Final Report: Predicting Forex Trends: A Comparative Study of Probabilistic Graphical Models and Beyond

Sinan He, Qingzhe Guo, Ziming Cui

Abstract:

This project investigates the potential of Probabilistic Graphical Models (PGMs) in predicting Forex trends and compares their performance to other popular machine learning models, such as Long Short-Term Memory (LSTM) and eXtreme Gradient Boosting (XGBoost). By examining the capabilities of PGMs and contrasting them with other widely-used machine learning models, we aim to identify the most effective approach for predicting Forex trends. The findings of this study will contribute to the development of more precise and dependable models for Forex trend prediction, ultimately benefiting various stakeholders in their decision-making processes.

1 Introduction and Motivation

Foreign exchange (Forex) is the global market for trading currencies, operating 24 hours a day, 5 days a week. It plays a critical role in the global economy, facilitating international trade, providing investment opportunities, determining exchange rates, enabling risk management, influencing central bank policy, and supporting tourism and remittances. Accurate prediction of Forex trends is essential for traders, financial institutions, and policy-makers, as it can optimize trading strategies, manage risk, and support decision-making processes.

In this project, we explore the application of Probabilistic Graphical Models (PGMs) for predicting Forex trends over a three-year period and assess their performance in comparison to other prominent machine learning models, such as LSTM and XGBoost. Success in this project is determined by the development of precise and reliable models for predicting medium-term Forex trends, which can be evaluated through performance metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) values.

1.1 Problem Statement

The problem we aim to solve is predicting medium-term Forex trends , which are crucial for various stakeholders, including traders, financial institutions, and policy-makers. Accurate predictions can enhance trading strategies, manage risk, and streamline decision-making processes.

1.2 Significance of the Problem

Forex trends are highly sensitive to numerous factors, such as macroeconomic events, geopolitical developments, and market sentiment. As international students, we are

directly affected by fluctuations in exchange rates, which influence our finances and the affordability of education and living expenses. Consequently, developing precise and reliable models for predicting Forex trends is beneficial not only for market participants but also for individuals impacted by currency fluctuations.

1.3 Research Objective

The primary objective of this project is to examine the application of Probabilistic Graphical Models (PGMs) for predicting medium-term Forex trends and evaluate their performance against other popular machine learning models, such as LSTM and XGBoost. We will investigate the capabilities of PGMs, including Hidden Markov Models (HMMs) and Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs), to determine the most appropriate approach for predicting Forex trends.

1.4 Success Criteria

A successful outcome in this project entails the development of accurate and reliable models for predicting medium-term Forex trends. We will assess the performance of these models using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared (R2) values. A successful model will demonstrate low error rates and high R2 values, indicating its effectiveness in predicting Forex trends. By considering these three evaluation metrics, we can gain insights into the accuracy, reliability, and explanatory power of our models. This comprehensive assessment allows us to identify the most suitable approach for predicting Forex trends and supports the development of effective trading strategies and decision-making processes.

2.Approach and Rationale

We chose to focus on PGMs due to their ability to provide a compact representation of complex probability distributions and facilitate efficient inference and learning. We will implement Hidden Markov Models (HMMs), Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs) and compare their performance with LSTM and XGBoost. To simplify the problem and limit the scope of the project, we will focus on predicting medium-term trends (e.g., a few years)., concentrate on major currency pairs (CNY/USD, EUR/USD, and USD/JPY), and use historical price data as the primary input, without incorporating external factors such as macroeconomic indicators or news sentiment.

2.1 Hidden Markov Models (HMMs)

Hidden Markov Models (HMMs) are a type of Probabilistic Graphical Model that represents stochastic processes with hidden (unobservable) states and observable

states. These models have a wide range of applications, including speech recognition, natural language processing, and financial market predictions, such as Forex trends. The main components of an HMM are:

- A finite set of hidden states, $N = S_1, S_2, \ldots, S_n$, which contain 'hidden' information that decides the conditions of the observable states. Hidden states can be described by a Markov process.
- A finite set of observations, $M = O_1, O_2, \ldots, O_m$, which are the states that can be directly observed and predicted.
- A state transition probability matrix, $A = a_{ij}$, where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$.
- An observation probability matrix, $B = b_j(k)$, where $b_j(k) = P(O_k | q_t = S_j)$.
- An initial state probability distribution, $\pi = \pi_i$, where $\pi_i = P(q_1 = S_i)$.

The HMM is described by the joint probability distribution:

$$P(O, Q) = P(q_1) \prod_{t=1}^{T-1} P(q_{t+1}|q_t) \prod_{t=1}^{T} P(o_t|q_t)$$

This equation consists of three main components:
$P(q_1)$: The initial probability distribution of the hidden states at the beginning of the process (time step 1).

$\prod_{t=1}^{T-1} P(q_{t+1}|q_t)$: The product of the state transition probabilities, representing the probability of transitioning from one hidden state to another across time steps.
$\prod_{t=1}^{T} P(o_t|q_t)$: The product of the observation probabilities, representing the probability of observing a particular state at each time step given the hidden state.
By multiplying these three components together, we obtain the joint probability distribution for both the hidden states and the observations in the HMM. This equation is fundamental to understanding the structure of an HMM and is used when learning the parameters of the model and making predictions

By multiplying these three components together, we obtain the joint probability distribution for both the hidden states and the observations in the HMM. This equation is fundamental to understanding the structure of an HMM and is used when learning the parameters of the model and making predictions.

In the context of predicting Forex trends, observable states are typically variables such as exchange rates or price movements for each time step, while hidden states

represent the underlying factors driving these changes that are not directly observable to traders. The HMMs can be used to model relationships between hidden states and observations, as demonstrated by a simplified example involving two friends A and B and their activities based on the weather.[1]

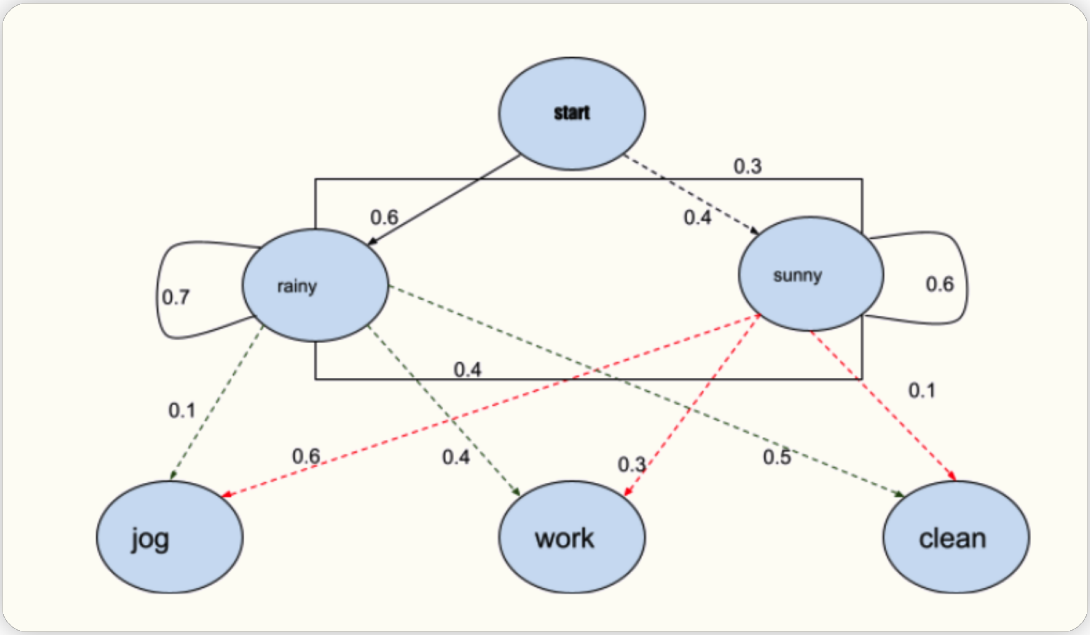


Figure 1.Simple HMM Example Illustrating the Relationship between Weather Conditions and Daily Activities

In this example, friend A performs daily activities such as jogging, working, and cleaning, depending on the weather conditions, while friend B cannot observe the weather but can estimate it based on A's activities. The weather, with two states Rainy and Sunny, serves as hidden states, while A's activities function as observations. B knows the HMM parameters and uses the transition probabilities to predict the weather on the next day and the emission probabilities to estimate A's activities.

In our Forex trend prediction project, we apply a similar approach to model the complex relationships between observable market variables and hidden factors influencing Forex trends. HMMs allow for the modeling of complex, time-dependent relationships between observable states and hidden factors influencing Forex trends.

By providing an introduction to HMMs and explaining their relevance in the context of Forex trend prediction, we lay the foundation for understanding the main concepts and mathematical formulations behind the models used in this project.

2.2 Gaussian Mixture Model-Hidden Markov Models (GMM-HMMs)

The Gaussian Mixture-Hidden Markov Model (GMM-HMM) is a powerful statistical model that combines the strengths of Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM). GMM-HMM has been widely used in various applications, such as automatic speech recognition (ASR), where it has demonstrated considerable success.

The GMM-HMM model aims to predict the most likely hidden state sequence given an observation sequence. In this model, GMM is employed to represent the observation probability distribution in the HMM framework. GMM is a statistical model that can model data using a combination of multiple Gaussian distributions. By incorporating GMM, the HMM can handle continuous data by modeling each hidden state as a Gaussian distribution. Consequently, the observation sequence is assumed to be generated by each hidden state according to a Gaussian mixture distribution.

The GMM-HMM model makes two basic assumptions:

1. The state at any time t depends on its state at time t-1 and is independent of any other moment:

$$P(S_t|S_{t-1}, O_{t-1),...,S_1,O_1)=P(S_t|S_{t-1_)}$$

2. The observation at any time t depends on its state at time t and is independent of any other moments:

$$P(O_t|S_T, O_T, ..., S_1, O_1) = P(O_t|S_t)$$

By leveraging the GMM-HMM framework, complex relationships between observable variables and hidden factors can be modeled effectively, making it a suitable choice for various applications, including Forex trend prediction.

In the context of Forex trend prediction, we can use the GMM-HMM model to analyze the relationships between the observable variables, such as the fraction change, and the hidden factors influencing the trends. To illustrate this, we have plotted the GMM-HMM distribution functions vs. the fraction change in the Forex data, where the number of hidden states is set to 4.

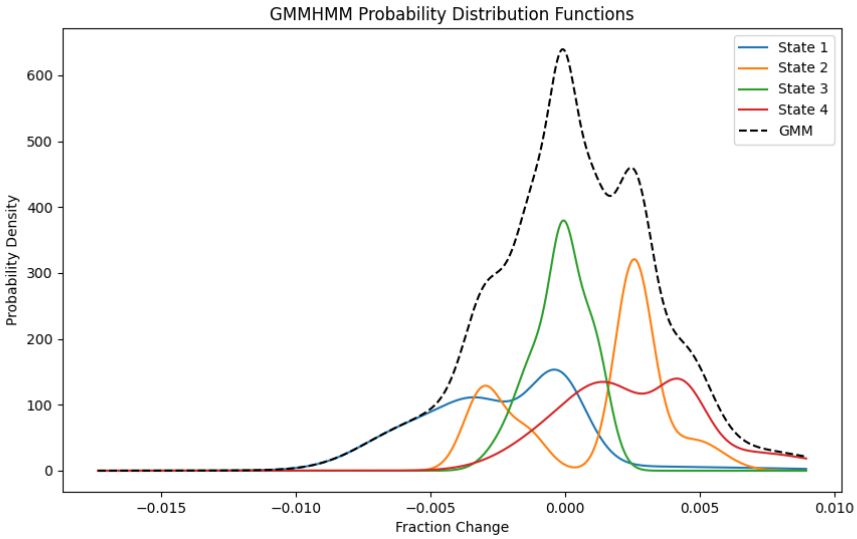


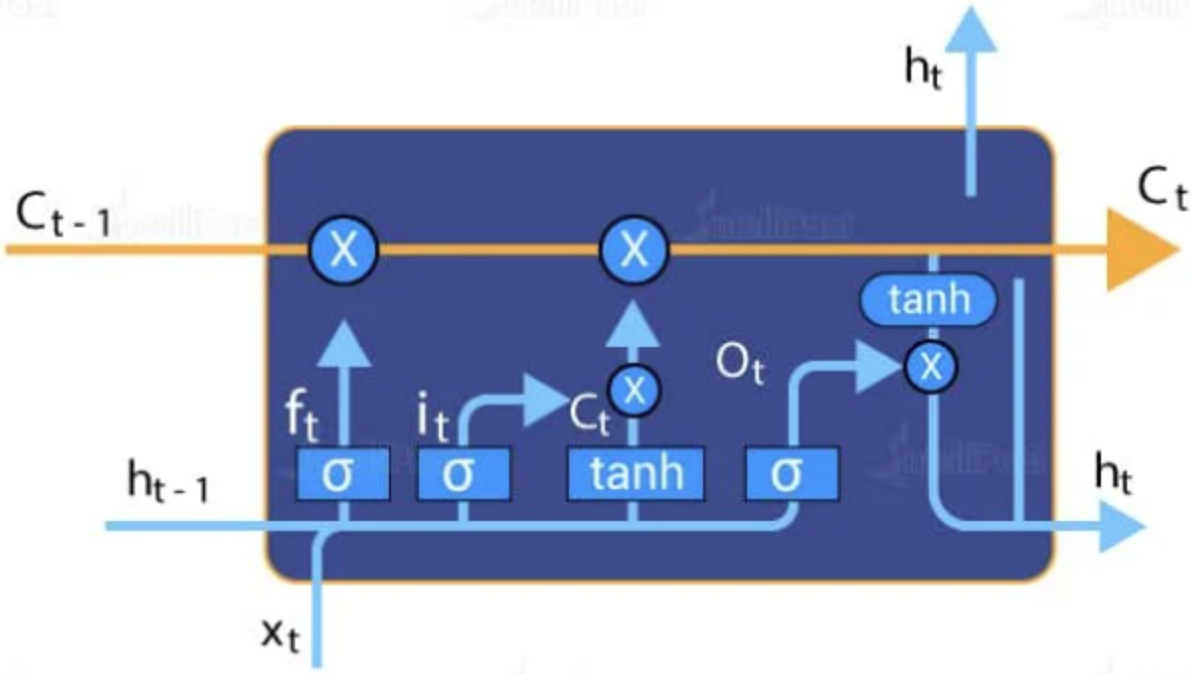
Figure 2 GMM-HMM Probability density Functions

The plot demonstrates how the GMM-HMM model captures the underlying structure of the Forex data by modeling the probability distributions of the fraction change with a combination of four Gaussian distributions. Each Gaussian distribution represents a hidden state in the GMM-HMM model. The overall GMM-HMM distribution function, which is the weighted sum of these Gaussian distributions, effectively models the observed fraction change in the Forex data. This allows us to uncover the hidden factors influencing the Forex trends and make more informed predictions about future market movements.

By visualizing the GMM-HMM distribution functions in relation to the fraction change, we can gain a better understanding of how the model captures the complex relationships between the observable variables and hidden factors in the Forex market. This graphical representation serves as a powerful tool for both understanding the model's structure and evaluating its performance in predicting Forex trends.

2.3 LSTM[2]

2.3.1 The theoretical basis



Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that was specifically designed to avoid the vanishing gradient problem in traditional RNNs. The vanishing gradient problem occurs when the gradients used to update the weights in a neural network decrease exponentially as they are propagated backward through time, leading to the inability of the network to learn long-term dependencies.

LSTM networks use a set of specialized units called memory cells that can store information over long periods of time and selectively forget or remember information based on a gating mechanism. The gating mechanism controls the flow of information through the network, allowing it to selectively remember or forget information from previous time steps.

LSTM networks consist of several layers of memory cells, with each layer being connected to the previous layer through a set of gates. The gates control the flow of information between the cells, allowing the network to selectively remember or forget information. The gates are implemented using sigmoid activation functions, which output values between 0 and 1, with 0 indicating to forget information and 1 indicating to remember information.

The mathematical basis of LSTM is based on the use of matrix multiplication, nonlinear activation functions, and backpropagation through time (BPTT) algorithm. During training, the weights of the network are adjusted using an optimization algorithm such as Stochastic Gradient Descent (SGD) to minimize the error between the predicted

output and the actual output. The gradients used in the optimization are computed using the BPTT algorithm, which involves the recursive computation of gradients through time.

The mathematical theoretical basis of LSTM involves the use of matrix multiplication, nonlinear activation functions, gating mechanisms, and BPTT algorithm to selectively remember or forget information and learn long-term dependencies in a sequential data.

2.3.2 Approach

This approach is to use a Long Short-Term Memory (LSTM) neural network to predict future prices of a forex currency. The rationale behind using an LSTM is that it is a type of recurrent neural network (RNN) that can handle sequential data, making it well-suited for predicting time series data such as stock or Forex prices.

The code begins by loading the necessary libraries, including PyTorch for building the LSTM model, NumPy and Pandas for data manipulation, and Matplotlib and Seaborn for data visualization. The data is loaded from a CSV file containing historical forex prices and preprocessed by scaling the values using a MinMaxScaler. We choose the past five years of foreign exchange price data as the dataset. It is a larger enough dataset, which can provide more diverse examples for the model to learn from and reduce the chances of overfitting.The preprocessed data is then split into training and testing datasets, and a sliding window approach is used to create sequences of data with a fixed number of time steps (i.e., the lookback period) to be used as input to the LSTM.

Next, the LSTM model is defined using PyTorch, with an input layer, two LSTM layers, and a fully connected output layer. The model is trained on the training dataset using backpropagation and the Adam optimizer, with the goal of minimizing the mean squared error (MSE) between the predicted and actual values. The training loss is recorded at each epoch and visualized using Seaborn.

After training, the model is used to predict future forex prices on the testing dataset, and the results are evaluated using the root mean squared error (RMSE) metric, which measures the difference between the predicted and actual values. The predictions are then visualized using Matplotlib and Plotly, with separate plots for the training and testing datasets.

At the same time, we also use regularization techniques such as dropout or weight decay such as dropout or weight decay, which can help prevent the model from becoming too complex and overfitting the training data.

The rationale behind this approach is to leverage the power of neural networks to capture complex patterns in time series data, and use this to make accurate predictions of future prices. By using an LSTM specifically, the model is able to handle the long-term dependencies and temporal dynamics present in financial data, making it a suitable tool for forecasting forex prices.

2.4 XGBoost[3][5]:

Before introducing the XGBoost algorithm, there are several concepts about it.

Boosting:

Boosting is an ensemble learning technique used in supervised machine learning to improve the accuracy and performance of a model. It works by combining several weak learners (base models) into a single strong learner. Weak learners are models that perform only slightly better than random chance, while a strong learner is a model with high accuracy and better generalization capabilities.

The main idea behind boosting is to iteratively train weak learners on different subsets of the training data or different weightings of the instances, with each new learner focusing on correcting the mistakes made by the previous learners in the ensemble. The final prediction is made by aggregating the predictions of all the individual weak learners, typically using a weighted vote or weighted average, depending on the problem being solved (classification or regression, respectively).

Gradient Boosting:

Gradient boosting is an ensemble learning technique that combines weak learners (typically decision trees) to create a strong learner. It works by iteratively adding new models to the ensemble, each model correcting the errors made by the previous ones. The objective is to minimize the overall prediction error by optimizing a differentiable loss function. The final prediction is made by aggregating the predictions of all the trees using a weighted sum.

XGBoost:

XGBoost is an optimized implementation of gradient boosting that adds regularization (L1 and L2) to control model complexity and prevent overfitting. It also includes several improvements, such as parallelization during tree construction, automatic handling of

missing values, and built-in cross-validation, which make it more efficient and effective than the basic gradient boosting algorithm. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

The algorithm flow:

- Start with a simple base model (usually a decision tree) that provides an initial prediction.
- For each iteration:
  a. Calculate the gradient of the loss function with respect to the current ensemble's prediction.
  b. Fit a new decision tree to the calculated gradient values.
  c. Add the new tree to the ensemble, applying a learning rate to control the tree's contribution.
- Combine all the decision trees in the ensemble to make the final prediction.
- Throughout the process, apply regularization techniques (L1 and L2) to prevent overfitting.

2.5 Model evaluation Metric

To assess the performance of our models in predicting Forex trends, we employ three widely-used evaluation metrics. These metrics provide a comprehensive understanding of the models' predictive accuracy and their ability to capture the underlying dynamics of the Forex market.

a. Mean Absolute Error (MAE): The MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It is calculated as the mean of the absolute differences between the predicted values and the actual values. A lower MAE indicates that the model is more accurate in its predictions. This metric is particularly useful for understanding the overall level of prediction errors and can help us identify models that consistently provide accurate forecasts.
b. Root Mean Squared Error (RMSE): The RMSE is a measure of the differences between the predicted and actual values, considering the square of the errors. By squaring the errors, RMSE gives higher weight to large errors, making it more sensitive to significant discrepancies in predictions. A lower RMSE value indicates better model performance. Comparing the RMSE values of different

models can help us identify the model that minimizes large errors and provides more reliable predictions.

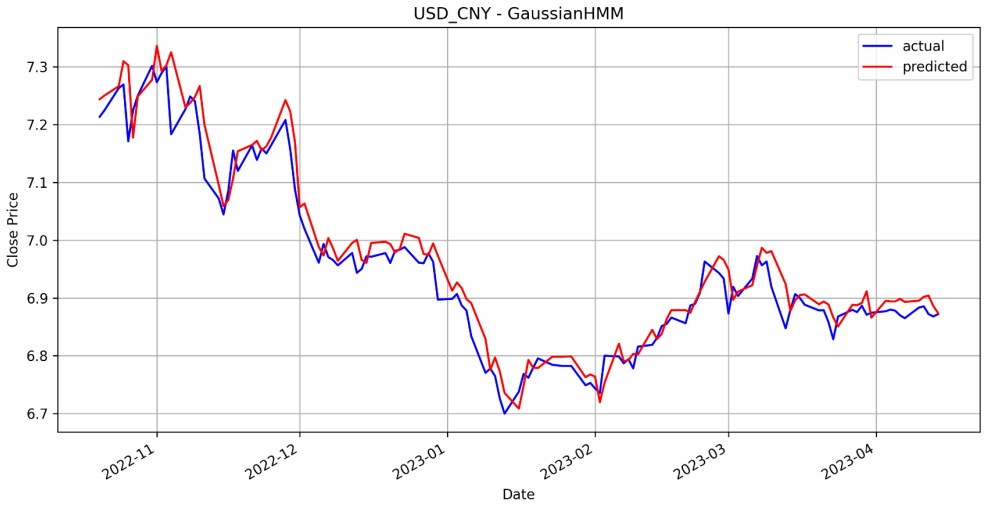
c. R-squared (R^2) value: The R^2 value, also known as the coefficient of determination, represents the proportion of the variance in the dependent variable that is predictable from the independent variables. In the context of Forex trend prediction, an R^2data, while a value close to 0 suggests that the model has limited predictive power. Comparing the R^2 values of different models can help us evaluate the models' ability to capture the underlying patterns in the Forex market.

2.6 Data Collection and Processing

To develop and evaluate our models, we need historical price data for selected currency pairs. We use the Alpha Vantage API to fetch daily Forex data for the currency pairs USD/EUR, USD/CNY, and USD/JPY. The data covers a period of 5 years and includes the open, high, low, and close prices for each trading day. The data is preprocessed by cleaning, normalizing, and extracting relevant features. We then split the preprocessed data into training and testing datasets at a 9:1 ratio. This data is used to train and evaluate models like HMMs, GMM-HMMs, LSTM,and XGBoost for predicting Forex trends.

3. Results and analysis

3.1 HMMs



MAE	RMSE	R^2
0.0255	0.0351	0.9471

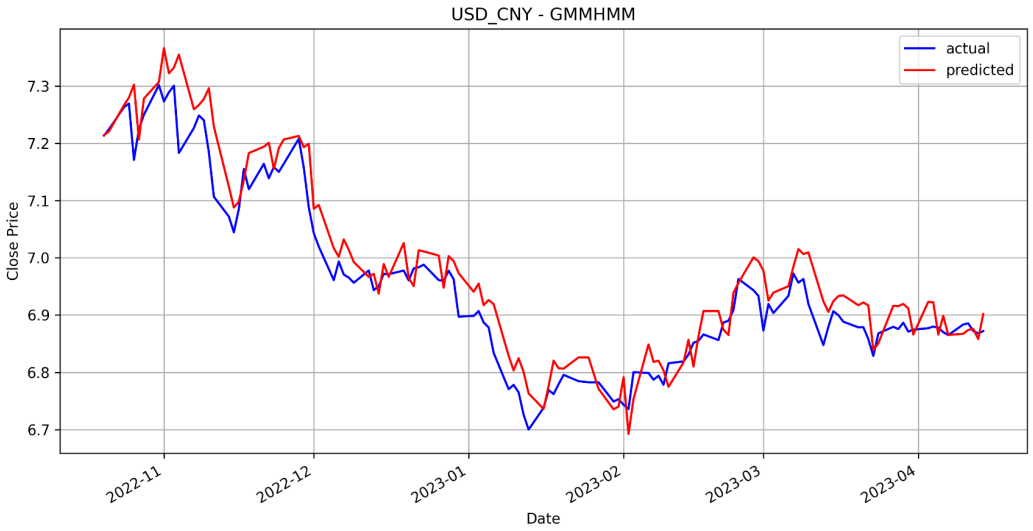
The results of the Forex prediction HMM demonstrate the potential of the HMM-based approach in capturing the dynamics of the currency market.Taking USD_CNY currency pair as an example, the model yielded a MAE of 0.0255, a RMSE of 0.0351, and an R^2 value of 0.9471, indicating a strong fit between the predicted and actual closing prices.

In the analysis of the HMM performance, it is essential to highlight the role of latent states in capturing the underlying structure of the time series data. By learning the hidden states, the HMM can infer the relationships between past and future price changes, making it suitable for Forex trend prediction. Moreover, the feature extraction process, which includes the calculation of fraction change, fraction high, and fraction low, contributes to the model's understanding of the data, improving its predictive capabilities.

However, there is room for improvement in the current approach. The choice of the number of hidden states, as well as the number of latency days, could be further optimized using techniques such as cross-validation. Additionally, exploring alternative feature extraction methods or incorporating additional data, such as technical indicators or macroeconomic factors, could lead to better predictions.

In conclusion, the HMM-based approach demonstrates its potential in Forex trend prediction for the USD_CNY currency pair. Nevertheless, further work should focus on refining the model and incorporating additional information to improve its performance. Comparing the HMM model with other time series models, such as LSTM would also provide valuable insights into the most effective approach for Forex trend prediction.

3.2 GMM-HMMS



MAE	RMSE	R^2
0.0467	0.0549	0.8706

The GMM-HMM model was implemented to predict the Forex trends. The performance of the GMM-HMM model on the test dataset for the USD_CNY currency pair resulted in a MAE of 0.0467, RMSE of 0.0549, and R2 of 0.8706. In comparison to the HMM model, the GMM-HMM model has a higher MAE and RMSE but a lower R2, indicating that the GMM-HMM model performs slightly worse in predicting the closing prices of the currency pair.

Both HMM and GMM-HMM models can be employed to model and predict financial time series, such as Forex trends. While the HMM model performed better in this specific case, the GMM-HMM model might offer better performance on other datasets or with more optimized hyperparameters.

The implemented approach worked to a certain extent, as the models could predict the Forex trends with reasonable accuracy. However, there is room for improvement. One way to improve the performance is by tuning the hyperparameters, such as the number of hidden states, number of mixture components, and other parameters related to the GMM-HMM model. The default hyperparameters I use for the experiments are ‘n_hidden_states’ (The number of hidden states in the GMM-HMM): 4, the same as ‘n_hidden_states’ of HMM, and for GMMHMM, The number of Gaussian components in the mixture model for each hidden state ‘n_mix’ is also 4. Other parameters related to

feature extraction and prediction: The number of previous days to consider when making a prediction for a specific day ‘n_latency_days’ is 10, Discretization granularity for the fraction change range ‘n_steps_frac_change’ is 50, and both Discretization granularity for the fraction high range ‘n_steps_frac_high’ and Discretization granularity for the fraction low range ‘n_steps_frac_low’ are 10. Tuning these parameters might get a better result.

The grid search technique can be applied to find the best combination of hyperparameters that result in the lowest prediction errors. Based on the MAE,the grid search yielded the following best combination of hyperparameters for the Gaussian HMM model:n_hidden_states: 6, n_latency_days: 12 with MAE 0.0228. And for the GMM-HMM model, the best combination is: n_hidden_states: 5 and n_latency_days: 14 with MAE 0.0402. The best combination of hyperparameters yielded 6 hidden states for the Gaussian HMM model and 5 for the GMM-HMM model, suggesting that GMM-HMM may capture the data structure more effectively with fewer states due to Gaussian mixture components. The optimal latency days were 12 and 14 for Gaussian HMM and GMM-HMM, respectively, indicating that more historical data in GMM-HMM could enhance its performance.

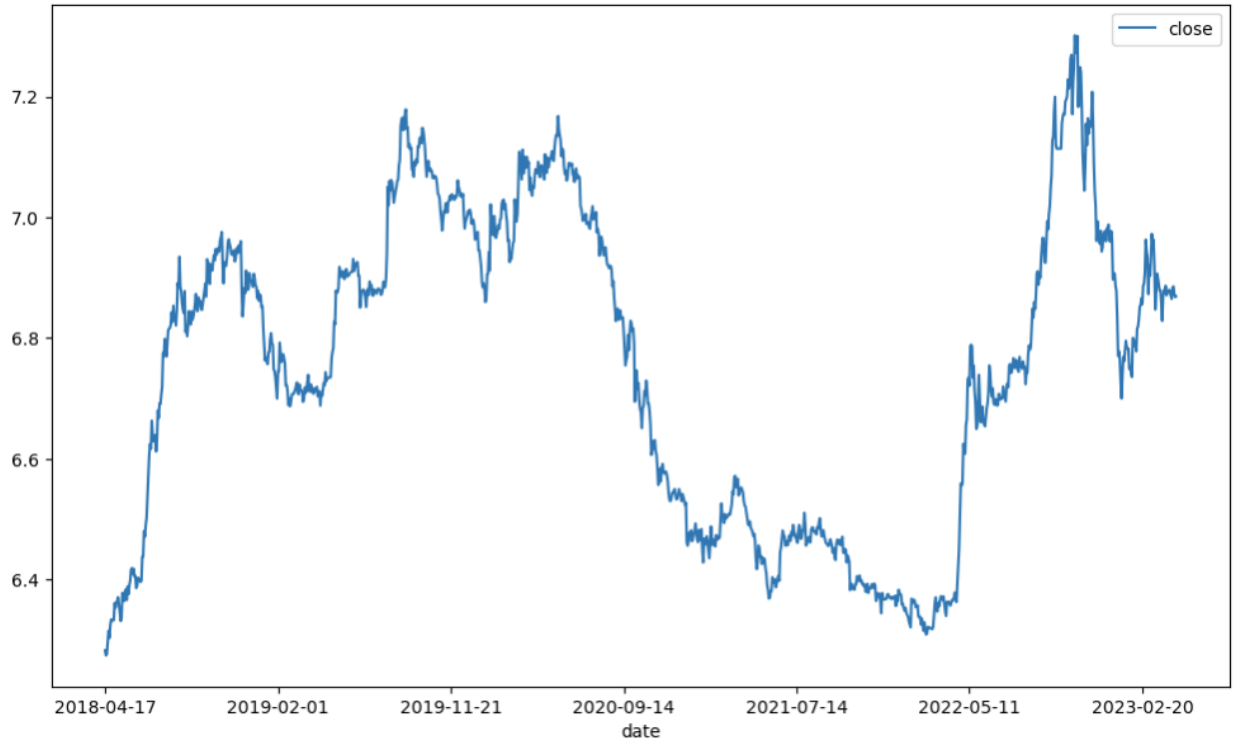
Evaluation using metrics like MAE, RMSE, and R2 is essential to assess the models' performance and identify improvement areas. Despite optimal hyperparameters, overfitting or underfitting may still occur. To boost performance, we can consider regularization, feature engineering, or alternative models such as LSTMs or ensemble models.

Throughout the project, it was learned that modeling complex financial time series is a challenging task, and there is no one-size-fits-all solution. The performance of different models may vary depending on the specific problem, dataset, and chosen hyperparameters.

To further explore and improve the project, alternative approaches such as deep learning models (e.g., LSTM, GRU), reinforcement learning, or ensemble methods could be considered. Additionally, incorporating more features, such as technical indicators, trading volumes, or macroeconomic data, might help in capturing the underlying patterns more effectively and improving the prediction performance. Another essential aspect to consider is the risk of overfitting. By employing techniques such as cross-validation and regularization, it is possible to reduce the risk of overfitting and build more robust predictive models.

3.3 LSTM

USD_CNY



USD_EUR

USD_JPY

My code demonstrates the use of PyTorch and LSTM models for time series prediction of Forex prices, and provides a detailed example of the various steps involved in building and evaluating such a model.

The following steps:

1. Imports necessary libraries such as Pandas, Numpy, Matplotlib, and PyTorch
2. Loads the historical Forex price data and preprocesses it by converting date strings to datetime objects and normalizing the data
3. Defines a function to create a dataset with a lookback window for time series prediction
4. Splits the preprocessed data into train and test sets and creates PyTorch DataLoader objects for each set
5. Defines and trains an LSTM model using PyTorch
6. Evaluates the trained LSTM model on the test set and calculates its performance using the root mean squared error (RMSE)
7. Plots the training loss and Forex price predictions made by the LSTM model
8. Uses Plotly to create an interactive plot of the Forex price predictions made by the LSTM model

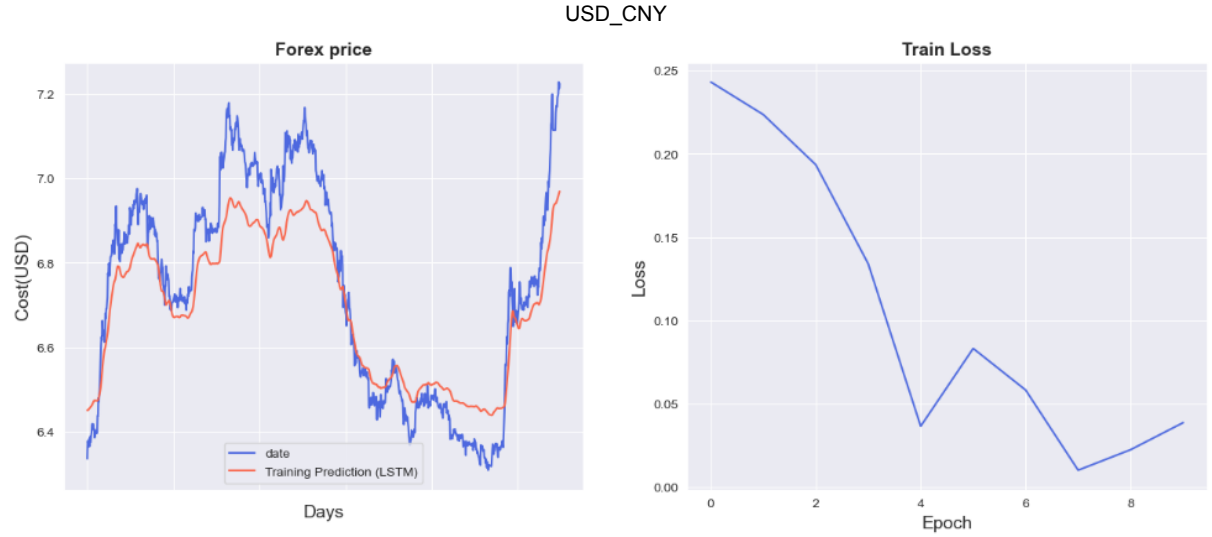
Result & Analysis

RMSE: My model's RMSE is 0.06, R^2 = 0.0508, and MAE = 0.34. These values indicate that my model is performing reasonably well in predicting.

The result of MAE, RMSE, and R^2 are shown below.

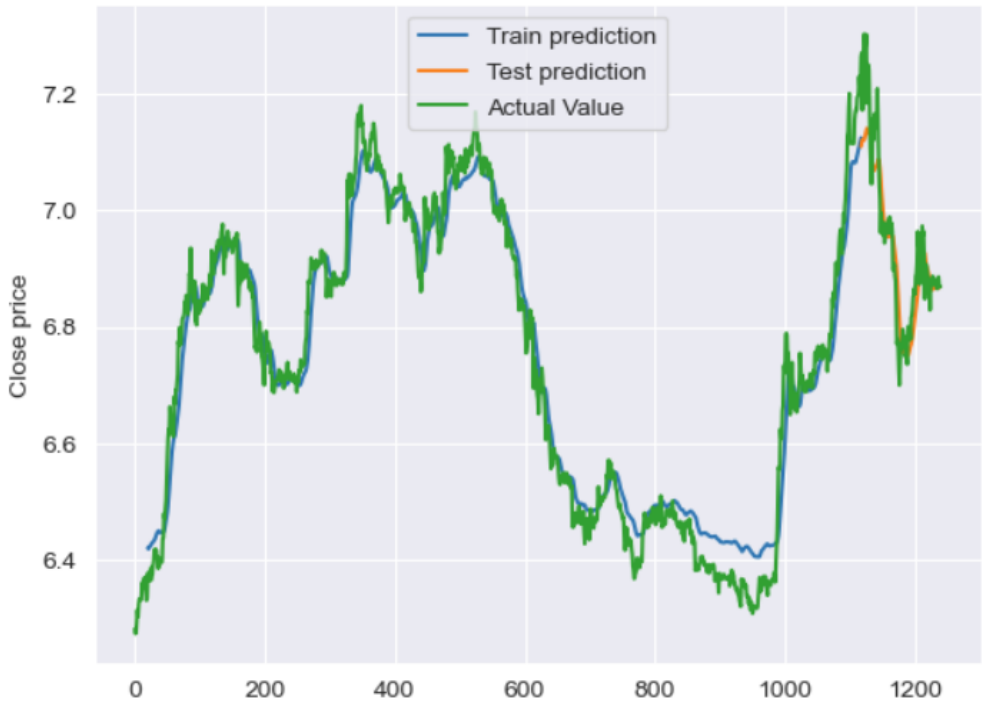
MAE	RMSE	R^2
0.34	0.06	0.0508

Loss: The loss function decreases as the number of epochs increase, which is expected in a well-performing LSTM model. The final loss value is relatively low, indicating a good fit of the model to the data.



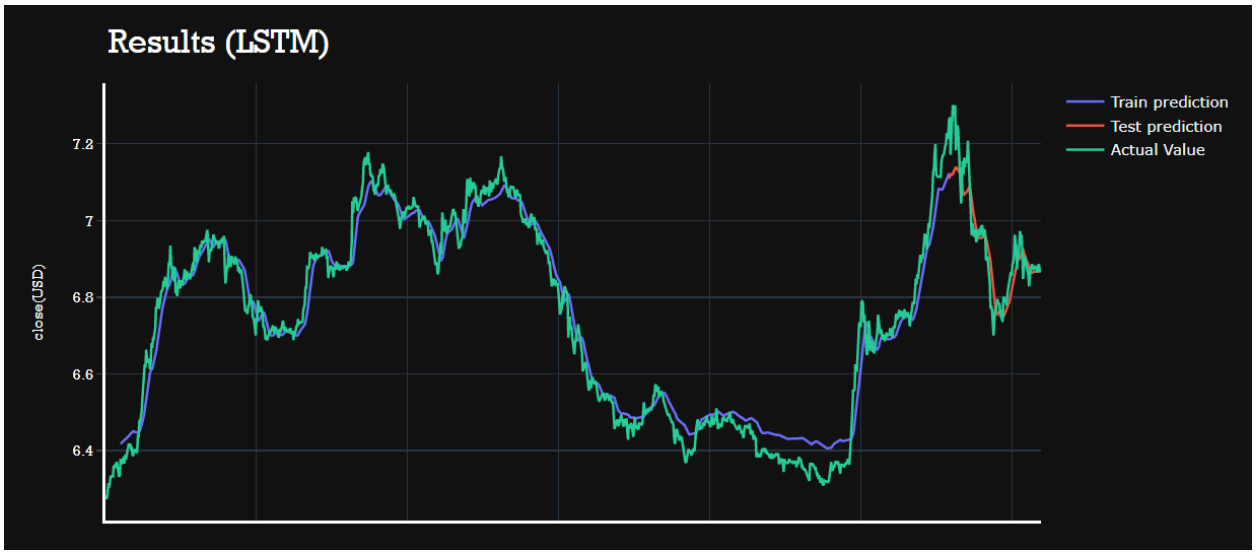
Plot: The plot shows that the actual prices closely follow the predicted prices, especially during the training period. During the test period, the actual prices show more variation than the predicted prices, but the overall trend is captured reasonably well by the model.

USD_CNY



Overfitting: Based on the results I provided, there are no clear signs of overfitting in my LSTM model. The model appears to generalize well to the test set, as evidenced by the relatively low test RMSE.

In conclussion, my LSTM model appears to be performing reasonably well in predicting forex prices, with low RMSE values and a good fit to the data.



For performance improvement, there are some approaches that can be tried including:

1. Using a larger dataset to train the model

2. Tuning the hyperparameters of the model, such as the learning rate and number of hidden layers
3. Using other types of models, such as hybrid models combining LSTMs with other types of neural networks or classical machine learning algorithms

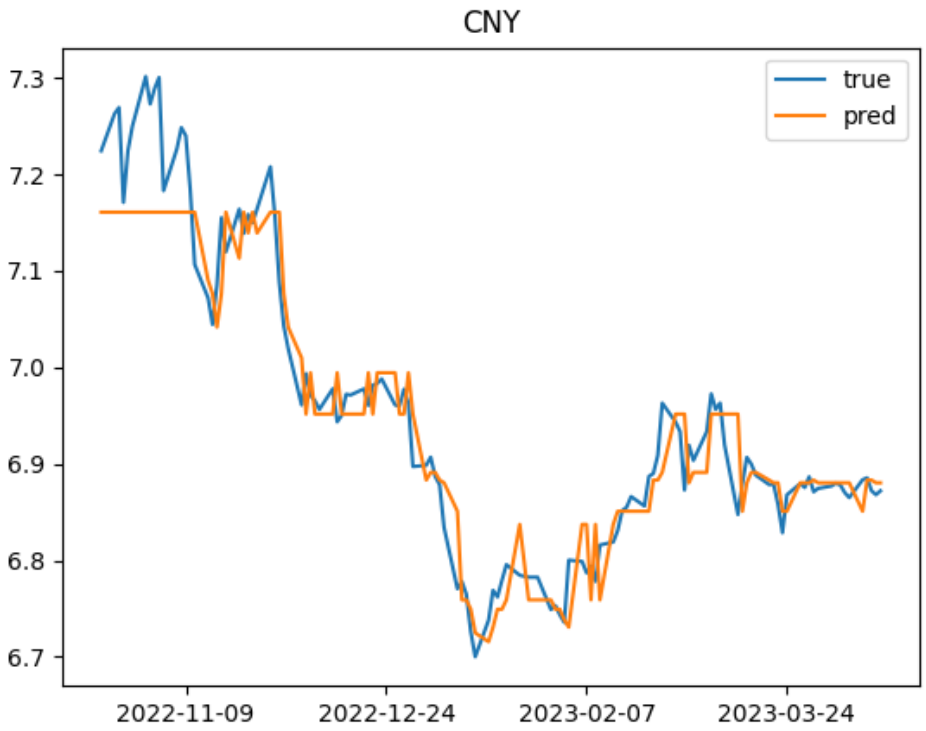
3.4 XGBoost

There are two different experiments for USD/CNY.

3.4.1 First experiment

In the first experiment, the 5-year dataset was split into training and testing datasets. The ratio of training dataset to testing dataset is 9:1.

The figure below shows the predicted and true values.



The result of MAE, RMSE, and R^2 are shown below.

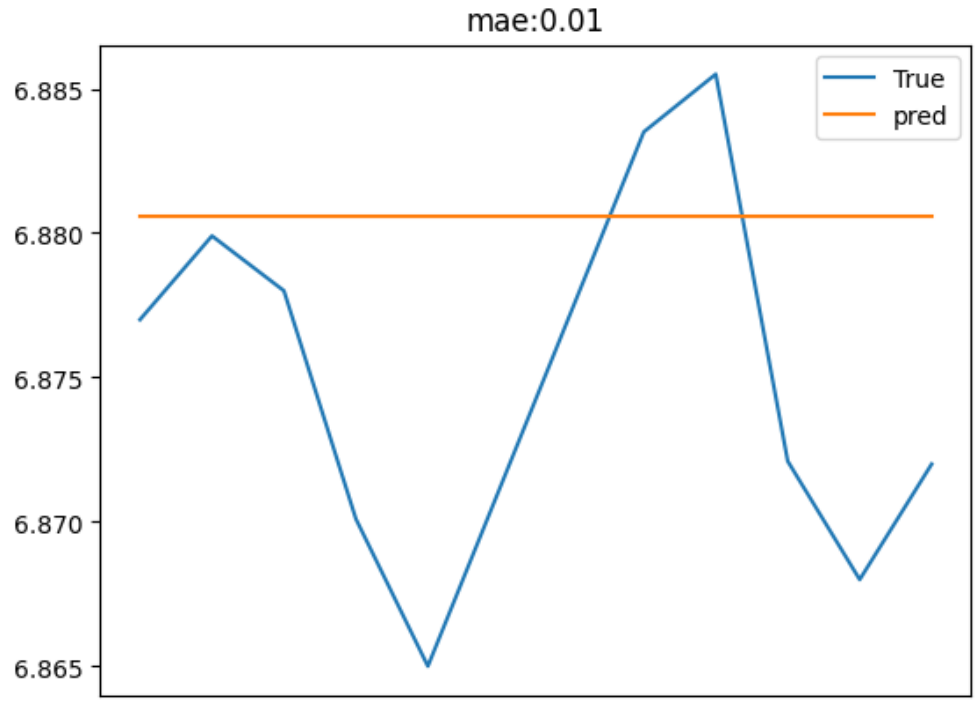
MAE	RMSE	R^2

0.03	0.0442	0.9148

The result of the Forex prediction shows the advantages of XGBoost in the long term Forex prediction. For this experiment, the MAE is 0.03, RMSE is 0.0442, and the R^2 is 0.9148. The XGBoost can predict a similar trend, although the accuracy is not high.

3.4.2 Second Experiment

In the second experiment, we only use data from 2023. The data from 1/2/2023 to 3/15/2023 is used as the training data set, and the data from 3/16/2023 to 4/16/2023 is used as the test data set.



The result of MAE, RMSE, and R^2 are shown below.

MAE	RMSE	R^2
0.01	0.0084	-0.7311

The second experiment only considers data from 2023. Forecasting cannot have a good performance on this prediction. For this experiment, the MAE is 0.01, RMSE is 0.0084, and the R^2 is -0.7311.

From these two experiments, we can see that XGBoost is more suitable for long-term prediction. Long-term data helps it capture more hidden state information, which makes XGBoost powerful inference.

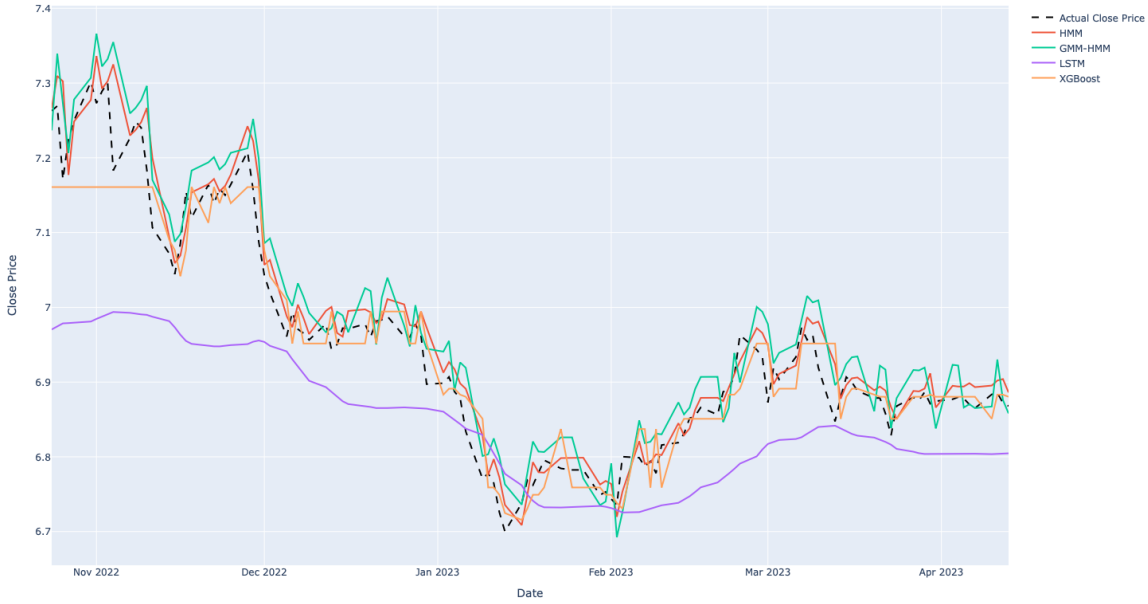
While the current approach shows promise for long-term prediction, there are opportunities for further improvement. Cross-validation techniques could be applied to optimize the selection of the number of hidden states and latency days. Moreover, investigating alternative feature extraction techniques and integrating additional data sources, such as technical indicators or macroeconomic factors, may lead to more accurate predictions. Also, as a machine learning approach, hyperparameter tuning also will improve the performance of the prediction. Using a smaller learning rate can lead to a more robust model, which may help XGBoost have a performance improvement on short-term prediction. The maximum depth of the tree(the simple classifier under XGBoost) and the number of estimators can also provide good performance.

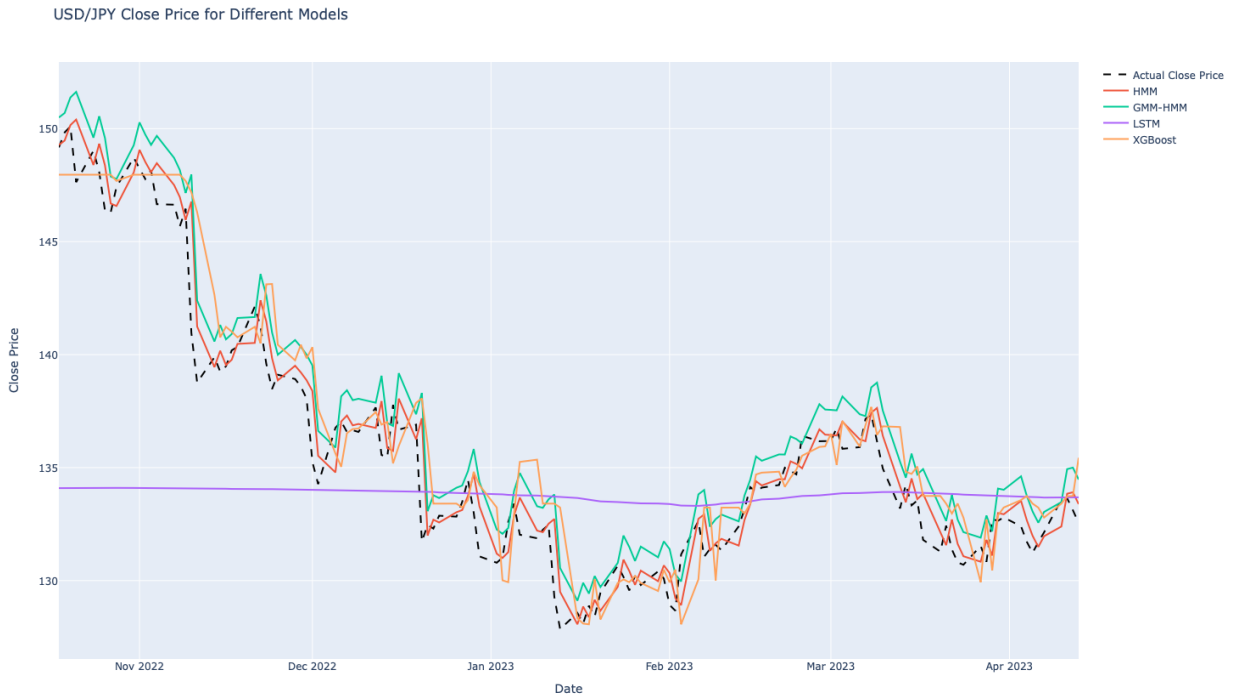
In summary, XGBoost demonstrates its potential for predicting foreign exchange. To improve the model further, future efforts should focus on hyperparameter tuning and incorporating cross-validation and up-to-date feature selection techniques to enhance the model's accuracy.

3.5 Performance Comparison

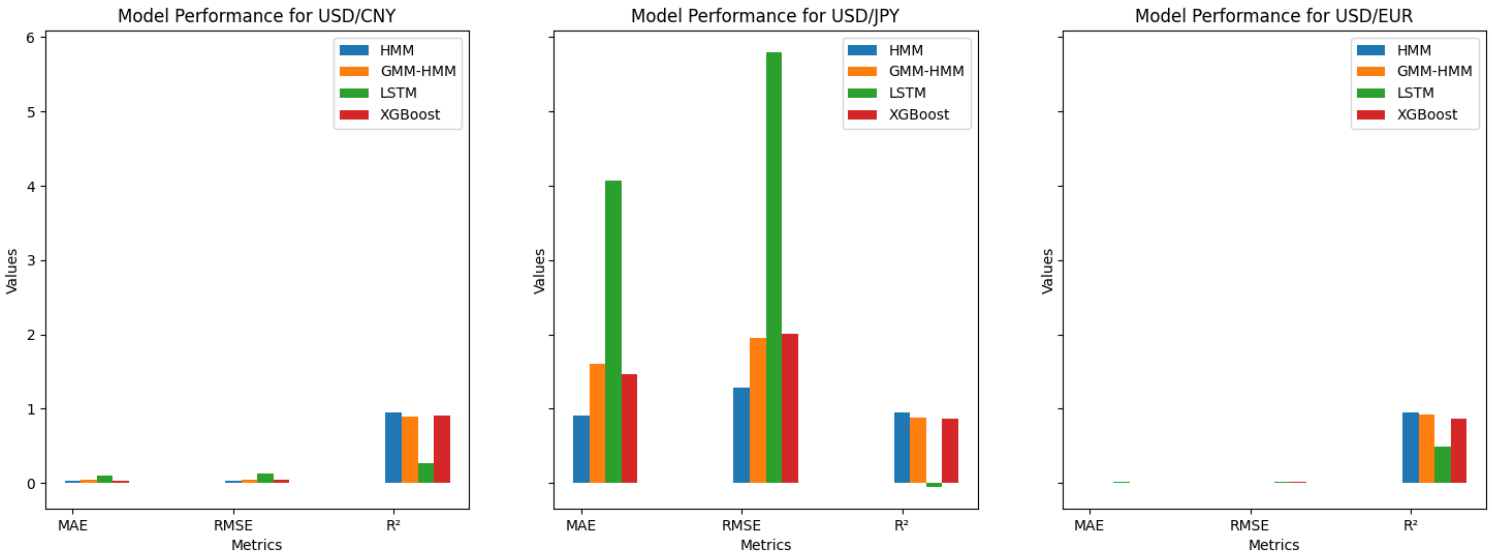
USD/EUR Close Price for Different Models

USD/CNY Close Price for Different Models





The results of the performance evaluation are as follows:



Our analysis reveals that the HMM model consistently performs well across all currency pairs, with lower MAE and RMSE values and higher R-squared scores. In contrast, LSTM exhibits the worst performance among the four models, with higher MAE and RMSE values and lower R-squared scores. GMM-HMM and XGBoost have comparable performance, with both models yielding better results than LSTM and sometimes outperforming each other.

The superior performance of the HMM model can be attributed to its ability to capture the underlying dynamics and regime changes in financial time series data. On the other hand, LSTM may struggle with the noisy and non-stationary nature of the data, leading to poorer performance. GMM-HMM and XGBoost, although not as effective as HMM, still manage to capture some patterns in the data and perform comparably.

While the LSTM model demonstrates the worst performance among the four models, several factors could contribute to this outcome:

a)Hyperparameter tuning: The performance of LSTM models can be sensitive to the choice of hyperparameters, such as the number of hidden layers, number of units in each layer, learning rate, and dropout rate. Insufficient tuning might have led to suboptimal model performance.
b)Model architecture: The architecture of the LSTM model, such as the use of bidirectional LSTMs, attention mechanisms, or the combination of LSTM with other neural network layers (e.g., convolutional layers), could have a significant impact on the model's effectiveness.
c)Feature engineering: LSTM models often benefit from the inclusion of additional relevant features. In our project, we only used the historical close price data. Incorporating other features, such as technical indicators, macroeconomic factors, or sentiment analysis data, might help the LSTM model capture more information and improve its predictions.
d)Data preprocessing: Financial time series data often exhibits non-stationarity and seasonality, which can negatively affect the performance of LSTM models. Applying appropriate preprocessing techniques, such as differencing, log transformation, or seasonal decomposition, could improve the model's ability to learn from the data.
Sequence length: The choice of sequence length (i.e., the number of time steps used as input) can also impact the LSTM model's performance. Using too short or too long sequences might hinder the model's ability to learn temporal dependencies in the data.

Although LSTM underperforms compared to the other models in our analysis, it is essential to consider the factors that may have contributed to this outcome. By addressing these issues, we might be able to improve the performance of the LSTM

model, making it more competitive with the other approaches. For future work, we recommend revisiting the LSTM model and thoroughly exploring the factors mentioned above to optimize its performance in predicting the close price of currency pairs.

4 Conclusion and Future work

Based on the performance of the four models across the three currency pairs, the key takeaways from this project are:

- HMM emerges as the most effective method for predicting the close price of the tested currency pairs.
- GMM-HMM and XGBoost show comparable performance, both outperforming LSTM and sometimes performing better or worse than each other.
- Although LSTMs are popular in time series forecasting, they may not be the best option for this particular problem.
- The approach we took in this project was successful for the HMM model, as it consistently produced accurate predictions with low error rates. However, there is room for improvement, particularly for LSTM, GMM-HMM, and XGBoost models.

To enhance the performance of the models, we can consider the following strategies:

- Further tuning of hyperparameters and model architectures, especially for GMM-HMM, LSTM, and XGBoost.
- Incorporating additional features, such as technical indicators, to help the models capture more information about the market dynamics.
- Ensembling multiple models or using a combination of different models to achieve more robust and accurate predictions.

This project has provided valuable insights into the effectiveness of different models for predicting the close price of currency pairs. We learned that traditional models like HMM can outperform more modern techniques like LSTM, GMM-HMM, and XGBoost in certain scenarios, emphasizing the importance of model selection. Moreover, careful evaluation and comparison of different models are crucial to determine their suitability for the problem at hand.

One advantage of HMMs and GMM-HMMs is their ability to effectively model complex, time-dependent relationships between observable states and hidden factors influencing Forex trends, which is particularly beneficial when dealing with financial time series data. However, HMMs and GMM-HMMs may be more sensitive to initial parameter

estimates and can be computationally intensive, especially for larger datasets. On the other hand, LSTM, a type of recurrent neural network, excels at capturing long-term dependencies in time series data and can handle large datasets, but its architecture can be complex and may require substantial training time. XGBoost, a popular gradient boosting algorithm, is known for its efficiency and scalability, but it may not be as effective at capturing intricate temporal dependencies compared to HMMs, GMM-HMMs, or LSTM. In summary, while HMMs and GMM-HMMs offer valuable insights into the hidden factors driving Forex trends, their performance should be evaluated alongside LSTM and XGBoost to determine the most suitable approach for the specific problem at hand, considering factors such as data size, computational resources, and the desired level of interpretability.

For future work, we could explore other time series models like ARIMA, SARIMA, or Facebook's Prophet library. Additionally, investigating the use of deep learning models like Convolutional Neural Networks (CNNs) or Transformer-based[4] models could offer alternative solutions. We could also implement ensemble methods or stack different models to leverage the strengths of each individual model, and perform feature engineering to include more relevant information for the models to learn from, such as macroeconomic indicators or sentiment analysis data from news articles. In terms of feature selection, foreign exchange, as a product of finance, will be affected by factors such as policies and international relations. We can get new features from new finance journals and social media.

References

[1] Verma, Y. (2021, October 15). *A guide to hidden markov model and its applications in NLP*. Analytics India Magazine. Retrieved April 18, 2023, from https://analyticsindiamag.com/a-guide-to-hidden-markov-model-and-its-applications-in-nlp/

[2] Liu, M., Huo, J., Wu, Y., & Wu, J. (2021). Stock market trend analysis using hidden Markov model and long short term memory. *arXiv preprint arXiv:2104.09700*.

[3] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., ... & Zhou, T. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), 1-4.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[5] https://en.wikipedia.org/wiki/XGBoost

[6] Bao, W., Yue, J., & Rao, Y. (2017). *A deep learning framework for financial time series using stacked autoencoders and long-short term memory*. PLoS One, 12(7), 1–24.

[7] Fischer, T., & Krauss, C. (2018). *Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research*, 270(2),654–669.