξ

$$\xi_t(i,j) = \frac{\alpha(t,i)\cdot a_{ij}\cdot b_j(o_{t+1})\cdot\beta(t+1,j)}{P(O|\lambda)}$$

- **At** $t = 1$:

$$\xi_1(1,1) \approx 0.180, \quad \xi_1(1,2) \approx 0.620$$
$$\xi_1(2,1) \approx 0.015, \quad \xi_1(2,2) \approx 0.185$$

Step 4: Update Parameters

New Transition Matrix A^*: $a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$

- For a_{11}^*:
  $a_{11}^* = \frac{0.180}{0.8} \approx 0.225$
- For a_{12}^*:
  $a_{12}^* = \frac{0.620}{0.8} \approx 0.775$
- For a_{21}^*:
  $a_{21}^* = \frac{0.015}{0.2} \approx 0.075$
- For a_{22}^*:
  $a_{22}^* = \frac{0.185}{0.2} \approx 0.925$

New Emission Matrix B^*: $b_j^*(o_k) = \frac{\sum_{t=1}^{T} \gamma_t(j)\cdot \mathrm{I}(o_t=o_k)}{\sum_{t=1}^{T} \gamma_t(j)}$

- For $b_1^*(\text{Umbrella})$:
  $b_1^*(\text{Umbrella}) = \frac{0.8}{0.8+0.196} \approx 0.803$
- For $b_2^*(\text{Sunglasses})$:
  $b_2^*(\text{Sunglasses}) = \frac{0.804}{0.2+0.804} \approx 0.800$

New Initial Probabilities π^*: $\pi^* = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$

Updated Model λ^*

$$\pi^* = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}, \quad A^* = \begin{bmatrix} 0.225 & 0.775 \\ 0.075 & 0.925 \end{bmatrix}, \quad B^* = \begin{bmatrix} 0.803 & 0.197 \\ 0.199 & 0.800 \end{bmatrix}$$

Interpretation

- The Baum-Welch algorithm updates λ to better fit the observation sequence O.
- Transitions to Sunshine q_2 are reinforced, and emissions align more closely with observations.
- Iterate until parameters converge (e.g., until A, B, π stabilize).

Step 2: Regime detection (Bhattacharya and Ray; Wang et al.; Zheng et al.)

We will use a Hidden Markov Model (HMM) to detect regime changes. An HMM is a mathematical model that allows us to understand the hidden states of a system by observing outputs or emissions. It is described by 5 main elements.

- **States (**Q**)**: these are the hidden states of the system. For example, the states of variation of crude oil prices:
  - Bullish (rising prices)
  - Bearish (falling prices)

- Stagnant (stable prices)

- **Symbols (Σ)**: these are emissions or observations. In our case, it will be either an increase in prices between months (1) or a decrease in price §(0).
- **Initial probability (Π)**: it represents the probability of starting in each hidden state (rising, falling or stagnant for the price of crude oil).
- **Transition probabilities (A)**: this matrix indicates the probabilities of transitioning from one hidden state to another.
- **Emission probabilities (B)**: this matrix describes the probability of observing a particular symbol given a specific hidden state.

These components define the HMM. These parameters are denoted by: $\lambda = (\Pi, A, B)$

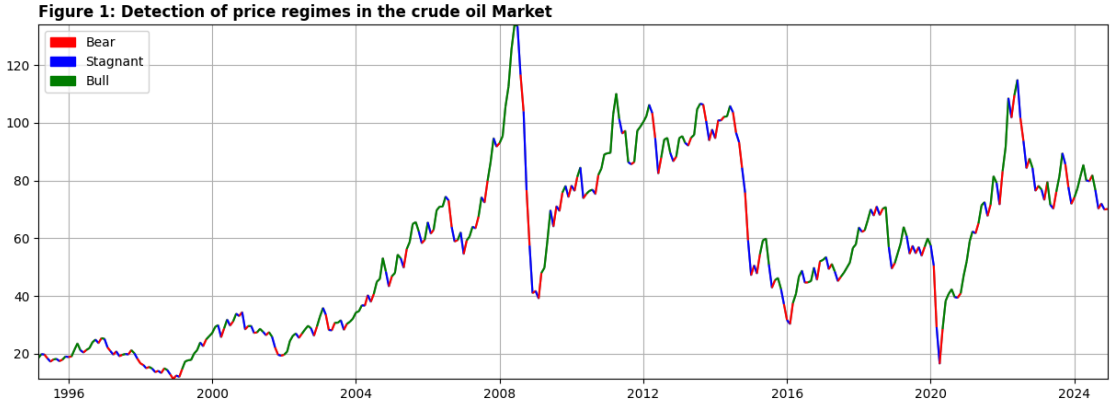
2.1. The dataset

We used the variable "WTISPLC" representing the price of crude oil.

2.2. Implementing HMM for regime detection

We created a function to perform the different steps of the HMM training. By fixing the random seed we obtained the reproductible parameters of the model. The next step was to reclassify the average differences of each regime (high, low, stagnant) in each sequence. A graph allows one to visualize the regime changes.

Figure 1: Detection of price regimes in the crude oil Market



This is the process of detecting hidden states in time-series data.

Regime detection was constructed using the Gaussian HMM model with a plot (Figure 1) of segmental coloring throughout the study period from 1995 to 2024 and using the logarithm of the crude oil price return. The bullish movement states are colored in green, the bearish states in red and the stagnant states in orange.

a) The Bull Market enthusiast

The analysis of bullish regimes on the crude oil price chart shows the relative frequency of bullish movements that start from the beginning of the time series and continue intermittently throughout the study period. Bullish movements are often immediately followed by bearish movements and sometimes

by stagnant states. There is a period of sustained increase from 2005 to 2008 indicating that bullish states were more important than bearish states during this period. The rise peaked at the end of 2007 with the crash of 2008. After the 2008 crisis, another period of sustained increase was observed until 2015 and then from 2022 to 2024.

b) The bear market observer

This more circumspect observer approaches the graph with a critical eye, seeking to identify periods of decline and uncertainty. His gaze falls on the red portions indicating the bearish states that most often follow the green portions. He notes that despite the incessant fluctuations between bearish, stagnant and bullish states, the overall trend is rather bullish with a price at the end of the study period 3 times higher than at the beginning of the period. His bearish analyst's gaze will focus mainly on 3 segments corresponding to the sudden drops in the price of crude oil: 2008 (subprime financial crisis), 2015-2016 and 2020 (COVID health crisis). The price of oil almost reached the lowest price in 2020 of the post-2000 period.

c) The stagnation specialist

This patient observer is dedicated to detecting periods of stagnation—those moments when the market seems to be holding its breath. His attention is drawn to the blue touches that appear either after a bullish or bearish state. Some stagnant states emerge during relatively prolonged bearish periods, corresponding to the 2008 financial crisis, the 2015-2016 oil price crash, and the 2000 dot-com bubble burst. These are periods of precarious equilibrium, often quickly followed by significant market variations.

Step 3: Hidden Markov model (Zheng et al.; Bhattacharya and Ray; Sean R. Eddy; Rabiner and Juang)

3.1. Definition and Properties

- Hidden Markov Model (HMM) is a probabilistic statistical model that can represent temporal or chronological systems using an underlying Markov process of hidden or unobservable states. Hidden states generate observable outputs or emissions that allow the behavior of the entire system to be inferred.
- A Markov process is a stochastic model in which the future evolution of the system depends only on its current state. This characteristic is known as the Markov property and is formally described by:
- Let $\left(X_t\right)_{t\geq 0}$ be a sequence of random variables representing the state of the system at time (t)
- The Markov property is stated as follows:
  $P\left(X_{t+1}=x_{t+1}|X_t=x_t, X_{t-1}=x_{t-1}, \ldots, X_0=x_0\right)= P\left(X_{t+1}=x_{t+1}|X_t=x_t\right)$
- This means that, conditional on the present state X_t, the distribution of X_{t+1} is independent of the previous states $X_{t-1}, X_{t-2}, \ldots, X_0$.

3.2. HMM Components

A HMM is defined by the following elements:

- **States (Q):** $Q = \{q_1, q_2, \ldots, q_N\}$, where each q_i represents a distinct regime. These are the hidden states of the system.
- **Observations (Σ):** The observed outputs are drawn from a finite set of symbols: $\Sigma = \{s_1, s_2, \ldots, s_M\}$. In many practical applications, such as market analysis, these can be binary indicators (e.g., 1 for a price increase and 0 for a price decrease).
- **Initial probabilities (Π):** the initial probability distribution over states is given by $ \Pi = {\pi_i}, \quad \pi_i = P(q_1 = q_i)$. This indicates the probability that the system starts in state q_i.
- **Transition probabilities (A):** $A = \{a_{ij}\}, \quad a_{ij} = P\left(q_{t+1} = q_j | q_t = q_i\right)$. This matrix governs the probability of the system transitioning from one state to another.
- **Emission probabilities (B):** Given a hidden state, the probability of observing a specific symbol is determined by $B = \{b_j(k)\}, \quad b_j(k) = P\left(s_k | q_j\right)$.
- The set of components defining an HMM is generally denoted: $\lambda = (\Pi, A, B)$

3.3. Inference in HMMs

The key tasks that make an HMM functional are:

- **Evaluation:** computing the likelihood of an observed sequence using the forward algorithm (forward, backward);
- **Decoding:** using the Viterbi algorithm to identify the most probable sequence of hidden states;
- **Learning:** estimating the parameters (Π, A, B) from the data using the Baum-Welch algorithm (Expectation-Maximization).
-

Step 4: Methods (*Homepage - U.S. Energy Information Administration (EIA)*; *World Bank Open Data | Data*; *Federal Reserve Economic Data | FRED | St. Louis Fed*; Bank; *Economic Policy Uncertainty Index*; *Yahoo Finance - Stock Market Live, Quotes, Business & Finance News*)

4.1. Macro strategy and identification of factors

4.1.1. Defining the macroeconomic data and identifying relevant indicators

4.1.1.1. Macroeconomic data

Exploring the macroeconomic landscape is crucial for analyzing the influence of the global economic system on the price of crude oil. We applied a methodical approach to collect information on the main macroeconomic variables related to the American market. This macroeconomic information should improve the performance of our predictive model. Several dimensions of the data were explored and

collected, including economic growth measured by gross domestic product (GDP), oil consumption, inflation and industrial production.

For GDP, we collected data on the GDP of OECD member countries and also that of non-OECD countries. Global economic growth could have a positive impact on oil consumption. Thus, more flourishing economies tend to increase energy consumption and, by extension, the stimulation of oil demand. As for oil consumption, the analysis of consumption trends in OECD and non-OECD countries will allow us to take this consumption into account in the evolution of oil prices. It should be noted that seasonal variations, technological progress and demographic changes are potential factors that could influence oil consumption.

Inflation is another leading macroeconomic indicator. We have chosen a measure of inflation relative to energy consumption. Thus, the energy consumer price index was used to take into account the possible impact of this index on oil prices. It should be noted, for example, that significant or galloping inflation can reduce the purchasing power of households or consumers, which will result in lower oil demand and therefore a downward adjustment in the price.

The Industrial Production Index (IPI) provides information on economic activity and oil demand within the industrial sector. Booming industrial production can stimulate the consumption of energy resources and more specifically that of oil.

The EUR/USD exchange rate represents the value of one euro in US dollars. It is a crucial economic indicator that reflects the relative strengths of the Eurozone and US economies. These relative strengths are influenced by interest rates, inflation, economic growth and geopolitical events. A strong euro with a high EUR/USD rate will promote imports to Europe and make exports more expensive. The trade balance is impacted by the exchange rate and the exchange rate can thus play a subtle role in the price of oil. A stronger euro makes oil more affordable for European buyers, which can stimulate oil demand which will subsequently lead to an upward adjustment in prices. This influence is all the more important since the Eurozone is a major importer of petroleum products. The fluctuation of the exchange rate must be taken into account when predicting the price of crude oil.

4.1.1.2. Geopolitical Considerations

Geopolitical factors impact movements related to oil production, demand, consumption and prices. Thus, conflicts, economic sanctions and decisions taken by nations playing a major role in oil exports often have a significant impact on oil prices. For example, it is worth noting the increase in oil prices related to political instability in oil-producing regions.

For our analysis, we collected geopolitical data on the production of petroleum products, crude oil production capacity, actual crude oil production and strategic reserves of OPEC and non-OPEC countries.

We also looked at a global index of geopolitical risk measurement called the geopolitical risk index (GPR). The GPR is a key barometer for assessing the impact of global instability on the price of crude oil. The GPR is determined by analyzing major geopolitical events mentioned in press articles with major international media coverage. Using natural language processing techniques, we can quantify the level of geopolitical agitation or risk. The more negative events are reported in newspapers, the higher the GPR and vice versa. Several events such as tensions between states, armed conflicts, economic sanctions often result in an increased perception of geopolitical risk and an increase in the GPR. This is due to an anticipation by the markets of the occurrence of a potential oil shortage. Periods of political stability

tend to reduce concerns on the markets with a stabilization or a consequent decrease in the GPR. Artificial intelligence has been used to obtain certain outlines in this section.

4.1.1.3. Microeconomic data and hidden forces

If the price of crude oil is influenced by complex interactions between macroeconomic data and geopolitical events, microeconomic data will also interact with the global market to truly impact oil market prices. The volatility of the price of crude oil is therefore potentially influenced by microeconomic data. These drivers act as active sensors and will capture variations in demand, supply, costs of scale for producers, refineries on the one hand and consumers on the other. We will analyze how national data on production, inputs, imports, capacity utilization, industrial production and price indices interact to impact crude oil prices.

- U.S. crude oil production data is a key indicator of understanding crude oil prices. An increase in production is an indicator of increased supply, which in the absence of strong demand will drive prices down. Conversely, a decrease in U.S. production due to, for example, oil well shutdowns or voluntary restrictions can decrease supply and drive prices up.
- Inputs refer to the amount of crude oil processed by refineries. An increase in inputs indicates greater demand for refined products such as gasoline and diesel. A decrease in inputs can signal a reduction in demand, with inventories of already refined products building up and a consequent drop in prices.
- International trade conditions the balance between supply and demand. The volume of crude oil imports into the United States reflects the country's ability to satisfy its domestic demand. High import dependence can be responsible for higher domestic market volatility in response to external shocks.
- Capacity utilization measures the percentage of oil production capacity that is actually used. High utilization rates indicate strong demand and tight supply, which can support prices. Short-term and long-term capacity utilization have different influences on market dynamics. Industrial production indices in the oil extraction, crude oil extraction and refining sectors provide a view of microeconomic activity. Strong production in the refining sector can support crude oil prices by assuming an increase in demand for refined products. The Producer Price Index (PPI) for oil and gas extraction measures the change in prices that producers receive for their production. An increase in the PPI is considered an indicator of rising crude oil prices.

4.1.1.4. Financial data

In addition to the price of crude oil (WTI), we have collected other financial data, in particular the prices of the S&P500 stock index and the inventories of petroleum products. High inventories of oil indicate an excess supply compared to current demand. This situation will tend to put downward pressure on oil prices. On the other hand, low inventories suggest strong demand and tighter supply, which can push up oil prices. Furthermore, the S&P500 stock index, considered as an indicator of overall economic health, can potentially influence oil prices. Thus, an increased performance of this stock index reflects greater investor confidence and an attractive economic outlook. In this case of increased confidence, the demand for oil can increase as well as its price. A correlation between the S&P 500 and oil prices is often

observed. Integrating these financial data into oil price prediction can help to better refine prediction models.

4.1.2. Data collection

Once the key indicators had been identified, we began the data collection process. This stage proved complex because the data came from disparate and diverse sources and were also presented in a variety of formats. We used the Application Programming Interfaces (APIs) of the Energy Information Administration (EIA), Federal Reserve Economic Data (FRED), World Bank, Economic Policy Uncertainty (EPU) , European Central Bank (ECB) and Yahoo Finance (Table 1).

- The EIA is a primary source of energy data such as production, consumption, stocks and oil prices.
- FRED) provides a wide range of US macroeconomic data, including interest rates, inflation rates and industrial production.
- The World Bank provides global macroeconomic and demographic data, including GDP.
- The EPU is the supplier of the GPR.
- The ECB is a source of European financial and economic data, including interest rates, inflation and exchange rates.
- Yahoo Finance provides financial data such as share prices and stock market indices.

We developed specific Python functions for each data source to automate data acquisition and structuring. Data integration was carried out taking into account the disparity in data frequency (monthly or quarterly) and data availability periods. Harmonisation procedures were put in place to ensure data consistency. For example, quarterly data has been extended to the months concerned. USD/EUR exchange rate data was not available for the 1995-1998 period, as the single European currency was introduced in 1999.

Table 1: Data table

Type	Sub-Type	Dataset ID	Description	Frequency	Source	Start Date	End Date	Unit of Measurement	Usual Range
Macroeconomic	GDP Growth	`OECD.GDP.GROWTH`	Real GDP for OECD countries	Quarterly	World Bank	01/01/1995	31/12/2024	Percentage (%)	-2% to 5%
	GDP Growth	`Non-OECD.GDP.GROWTH`	Real GDP for non-OECD countries	Quarterly	World Bank	01/01/1995	31/12/2024	Percentage (%)	-3% to 7%
	Petroleum Consumption	`STEO.PATC_OECD.M`	Petroleum consumption in OECD countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	40-50 MMb/d
	Petroleum Consumption	`STEO.PATC_NON_OECD.M`	Petroleum consumption in non-OECD countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	50-60 MMb/d
	Forex Rates	`exchange_rate`	Global foreign exchange rates (USD/EUR), reflecting macroeconomic stability	Monthly	ECB	01/01/1999	31/12/2024	USD/EUR	0.8-1.2 USD/EUR
	Inflation	`CPIENGSL`	Consumer Price Index (CPI) for energy, reflecting inflation in energy prices	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
	Industrial Production	`INDPRO`	Industrial Production Index, measuring output in the industrial sector	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
Geopolitical	Petroleum Production	`STEO.PAPR_NONOPEC.M`	Petroleum production in non-OPEC countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	50-60 MMb/d
	Petroleum Production	`STEO.PAPR_OPEC.M`	Petroleum production in OPEC countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	30-40 MMb/d
	Crude Oil Production	`STEO.COPS_OPEC.M`	Crude oil production in OPEC countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	25-35 MMb/d
	Crude Oil Production Capacity	`STEO.COPC_OPEC.M`	Crude oil production capacity in OPEC countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	30-40 MMb/d
	Strategic Reserves	`STEO.T3_STCHANGE_OOECD.M`	Strategic petroleum reserve changes in OECD countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels	±10 million
	Strategic Reserves	`STEO.T3_STCHANGE_NOECD.M`	Strategic petroleum reserve changes in non-OECD countries	Monthly	EIA	01/01/1995	31/12/2024	Million barrels	±5 million
	Geopolitical Risk	`GPR`	Geopolitical Risk Index	Monthly	EPU	01/01/1995	31/12/2024	Index	50-150
Microeconomic	Crude Oil Production	`STEO.COPRPUS.M`	Crude oil production in the United States	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	10-15 MMb/d
	Crude Oil Inputs	`STEO.CORIPUS.M`	Crude oil inputs to refineries in the United States	Monthly	EIA	01/01/1995	31/12/2024	Million barrels per day (MMb/d)	15-20 MMb/d
	Crude Oil Imports	`PET.MCRIMXX2.M`	Crude oil imports by country of origin	Monthly	EIA	01/01/1995	31/12/2024	Thousand barrels	200-300 thousand
	Capacity Utilization	`CAPG211S`	Capacity utilization in the oil and gas extraction industry	Monthly	FRED	01/01/1995	31/12/2024	Percentage (%)	70-90%
	Capacity Utilization	`CAPUTLG211S`	Capacity utilization in the oil and gas extraction industry (long-term)	Monthly	FRED	01/01/1995	31/12/2024	Percentage (%)	70-90%
	Industrial Production	`IPG211S`	Industrial production index for oil and gas extraction	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
	Industrial Production	`IPG211111CN`	Industrial production index for crude oil extraction	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
	Industrial Production	`IPN213111N`	Industrial production index for petroleum refineries	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
	Producer Price Index	`PCU211211`	Producer Price Index (PPI) for oil and gas extraction	Monthly	FRED	01/01/1995	31/12/2024	Index (Base = 100)	80-120
Financial	Crude Oil Prices	`WTISPLC`	West Texas Intermediate (WTI) crude oil spot price, a key financial benchmark	Monthly	EIA	01/01/1995	31/12/2024	USD per barrel	20-120 USD
	Petroleum Stocks	`STEO.PASC_OECD_T3.M`	Petroleum stocks in OECD countries, influencing oil prices	Monthly	EIA	01/01/1995	31/12/2024	Million barrels	2500-3000 million
	Market Indices	`SP500 returns`	S&P 500 Index, reflecting overall market performance	Monthly	Yahoo	01/01/1995	31/12/2024	Percentage (%)	-10% to 10%
	Forecasted Prices	`forecast_oil_price`	Forecasted oil price, used for financial planning and analysis	Monthly	EIA	01/01/1995	31/12/2024	USD per barrel	20-120 USD

4.1.3. Data cleaning

Extreme outliers were identified via the interquartile range (IQR) method with a strict threshold (boundaries: Q1 - 3×IQR and Q3 + 3×IQR), Figure 5. The approach revealed extreme outliers mostly explain by specific type of crisis:

- Geopolitical Risk Index (GPR): peak in 2001 and 2022 reflecting geopolitical tensions;
- Non-OECD Gross Domestic Product (Non-OECD.GDP.GROWTH); troughs in 2008 and 2020 marking recessions;
- Strategic Petroleum Reserve Changes (STEO.T3_STCHANGE_NOECD.M): COVID-19 related variations;
- Capacity utilization in the oil and gas extraction industry (long-term) (CAPUTLG211S) with post-crisis falls of 2008; These outliers, uncorrected, capture real events (crises, wars), considered essential for the analysis of oil shocks.

The outliers have not been corrected as analysis shows that these data are associated with major crises and reflect systemic risks that can impact the price of crude oil. We applied KNN imputation to the covariates for the imputation of missing values. The extreme outliers have not been corrected since these values seem to correspond to periods of crises or major phenomena.

4.2. Regime process and working of codes from hmms

4.2.1. Introduction to Regime Detection with Hidden Markov Models

Understanding the dynamics of financial time series requires the identification of distinct periods or ‘regimes’. Regimes are characterised by different statistical properties. For example, in the case of West Texas Intermediate (WTI) crude oil analysed in this work, its price exhibits periods of high volatility, with an upward trend interspersed with periods or phases of stagnation or decline. Hidden Markov Models (HMMs) offer a powerful framework for uncovering these latent patterns. HMMs are stochastic mathematical models that assume that observed data are generated by a sequence of hidden or unobservable underlying states. These models will learn both the characteristics of each state and the transition probabilities between states. This learning process enables us to understand the temporal evolution of the system. This part of the work details the process of applying an HMM to detect regime shifts in the price of WTI crude oil. The roles and reasoning behind each block of code used will be explained. The code base comes from the hmms package, which is used to run the HMM. This framework is particularly useful for identifying recurring patterns and behavioral changes in time series, detecting market phases such as bull or bear markets when modeling financial data, improving forecasts by considering the current regime, and managing risks through state-dependent risk assessment.

4.2.2. Code Overview and Data Preparation

We have created a python function that automates the entire process of regime detection. A specific code has also been provided for the graphical representation of regime changes. Thus the function will allow (load_and_detect_regimes) to automate the process of loading data, training an HMM and assigning regime labels to the crude oil price. In the following, the code will be broken down section by section while explaining the execution details.

The next section allows us to import the libraries needed for training the model. These are numpy for numerical calculations, pandas for data manipulation, random for generating random seeds, hmms for HMM-related functions, and matplotlib for plotting and visualizing regimes. We set the random state (default = 42) to ensure reproducible results when training the model.

```
import pandas as pd
import numpy as np
import hmms
import datetime as dt
import random

def load_and_detect_regimes(random_seed=42):
    """
    Load the dataset, process the data, train an HMM, and detect regimes.
    Returns the processed DataFrame with regimes for plotting.

    Parameters:
        random_seed (int): Random seed for reproducibility. Default is 42.
    """
    # Set random seeds for reproducibility
    np.random.seed(random_seed)
    random.seed(random_seed)
```

This code block is used to load the entire cleaned CSV database from the specified URL. Then the "date" column is forced as the index of the dataset after converting it to the correct datetime format. Using the datetime object is useful for time series calculations.

```
# Load dataset
    path =
"https://raw.githubusercontent.com/pefura/WQU_Project/refs/heads/main/sterilized_df_oil.csv"
    sterilized_df = pd.read_csv(path)

    # Set the 'date' column as the index
    sterilized_df['date'] = pd.to_datetime(sterilized_df['date'])
    sterilized_df.set_index('date', inplace=True)
```

4.2.3. Features extraction and emission sequence creation

This section extracts the “WTISPLC” data corresponding to the spot price of WTI crude oil, calculates the monthly price differences (diff), and then transforms these differences into a binary emission sequence. For this representation, the value 1 indicates a price increase while the value 0 represents a price decrease. This simplifies the training for the HMM by focusing on directional changes.

```
# Extract the 'WTISPLC' column and compute monthly differences
    wti_monthly = sterilized_df[['WTISPLC']].copy()
    wti_monthly['diff'] = wti_monthly['WTISPLC'].diff()

    # Create the emission sequence (1 for increase, 0 for decrease)
    emission_seq = np.array(wti_monthly['diff'][1:].apply(lambda x: 1 if x > 0 else 0).values)
```

The emission_seq is divided into segments of length 32 to or less before introducing to Baum-Welch algorithm to improve the robustness of the HMM model and reduce model overfitting. This is a common practice when dealing with long time series.

```
# Split the emission sequence into chunks of length 32 or less
    emission_seq_arr = np.array_split(emission_seq, 32)
```

4.2.4. HMM initialization, training, and state Prediction

We initialized an HMM with 3 hidden states representing the different crude oil price regimes (bullish, bearish, stagnant) and 2 possible symbols (increase or decrease in prices). The HMM model is trained by the baum_welch function using the Baum-Welch algorithm. This algorithm is also known as the expectation-maximization (EM) algorithm. It iteratively adjusts the model parameters to maximize the probability of the observed emission sequence. The model parameters are then printed to allow the model components to be appreciated, in particular the transition probabilities between states and the emission probabilities within each state.

```
# Initialize and train the HMM
    dhmm_wti = hmms.DtHMM.random(3, 2)  # 3 states, 2 symbols
    dhmm_wti.baum_welch(emission_seq_arr, 100)  # Train with 100 iterations

    # Print the learned parameters
    hmms.print_parameters(dhmm_wti)
```

The Viterbi algorithm is then applied to select the most probable hidden state sequence given the observed emission sequence. This is performed here by the "viterbi" function. This algorithm works by computing the hidden state path that maximizes the emission probability of the observed data. The log-likelihood of the most probable sequence (log_ptrob) and the predicted state sequence for the time series (state_seq) are obtained at the end of the Viterbi algorithm recursivity.

```
# Predict the most likely sequence of states using the Viterbi algorithm
    np.int = int  # Dependency fix
    log_prob, state_seq = dhmm_wti.viterbi(emission_seq)
```

4.2.5. Regime mapping and function output

The last section of the function that encodes the HMM adds the regime prediction to the crude oil price dataframe (wti_monthly). A remapping based on the average price differences in each regime is applied to the regimes to obtain more meaningful labels. The first row of the final result that contained a missing value due to the calculation of monthly price differences is removed. The final result of the function is a processed dataframe ready for further analysis or for visualization of the different regimes.

```
# Add the predicted regimes to the DataFrame
    wti_monthly = wti_monthly.iloc[1:]  # Remove the first row (NaN due to diff)
    wti_monthly['regime'] = state_seq

    # Map regimes based on mean price differences
    diff_mean = wti_monthly.groupby('regime')['diff'].mean()
    init_regime = diff_mean.index.tolist()
    ordered_regime = diff_mean.sort_values().index.tolist()
    mapped_regimes = dict(zip(ordered_regime, init_regime))
    wti_monthly['regime'] = wti_monthly['regime'].map(mapped_regimes)

    # Return the processed DataFrame for plotting
    return wti_monthly
```

The results are displayed by calling the function and indicating the chosen random seed.

```
# Call the function with a fixed random seed
wti_monthly = load_and_detect_regimes(random_seed=10)

# Display the tail of the DataFrame
print(wti_monthly.tail())
```

Here are the results obtained with a random state of 10.

```
Initial probabilities (π) :
          0
0  0.338726
1  0.294118
2  0.367156
Transition probabilities matrix (A):
          0         1         2
0  0.541537  0.018179  0.440285
1  0.572781  0.183826  0.243393
2  0.218531  0.739396  0.042073
Emission probabilities matrix (B):
          0         1
0  0.054984  0.945016
1  0.994690  0.005310
2  0.407086  0.592914
            WTISPLC  diff  regime
date
2024-08-01    76.68 -5.12       1
2024-09-01    70.24 -6.44       0
2024-10-01    71.99  1.75       1
2024-11-01    69.95 -2.04       0
2024-12-01    70.12  0.17       2
```

4.2.6. Visualization of regimes

A visualization code for crude oil price regimes detected by the HMM is finally provided. The regimes are superimposed on historical price data. The regimes are indicated by segments colored with red for a bear market, blue for a bull market and green for a stagnant market. Using the LineCollection module allows to color each segment according to the dominant regime. The transitions between regimes are thus clearly visualized.

```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.patches as mpatches
from matplotlib.collections import LineCollection
from matplotlib.colors import Colormap, ListedColormap, BoundaryNorm
import seaborn as sns# Create the plot
fig, ax1 = plt.subplots(figsize=(15, 5))
# Plot the raw data
wti_monthly.index = pd.to_datetime(wti_monthly.index)
ax1.plot(wti_monthly.index, wti_monthly['WTISPLC'], color='black', alpha=0.5, label='WTISPLC')
# Make 0 (Bear) - red, 1 (Stagnant) - blue, 2 (Bull) - green
cmap = ListedColormap(['r', 'b', 'g'])
norm = BoundaryNorm(range(4), cmap.N)  # Use range(4) to include 3 regimes (0, 1, 2)
# Convert dates to numerical format for plotting
inxval = mdates.date2num(wti_monthly.index)
# Create segments for LineCollection
points = np.array([inxval, wti_monthly['WTISPLC']]).T.reshape(-1, 1, 2)
segments = np.concatenate([points[:-1], points[1:]], axis=1)
# Create LineCollection and set colors based on 'regime'
lc = LineCollection(segments, cmap=cmap, norm=norm)
lc.set_array(wti_monthly['regime'])
plt.gca().add_collection(lc)
# Set plot limits
plt.xlim(wti_monthly.index.min(), wti_monthly.index.max())
plt.ylim(wti_monthly['WTISPLC'].min(), wti_monthly['WTISPLC'].max())

# Add legend
r_patch = mpatches.Patch(color='red', label='Bear')
b_patch = mpatches.Patch(color='blue', label='Stagnant')
g_patch = mpatches.Patch(color='green', label='Bull')
plt.legend(handles=[r_patch, b_patch, g_patch])
# Add title (bold and left-aligned)
plt.title('Figure 1: Detection of price regimes in the crude oil Market', fontweight='bold', loc='left')
# Add grid
plt.grid()

# Show plot
plt.show()
```

4.3. Training and validating Bayesian networks for financial regime prediction

4.3.1. Introduction to Bayesian Network Learning for financial time series

Bayesian Networks (BNs) are powerful probabilistic graphical models that represent dependencies between a set of variables. BNs are used in financial time series to model relationships between different financial indicators in order to predict future market behavior. To train a BN, we need two steps:

- Structure learning with identification of the network's graphical structure
- Parameter learning with estimation of conditional probability distributions. In this section we will explain a relevant training approach of BN using the p library.

4.3.2. Core Functionalities: Set seeds and data preparation

The initial part of the code imports the necessary libraries and defines the set_seeds function for reproducibility. The pgmpy modules are used for training and BN estimation.

```
import numpy as np
import pandas as pd
from pgmpy.estimators import HillClimbSearch, K2Score
from pgmpy.models import BayesianModel
from pgmpy.estimators import BayesianEstimator
from tqdm import tqdm
import hmms
import random

# Set random seeds for reproducibility
def set_seeds(random_seed=42):
    np.random.seed(random_seed)
    random.seed(random_seed)
```

4.3.3. Data splitting

This function splits the dataset into three datasets: training, validation, and test based on the chosen ratios. The training data is used to learn the structure and parameters of the Bayesian network. The test data is used to verify the correct functioning of the model.

```
# 1. Data Preparation and Splitting
def split_data(df, train_ratio=0.8, valid_ratio=0.1):
    """
    Split dataset into train, validation, and test sets
    Returns: (train_df, valid_df, test_df)
    """
    total_size = len(df)
    train_size = int(total_size * train_ratio)
    valid_size = int(total_size * valid_ratio)

    train_df = df.iloc[:train_size].copy()
    valid_df = df.iloc[train_size:train_size+valid_size].copy()
    test_df = df.iloc[train_size+valid_size:].copy()

    return train_df, valid_df, test_df
```

4.3.4. Time series discretization

The application of Bayesian networks often requires the discretization of continuous time series data. This function uses Hidden Markov Models (HMMs) to convert the continuous time series into discrete states. The algorithms used for HMM are described above.

```python
def discretize_time_series(train_df, ts_names, n_states=3, n_symbols=2, bw_iterations=100)
    """
    Convert continuous time series to discrete states using HMM
    Returns: (discretized_df, hmm_models)
    """
    discretized_df = pd.DataFrame(index=train_df.index[1:])  # Skip first NaN
    hmm_store = {}

    for ts_name in tqdm(ts_names, desc="Discretizing series"):
        # Calculate price changes
        price_diff = train_df[ts_name].diff().dropna()

        # Create emission sequence (1=price increase, 0=decrease)
        emissions = np.where(price_diff > 0, 1, 0)

        # Initialize and train HMM
        hmm = hmms.DtHMM.random(n_states, n_symbols)
        hmm.baum_welch([emissions], bw_iterations)  # Train on single sequence

        # Get most likely state sequence
        _, states = hmm.viterbi(emissions)
        discretized_df[ts_name] = states

        # Store trained HMM
        hmm_store[ts_name] = hmm

    return discretized_df, hmm_store
```

4.3.5. Bayesian Network structure learning

We now have the discretized data and can start training the Bayesian network structure. We use the learn_bayesian_network function which will apply a Hill Climb search algorithm with the K2 score to determine the most likely network structure. This heuristic algorithm will find the maximum value of the likelihood function.

First, a basic Bayesian model is initialized with the provided list of time series and initial nodes (initial_edges). The initial edges can be used to provide domain knowledge. The estimator and scorer are then configured as variables. Finally, the estimator performs function training. Parameters are then learned to better improve predictions. The parameters are learned as follows:

- Maximum Likelihood Estimator (MLE): the parameters that give the model the highest probability of seeing the original data.
- Bayesian Estimator - for regularization to the training function and to provide prior knowledge to the model.

```python
# 3. Bayesian Network Learning with Hill Climbing
def learn_bayesian_network(discretized_df, ts_names_filtered, initial_edges):
    """
    Learn Bayesian network structure using hill climbing
    Returns: trained Bayesian model
    """
    # Initialize model with domain knowledge
    model = BayesianModel()
    model.add_nodes_from(ts_names_filtered)
    model.add_edges_from(initial_edges)
    # Configure search algorithm
    hc = HillClimbSearch(discretized_df)
    score = K2Score(discretized_df)
    # Perform structure learning
    optimized_model = hc.estimate(
        scoring_method=score,
        start_dag=model,
        max_indegree=4,
        max_iter=100
    )
    # Learn parameters
    optimized_model.fit(
        discretized_df,
        estimator=BayesianEstimator,
        prior_type="K2",
        state_names={col: [0,1,2] for col in ts_names_filtered}
    )
    return optimized_model
```

4.3.6. Cross-validation and parameter testing

Implementing a comprehensive cross-validation strategy with backward chaining and other parameter testing techniques is a challenge in Bayesian networks. It requires careful adaptation to the nature of financial time series data.

The code applied for verification uses VariableElimination to test whether the model correctly predicts values, based on information already available. Model validation involves comparing predictions with actual values. The goal is to create a robust model by combining network structure learning, parameter estimation, and cross-validation.

```
from pgmpy.inference import VariableElimination

# Using VariableElimination for inference
infer = VariableElimination(optimized_model)

# Provide evidence and perform inference
evidence = {'SP500': 0, 'WTISPLC': 2}  # Example evidence values
result = infer.query(variables=['Crude Oil Production'], evidence=evidence)

print(result)
```

4.3.7. Graphing BNs

The network can be plotted using nteworkx package.

```
import networkx as nx
import pylab as plt
plt.figure(figsize=(12,8))

# Initialize graph
network_graph = nx.Graph()
network_graph.add_edges_from(bayesian_net.edges())

# Draw edges and nodes
pos = nx.spring_layout(network_graph)
nx.draw_networkx_nodes(network_graph, pos, node_size = 50)
nx.draw_networkx_edges(network_graph, pos, arrowstyle='-|>', arrows=True, arrowsize=10, edge_color='black')

# Add vertex labels
label_pos = {node: (x + 0.1, y + 0.05) for node, (x, y) in pos.items()}
nx.draw_networkx_labels(network_graph, label_pos, font_size=8, font_color='black')

# Display edge weights
edge_labels = {edge: f"{K2Score(train_discretized).local_score(edge[1], parents=[edge[0]]):.2f}" for edge in network_graph.edges()}
nx.draw_networkx_edge_labels(network_graph, pos, edge_labels=edge_labels, font_size=6)

plt.show()
```

Step 5 : Combination of sections from step 4 done

Step 6: Regime detection in full dataset

6.1. Time series transformation

In order to build Hidden Markov Models for detecting regimes in time-series data, we need to transform the time-series data into output emissions. Given the monthly crude oil price data, we could extract the price difference between consecutive months information and discretize it into two possibilities; 1 represents the positive price difference (increasing) from the previous month, and 0 otherwise. The following steps will be performed to transform the time-series data into the emission sequence:

Step 1: Calculate the price difference from the previous month for each data point at time t

$$price\_diff_t = price_t - price_{t-1}, \quad \text{for } t = 1, \ldots, T$$

Make sure to exclude the first observation since the price difference value only can be calculated from the second observation onwards. The size of the transformed time series should be equal to $T - 1$.
Step 2: Discretize the price difference into the desired output emissions, where 1 indicates an increase in the price from the previous month, and 0 otherwise.
If $price\_diff_t > 0$, then $emission\_sequence_t = 1$,
else $emission\_sequence_t = 0$

Step 3: Given the constraint of the hmms.DtHMM model can not take an array with length longer than 32, we need to split out the array in arrays each of length 32 or less.
$emission\_sequence = array\_split(emission\_sequence,\ split\ size[32])$

6.2. Parameters learning

The Baum-Welch algorithm will be used to learn the parameters of the HMM. Firstly, we will create and initialize a model with random parameters, with three predefined states (bull, bear, or stagnant), and two possible symbols or observations (increase or decrease). Next, we will feed the prepared emission sequence from the data transformation step to train this model. The outcome of this process is the following estimated parameters (λ):

1. Initial probabilities of hidden states (π)
2. $N\ x\ N$ Transition probabilities matrix (from $state_i$ to $state_j$)
3. $N\ x\ M$ State-Emission probabilities matrix (from $state_i$ to $emission_j$)

   Outcome: model λ(π, a, b)

6.3. Generating sequence of hidden states

Now we already have all of the required ingredients: the emitted observation sequence from time-series data transformation step and estimated parameters () using Baum-Welch algorithm. We have enough information to run the Viterbi algorithm to identify and generate the most likely state-transition path (in this case, the regimes) through the selected time-series. The outcome of this step is a state sequence with size equal to the original observation that represents the underlying regimes(i.e. bear, bull, and stagnant). Outcome: state sequences = {s1, s2, ... , sT}

Step 7: Identification of the latent meaning behind each hidden state(Vanluyten et al.; Zhang et al.)

Since there is no information about which hidden states represent the corresponding regime, we need to interpret the meaning of these states and assign them to the correct regimes. In this case, we are using simple statistical measures such as the average of the price difference to summarize the state characteristics. The bear regime indicates the expected price difference tends to decrease (negative value), the bull regime indicates the expected price difference tends to increase (positive value), and the stagnant regime has the expected price difference close to zero. Using these properties, we can assign

each state to the correct regime. The following steps are performed to recognize and analyze the meaning of the hidden states:

Step 1: assign the most-likely state sequences generated using the Viterbi algorithm as the state series to the transformed time series.

Step 2: calculate the average of the price_difference for each hidden state.

Step 3: sort the step 2 result

Step 4: infer and assign the hidden state to the corresponding regime label:

- Bear regime: The state with the lowest average of price_difference(most likely in negative values).
- Stagnant regime: The state with the average of price_differenceclose to zero (not the lowest nor the largest).
- Bull regime: The state with the largest average o f price_difference(most likely in positive values).

Step 8: Reading the corresponding paper section for Hill Climb

- Section read.

Step 9: Implementation of the minimal working example of a Hill Climb search and identification of the latent meaning behind each hidden state

9.1. Implementation of the minimal working example of a Hill Climb search

Figure 2: Visualization of learned Bayesian Network structure and edge scores

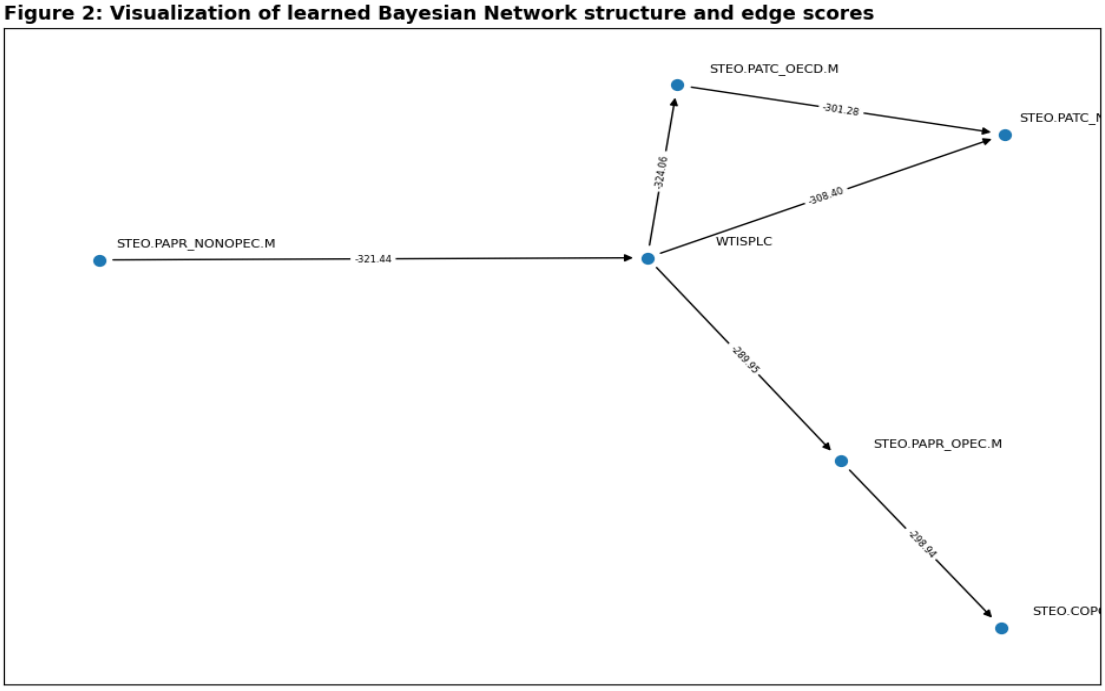


Figure 2 illustrates a trained Bayesian network that shows probabilistic relationships relevant to crude oil price analysis. The nodes represent variables such as the WTI spot price ("WTISPLC") and production data from OPEC and non-OECD countries. Arrows indicate dependencies, with numerical values representing the strength of each relationship as measured by the K2 score. Key relationships highlight the influence of production data from various sources (OPEC and non-OPEC) on the WTISPLC

9.2. Identification of the latent meaning behind each hidden state

Table 2: HMM State Descriptions

Time Series	State	Mean Price Change	Description
STEO.PAPR_NONOPEC.M	0	-0.326	Decrease in Non-OPEC production; low variation.
STEO.PAPR_NONOPEC.M	1	0.228	Increase in Non-OPEC production; moderately volatile.
STEO.PAPR_NONOPEC.M	2	0.236	Similar to state 1; most occurrences.
STEO.PAPR_OPEC.M	0	-0.304	OPEC production reduction.
STEO.PAPR_OPEC.M	1	0.308	Increase in OPEC production; highly volatile.
STEO.PAPR_OPEC.M	2	-0.338	Production reduction; few occurrences.
WTISPLC	0	2.04	Large increase in WTI price; high variability.
WTISPLC	1	-3.22	Large decrease in WTI price; high variability.
WTISPLC	2	2.62	Increase in WTI price.

The hidden states of our HMM model are summarized in Table 2. State 0 indicates a decline, state 1 reflects price increases, and state 2 represents stagnation. Non-OPEC production is more stable than OPEC production, and WTI prices exhibit high variability in both increases and decreases.
Insights

- Non-OPEC: State 1 exhibits the most volatile increases, while state 2 remains relatively stable;
- OPEC: State 1 exhibits large increases, with state 2 representing rare and large reductions;
- WTI prices: The high variability of state 0 (large increases) and state 1 (large decreases) highlights the sensitivity of the market.

References

Bank, European Central. "European Central Bank." *European Central Bank*, https://www.ecb.europa.eu/home/html/index.en.html. Accessed 10 Feb. 2025.

Bhattacharya, Chandrachur, and Asok Ray. "Data-Driven Detection and Classification of Regimes in Chaotic Systems Via Hidden Markov Modeling." *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 021009, Aug. 2020. *Silverchair*, https://doi.org/10.1115/1.4047817.

Economic Policy Uncertainty Index. https://www.policyuncertainty.com/. Accessed 10 Feb. 2025.

Eddy, Sean R. "Hidden Markov Models." *Current Opinion in Structural Biology*, vol. 6, no. 3, June 1996, pp. 361–65. *ScienceDirect*, https://doi.org/10.1016/S0959-440X(96)80056-X.

Eddy, Sean R. "What Is a Hidden Markov Model?" *Nature Biotechnology*, vol. 22, no. 10, Oct. 2004, pp. 1315–16. *www.nature.com*, https://doi.org/10.1038/nbt1004-1315.

Federal Reserve Economic Data | FRED | St. Louis Fed. https://fred.stlouisfed.org/. Accessed 10 Feb. 2025.

Homepage - U.S. Energy Information Administration (EIA). https://www.eia.gov/index.php. Accessed 10 Feb. 2025.

Rabiner, L., and B. Juang. "An Introduction to Hidden Markov Models." *IEEE ASSP Magazine*, vol. 3, no. 1, Jan. 1986, pp. 4–16. *IEEE Xplore*, https://doi.org/10.1109/MASSP.1986.1165342.

Vanluyten, Bart, et al. "A New Approach for the Identification of Hidden Markov Models." *2007 46th IEEE Conference on Decision and Control*, 2007, pp. 4901–05. *IEEE Xplore*, https://doi.org/10.1109/CDC.2007.4434912.

Wang, Matthew, et al. "Regime-Switching Factor Investing with Hidden Markov Models." *Journal of Risk and Financial Management*, vol. 13, no. 12, 12, Dec. 2020, p. 311. *www.mdpi.com*, https://doi.org/10.3390/jrfm13120311.

World Bank Open Data | Data. https://data.worldbank.org/. Accessed 10 Feb. 2025.

Yahoo Finance - Stock Market Live, Quotes, Business & Finance News. https://finance.yahoo.com/. Accessed 22 Aug. 2024.

Zhang, Deyi, et al. "Bayesian Identification of Hidden Markov Models and Their Use for Condition-Based Monitoring." *IEEE Transactions on Reliability*, vol. 65, no. 3, Sept. 2016, pp. 1471–82. *IEEE Xplore*, https://doi.org/10.1109/TR.2016.2570561.

Zheng, Kai, et al. "Regime Switching Model Estimation: Spectral Clustering Hidden Markov Model." *Annals of Operations Research*, vol. 303, no. 1, Aug. 2021, pp. 297–319. *Springer Link*, https://doi.org/10.1007/s10479-019-03140-2.