```
#Coding the Likelihood Function
import numpy as np
from scipy.stats import norm

def ar7_conditional_likelihood(params, data):
    """
    Calculate the conditional log likelihood of an AR(7) model.

    Args:
        params (list): Parameters [phi1, phi2, ..., phi7, sigma^2].
        data (array): Array of observed data.

    Returns:
        float: Conditional log likelihood value.
    """
    phi = params[:-1]
    sigma_squared = params[-1]
    n = len(data)
    conditional_ll = 0

    # Calculate conditional log likelihood for each observation
starting from the 8th
    for t in range(7, n):
        # Calculate conditional mean based on past 7 observations
        mean = np.dot(phi, data[t-7:t][::-1])
        # Calculate log likelihood contribution of current observation
        conditional_ll += norm.logpdf(data[t], loc=mean,
scale=np.sqrt(sigma_squared))

    return conditional_ll

def ar7_unconditional_likelihood(params, data):
    """
    Calculate the unconditional log likelihood of an AR(7) model.

    Args:
        params (list): Parameters [phi1, phi2, ..., phi7, sigma^2].
        data (array): Array of observed data.

    Returns:
        float: Unconditional log likelihood value.
    """
    phi = params[:-1]
    sigma_squared = params[-1]
    n = len(data)
    unconditional_ll = 0

    # Calculate the unconditional mean of the AR(7) process
    mu = np.dot(phi, data[:7][::-1])
```

```
    # Calculate unconditional log likelihood for each observation
    for t in range(n):
        # Calculate log likelihood contribution of current observation
using unconditional mean
        unconditional_ll += norm.logpdf(data[t], loc=mu,
scale=np.sqrt(sigma_squared))

    return unconditional_ll

# Example:
# Define some sample data and initial parameters
data = np.random.normal(size=100)  # Sample data
initial_params = [0.5, -0.2, 0.3, -0.1, 0.2, -0.4, 0.1, 1.0]  #
Initial parameters

# Calculate conditional log likelihood
conditional_ll = ar7_conditional_likelihood(initial_params, data)
print("Conditional Log Likelihood:", conditional_ll)

# Calculate unconditional log likelihood
unconditional_ll = ar7_unconditional_likelihood(initial_params, data)
print("Unconditional Log Likelihood:", unconditional_ll)
```

```
Conditional Log Likelihood: -188.55975121418842
Unconditional Log Likelihood: -158.3866139766513
```

```
#Maximizing the likelihood
from scipy.optimize import minimize

# Define negative log likelihood functions (to be minimized)
def neg_log_likelihood_conditional(params, data):
    return -ar7_conditional_likelihood(params, data)

def neg_log_likelihood_unconditional(params, data):
    return -ar7_unconditional_likelihood(params, data)

# Example usage:
# Define some sample data
data = np.random.normal(size=100)  # Sample data

# Initial parameter guess
initial_params = [0.5, -0.2, 0.3, -0.1, 0.2, -0.4, 0.1, 1.0]

# Maximize conditional likelihood
result_conditional = minimize(neg_log_likelihood_conditional,
initial_params, args=(data,), method='Nelder-Mead')
print("Conditional Likelihood Parameters:", result_conditional.x)

# Maximize unconditional likelihood
result_unconditional = minimize(neg_log_likelihood_unconditional,
```

```
initial_params, args=(data,), method='Nelder-Mead')
print("Unconditional Likelihood Parameters:", result_unconditional.x)
```

```
Conditional Likelihood Parameters: [ 0.11910609  0.03613495
0.10638868 -0.01024039 -0.0283631  -0.21840062
  0.04941569  1.04463373]
Unconditional Likelihood Parameters: [ 0.55858841 -0.20696262
0.36436363 -0.09779754  0.1884641  -0.17724524
  0.11702908  1.08733089]
```

```
#Parameter Estimation
import numpy as np
import pandas as pd
from scipy.optimize import minimize

# Step 1: Data Preprocessing
# Load FRED-MD dataset
data = pd.read_csv('/content/current.csv')  # Replace
'path_to_your_dataset.csv' with the actual path

# Assuming 'INDPRO' is the column name for the INDPRO variable
indpro = data['INDPRO']

# Calculate monthly log differences
log_diff_indpro = np.log(indpro).diff().dropna()

# Step 2: Likelihood Functions
def ar7_conditional_likelihood(params, data):
    phi, sigma_sq = params[:7], params[7]
    n = len(data)
    ll = 0
    for t in range(7, n):
        yt = data[t]
        mu_t = np.dot(phi, data[t-7:t][::-1])  # Reverse the order to
align with AR(7) formula
        ll += -0.5 * (np.log(2 * np.pi * sigma_sq) + (yt - mu_t)**2 /
sigma_sq)
    return ll

def ar7_unconditional_likelihood(params, data):
    phi, sigma_sq = params[:7], params[7]
    n = len(data)
    mu0 = np.mean(data[:7])  # Initial mean
    ll = -0.5 * (7 * np.log(2 * np.pi * sigma_sq) + np.sum((data[:7] -
mu0)**2) / sigma_sq)
    for t in range(7, n):
        yt = data[t]
        mu_t = np.dot(phi, data[t-7:t][::-1])  # Reverse the order to
align with AR(7) formula
        ll += -0.5 * (np.log(2 * np.pi * sigma_sq) + (yt - mu_t)**2 /
```

```
sigma_sq)
    return ll

# Step 3: Parameter Estimation
# Initial parameter guess
initial_params = np.array([0.5, -0.2, 0.3, -0.1, 0.2, -0.4, 0.1, 1.0])

# Maximize conditional likelihood
result_conditional = minimize(ar7_conditional_likelihood,
initial_params, args=(log_diff_indpro,), method='Nelder-Mead')
print("Maximized Conditional Likelihood Parameters:",
result_conditional.x)

# Maximize unconditional likelihood
result_unconditional = minimize(ar7_unconditional_likelihood,
initial_params, args=(log_diff_indpro,), method='Nelder-Mead')
print("Maximized Unconditional Likelihood Parameters:",
result_unconditional.x)
```

```
Maximized Conditional Likelihood Parameters: [-1.05849043e+50
1.06746350e+49 -1.73462824e+49  2.64622188e+49
  2.68764340e+49 -4.69880006e+49 -3.21491762e+49  6.97645179e+50]
Maximized Unconditional Likelihood Parameters: [-1.05849043e+50
1.06746350e+49 -1.73462824e+49  2.64622188e+49
  2.68764340e+49 -4.69880006e+49 -3.21491762e+49  6.97645179e+50]
```

```
#Forecasting
import numpy as np

# Function to forecast future values
def forecast_ar7(params, data, steps):
    # Extract parameters
    phi = params[:7]
    sigma = params[7]

    # Initialize forecast array with observed data
    forecast = np.copy(data)

    # Forecast future values
    for i in range(steps):
        forecast_next = np.dot(phi, forecast[-7:]) +
np.random.normal(0, sigma)
        forecast = np.append(forecast, forecast_next)

    return forecast

# Forecast using conditional likelihood parameters
conditional_forecast = forecast_ar7(result_conditional.x,
log_diff_indpro, 8)
print("Conditional Forecast:", conditional_forecast[-8:])
```

```
# Forecast using unconditional likelihood parameters
unconditional_forecast = forecast_ar7(result_unconditional.x,
log_diff_indpro, 8)
print("Unconditional Forecast:", unconditional_forecast[-8:])
```

```
Conditional Forecast: [-4.85355126e+050  1.56037675e+100 -
5.01648269e+149  1.61275786e+199
 -5.18488365e+248  1.66689738e+298             -inf              nan]
Unconditional Forecast: [-7.80827157e+050  2.51029498e+100 -
8.07039157e+149  2.59456440e+199
 -8.34131081e+248  2.68166271e+298             -inf              nan]
```