Experiment No. 1
Prediction of Gold Price Using Hidden Markov Model
Date of Performance:
Date of Submission:

Aim: Prediction of Gold Price Using Hidden Markov Model

Objective: Ablility to implement Hidden Markov Model for prediction

Theory:

The hidden Markov Model (HMM) is a statistical model that is used to describe the probabilistic relationship between a sequence of observations and a sequence of hidden states. It is often used in situations where the underlying system or process that generates the observations is unknown or hidden, hence it has the name "Hidden Markov Model."
It is used to predict future observations or classify sequences, based on the underlying hidden process that generates the data.
An HMM consists of two types of variables: hidden states and observations.

- The hidden states are the underlying variables that generate the observed data, but they are not directly observable.
- The observations are the variables that are measured and observed.

The relationship between the hidden states and the observations is modeled using a probability distribution. The Hidden Markov Model (HMM) is the relationship between the hidden states and the observations using two sets of probabilities: the transition probabilities and the emission probabilities.

- The transition probabilities describe the probability of transitioning from one hidden state to another.
- The emission probabilities describe the probability of observing an output given a hidden state.

Hidden Markov Model Algorithm
The Hidden Markov Model (HMM) algorithm can be implemented using the following steps:
Step 1: Define the state space and observation space
The state space is the set of all possible hidden states, and the observation space is the set of all possible observations.

Step 2: Define the initial state distribution
This is the probability distribution over the initial state.

Step 3: Define the state transition probabilities
These are the probabilities of transitioning from one state to another. This forms the transition matrix, which describes the probability of moving from one state to another.

Step 4: Define the observation likelihoods:
These are the probabilities of generating each observation from each state. This forms the emission matrix, which describes the probability of generating each observation from each state.

Step 5: Train the model
The parameters of the state transition probabilities and the observation likelihoods are estimated using the Baum-Welch algorithm, or the forward-backward algorithm. This is done by iteratively updating the parameters until convergence.

Step 6: Decode the most likely sequence of hidden states
Given the observed data, the Viterbi algorithm is used to compute the most likely sequence of hidden states. This can be used to predict future observations, classify sequences, or detect patterns in sequential data.

Step 7: Evaluate the model
The performance of the HMM can be evaluated using various metrics, such as accuracy, precision, recall, or F1 score.

To summarise, the HMM algorithm involves defining the state space, observation space, and the parameters of the state transition probabilities and observation likelihoods, training the model using the Baum-Welch algorithm or the forward-backward algorithm, decoding the most likely sequence of hidden states using the Viterbi algorithm, and evaluating the performance of the model.

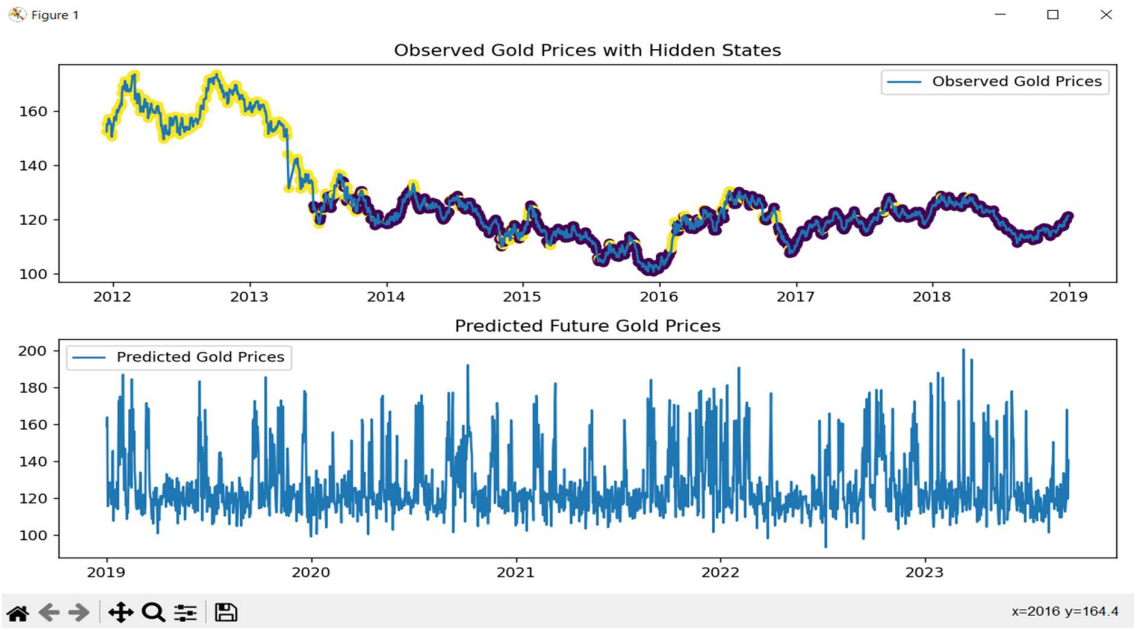
Implementation:

Code:

```
import pandas as pd
import numpy as np
from hmmlearn import hmm
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv(r"C:\Users\Lenovo\Downloads\FINAL_USO.csv")

# Preprocess the data
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)
data.dropna(inplace=True)  # Handle missing values

# Select relevant features
selected_features = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']

# Normalize the data
normalized_data = (data[selected_features] - data[selected_features].mean()) / data[selected_features].std()

# Define and train the HMM
model = hmm.GaussianHMM(n_components=2, covariance_type="full", n_iter=100)
model.fit(normalized_data)

# Save the model
from joblib import dump
dump(model, 'hmm_model.joblib')

# Predict hidden states
hidden_states = model.predict(normalized_data)

# Predict future gold prices
future_gold_prices, _ = model.sample(len(data))
future_gold_prices = future_gold_prices[:, selected_features.index('Close')]  # Considering only the 'Close' price
future_gold_prices = future_gold_prices * data['Close'].std() + data['Close'].mean()

# Create index for the predicted values
future_index = pd.date_range(start=data.index[-1] + pd.DateOffset(days=1), periods=len(data))

# Plot the results
plt.figure(figsize=(10, 6))
```

```
# Plot observed gold prices with hidden states
plt.subplot(2, 1, 1)
plt.plot(data.index, data['Close'], label="Observed Gold Prices")
plt.scatter(data.index, data['Close'], c=hidden_states, cmap='viridis', marker='o')
plt.title("Observed Gold Prices with Hidden States")
plt.legend()

# Plot predicted future gold prices
plt.subplot(2, 1, 2)
plt.plot(future_index, future_gold_prices, label="Predicted Gold Prices")
plt.title("Predicted Future Gold Prices")
plt.legend()

plt.tight_layout()
plt.show()
```

Output:



Conclusion:

The prediction accuracy of the Hidden Markov Model (HMM) algorithm for forecasting gold prices depends on various factors, including the quality of input data, model parameters, and market dynamics. Continuous evaluation and comparison against actual prices are essential to assess its effectiveness accurately.