Forecasting USD/EUR Exchange Rate using Arima, GARCH, VAR and RNN models

YIJING

1. Introduction

Euro was first introduced as a currency in January 1999 and was started to be used in retail transactions in January 2002. Euro has become one of the most important currencies in the foreign exchange market. The fluctuation of the euro will not only influence the economy of member countries but also the economy of other countries in the world. The fluctuation of USD/EUR exchange rate could create economic uncertainty, affect capital flows and international trade. Thus, it is crucial to forecast the change of USD/EUR exchange rate more precisely.

In this project I will use ARIMA， GARCH and VAR (Vector Autoregression Model) models to forecast the change of the USD/EUR exchange rate. Considering the macroeconomics effect, I use GDP, CPI, crude oil prices, gold prices and money supply as indicators in the VAR model. Considering that the rate is noise and non-stationary, I assume that the historical data is the result of incorporation of all the behaviors including inflation, economic growth, changes of interest rates, etc.

Then, I will build RNN (Recurrent Neural Network), which is a kind of supervised learning method, to predict the values. I plan to achieve this by using Python and a couple of libraries like, Pandas (to handle time series data), Matplotlib (for plotting various graphs), FFNET (neural network library). Our evaluation technique involves observing the actual currency value and calculating the mean squared error.

Thus, I will use historical data from 01/01/2015 to 29/02/2020 to build several forecasting models. Finally, I can select the best model based on the prediction of the future trend and calculation the accuracy of models with Root Mean Squared Error (RMSE).

2. Data Description

I obtain the data of the exchange rate of USD/EUR from the site of The European Central Bank. I obtain the data of indicators from fred.stlouisfred.org. The time frame of our dataset is 1 month starting from Jan. 2015 to Feb. 2020.

In the project, I consider money supply (in US and Europe), GDP (in US and Europe), CPI (in US and Europe), Philadelphia Gold and Silver Index (XAU) and West Texas Intermediate

(WTI) to forecast the exchange rate of USD/EUR. Where CPI is used to reflect inflation rate, XAU/USD represents the gold price and WTI/USD is oil price.

3. Theory and Methodology

3.1 Time Series Models

3.1.1 ARIMA model

The purpose of ARIMA (AutoRegressive Integrated Moving Average) models is to generalize ARMA models to analysis non-stationary time series data, by combining differencing, moving average terms and autoregresive terms. I can consider ARIMA to be a further generalization of ARMA.

I can write the general ARIMA(p, d, q) model has p autoregressive terms and q moving average terms, with d degree of differencing in the form:

$$Y_t^{(d)} = c + \sum_{i=1}^{p} \phi_i \, Y_{t-i} + \sum_{j=1}^{q} \theta_i \, \epsilon_{t-j}$$

3.1.2 ARCH model

Engle (1982) proposed the ARCH model (Auto-regressive Conditional Heteoskedastic Model).

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \ldots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \, \epsilon_{t-1}^2$$

Baillie and Bollerslev (1989) explained the variation on error terms has been changed from the constant to be a random sequence. Teräsvirta (2006) pointed out, ϵ_t has a conditional mean and variance based on the information set I_{t-1}.

$$E(\epsilon_t | I_{t-1}) = 0$$

$$\sigma_t^2 = E(\epsilon_t^2 | I_{t-1})$$

Here,

$$\epsilon_t = z_t \sigma_t$$

$$z_t \sim N(0,1)$$

So $\{\epsilon_t\}$ is a normal distribution which mean equals to zero and variance equals to σ_t^2,

$$\epsilon_t \sim N(0, \sigma_t^2)$$

Assume $\alpha_0 > 0$, and $\alpha_i >= 0$, $i = 1,\dots,q$, $\alpha_1 + \dots + \alpha_q < 1$ for ensuring $\{\sigma_t^2\}$ as weak stationary.

3.1.3 Generalized-ARCH model (GARCH)

Bollerslev (1986) and Taylor (1986) proposed the so-called generalized ARCH (GARCH) model for substituting the ARCH model.

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i\, \sigma_{t-1}^2 + \sum_{j=1}^{p} \beta_j\, \sigma_{t-j}^2$$

Alexander and Lazar (2006) assume $\alpha_0 > 0; \alpha_i >= 0, i = 1,\dots,q; \beta >= 0, j = 1,\dots,p$; $\sum_{i=1}^{q} \alpha_i + \sum_{i=1}^{p} \beta_j < 1$ for ensuring $\{\sigma_t^2\}$ as weak stationary. Enocksson and Skoog(2012) pointed out some limitations on GARCH model. The most important one is GARCH model cannot capture the asymmetric performance.

3.1.4 Multivariate GARCH model (MGARCH)

Multivariate GARCH models allow the conditional covariance matrix of the dependent variables to follow a flexible dynamic structure and allow the conditional mean to follow a vector autoregressive (VAR) structure.

The general MGARCH model can be written as

$$y_t = Cx_t + \epsilon_t$$

$$\epsilon_t = H_t^{1/2} v_t$$

where y_t is a m-vector of dependent variables, C is a m × k parameter matrix, x_t is a k-vector of explanatory variables, possibly including lags of y_t, $H_t^{1/2}$ is the Cholesky factor of the time-varying conditional covariance matrix H_t, and v_t is a m-vector of zero-mean, unit-variance i.i.d. innovations.

In this general framework, H_t is a matrix generalization of univariate GARCH models. For example, a general MGARCH (1,1)) model may be written as:

$$vech(H_t) = s + A * vech(\epsilon_{t-1}\epsilon_t') + B * vech(H_{t-1})$$

where the vech(·) function returns a vector containing the unique elements of its matrix argument. The various parameterizations of MGARCH provide alternative restrictions on H, the conditional covariance matrix, which must be positive definite for all t.

3.1.5 Vector Autoregression Model (VAR)

The VAR model is to use all current variables in the model to do regression on several lag variables of all variables. The VAR model promotes the univariate autoregressive model (AR model) by allowing multiple evolutionary variables and is often used in the joint model of multiple asset returns.

For a p-order vector autoregressive model containing n variables, it is written as VAR(p). The formula is:

$$Y_t = c + A_1 Y_{t\text{-}1} + \ldots + A_P Y_{t\text{-}P} + BX_t + \varepsilon_t$$

Among them, Yt is a k-dimensional endogenous variable vector, Xt is a d-dimensional exogenous variable vector, p is a lag order, T is the number of samples, k1*k2-dimensional matrix A1, A2 ,. . . . , Ap and k * d-dimensional matrix B is the coefficient matrix to be estimated, and t is the k-dimensional disturbance vector.

The VAR model is a model that involves the interaction between multiple variables. It can use the lag information of the sequence itself and the lag information of the related series with which it interacts, which can be effectively used for prediction.

3.1.6 Vector Error Correction Model (VEC)

The VEC model is a VAR model with co-integration constraints. It is mostly used in the modeling of non-stationary time series with co-integration relationships. According to Johnson's definition, the VAR model of n-dimensional vector Xt (including the p-order lag variables) can be expressed as :

$$X_t = A_1 X_{t\text{-}1} + A_2 X_{t\text{-}2} + \ldots + A_P Y_{t\text{-}P} + \varepsilon_t$$

Each Ai is an n * n parameter matrix. et is an independent and uniformly distributed n-dimensional vector. The above equation can be written as a vector error correction model (VECM):

$$\Delta X_t = BX_{t\text{-}1} \sum_{i=1}^{p-1} \left(- \sum_{j=i+1}^{p} A_j\right) \Delta X_{t\text{-}i} + \varepsilon_t$$

The advantage of the VEC model is that it reflects the long-term static relationship and the short-term dynamic relationship at the same time.

The premise of VEC modeling is that the sequence is integrated of the same order.

3.2 RNN Neural Network Model

Recurrent neural network (RNN) is a kind of neural network specially dealing with time series problems. It can extract information of time series, allow information persistence, and use previous knowledge to infer follow-up patterns. Figure 1 is a RNN model structure with a simplified left-hand structure on the right. Figure 2 shows the structure of RNN.

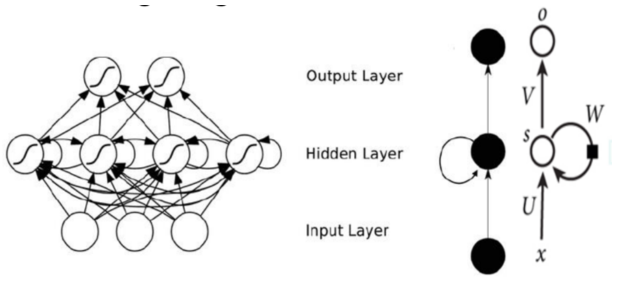


Figure 1. RNN Neural Network Model Structure.

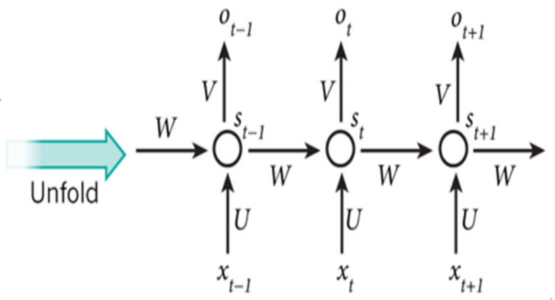


Figure 2. RNN deployment structure.

The main characteristic of RNN is that it can deal with uncertain input and get certain output. However, due to the increase of information association, RNN will not be able to learn the relationship between information, and then lose all its influence. In order to deal with the forgetfulness of RNN, LSTM structure has been created.

3.3 Long Short-Term Memory Model

In practical applications, RNN is difficult to deal with long-distance dependence. An improved cyclic neural network, Long Short-Term Memory Network (LSTM), is proposed.

LSTM is a kind of time recursive neural network (RNN), which can learn long-term dependency information. The LSTM has an additional "processor" called cell in the algorithm to determine whether information is useful or not. One cell has three doors called the input gate i_t, the forgetting gate f_t and the output gate o_t, respectively, which make the weight of the self-circulation change dynamically at different moments when parameters are fixed. At time t, every cell records the cell state C_t, which will impact the rear cell block. The cell will read inputs from two external sources: the previous hidden state h_{t-1} and the input vector x_t . The whole process is shown in Figure 3.

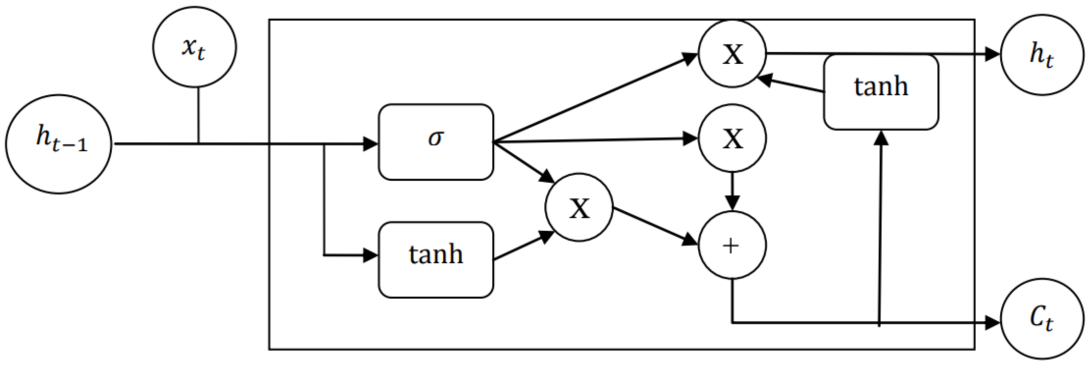


Figure 3. Overall processed for LSTM model

Firstly, I need to determine which information should be abandoned from the cell state. The forgetting gate reads h_{t-1} and x_t, and assigns a number for each value in the former cell state c_{t-1} as the output. "0" means "complete abandonment" and "1" means "complete reservation".

$$f_t = \sigma[W_f(h_{t-1}, x_t) + b_f]$$

Then, I need to decide the new information which will be stored in the cell state. Sigmoid function will decide what value I need to update i_t and tanh function will create a new candidate vector $\widetilde{C_t}$.

$$i_t = \sigma[W_i(h_{t-1}, x_t) + b_i]$$

$$\widetilde{C_t} = tanh[W_c(h_{t-1}, x_t) + b_c]$$

Afterwards, the old cell state should be updated. The first step has already determined the number f_t, which needs to multiply the old state C_{t-1}, which decides the information I need to discard. Adding the result of the second step, I can gain the new cell state C_t,

$$C_t = f_t * C_{t-1} + i_c * \widetilde{C_t}$$

Finally, the output value needs to be determined. First, I use sigmoid function to decide which part of the state cell o_t will be output. Then, o_t need multiple with the cell state that processed by tanh function as the final output h_t .

$$o_t = \sigma[W_o(h_{t-1}, x_t) + b_o]$$

$$h_t = o_t * tanh(C_t)$$

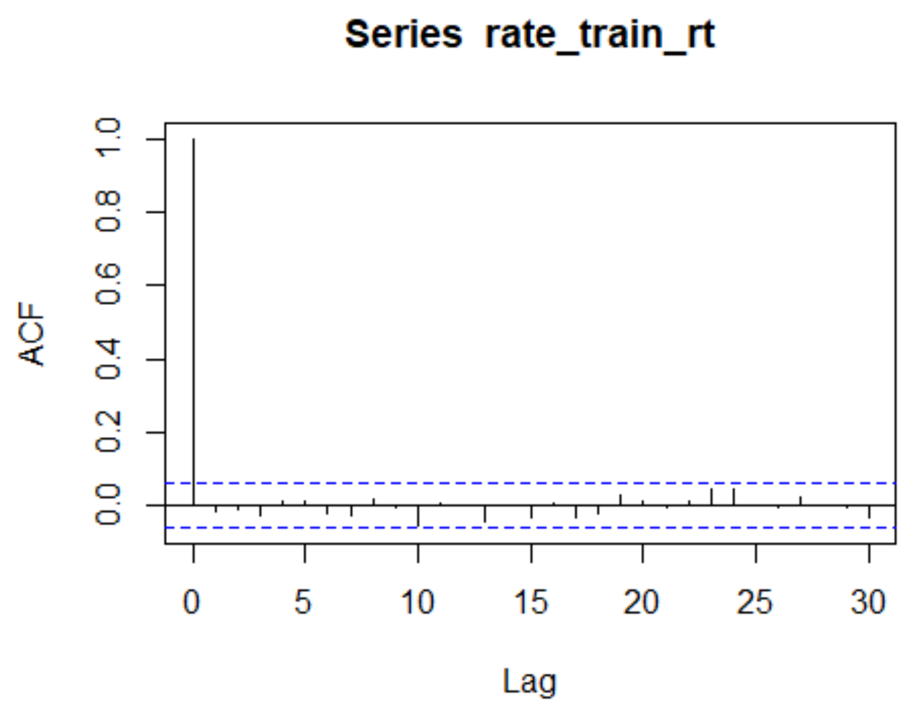
In the above equation, $\sigma()$ is the sigmoid function and tanh() is the tanh function. W_f, W_i, W_c are peep hole connection. b_f, b_i, b_c are the input weight matrix.

4. Results and Discussion

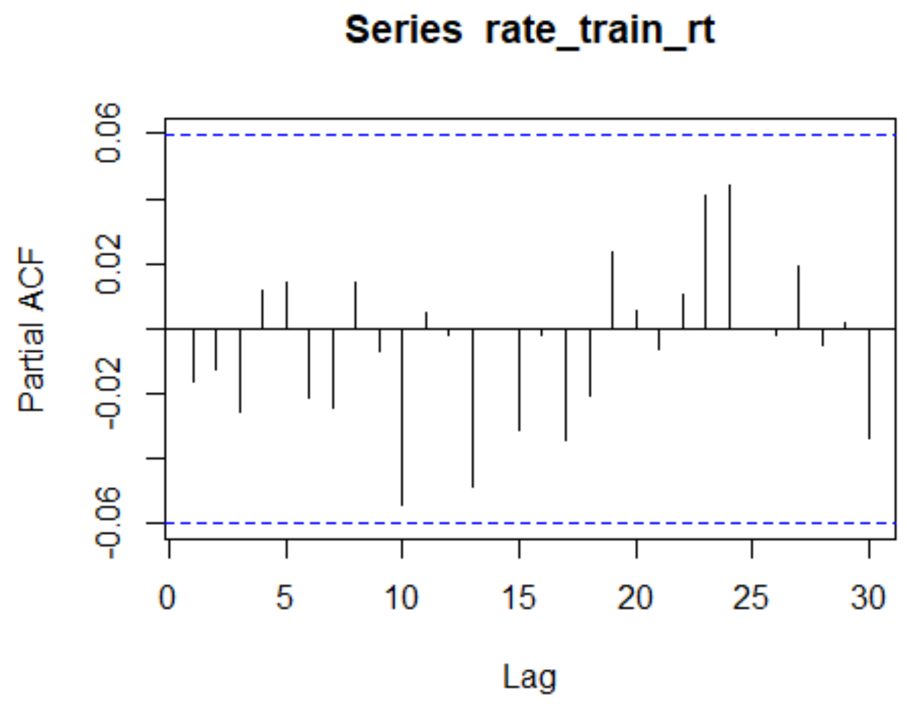
4.1 ARIMA & GARCH

Use “auto.arima” to fit a model for daily return of currency rate.
Then check out ARCH effect to build GARCH model.

```
md_arima_1 = auto.arima(rate_train)
md_arima_1

## Series: rate_train
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.8903  -0.9086
## s.e.  0.0926   0.0844
##
## sigma^2 estimated as 2.45e-05:  log likelihood=4197.75
## AIC=-8389.5   AICc=-8389.48   BIC=-8374.55

rate_train_rt = dailyReturn(rate_train, type = "log")
rate_test_rt = dailyReturn(rate_test, type = "log")
adfTest(rate_train_rt)

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -23.699
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:25:24 2020 by user: yangm

acf(rate_train_rt)
```



```r
pacf(rate_train_rt)
```

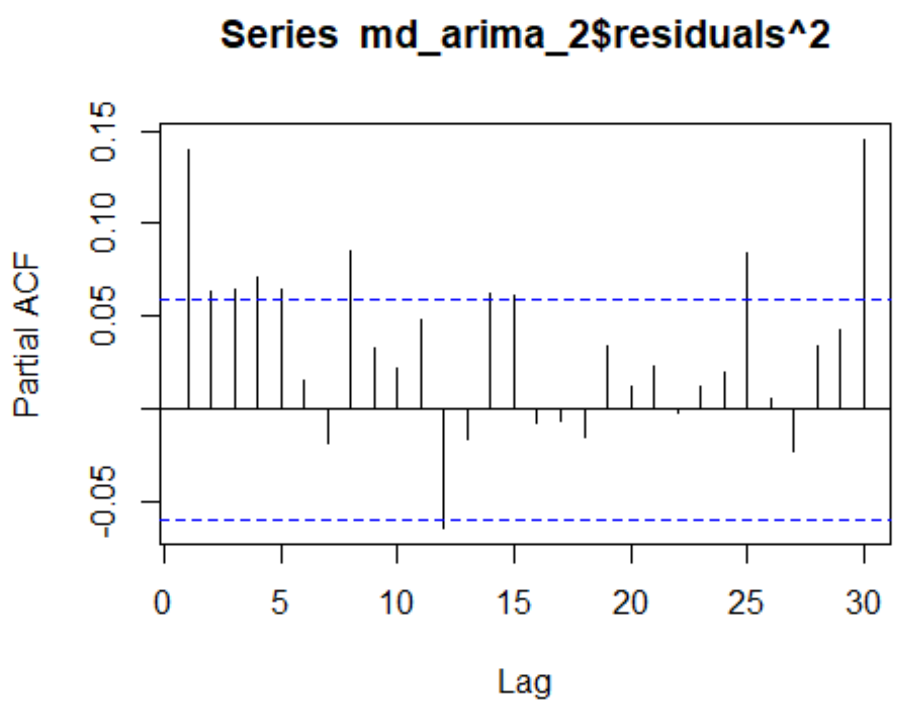


```r
md_arima_2 = auto.arima(rate_train_rt)
md_arima_2
```

```
## Series: rate_train_rt
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1
##       0.8742  -0.0013  -0.8919
## s.e.  0.1063   0.0314   0.1021
##
## sigma^2 estimated as 3.077e-05:  log likelihood=4079.17
## AIC=-8150.34   AICc=-8150.3   BIC=-8130.4
```

```r
#pred_arima = predict(md_arima_2, 263)
#rmse(predicted = as.numeric(unlist(pred_arima[1])), actual = as.numeric(unli
st(rate_test_rt)))
```

```
##
##  Box-Ljung test
##
## data:  md_arima_2$residuals^2
## X-squared = 76.58, df = 10, p-value = 2.341e-12
```



4.2 Vector Autoregression Model (VAR)

In this part, I add two key daily economic indicators: Philadelphia Gold and Silver Index (XAU) and West Texas Intermediate (WTI) to do prediction of USD/EUR exchange rate based on its relationship with others.

The premise of building a VAR model is that there is correlation between the series, then it needs to do the correlation analysis to five factors (CPI, GDP, M1, XAU, WTI) that I assume will affect USD/EUR exchange rate. The correlation coefficient of the factors (XAU, WTI) that I finally picked are basically above 0.8, so there is a strong correlation between these picked factors, which can be carried out VAR modeling.

I do the ADF test on all the factors, and both XAU and WTI have unit roots. XAU and WTI then show to be stationary after one differencing and log transformation in the Figure 4.

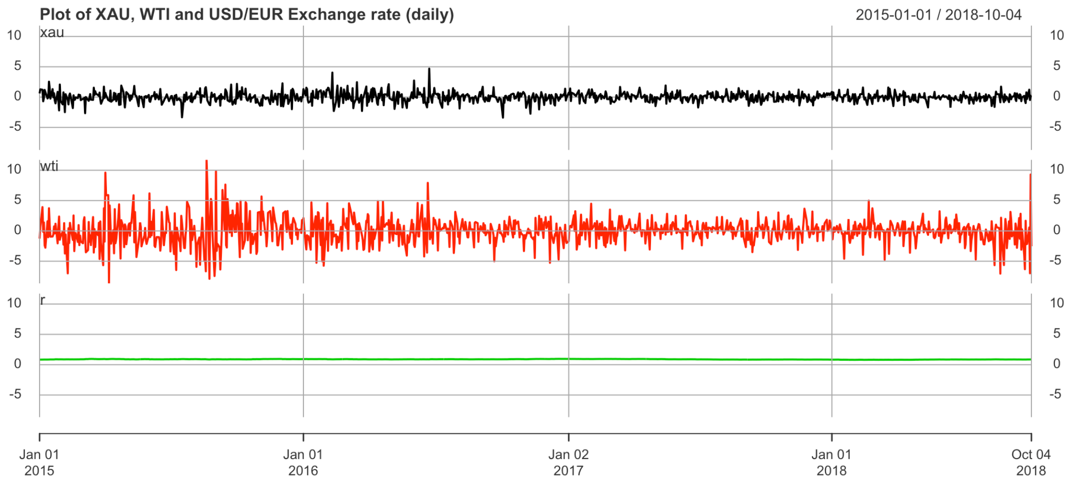


Figure 4. Plot of XAU, WTI and USD/EUR Exchange rate

The p order for the VAR model is 3, and a better VAR model is restricted model with degree of freedom 5：

$$r_t = \begin{pmatrix} 0 \\ 0 \\ 0.008 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -28.857 \\ -0.022 & 0 & 0.991 \end{pmatrix} r_{t-1} + \begin{pmatrix} 0 & 0 & 12.2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} r_{t-2} + \begin{pmatrix} 0.079 & 0 & -12.2 \\ 0 & 0 & 28.8 \\ 0 & 0.0002 & 0 \end{pmatrix} r_{t-3} + a_t$$

From the simplified model, the exchange rate of USD/EUR influenced by XAU and WTI is subject to the following model:

$$r_{3t} = 0.008\text{-}0.022 r_{1,t-1} + 0.991 r_{3,t-1} + 0.0002 r_{2,t-3} + a_{3t}$$

The established constrained VAR (3) model is based on the data from Jan. 2015 to Oct. 2018, and then I use the established model for forecasting test data from end of Oct. 2018 to Feb. 2020. The RMSE for this period achieved 0.00766455.

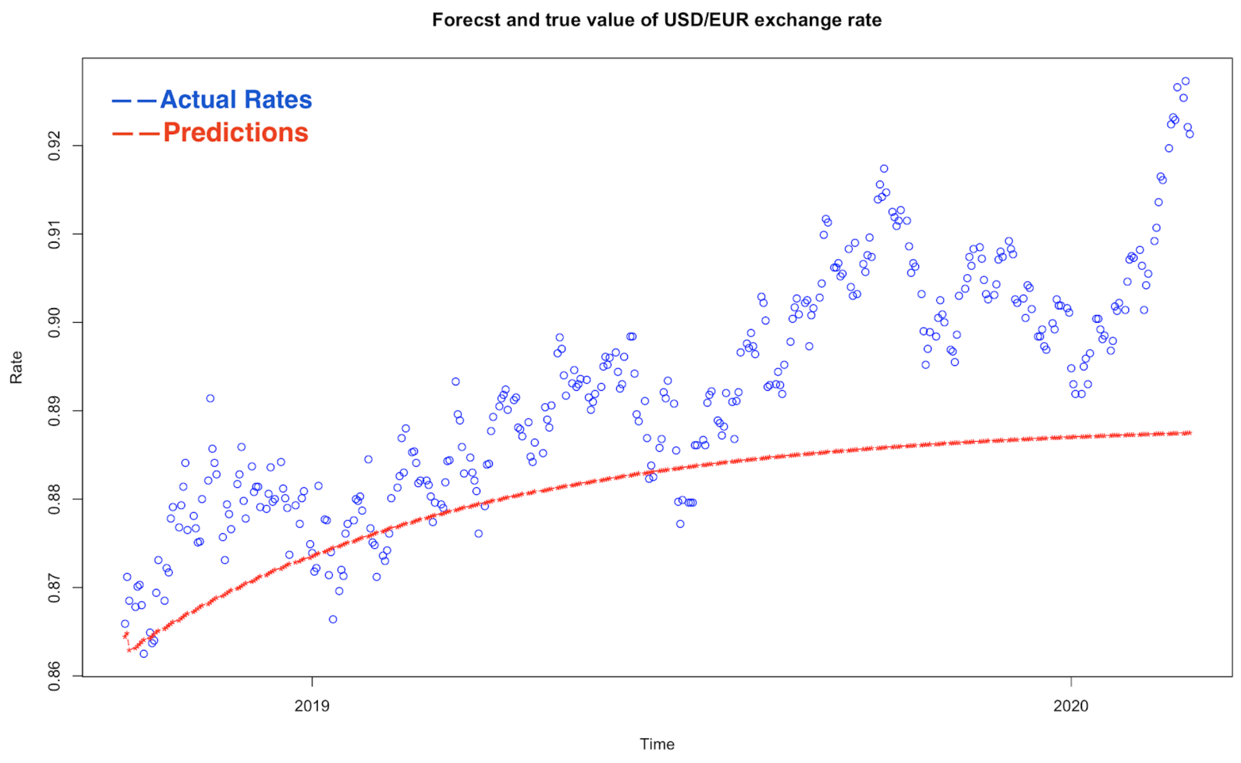


Figure 5. Graph of USD/EUR exchange rate predictions on test data

4.3 VEC

From the result of granger test, only GDP_EU_rt is significant, while it is a first order differential. So I tried cointegration test.

```
##
##  Durbin-Watson test
##
## data:  md_lm
## DW = 1.5607, p-value = 0.03119
## alternative hypothesis: true autocorrelation is greater than 0

##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -5.6952
```

```r
vecm = ca.jo(x, K = 2, ecdet = "const")
summary(vecm)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear tren
d and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 5.430146e-01 1.116981e-01 5.204170e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  5.69  7.52  9.24 12.97
## r = 0  | 37.59 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 GDP_EU.l2       RATE.l2      constant
## GDP_EU.l2  1.0000000000  1.0000000000  1.0000000000
## RATE.l2    0.0720521911 -0.0135857305 -0.0044551604
## constant  -0.0001026826  0.0002173345 -0.0009482182
##
## Weights W:
## (This is the loading matrix)
##
##               GDP_EU.l2     RATE.l2      constant
## GDP_EU.d  -0.01118603 -0.03512347  3.263710e-17
## RATE.d   -14.86184372  3.24985614 -1.146792e-14
```

To construct the VAR model for the selected endogenous variables, the co-integration test is needed. Apply durbin-watson test to linear model. As the value of p-value is small enough, indicating that the residual sequence is not independent at the significance level of 5% and has autocorrelation. According to the stationariness test results of the residual series, the null hypothesis that the residual series has unit root is rejected at the significance level of 5%, that is, the residual series is stable, indicating that there is a co-integration relationship between currency rate and GDP of EU.
From the result of Johansen procedure, I know the rank of the co-integration vector is 1. Then used cajorls() to estimate the coefficient matrix of the VEC model. Then VEC model is converted to horizontal VAR model to predict currency rate.

4.4 DCC-GARCH

Calculate the maximum, minimum, median, skewness, kurtosis and extremum

There is no ARCH effect of the monthly time series data, I decided to use the daily data, because ARCH effect shows in all three time series.

4.5 Long Short-Term Memory Model

The Long Short-Term Memory recurrent neural network has the promise of learning long sequences of observations. Therefore, it is a perfect match to deal with the problem of forecasting exchange rate. I will discover how to develop an LSTM forecast model for a univariate time series forecasting problem by following steps.

Step 1: USD/EUR Exchange rate dataset

This dataset describes the daily exchange rate between USD and EUR from 01/01/2015 to 29/02/2020. The units are exchange rate and there are 1343 observations. A line plot of the series in shown in Figure 5.

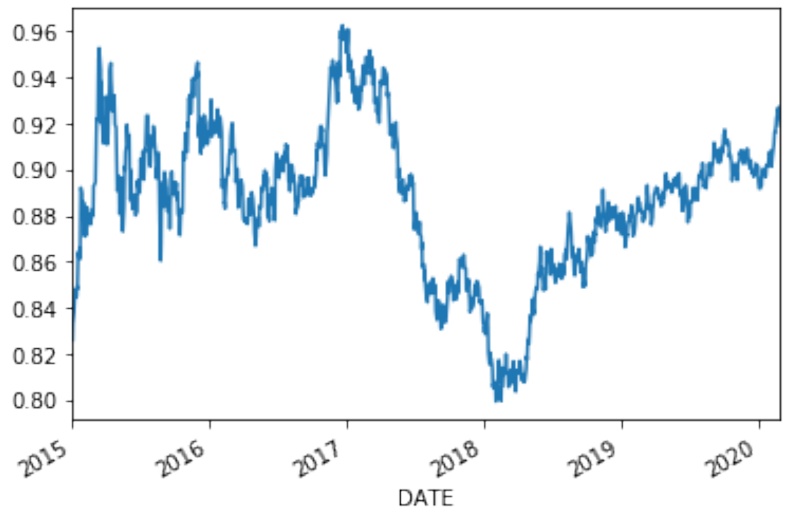


Figure 5. Line plot of the dataset

Step 2: Data preprocessing

Before I can fit an LSTM model to the dataset, I must transform the data. Preprocessing is broken down into three parts:

1) transforming the time series into a supervised learning problem
2) making series stationary
3) scaling the data
4) splitting into a training set and a test set

Firstly, I should transform the time series into a supervised learning problem. The LSTM model in Keras assumes that your data is divided into input (X) and output (y) components. For a time series problem, I can achieve this by using the observation from the last time step (t-1) as the input and the observation at the current time step (t) as the output. Then I can then concatenate these two series together to create a DataFrame ready for supervised learning.

Then, I need to transform the time series so that it is stationary. Based on analysis in time series models, exchange rate is not stationary. This means that there is a structure in the data that is dependent on the time. However, differenced series with lag=1 is stationary. So in order to result in more skillful forecasts, I need to difference the data with lag=1.

Besides, LSTMs expect data to be within the scale of the activation function used by the network. The default activation function for LSTMs is the hyperbolic tangent (tanh), which outputs values between -1 and 1. This is the preferred range for the time series data. So, I can transform the dataset to the range [-1, 1] using the MinMaxScaler class.

Finally, the first three years of data will be taken for the training dataset and the remaining one year of data will be used for the test set. So, there are 978, 365 observations respectively.

Step 3: LSTM model Development

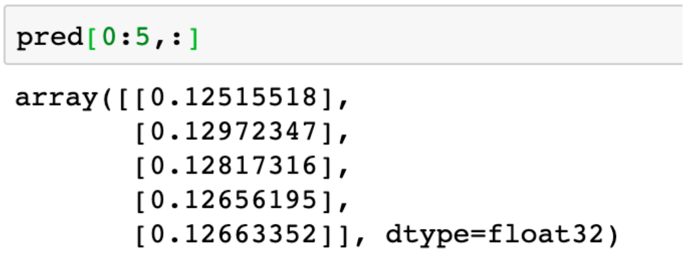
I have some flexibility in how the exchange rate dataset is framed for the network. I will keep it simple and frame the problem as each time step in the original sequence is one separate sample, with one timestep and one feature.

When I create a single LSTM hidden layer, input shape and the number of neurons I use should be specified. I identify the expected number of observations to read each batch is 1, which means every batch includes every sample. So there are 1343 batches in total. An important parameter in defining the LSTM layer is the number of neurons. This is a reasonably simple problem and a number between 1 and 5 should be sufficient. So I identify it as 4. Once the network is specified, it must be compiled into an efficient symbolic representation using a backend mathematical library, such as TensorFlow. In compiling the network, I must specify a loss function and optimization algorithm. I will use “mean_squared_error” as the loss function as it closely matches RMSE that I will be interested in, and the efficient ADAM optimization algorithm. In our code, I define a function called fit_lstm() that trains and returns an LSTM model. As arguments, it takes the training dataset in a supervised learning format, a batch size, a number of epochs, and a number of neurons. I don’t tune the network in our project. It may be included in further improvement. This time I will use the following configuration :

Batch size=1; Epochs=30; Neurons=4

Step 4: LSTM Forecast

After fitting the LSTM model to the training data, it can be used to forecast. For simplicity, I decide to fit the model once on all of the training data, then predict each new time step one at a time from the test data. That is to say, the model and predicted results are unchanged. To make a forecast, I can call the predict() function on the model. The function will separate out the input data from the test row, reshape it, and return the predicted values for each of the 365 days in the test series. Predicted results are partly shown in Figure 6.



```
pred[0:5,:]
```

```
array([[0.12515518],
       [0.12972347],
       [0.12817316],
       [0.12656195],
       [0.12663352]], dtype=float32)
```

Figure 6. Part of predicted values

Step 5: Evaluation

I decided to use Root Mean Square Error (RMSE) as an evaluation measurement method. It is a standard way to measure the error of a model in predicting quantitative data. Formally it is defined as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y_i} - y_i)^2}{n}}$$

$\hat{y_i}$ is predicted values, y_i is observed values, n is the number of observations.

Through calculation, RMSE=0.03. Figure 7 is the line plot of predicted values and true values. Two lines are so close and RMSE is small. Therefore, I conclude that our LSTM model has good performance in predicting the USD/EUR daily exchange rate.

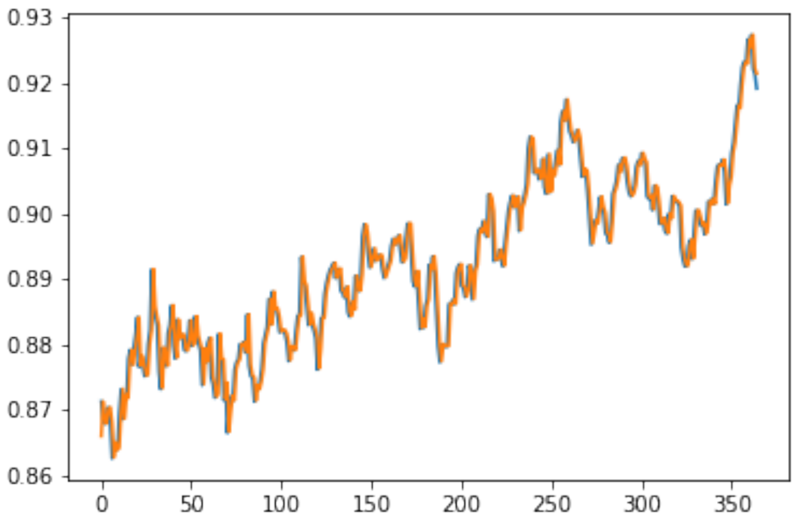
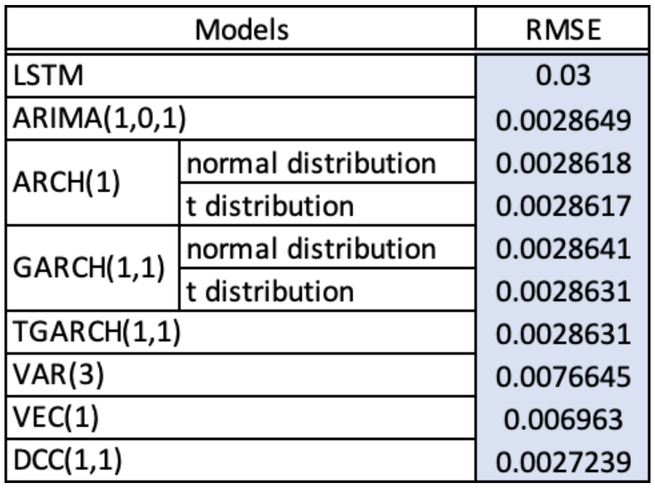


Figure 7. Line plot of predicted values and true values

5. Conclusion

Following are the RSME of each model.



Models		RMSE
LSTM		0.03
ARIMA(1,0,1)		0.0028649
ARCH(1)	normal distribution	0.0028618
	t distribution	0.0028617
GARCH(1,1)	normal distribution	0.0028641
	t distribution	0.0028631
TGARCH(1,1)		0.0028631
VAR(3)		0.0076645
VEC(1)		0.006963
DCC(1,1)		0.0027239