Stock Market Prediction using Hidden Markov Models

Introduction:

Stock market prediction is a complex and challenging task with significant financial implications. Accurately predicting future stock prices can help investors make informed decisions, manage risks, and potentially maximise returns.

This project aims to develop a stock market prediction model using Hidden Markov Models (HMMs), a powerful statistical technique for modelling time-series data with underlying hidden states.

The model will be trained on historical stock data to learn the patterns and relationships between different market indicators, and then used to forecast future price movements.

Data Acquisition:

The data for this project will be obtained from Yahoo Finance, a widely used source of financial data, using the `yfinance` library in Python.

We will focus on the S&P 500 index (`^GSPC`), a leading indicator of the overall U.S. stock market performance. This index represents a diversified portfolio of 500 large-cap U.S. companies.

The data will cover a period of over 20 years, from January 1, 2000, to August 1, 2021, providing a substantial historical dataset for training and testing the model.

The code snippet for downloading the data using `yfinance` is as follows:

import yfinance as yf

data = yf.download("^GSPC", start="2000-01-01", end="2021-08-01")

```
data.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2000-01-03	1469.250000	1478.000000	1438.359985	1455.219971	1455.219971	931800000
2000-01-04	1455.219971	1455.219971	1397.430054	1399.420044	1399.420044	1009000000
2000-01-05	1399.420044	1413.270020	1377.680054	1402.109985	1402.109985	1085500000
2000-01-06	1402.109985	1411.900024	1392.099976	1403.449951	1403.449951	1092300000
2000-01-07	1403.449951	1441.469971	1400.729980	1441.469971	1441.469971	1225200000

```
data.shape
```

```
(5429, 6)
```

```
train_size = int(0.8*data.shape[0])
print(train_size)
```

```
4343
```

Feature engineering:

Feature engineering is crucial for improving the performance of machine learning models. It involves creating new features from existing data to better represent the underlying patterns and relationships.

In this project, we will engineer three features to capture the daily price movements and volatility:

- `delOpenClose`: Represents the fractional change between the opening and closing prices of a day. Calculated as: (Close Price - Open Price) / Open Price.
- `delHighOpen`: Represents the fractional change between the high and opening prices of a day. Calculated as: (High Price - Open Price) / Open Price.
- `delLowOpen`: Represents the fractional change between the opening and low prices of a day. Calculated as: (Open Price - Low Price) / Open Price.

These features provide a more nuanced view of the daily price action than simply using raw prices.

The code snippets for the `augment_features` and `extract_features` functions, used to create and extract these features, are included in the notebook.

Model Selection:

Hidden Markov Models (HMMs) are well-suited for modelling sequential data like stock prices. They assume that the observed data is generated by an underlying unobservable Markov process with hidden states.

In the context of stock market prediction, these hidden states can represent different market regimes, such as bullish, bearish, or sideways trends.

The HMM learns the transition probabilities between these states and the emission probabilities of observing specific price movements given the current state.

We will use the `hmmlearn` library in Python to implement the GaussianHMM model, a type of HMM that assumes the observed data is generated from a Gaussian distribution.

Model Training:

The training data is first preprocessed using the feature engineering functions.

The GaussianHMM model is then initialised with a specified number of hidden states (e.g., 10).

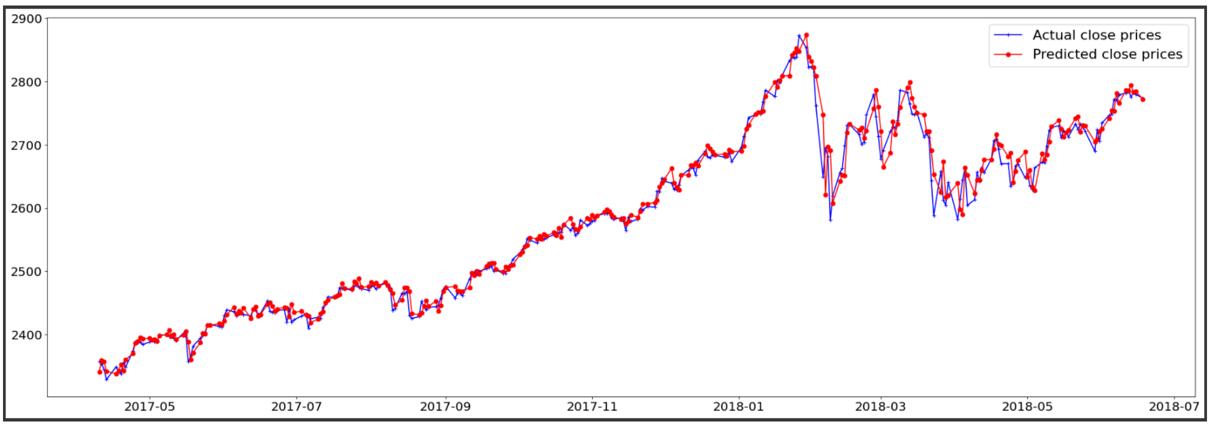
The model is trained on the extracted features from the training data using the `fit` method, which estimates the model parameters (transition and emission probabilities).

Prediction:

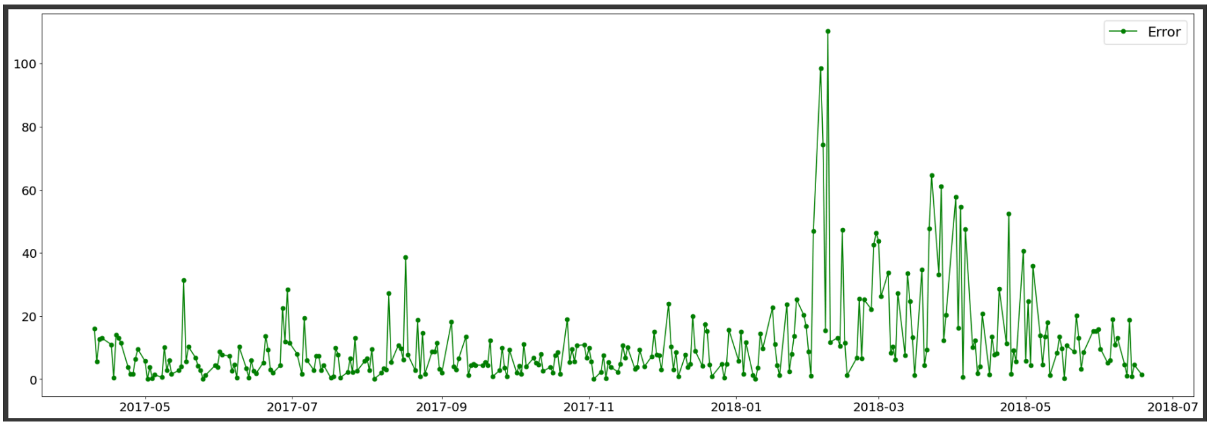
The prediction process involves several steps:

- Define a sample space of possible outcomes for the engineered features. This space is discretized into intervals for computational efficiency.
- For each day to be predicted, extract the features from the previous `num_latent_days` (e.g., 50) of data.
- For each possible outcome in the sample space, append it to the previous days' features and calculate the score (log-likelihood) using the trained HMM model.
- The outcome with the highest score is considered the most probable outcome for the next day.
- The predicted closing price is calculated using the predicted outcome for `delOpenClose` and the opening price of that day: `Predicted Close Price = Opening Price

Graph between Actual close price and Predicted close price:



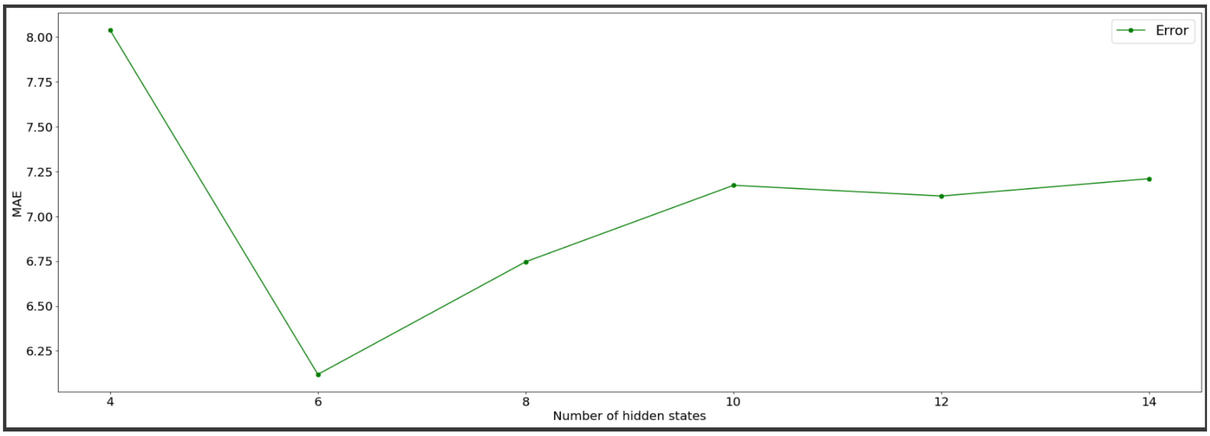
Error:




```
print("Max error observed = " + str(ae.max()))
print("Min error observed = " + str(ae.min()))
print("Mean error observed = " + str(ae.mean()))
```

```
Max error observed = 110.22855246772042
Min error observed = 0.046184990517758706
Mean error observed = 11.31082803576017
```

Number of Hidden states and Mean Absolute Error:



Number of intervals for Features and Mean Absolute Error: