GROUP WORK PROJECT # 2 MScFE 660: RISK MANAGEMENT
Group Number: 5638

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Shailza Virmani	India	virmanishailza@gmail.com	
Zhe Zhang	Hong Kong	zhezhangcs@gmail.com	
Anubhav Mishra	India	anubhav0by0@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	**Shailza Virmani**
Team member 2	**Zhe Zhang**
Team member 3	**Anubhav Mishra**

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.
N/A

GROUP WORK PROJECT # 2 MScFE 660: RISK MANAGEMENT
Group Number: 5638

Step 1

The Forward Algorithm

Input:

- O: Partially observed sequence of length T
- λ: Model parameters (transition probabilities, emission probabilities, initial state distribution)

Output:

- α: Forward variables

Procedure ForwardAlgorithm(O, λ): N <- number of hidden states T <- length of the observed sequence O

```
Initialize α matrix of size (T, N) with zeros

// Base case: Compute α1(i)
for i = 1 to N do:
    α[1, i] <- λ.pi[i] * λ.b[i, O[1]]

// General case: Compute αt+1(i)
for t = 1 to T-1 do:
    for i = 1 to N do:
        sum <- 0
        for j = 1 to N do:
            sum <- sum + α[t, j] * λ.a[j, i]
        α[t+1, i] <- λ.b[i, O[t+1]] * sum

return α
```

Input:

observed_sequence = [0, 1, 0]

initial_state_distribution = [0.6, 0.4]

transition_matrix = [[0.7, 0.3], [0.4, 0.6]]

emission_matrix = [[0.1, 0.9], [0.5, 0.5]]

Output (alpha):

[0.06, 0.2]

[0.1098, 0.069]

[0.010446, 0.03717]

The Backward Algorithm

Input:

- O: Partially observed sequence of length T
- λ: Model parameters (transition probabilities, emission probabilities, initial state distribution)

Output:

- β: Backward variables

Procedure BackwardAlgorithm(O, λ): N <- number of hidden states T <- length of the observed sequence O

```
Initialize β matrix of size (T, N) with zeros

// Base case: Set βT(i) = 1 for all i
for i = 1 to N do:
    β[T, i] <- 1

// General case: Compute βt(i)
for t = T-1 downto 1 do:
    for i = 1 to N do:
        sum <- 0
        for j = 1 to N do:
            sum <- sum + λ.a[i, j] * λ.b[j, O[t+1]] * β[t+1, j]
        β[t, i] <- sum

return β
```

Input:

observed_sequence = [0, 1, 0]

initial_state_distribution = [0.6, 0.4]

transition_matrix = [[0.7, 0.3], [0.4, 0.6]]

emission_matrix = [[0.1, 0.9], [0.5, 0.5]]

Output (beta):

[0.19, 0.1812]

[0.22, 0.34]

[1, 1]

Backward Viterbi Algorithm

```
Function to perform the backward Viterbi algorithm
function backward_viterbi(obs, states, start_prob, trans_prob, emit_prob):
T = length of obs
N = length of states

Create a 2D array beta with dimensions T x N

for each state i from 0 to N-1:
    Set beta[T-1][i] to 1

for each time step t from T-2 down to 0:
    for each state i from 0 to N-1:
        sum_beta = 0
        for each state j from 0 to N-1:
            sum_beta = sum_beta + beta[t+1][j] * trans_prob[i][j] * emit_prob[j][obs[t+1]]
        Set beta[t][i] to sum_beta

prob_obs_given_model = 0
for each state i from 0 to N-1:
    prob_obs_given_model = prob_obs_given_model + start_prob[i] * emit_prob[i][obs[0]] * beta[0][i]

return beta, prob_obs_given_model
```

Input:

obs = [0, 1, 0]

states = [0, 1]

start_prob = [0.5, 0.5]

trans_prob = [[0.7, 0.3], [0.4, 0.6]]

emit_prob = [[0.5, 0.5], [0.8, 0.2]]

Output:

Backward Matrix:

[0.2473, 0.1996]

[0.59, 0.679999999999]

[1, 1]

The Baum-Welch algorithm

The Baum Welch algorithm is utilized to determine the ideal parameters of the model so that we end up maximizing the probability of generating an observed sequence of emissions. O = O_1O_2…. OT i.e. in determining **$\lambda^* = \text{argmax}_{\lambda} P(O|\lambda)$**

Thus, the idea is to derive the value of A, the state transition matrix and B, the emission matrix. The Baum-welch algorithm is a special case of the Expectation Maximization(EM) algorithm [4] where we compute the expected state occupancy count Υ and the expected state transition count δ from the earlier A and B probabilities. In the M-step, we use Υ and δ to recompute the A and B probabilities.

Base state

State transition matrix A = a_{11}, …., a_{ij}, a_{NN}

Emission matrix B = $b_i(O_t)$

Iteration

α is the forward probability obtained in the Forward Algorithm

β is the backward probability obtained via the backward algorithm

E-step:

- $\Upsilon_t(i) = \Sigma_{j=1}^{N}\xi_t(i,j)$
- $\Upsilon_t = \alpha_t(i)\beta_t(i)/\Sigma_{j=0}^{N}\alpha_t(j)\beta_t(j)$

M-step:

- $\pi_i = \Upsilon_1(i)$
- $a_{ij} = \Sigma^{T-1}_{t=1}\xi_t(I, j)/\Sigma^{N}_{j=1}\Sigma^{T-1}_{t=1}\xi_t(i, j)$
- $b_i(v_k) = \Sigma^{T}_{t=1, Ot = Vk}\Upsilon_t(i)/\Sigma^{T}_{t=1}\Upsilon_t(i)$

Step 2

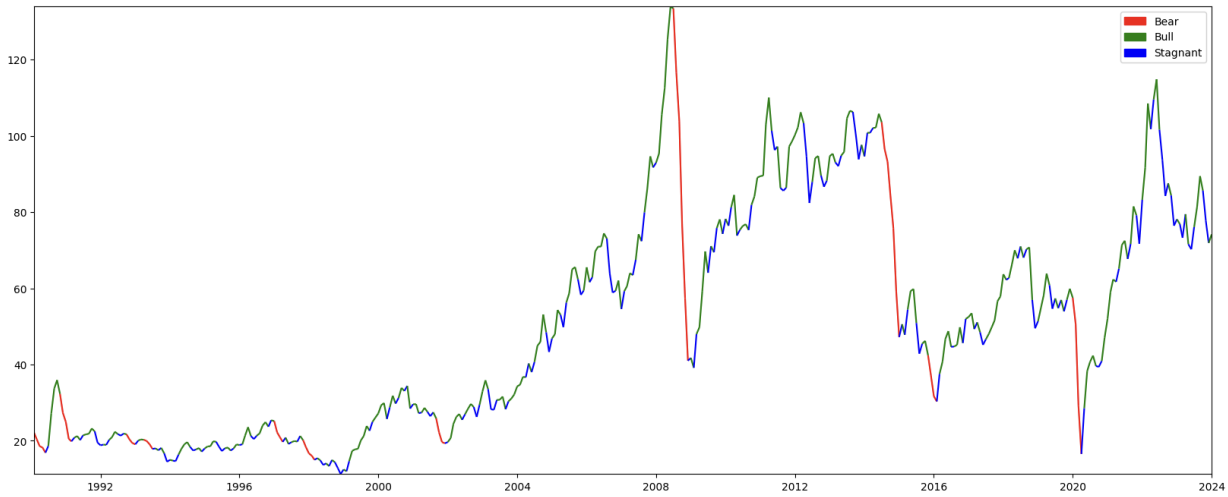

Figure: Identify the bull, bear, and stagnant regimes

Bull regimes

There was a bull regime in 2007 where the crude oil price doubled in a year, possibly because

- Ongoing conflicts in Iraq and tensions with Iran heightened market uncertainty, pushing prices upward.
- Rapid industrialization and increased consumption drove the need for energy, putting upward pressure on prices.
- Financial markets witnessed significant speculative activity in oil futures contracts during this period. Investors, including hedge funds and institutional traders, entered the oil market, seeking to profit from rising prices. This speculative activity created additional upward pressure on oil prices, amplifying the bull regime.

Bear regimes

Here are the two examples:

2008 Financial Crisis:

- The situation of the financial markets and the price of crude oil decline simultaneously.

- It led to a decrease in the world's demand for oil, an increase in market turbulence, and a general downturn in economic activity. Resulting in a price drop.
- Speculative trading, surplus supply and changes in the value of the US dollar all affected the price of the oil

COVID-19 Pandemic(2020):

- It was the biggest collapse in oil price history due to a drop in prices because of a pandemic.
- The travel restrictions imposed during lockdown resulted in decrease in oil demand and further affecting oil prices.
- Oil prices also got worse due to the price war between Saudi Arabia and Russia which resulted in dropping some futures contracts to negative in April 2020.

Stagnant regimes

From 1992 to 1996, the price of oil is stable for a long time. Possible reasons include

- During early 1990s, the places where oil is produced didn't have many conflicts or problems during that time.
- Countries in OPEC agreed to produce less oil so that there wouldn't be too much of it. They wanted to make sure the price stayed the same.
- The United States also started producing more of its own oil during that time. This helped balance out the effects of OPEC's production cuts and kept the price of oil fairly steady.

Step 3

Definition of Hidden Markov model

The Hidden Markov Model (HMM) is a model that represents a system that contains hidden states where the system evolves over time while the observers can only view the final emission/output of the model and cannot see the states [1]. HMM depicts the characteristics of Markov chains and the future state of a process is dependent only on the current state and not a sequence of steps.

Hidden Markov Models contain the following components:

- States: which represent the possible conditions of the system that are not visible externally
- Observations: The outcomes that are visible and are inherently dependent on the hidden states
- Transition probabilities: They are 2-dimensional matrices that capture the probability of movement from one state to the other
- Emission probabilities: The emission or observation matrix captures the probability of an observation being generated from a state
- Stationary state: indicates the likelihood of the system starting in a particular state

Operation of the HMM:

- Evaluation: With the observed sequence of data and the model parameters, we compute the probability of the observed sequence which is solved by the Forward-Backward algorithm
- Decoding: With the observed sequence and the model parameters, we determine the most likely sequence of hidden states (performed by the Viterbi algorithm)
- Learning: With the observed sequence and the number of states in the model, we estimate the model parameters (A and B) which are the transition and emission matrices leveraging the Baum-Welch algorithm which is a special case of Expectation Maximization algorithm.

Step 4-5

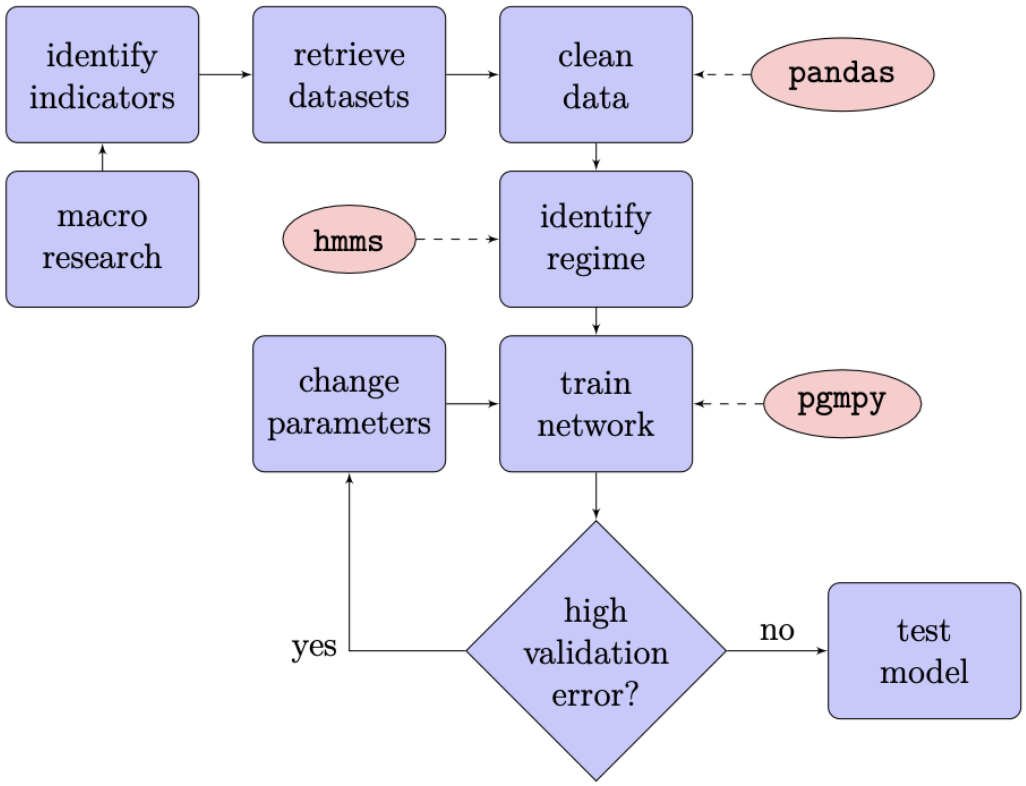

Figure: An illustration of the design process for the Crude Oil Trading model

Macro research

Macro research of oil trading involves the interpretation and prediction of large-scale events related to geopolitics, monetary policy, privatizations of national petroleum and gas companies, supply and demand factors, and natural disasters.

Macro factors include oil production, consumption, inventories and geopolitical events. OPEC plays a pivotal role in the stability of oil markets, with its member countries actively seeking to control oil production through production targets, while non-OPEC producers such as the companies in the United States make independent decisions about oil production and usually play the role of price taker. Both OECD and non-OECD countries can affect the oil demand. Structural conditions in OECD economies, such as larger technology and services sectors, may result in different relationships between oil prices, economic growth, and oil consumption compared to non-OECD countries. Developing countries tend to

have a greater proportion of their economies in energy-intensive manufacturing industries, and rising incomes and vehicle ownership contribute to increased oil consumption. Besides, Inventories act as a balancing point between supply and demand, with crude oil and petroleum products stored during periods of over-production or used during periods of over-consumption. Oil inventories can also be regarded as a measure of uncertainty in supply and demand.

Other macro-related factors such as the Federal Reserve Bank's monetary policy, macroeconomic atmosphere, industrial growth, and industrial production can also influence the price of crude oil.

Indicator identification

Based on the macro research results, several key factors can be considered as follows.

- Macro Factors, include GDP growth rates, inflation, interest rates, fiscal policies, and geopolitical events. Changes in economic growth rates can influence oil demand. Inflation and interest rates affect production costs in the oil industry and the oil demand. Geopolitical events, such as conflicts in the Mideast area can disrupt the oil supply.
- Supply and Demand Factors, include non-OPEC and OPEC crude oil production, oil consumption, OECD inventories, and oil trading activity in financial markets. Besides, changes in production levels and consumption patterns can provide insights into market imbalances and potential price movements.
- Energy Market Indicators, include renewable energy penetration, natural gas prices, coal consumption trends, and nuclear energy production.
- Financial Indicators associated with oil trading and market sentiment, such as oil futures prices, trading volumes, open interest, and volatility. Usually, when there is a large event related to oil price, the trading volume will be large and the intra-day volatility will be much higher.

Dataset retrieval

The Energy Information Administration (EIA) and the Federal Reserve Economic Data (FRED) are two major sources that we can rely on for comprehensive analysis of energy markets

- The EIA provides extensive data related to the energy markets including our target, crude oil market. The datasets, such as the International Energy Data, and the Short-Term Energy Outlook,

offer essential information on the physical market factors, the latest news, reports, and forecasts. These datasets include data on crude oil production, consumption, inventories, and projections for future trends. By accessing and analyzing these datasets, we can gain valuable insights into supply and demand dynamics, storage levels, and the overall outlook for the oil market.

- FRED serves as a primary source for macroeconomic data, which offers an extensive collection of economic data, including indicators related to GDP, M2, inflation, employment ratio, and interest rates. We can access FRED's database online or obtain the data through Python API. For example, we can access datasets on industrial production for all kinds of goods, capacity utilization of oil products, and manufacturing activity to assess the overall health of the economy and anticipate changes in oil demand. Additionally, FRED provides data on monetary policy indicators such as long-term rates and short-term rates, that can influence the cost of capital and investment decisions in the oil industry.

By combining the datasets from EIA and FRED, we can integrate macroeconomic variables with oil market analysis and do comprehensive assessments of the factors driving oil market dynamics. These datasets enable us to identify trends, correlations, and causal relationships and make predictions and strategic decisions in the oil market.

Data cleaning

Data cleaning involves addressing various data inconsistencies and preparing the data for analysis. Data obtained from sources like the EIA and FRED often contain missing values (possibly due to the release problems in certain years), inconsistent date ranges and indices formatted as strings instead of datetime. To tackle these issues, pandas, an open-source Python library, can be utilized. Pandas provides fast, flexible, and expressive data structures, such as dataframes, and offers functions to handle missing values, ensure consistent date ranges, convert indices to the appropriate format (e.g., datetime), and merge data with congruent frequencies. By leveraging pandas, we can clean the data and create training, validation, and testing datasets that could be effectively used with pgmpy for further analysis.

Regime process

Regime analysis is a vital tool to analyze varying states or conditions of the financial markets over time. Regimes represent different market conditions with unique trends, sentiments and behaviors. These include a range of situations, from bullish phases of upward price momentum and investor optimism to bearish periods marked by downward trends and negative sentiments. Additionally, regimes may include phases of stability or consolidation, during which prices show minimal changes and market players choose a cautious stance.

Comprehending diverse market regimes is crucial for conducting efficient market dynamics analysis and arriving at well-informed investing choices. Investors who comprehend the present framework might modify their strategy to capitalize on opportunities and reduce dangers. Investors may employ aggressive tactics to take advantage of rising trends in bullish regimes, while defensive tactics like hedging or limiting exposure may be prudent in bearish regimes to guard against possible losses.

In risk assessment and portfolio management, regime analysis is equally crucial. Through the incorporation of regime-based signals and indicators into risk models, investors can more accurately evaluate the probability of unfavorable market fluctuations and modify their portfolio allocations accordingly. Studying the regime also makes it easier to create dynamic asset allocation plans that adapt to shifting market conditions and ultimately increase portfolio returns and strength.

Identification and Classification:

The capacity to recognize and classify financial market regimes is necessary for comprehending market movements and making informed investment decisions. The phases in the procedure include identifying patterns and trends in the market data and grouping them into various regimes based on significant attributes like volatility, trend direction, and trend pattern.

- **Process of Identifying and classifying regimes**

  It starts with analyzing past market information, such as price movements, trading volumes, and other relevant indicators. Statistical techniques, algorithms in machine learning, and econometric models are often used to detect patterns and shifts between various market conditions. Different quantitative techniques [such as clustering algorithms, regime-switching models, and Hidden Markov Models (HMMs)] are used to categorize market data into distinct regimes.

- **Characteristics of Market Regimes:**
    - **Volatility:**

      It displays the size and regularity of price changes. While low volatility regimes are linked to consistent and calm price movements, high volatility regimes are linked to sudden and erratic market fluctuations.
    - **Trend Direction:**

      The direction of market developments within each regime is a significant additional factor. Upward price patterns driven by confidence and a favorable investor attitude characterize bullish periods. Conversely, negative price patterns, pessimism, and an adverse market outlook characterize bear markets.
    - **Trading Patterns:**

      Different trading behaviors and investor actions can help in categorizing regimes. Bullish periods typically feature higher trading volumes, more purchases and an expanding market range, showing broad-based market participation. On the other hand, during bearish regimes, trading volumes may decrease, selling pressure may increase, and market breadth may shrink, signaling investor caution and risk aversion.
- **Examples of Market Regimes:**
    - **Bull Regime:**

      The higher trend in prices can be attributed to favorable economic fundamentals, robust investor confidence, and optimism. Increasing trading volumes, rising stock values, and widespread participation across industries are characteristics of bull markets.
    - **Bear Regime:**

      Prices trend downwards, showing negativity, heightened uncertainty, and worsening economic conditions. Bear markets are defined by declining stock prices, increasing volatility, and a general push for selling by investors attempting to reduce losses.
    - **Stagnant Regime:**

      Prices show minimal fluctuation and lack defined trends. Low volatility, trading inside a defined range, and a lack of clear investor mood are characteristics of stagnant markets. These conditions are usually brought on by a lack of significant market drivers or contradictory economic data.

Role of Hidden Markov Models (HMMs):

HMMs are effective statistical instruments that are widely used in different sectors, such as finance, for modeling sequential data and comprehending underlying processes with hidden states. In the context of regime analysis, HMMs provide a complex structure for capturing and modeling the dynamic transitions between various market regimes over time.

- **Principles behind HMMs:**
    - **Hidden States:**
      Unobservable states of the system, which represent different market regimes. Every hidden state is associated with a probability distribution over observable outcomes.
    - **Observable Emissions:**
      These are the observable events or data points produced by the hidden states. Observations are assumed to be conditionally independent given the hidden states.
    - **Transitioning Probabilities:**
      These represent the probability of transitioning between hidden states over time. Transition probabilities are usually derived using historical data and show how regimes change over time.

In the context of analyzing crude oil market data, the Hidden Markov Model (HMM) algorithm is used to identify different regimes, such as bull, bear and stagnant. Here's an explanation of how the HMM algorithm is implemented for regime identification.

- **Model Initialization:**
    - The desired number of hidden states, each of which represents a distinct regime in the crude oil market, is initialized into the HMM model.
    - Based on the features of previous pricing data, initial probability for each state (π), state transition probabilities (A), and emission probabilities (B) are determined.
    - The code snippet "dhmm_r = hmms.DtHMM.random(3, 2)" initializes a discrete time HMM with three hidden states and two output variables.
- **Data Processing:**

- The code processes the input data, which is the time series data showing the price changes in the crude oil market.
- Price changes between consecutive months are calculated to determine symbols (emissions) for the HMM.
- The emission sequence uses 1 to indicate positive price changes, and 0 for negative or zero changes.

- **Training the Model:**
  - The HMM model is trained using historical price data with the Baum-Welch algorithm, which estimates model parameters based on observed price movements.
  - The model learns the hidden patterns and shifts from the data used for training.
- **Regime Detection:**
  - After the model has been trained, it can be applied to new data to determine the hidden states using the observed price movements.
  - By analyzing the sequence of hidden states, the algorithm is able to detect bull, bear, and stagnant trends within the crude oil market.
- **Specific Functions/Methods:**

Fit Method	used to train the HMM model about regime trends using previous pricing data.
Predict Method	uses observable price movements to predict the order of hidden states, or regimes.
Decode Method	based on the observed pricing data, decodes the most likely order of hidden states.

- **Model Evaluation:**
  - Assessing the effectiveness of the trained HMM on validation data to tune the model parameters.
  - The effectiveness of the regime detection process can be assessed by evaluating the final performance of the tuned model with the testing data.

	Explanation	**Code**
Import library	To import hmms library	python import hmms
Initialize HMM model	Initialize the HMM with the ideal states and symbols	python model = hmms.DtHMM(n_states=3, n_symbols=2)
Learning HMM parameters	Use Baum-Welch algorithm	python model.baum_welch(emissions, n_iter=100)
Hidden states	Use Viterbi algorithm	python states = model.viterbi(emissions)
Interpret Regimes	Analyze based on hidden states	python analyze_regimes(states)

Key Features of enabling regime identification:

- **Baum-Welch Algorithm:**
  The usage of the Baum-Welch algorithm in the code allows the autonomous acquisition of the HMM parameters from the crude oil price data, facilitating in regime identification.
- **State Transition Probabilities:**
  The HMM state transition probabilities model the shifts between the regimes, enabling the identification of the regime changes in the crude oil market.
- **Emission Sequence Analysis:**
  The code utilizes emission sequence obtained from price changes to probabilistically allocate different regimes to the data points, aiding in identifying regimes.

Network Training, Testing and Validation – PGMPY

Similar to any other machine learning model, we will be feeding the input data to the Bayesian model to allow for data-driven decision making. The input data is split into the training data, the validation data and the testing data. A brief about the different categories of data that is handled by the model is given below:

- Training Data - The data on which the model is trained and where the belief network is learnt using the Hill Climbing algorithm
- Validation Data - The dataset on which the parameters of the belief network are tuned. The tuning of the model is based on the score that is provided importance by the practitioner which could be different Bayesian Dirichlet functions like BDeu or K2 or the Bayesian Information Criterion (BIC). We merge the training and the validation dataset using the K-fold cross-validation technique which uses a leave one out cross-validation
- Testing data - The performance of the fully trained model is assessed using the testing data. Using this dataset we assess the error rate of the model post-validation. It should be noted that the model should not be tuned based on the performance of the test score which is a case of data leakage.

If we refer to the figure, we understand that the data needs to be discretized prior to the preprocessing step of splitting the data into the training, the validation and the test set for the application of the pgmpy package. Thus, the EIA and FRED data which are continuous in nature are discretized by identifying the hidden states which are in the form of regimes so that the belief networks can be identified. Thus, in a bullish regime, the market participants are expected to be in long exposure to the underlying and vice versa during the bearish regime. A regime in a time series of data indicates the helps us identify periods when the data depicts similar characteristics which could be in the form of bullish/bearish trend or high/low volatility. The periods or regimes are latent in nature and only the emissions are observed. Thus the identification can only be done by Hidden Markov Models. The HMM application for regime detection comprises of four stages:

1. Transforming - generating an emission sequence by first order integration and replacing the returns by parity of return

2. Learning - This step involves the identification of optimal parameters of the HMM using the Baum-Welch algorithm and thus identifying the initial state probabilities, the transition matrix and the emission matrix
3. Finding - In this step we find the most likely sequence of hidden states which is generated from the sequence of emission using the Viterbi algorithm
4. Identifying - The latent meaning behind each sequence number is identified by finding their corresponding arithmetic mean

For the understanding of the oil market we use a structure learning algorithm for constructing a causal network of the oil market which can be driven by either a constraint based structural learning algorithm or a score based structural learning algorithm. We use the latter using the Hill climbing algorithm which is a greedy iterative search algorithm used to maximize the network score and thus learn the structure of the oil market and predict the future. We start the hill climbing algorithm using a structure created by professional experts where we leverage the knowledge of demand and supply of the oil markets so that we do not get stuck at the local maxima and that the initial model would not vary significantly from the validation model. Thus we use the training data to estimate the model and fit the data to the estimator to learn the parameters.

We validate the model during the validation stage using a scoring methodology so that the errors are minimal and we continue tuning the model iteratively so that we are happy with the performance of the model on the validation set. We judge the network based on the quality of network structure and how linked it is and thus the Hill Climbing algorithm converges to a local maxima.

Step 6

Transform the time series

We transform the time series to identify the regime and model the underlying dynamics using Hidden Markov Models (HMMs) as follows.

1. Import libraries such as pandas, numpy, and hmms and Load the time series data.
2. Transform the time series data into an emission sequence that represents the changes or transitions in the data. Calculate the difference between consecutive observations in the time series and replace these differences with binary values. Assign a value of 1 if the change is positive (indicating an increase) and 0 if the change is negative (indicating a decrease).
3. Initialize the HMM, by specifying the number of hidden states and the number of possible symbols that can be emitted by each state. The number of hidden states is determined based on the complexity of the time series and the number of regimes we expect to identify.
4. Use the Baum-Welch algorithm to learn the parameters of the HMM that generates the observed sequence. The Baum-Welch algorithm is an iterative algorithm that updates the model's parameters to maximize the likelihood of the observed data.
5. To visualize the identified regimes, we create a data frame that combines the original time series data and the associated regime for each time point. Plot the multicolored time series graph to observe how well the regimes have been identified. Assign different colors to different regimes (e.g., bear, bull, and stagnant) to visually distinguish them.
6. Apply a similar process to discretize the entire training dataset. Iterate through all the variables in the dataset, learn the HMM parameters for each variable, and store them. Construct a discretized training dataframe that contains the discretized values for each variable.
7. To ensure the correct identification of regimes in all datasets, we plot the regime-switching models for each variable to see whether the bull, bear and stagnant states have been correctly identified. Plot the original time series data with different colors representing the identified regimes.

Learning Parameters

We must train the HMM model in order to obtain the HMM parameters, after converting the time series data and obtaining the observation data. As previously discussed in Step 3, in order to acquire the three model parameters—the starting state probabilities (π), the state transition probability matrix A (A), and the emission matrix B—we must apply the Baum-Welch algorithm, which is an Expectation-Maximization technique.

The primary input for the hmm model will be the observed emission sequence that we obtained in the previous step:

" dhmm_r = hmms.DtHMM.random(3 , 2); "

" dhmm_r.baum_welch(e_seq, 100); "

The observed emission sequence, or e_seq, was entered with 100 iterations after the author defined a hmm model structure using the two scripts mentioned above. By iteratively completing the M-step (maximization) and the E-step (computation of expectation), the Baum-Welch algorithm gradually improves the parameter estimates until the parameter set that maximizes the likelihood of the observed data is found:

" hmms.print_parameters(dhmm_r); "

Viterbi Algorithm

The Viterbi algorithm is a dynamic programming method for calculating the maximum a posteriori probability estimate of the most probable series of hidden states, known as the Viterbi path, that leads to a series of observed events, particularly when applied to Markov information sources and hidden Markov models (HMM). The Viterbi algorithm operates in an iterative manner using the set of formulae below:

$$\begin{aligned}
\mu(X_0) &= P[Y_0|X_0]P[X_0] \\
\mu(X_1) &= \max_{X_0} \mu(X_0)P[X_1|X_0]P[Y_1|X_1] \\
\mu(X_2) &= \max_{X_1} \mu(X_1)P[X_2|X_1]P[Y_2|X_2] \\
\mu(X_3) &= \max_{X_2} \mu(X_2)P[X_3|X_2]P[Y_3|X_3]
\end{aligned}$$

The first formula generates the initial states of probability and the subsequent formula helps decide whether we are constrained to the initial states [1, 2]. The second equation identifies the best starting state that maximizes the product of the equation and the first state is left as a free parameter in the third equation [4].

Find the most likely sequence of hidden states

```
Initial probabilities () :


          0
0  0.209758
1  0.650148
2  0.140094


Transition probabilities matrix (A):


          0         1         2
0  0.383857  0.550736  0.065407
1  0.056177  0.435758  0.508065
2  0.488859  0.000035  0.511106


Emission probabilities matrix (B):


          0         1
0  0.999584  0.000416
1  0.000149  0.999851
2  0.286833  0.713167
```

Step 7

Latent meaning behind each hidden state

Latent meaning behind each hidden state is indexed according to their average value. This helps in understanding the potential underlying regimes that produce the series of observed emissions. By linking each hidden state with its average value, researchers can interpret the importance and features of various states within the Hidden Markov Model, giving insights into the patterns and behaviors of the modeled system.

The indexing process enables a clearer understanding of the hidden states and eases the recognition of patterns, trends and shifts within the time-series data. Additionally, the Viterbi algorithm is frequently used to determine the most probable sequence of hidden states which produced observed outputs. By determining the sequence of hidden states , researchers can reveal the underlying regimes or states of the system's corresponding patterns or behaviors. This process of identifying and interpreting hidden states through their average values is crucial for the composition and changes within the system being analyzed, like the determinants impacting crude oil prices in the research investigation.

Step 8-9

Minimal working example of a Hill Climb search

We generate 8 random series with 1500 samples labeled through A to H, and make data A equals data B plus data C and make data H equals data G minus data A, following the setting of the given thesis. Then, we run the Hill Climb search using the pgmpy package and draw the causal graph using the networks package. The results are shown in the following figure. We can find that the Hill Climb algorithm successfully finds the causal relationship.

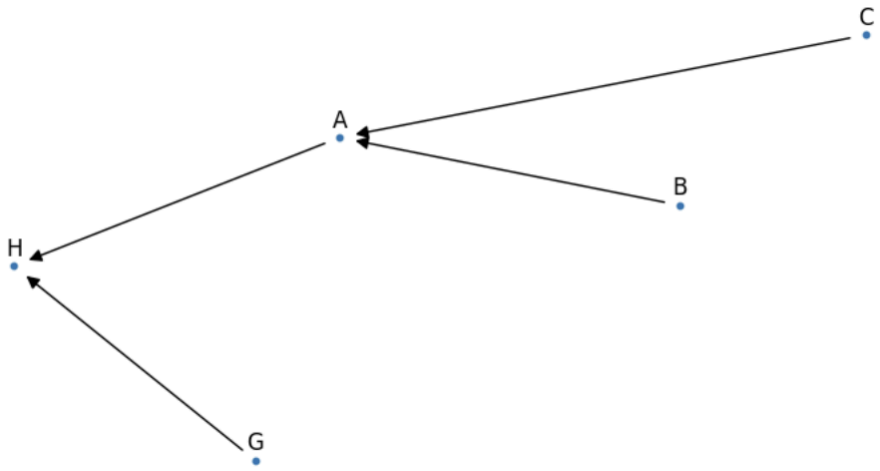

Figure: Causal network learned from Hill Climbing Search

Identify the latent behind each hidden state

'A' represents a hidden state that is influenced by the variables 'B' and 'C' (as indicated by the edges ('B', 'A') and ('C', 'A')). This hidden state presents a combination or aggregation of the values of 'B' and 'C', possibly representing a cumulative effect of these variables.

'H' represents a hidden state that is influenced by the variables 'A' and 'G' (as indicated by the edges ('A', 'H') and ('G', 'H')). This hidden state presents the difference between the value of 'G' and the value of 'A', suggesting that it captures the discrepancy between these two variables.

References

[1] Alvi, Danish. “Application of Probabilistic Graphical Models in Forecasting Crude Oil Price”, Dissertation, 2018.

[2] Baum, Leonard E. and John A. Eagon. “An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology.” Bulletin of the American Mathematical Society 73 (1967): 360-363, 1967.

[3] Baum, Leonard E. “An inequality and associated maximization technique in statistical estimation of probabilistic functions of markov processes”. Inequalities III: Proceedings of the Third Symposium on Inequalities, page 1--8, 1972.

[4] Dempster, A.P., Laird, N.M. and Rubin, D.B. “Maximum Likelihood from Incomplete Data Via the EM Algorithm”. Journal of the Royal Statistical Society: Series B (Methodological), 39: 1-22, 1977.