ผลการวิจัย และการอภิปราย

ในบทนี้จะกล่าวถึงชุดข้อมูลที่ใช้ในการวิจัย และการประเมินประสิทธิภาพของกระบวนพยากรณ์อัตราแลกเปลี่ยนค่าเงินไทยบาทต่อดอลลาร์สหรัฐอเมริกา THB/USD โดยแสดงรายละเอียดออกเป็น 2 ส่วน คือส่วนชุดข้อมูลที่ใช้ในงานวิจัย และส่วนการประเมินประสิทธิภาพ และการวิจารณ์ผลทีได้

4.1 ประเมินประสิทธิภาพ และการวิจารณ์ผลที่ได้ด้วยแบบจำลอง SARIMAX

การประเมินประสิทธิภาพแบบจำลอง SARIMAX จะเปรียบเทียบชุดข้อมูลตัวแปรภายนอกประเภทต่าง ๆ ค่าตัวแปรเชิงฤดูกาล ความสำคัญของการใช้ตัวแปรภายนอกกับผลการพยากรณ์ โดยสามารถแยกผลการทดสอบได้ดังนี้

4.1.1 ผลการทดสอบค่าพารามิเตอร์ที่เหมาะสมแบบจำลอง SARIMAX

จากแบบประเมิน 2 วิธีที่กล่าวไว้ในบทที่ 2 นำมาทดสอบในการพยากรณ์อัตราแลกเปลี่ยนสกุลเงิน THB/USD ในระยะเวลา ... เดือน เพื่อพยากรณ์ค่าเงินไทยบาท และดอลลาร์สหรัฐด้วยวิธี SARIMAX โดยเพิ่มตัวแปรปัจจัยภายนอก 1 ตัว คือ อัตรานอกเบี้ยนโยบาย

ใช้แบบจำลอง SARIMAX(0,1,1)(1,0,1,12) แทนค่าด้วยตัวแปร p=0, d=0, q=0, P=1, D=0, Q=1, Seasonality = 12เหตุผลในการเลือกใช้การแปรผลตามฤดูกาลเป็น 12 เนื่องจากข้อมูลที่นำมาจะเหมาะสมกับการใช้การแปรผันตาม

4.1.2 ผลการทดสอบนัยสำคัญของตัวแปรภายนอกด้วยแบบจำลอง SARIMAX

ชุดข้อมูลตัวแปรภายนอกที่ใช้ทดสอบในงานวิจัยนี้เพื่อพยากรณ์อัตราแลกเปลี่ยนค่าเงินไทยบาทต่อดอลลาร์สหรัฐ คือ ดอกเบี้ยนโยบาย ผลการทดลองแสดงให้เห็นว่าการใช้ตัวแปรภายนอกมีผลให้ค่าพยากรณ์อัตราแลกเปลี่ยนค่าเงินไทยบาทต่อค่าเงินดอลลาร์สหรัฐใกล้เคียงกับค่าที่ถูกต้องมากขึ้นนั้นหมายความว่าความคลาดเคลื่อนในการพยากรณ์ลดลง

ตารางประเมินประสิทธิภาพระหว่างการใช้ และไม่ใช้ตัวแปรภายนอกด้วยวิธี ARIMAX

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No predict | Actual | Predict with Exogenous | MAPE | Predict without Exogenous | MAPE |
| 1 | 30.379783 | 32.811551 | 1.71 % | 30.506989 | 1.51 % |

4.1.3 ผลการทดสอบชุดข้อมูลตัวแปรภายนอกด้วยแบบจำลอง SARIMAX

งานวิจัยของ... ซึ่งงานวิจัยนี้ได้ใช้แนวคิดในการนำข้อมูลตัวแปรภายนอกเช่นกัน บลาๆ

ผลการทดลองค่า RMASE และ MAPE ตัวแปรชนิดต่าง ๆ ด้วยวิธี SARIMAX

|  |  |  |  |
| --- | --- | --- | --- |
| ชนิดข้อมูลตัวแปรภายนอก | จำนวนเดือนที่พยากรณ์ | RMSE | MAPE |
| ดอกบี้ยนโยบาย | 229 | 0.10230 | 5.34 % |

4.1.4 ผลการทดสอบสมมติฐานตัวแปรภายนอกด้วยแบบจำลอง SARIMAX

ตารางค่าพยากรณ์สมมติฐานตัวแปรภายนอก SARIMAX

|  |  |  |
| --- | --- | --- |
| ครั้งที่พยากรณ์ | ค่าจริง | ดอกเบี้ยนโยบาย |
| 1 | 0.500000 | 0.706916 |
| 2 | 0.500000 | 0.388000 |
| 3 | 0.500000 | 0.460768 |
| 4 | 0.500000 | 0.460768 |
| 5 | 0.500000 | 0.495275 |
| 6 | 0.500000 | 0.498352 |
| 7 | 0.500000 | 0.499424 |
| 8 | 0.500000 | 0.499799 |
| 9 | 0.500000 | 0.499930 |
| 10 | 0.500000 | 0.499975 |

แสดงค่าพยากรณ์อัตราแลกเปลี่ยนค่าเงินปอนด์ต่อค่าเงินดอลลาร์สหรัฐโดยใช้ตัวแปรภายนอก ซึ่งเมื่อคำนวณค่าความผิดพลาดด้วยวิธี RMSE และ MAPE แล้ว การใช้ดอกเบี้ยนโยบายเป็นตัวแปรภายนอกจะได้ค่าความผิดพลาดน้อยที่สุดที่ RMSE เท่ากับ 0.10230 และ MAPE เท่ากับ 5.34 % ในตาราง ... ที่ได้ทำการสรุปไว้

รูปภาพประกอบด้วย ข้อความ, ภาพหน้าจอ, แผนภาพ, แผนที่

คำอธิบายที่สร้างโดยอัตโนมัติ

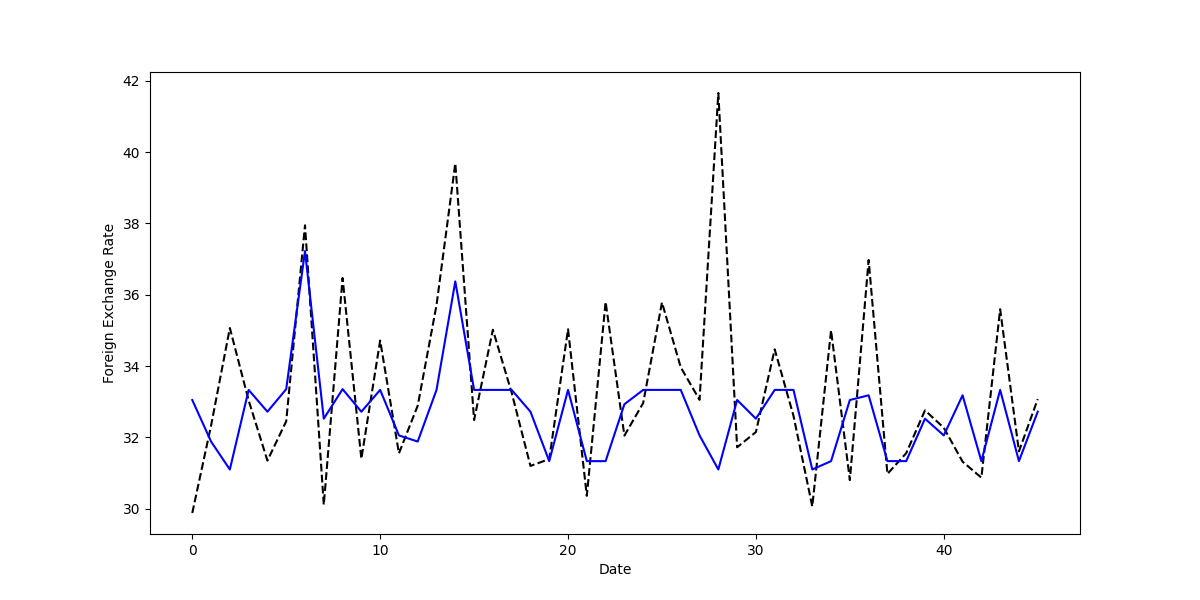
4.1.5 SVR

วิเคราะห์ความสัมพันธ์ระหว่างอัตราแลกเปลี่ยนไทยบาทต่อดอลลาร์สหรัฐ โดยสร้างโมเดล Ordinary Least Squares (OLS) และได้ค่า R-squared และ MSE คือ

|  |  |  |  |
| --- | --- | --- | --- |
| ชนิดข้อมูลตัวแปรภายนอก | จำนวนเดือนที่พยากรณ์ | R-squared | MSE |
| ดอกเบี้ยนโยบาย | 229 | 0.0755 | 6.4221 |

สร้างโมเดล SVR โดยกำหนดพารามิเตอร์ C = 1.0 และ epsilon = 0.1

ประเมินผลโมเดลด้วย RMSE และ MAPE การใช้ดอกเบี้ยนโยบายเป็นตัวแปรภายนอกจะได้ค่าความผิดพลาดน้อยที่สุดที่ RMSE เท่ากับ 2.38863 และ MAPE เท่ากับ 4.63 %



Script code ARIMA & SARIMAX

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import seaborn as sns

%matplotlib inline

import statsmodels.api as sm

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.tsa.arima.model import ARIMA

from statsmodels.tsa.statespace.sarimax import SARIMAX

from pmdarima import auto\_arima

from statsmodels.graphics.tsaplots import plot\_acf , plot\_pacf

from pandas.plotting import autocorrelation\_plot

from statsmodels.tsa.stattools import adfuller

from math import sqrt

from sklearn import preprocessing

from sklearn.metrics import r2\_score , mean\_absolute\_error , mean\_absolute\_percentage\_error , mean\_squared\_error

import pickle

import warnings

warnings.filterwarnings('ignore')

os.chdir('C:\\Users\\....\\Final\_PJ\\Data')

Forex = pd.read\_csv('USDTHB\_N.csv')

Forex.head()

Forex.shape

Forex.isnull().sum()

Forex.duplicated().sum()

Forex.dtypes

Forex.describe()

Forex['Date'] = pd.to\_datetime(Forex['Date'])

Forex.set\_index('Date',inplace = True)

Forex.plot(figsize = (10,5))

plt.title('Foreign Exchange Rate - THB to USD')

plt.savefig('Foreign Exchange Rate - THB to USD.png')

plt.show()

Forex\_week = Forex.resample('W').mean()

print('Count of The Weekly Data Frame : ',Forex\_week.shape[0])

Forex\_week.head()

Forex\_week.plot(figsize = (10,5))

plt.title('Foregin Exchange Rate (weekly) - THB to USD')

plt.savefig('Foregin Exchange Rate (weekly) - THB to USD.png')

plt.show()

Forex\_month = Forex.resample('M').mean()

print('Count of The Monthly Data Frame : ',Forex\_month.shape[0])

Forex\_month.head()

Forex\_month.plot(figsize = (10,5))

plt.title('Foregin Exchange Rate (Monthly) - THB to USD')

plt.savefig('Foregin Exchange Rate (Monthly) - THB to USD')

plt.show()

Forex\_year = Forex.resample('Y').mean()

print('Count of The Yearly Data Frame : ',Forex\_year.shape[0])

Forex\_year.head()

Forex\_year.plot(figsize = (10,5))

plt.title('Foregin Exchange Rate (Yearly) - THB to USD')

plt.savefig('Foregin Exchange Rate (Yearly) - THB to USD.png')

plt.show()

plt.rcParams['figure.figsize'] = (15,7)

sns.scatterplot(x = Forex\_month.index , y = Forex\_month.Value , color = 'black')

plt.title('Foreign Exchange Rate (weekly) - THB to USD [Scatter Plot]')

plt.savefig('Foreign Exchange Rate (weekly) - THB to USD [Scatter Plot].png')

plt.show()

sns.barplot(data = Forex\_month,x = Forex\_month.index , y = Forex\_month.Value)

plt.title('Data Spread of Foreign Exchange Rate through Bar Plot')

plt.savefig('Data Spread of Foreign Exchange Rate through Bar Plot.png')

plt.show()

sns.distplot(Forex\_month)

plt.title('Distribution of data in Foreign Exchange Rate (Weekly) - THB to USD')

plt.savefig('Distribution of data in Foreign Exchange Rate (Weekly) - THB to USD.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

Forex\_month.hist(ax = ax1)

Forex\_month.plot(kind = 'kde' , ax = ax2,c = 'r')

plt.title('Data Distribution of Foreign Exchange Rate')

plt.savefig('Data Distribution of Foreign Exchange Rate.png')

plt.show()

plt.rcParams['figure.figsize']=(12,6)

decomposition = seasonal\_decompose(Forex\_month , period = 12 , model = 'additive')

decomposition.plot()

plt.savefig('Discription , trend , seasonal , residuals.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

ax1 = plot\_pacf(Forex\_month , lags = 5 , ax = ax1)

ax2 = plot\_acf(Forex\_month , lags = 5 , ax = ax2)

plt.savefig('Partial Autocorrelation and Autocorrelation.png')

plt.show()

def adf\_check(time\_series):

result = adfuller(time\_series , autolag = 'AIC')

label = pd.Series(result[0:4], index=['Test Statistic','p-value','Number of Lags Used','Number of Observations Used'])

for key,value in result[4].items():

label['Critical Value (%s)'%key] = value

print(label)

if result[1] <= 0.05:

print('Strong evidence against the null hypothesis, hence REJECT null hypothesis and the series is Stationary')

else:

print ('Weak evidence against the null hypothesis, hence ACCEPT null hypothesis and the series is Not Stationary ')

adf\_check(Forex\_month)

Forex1\_month = Forex\_month.diff().dropna()

print('Count of monthlyly First Difference',Forex1\_month.shape[0])

Forex1\_month.head()

adf\_check(Forex1\_month)

Forex1\_month.plot(figsize = (10,5))

plt.title('Foreign Exchange Rate Weekly (First Difference) - THB to USD')

plt.savefig('Foreign Exchange Rate Weekly(First Difference) - THB to USD.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

ax1 = autocorrelation\_plot(Forex1\_month , ax = ax1)

ax1.set\_title('Non-Stationary Data')

ax2 = autocorrelation\_plot(Forex1\_month , ax = ax2)

ax2.set\_title('Stationary Data')

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Stationary data and Non-Stationary data.png')

plt.show()

model = auto\_arima(Forex\_month , m = 12, d = 1 ,seasonal = False , max\_order = 8 , test = 'adf' , trace = True)

model.summary()

model = ARIMA(Forex\_month , order = (0,1,1))

result = model.fit()

result.summary()

result.plot\_diagnostics(figsize = (10,5))

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Diagnostic plot of best model.png')

plt.show()

predictions = result.predict(typ = 'levels')

print('Evaluation Result for whole data : ','\n')

print('R2 Score for whole data : {0:.2f} %'.format(100\*r2\_score(Forex\_month['Value'],predictions)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(Forex\_month['Value'],predictions),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(Forex\_month['Value'],predictions),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(Forex\_month['Value'],predictions)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(Forex\_month['Value'],predictions)))

Final\_data = pd.concat([Forex\_month,Forex1\_month,predictions],axis=1)

Final\_data.columns = ['Foreign Exchange Rate (monthly)','Monthly First Difference','Predicted Exchange Rate']

#Final\_data.to\_csv('Foreign Exchange Rate with Prediction (THB To USD).csv')

Final\_data.head()

size = int(len(Forex\_month)\*0.80)

train , test = Forex\_month[0:size]['Value'] , Forex\_month[size:(len(Forex\_month))]['Value']

print('Counts of Train Data : ',train.shape[0])

print('Counts of Test Data : ',test.shape[0])

train\_values = [x for x in train]

prediction = []

print('Printing Predictied vs Expected Values....')

print('\n')

for t in range(len(test)):

model = ARIMA(train\_values , order = (0,1,1))

model\_fit = model.fit()

output = model\_fit.forecast()

pred\_out = output[0]

prediction.append(float(pred\_out))

test\_in = test[t]

train\_values.append(test\_in)

print('Predicted = %f , Actual = %f' % (pred\_out , test\_in))

print('Evaluation Result for Test data : ','\n')

print('R2 Score for Test data : {0:.2f} %'.format(100\*r2\_score(test,prediction)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(test,prediction),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(test,prediction),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(test,prediction)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(test,prediction)))

predictions\_df = pd.Series(prediction, index = test.index)

plt.rcParams['figure.figsize'] = (12,6)

fig, ax = plt.subplots()

ax.set(title='Foreign Exchange Rate Prediction, THB to USD', xlabel='Date', ylabel='Foreign Exchange Rate')

ax.plot(Forex\_month, 'o', label='Actual')

ax.plot(predictions\_df, 'r', label='forecast')

legend = ax.legend(loc='upper left')

legend.get\_frame().set\_facecolor('w')

plt.savefig('Foreign Exchange Rate Prediction - THB to USD.png')

Pr = pd.read\_csv('Policy\_rate\_data.csv')

Pr.head()

Pr.shape

Pr.isnull().sum()

Pr.duplicated().sum()

Pr.dtypes

Pr.describe()

Pr['Date'] = pd.to\_datetime(Pr['Date'])

Pr.set\_index('Date',inplace = True)

Pr.plot(figsize = (10,5))

plt.title('Policy Rate - THB')

plt.savefig('Policy Rate - THB to USD.png')

plt.show()

Pr\_month = Pr.resample('M').mean()

print('Count of The Monthly Data Frame : ',Pr\_month.shape[0])

Pr\_month.head()

Pr\_month.plot(figsize = (10,5))

plt.title('Policy Rate (Monthly) - THB')

plt.savefig('Policy Rate (Monthly) - THB to USD')

plt.show()

Pr\_year = Pr.resample('Y').mean()

print('Count of The Yearly Data Frame : ',Pr\_year.shape[0])

Pr\_year.head()

Pr\_year.plot(figsize = (10,5))

plt.title('Policy Rate (Yearly) - THB')

plt.savefig('Policy Rate (Yearly) - THB to USD.png')

plt.show()

plt.rcParams['figure.figsize'] = (15,7)

sns.scatterplot(x = Pr\_month.index.to\_numpy().ravel() , y = Pr\_month.values.ravel() , color = 'black')

plt.title('Policy Rate (monthly) - THB [Scatter Plot]')

plt.savefig('Policy Rate (monthly) - THB to USD [Scatter Plot].png')

plt.show()

sns.barplot(data = Pr\_month,x = Pr\_month.index , y = Pr\_month.values.ravel())

plt.title('Data Spread of Policy Rate through Bar Plot')

plt.savefig('Data Spread of Policy Rate through Bar Plot.png')

plt.show()

sns.distplot(Pr\_month)

plt.title('Distribution of data in Policy Rate (Monthly) - THB')

plt.savefig('Distribution of data in Policy Rate (Monthly) - THB to USD.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

Pr\_month.hist(ax = ax1)

Pr\_month.plot(kind = 'kde' , ax = ax2,c = 'r')

plt.title('Data Distribution of Policy Rate')

plt.savefig('Data Distribution of Policy Rate.png')

plt.show()

plt.rcParams['figure.figsize']=(12,6)

decomposition = seasonal\_decompose(Pr\_month , period = 12 , model = 'additive')

decomposition.plot()

plt.savefig('Discription , trend , seasonal , residuals.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

ax1 = plot\_pacf(Pr\_month , lags = 5 , ax = ax1)

ax2 = plot\_acf(Pr\_month , lags = 5 , ax = ax2)

plt.savefig('Partial Autocorrelation and Autocorrelation.png')

plt.show()

def adf\_check(time\_series):

result = adfuller(time\_series , autolag = 'AIC')

label = pd.Series(result[0:4], index=['Test Statistic','p-value','Number of Lags Used','Number of Observations Used'])

for key,value in result[4].items():

label['Critical Value (%s)'%key] = value

print(label)

if result[1] <= 0.05:

print('Strong evidence against the null hypothesis, hence REJECT null hypothesis and the series is Stationary')

else:

print ('Weak evidence against the null hypothesis, hence ACCEPT null hypothesis and the series is Not Stationary ')

adf\_check(Pr\_month)

Pr1\_month = Pr\_month.diff().dropna()

print('Count of monthly Frist Diference', Pr1\_month.shape[0])

Pr1\_month.head()

adf\_check(Pr1\_month)

Pr1\_month.plot(figsize = (10,5))

plt.title('Policy Rate Monthly (Frist Diference) - THB')

plt.savefig('Policy Rate Monthly (Frist Diference) - THB.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

ax1 = autocorrelation\_plot(Pr1\_month , ax = ax1)

ax1.set\_title('Non-Stationary Data')

ax2 = autocorrelation\_plot(Pr1\_month , ax = ax2)

ax2.set\_title('Stationary Data')

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Stationary data and Non-Stationary data.png')

plt.show()

model = auto\_arima(Pr\_month, m = 12, d = 1, seasonal = False, max\_order = 8, test = 'adf', trace = True)

model.summary()

model = ARIMA(Pr\_month, order = (1,1,1))

result = model.fit()

result.summary()

result.plot\_diagnostics(figsize = (10,5))

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Diagnostic Policy Rate plot of best model.png')

plt.show()

predictions = result.predict(typ = 'levels')

print('Evaluation Result for whole data : ','\n')

print('R2 Score for whole data : {0:.2f} %'.format(100\*r2\_score(Pr\_month['Policy rate'],predictions)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(Pr\_month['Policy rate'],predictions),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(Pr\_month['Policy rate'],predictions),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(Pr\_month['Policy rate'],predictions)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(Pr\_month['Policy rate'],predictions)))

Final\_data = pd.concat([Pr\_month,Pr1\_month,predictions],axis=1)

Final\_data.columns = ['Policy Rate (monthly)','Monthly First Difference','Predicted Policy Rate']

Final\_data.to\_csv('FPolicy Rate with Prediction (THB To USD).csv')

Final\_data.head()

size = int(len(Pr\_month)\*0.80)

train , test = Pr\_month[0:size]['Policy rate'] , Pr\_month[size:(len(Pr\_month))]['Policy rate']

print('Counts of Train Data : ',train.shape[0])

print('Counts of Train Data : ',test.shape[0])

train\_values = [x for x in train]

prediction = []

print('Printing Predictied vs Expected Values....')

print('\n')

for t in range(len(test)):

model = ARIMA(train\_values , order = (1,1,1))

model\_fit = model.fit()

output = model\_fit.forecast()

pred\_out = output[0]

prediction.append(float(pred\_out))

test\_in = test[t]

train\_values.append(test\_in)

print('Predicted = %f , Actual = %f' % (pred\_out , test\_in))

print('Evaluation Result for Test data : ','\n')

print('R2 Score for Test data : {0:.2f} %'.format(100\*r2\_score(test,prediction)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(test,prediction),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(test,prediction),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(test,prediction)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(test,prediction)))

predictions\_df = pd.Series(prediction, index = test.index)

plt.rcParams['figure.figsize'] = (12,6)

fig, ax = plt.subplots()

ax.set(title='Policy Rate Prediction, THB', xlabel='Date', ylabel='Policy Rate')

ax.plot(Pr\_month, 'o', label='Actual')

ax.plot(predictions\_df, 'r', label='forecast')

legend = ax.legend(loc='upper left')

legend.get\_frame().set\_facecolor('w')

plt.savefig('Policy Rate Prediction - THB to USD.png')

print(Forex.columns)

print(Pr.columns)

merged\_df = Forex\_month.merge(Pr\_month, how='inner', on='Date')

merged\_df.head()

merged\_df.shape

merged\_df.isnull().sum()

merged\_df.duplicated().sum()

merged\_df.dtypes

merged\_df.describe()

merged\_df.plot(figsize = (10,5))

plt.title('Foreign Exchange Rate & Policy Rate - THB to USD')

plt.savefig('Foreign Exchange Rate & Policy Rate - THB to USD')

plt.show()

plt.rcParams['figure.figsize'] = (15,7)

sns.scatterplot(x = merged\_df.index , y = merged\_df.Value , color = 'black')

plt.title('Foreign Exchange Rate & Policy Rate - THB to USD [Scatter Plot]')

plt.savefig('Foreign Exchange Rate & Policy Rate - THB to USD [Scatter Plot].png')

plt.show()

sns.barplot(data = merged\_df, x = merged\_df.index , y = merged\_df.Value)

plt.title('Data Spread of Foreign Exchange Rate & Policy Rate through Bar Plot')

plt.savefig('Data Spread of Foreign Exchange Rate & Policy Rate through Bar Plot.png')

plt.show()

print(merged\_df.Value)

sns.distplot(merged\_df)

plt.title('Distribution of data in Foreign Exchange Rate & Policy Rate - THB to USD')

plt.savefig('Distribution of data in Foreign Exchange Rate & Policy Rate - THB to USD.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2, ncols = 1, sharex = False, sharey = False, figsize = (10,7))

merged\_df.Value.hist(ax = ax1)

merged\_df.Value.plot(kind = 'kde' , ax = ax2,c = 'r')

plt.title('Data Distribution of Foreign Exchange Rate & Policy Rate')

plt.savefig('Data Distribution of Foreign Exchange Rate & Policy Rate.png')

plt.show()

plt.rcParams['figure.figsize']=(12,6)

decomposition = seasonal\_decompose(merged\_df.Value , period = 12 , model = 'additive')

decomposition.plot()

plt.savefig('Discription , trend , seasonal , residuals.png')

plt.show()

fig , (ax1,ax2) = plt.subplots(nrows = 2 ,ncols = 1,sharex = False , sharey = False , figsize = (10,5))

ax1 = plot\_pacf(merged\_df.Value , lags = 5 , ax = ax1)

ax2 = plot\_acf(merged\_df.Value , lags = 5 , ax = ax2)

plt.savefig('Partial Autocorrelation and Autocorrelation.png')

plt.show()

def adf\_check(time\_series):

result = adfuller(time\_series , autolag = 'AIC')

label = pd.Series(result[0:4], index=['Test Statistic','p-value','Number of Lags Used','Number of Observations Used'])

for key,value in result[4].items():

label['Critical Value (%s)'%key] = value

print(label)

if result[1] <= 0.05:

print('Strong evidence against the null hypothesis, hence REJECT null hypothesis and the series is Stationary')

else:

print ('Weak evidence against the null hypothesis, hence ACCEPT null hypothesis and the series is Not Stationary ')

adf\_check(merged\_df.Value)

merged\_df1 = merged\_df.diff().dropna()

print('Count of merged policy rate', merged\_df1.shape[0])

merged\_df1.head()

adf\_check(merged\_df1['Value'])

adf\_check(merged\_df1['Policy rate'])

merged\_df1.plot(figsize = (10,5))

plt.title('Foreign Exchange Rate with Policy rate')

plt.show()

fig, (ax1,ax2) = plt.subplots(nrows = 2, ncols = 1, sharex = False, sharey = False, figsize = (10,5))

ax1 = autocorrelation\_plot(merged\_df1, ax = ax1)

ax1.set\_title('Non-Stationary Data')

ax2 = autocorrelation\_plot(merged\_df1 , ax = ax2)

ax2.set\_title('Stationary Data')

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Stationary data and Non-Stationary data.png')

plt.show()

model\_sarimax = auto\_arima(merged\_df['Value'],

exog = merged\_df['Policy rate'],

m = 12, d = 1,

seasonal = True,

max\_order = 8,

test = 'adf',

trace = True)

model\_sarimax.summary()

model\_sarimax1 = SARIMAX(merged\_df['Value'],

order = (0, 1, 1),

seasonal\_order = (1, 0, 1, 12),

exog = merged\_df['Policy rate'],

freq = 'M',

enforce\_stationarity=False,

enforce\_invertibility=False)

result\_SARIMAX = model\_sarimax1.fit(disp = False)

result\_SARIMAX.summary()

result\_SARIMAX.plot\_diagnostics(figsize = (10,5))

plt.subplots\_adjust(hspace = 0.5)

plt.savefig('Diagnostic plot of best SARIMAX model.png')

plt.show()

predictions\_sarimax = result\_SARIMAX.predict(typ = 'levels')

print('Evaluation Result for whole data : ','\n')

print('R2 Score for whole data : {0:.2f} %'.format(100\*r2\_score(merged\_df['Value'],predictions\_sarimax)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(merged\_df['Value'],predictions\_sarimax),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(merged\_df['Value'],predictions\_sarimax),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(merged\_df['Value'],predictions\_sarimax)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(merged\_df['Value'],predictions\_sarimax)))

Final\_data = pd.concat([merged\_df, merged\_df1,

predictions\_sarimax],axis=1)

Final\_data.columns = ['Foreign Exchange Rate (monthly)',

'Policr Rate (monthly)',

'Monthly First Difference',

'Predicted Policy Rate',

'Predicted Exchange Rate']

Final\_data.to\_csv('Foreign Exchange Rate with Prediction (THB To USD).csv')

Final\_data.head()

train\_size = int(0.8 \* len(merged\_df))

test\_size = len(merged\_df) - train\_size

train\_set = merged\_df[:train\_size]

test\_set = merged\_df[train\_size:]

print('Counts of Train Data : ',train.shape[0])

print('Counts of Test Data : ',test.shape[0])

print(train\_set)

print(test\_set)

train\_values\_Poli\_add\_Date = train\_set.loc[train\_set.index]

print(train\_values\_Poli\_add\_Date)

train\_values = train\_set['Value']

train\_values

train\_values\_Poli = [y for y in train\_set['Policy rate']]

train\_values\_Poli\_copy = train\_set.copy()

train\_values\_Poli = train\_set['Policy rate'].to\_frame()

prediction = []

print('Printing Predictied vs Expected Values....')

print('\n')

for t, value in enumerate(test):

model = SARIMAX(endog = train\_values,

order = (0, 1, 1),

seasonal\_order = (1, 0, 1, 12),

exog = train\_values\_Poli,

freq = 'M',

enforce\_stationarity=False,

enforce\_invertibility=False)

model\_fit = model.fit()

policy\_model\_arima = ARIMA(train\_values\_Poli['Policy rate'],

order = (1,1,1))

policy\_model\_arima\_fit = policy\_model\_arima.fit()

future\_policy\_rates\_arima = policy\_model\_arima\_fit.forecast()

output = model\_fit.forecast(exog = future\_policy\_rates\_arima)

pred\_out = output[0]

prediction.append(float(pred\_out))

train\_values = []

train\_values.append(test[t])

print('Predicted = %f, Actual = %f' % (pred\_out, train\_values[-1]))

Script code SVR Model

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import os

import seaborn as sns

%matplotlib inline

import statsmodels.api as sm

from statsmodels.tsa.seasonal import seasonal\_decompose

from statsmodels.tsa.arima.model import ARIMA

from statsmodels.tsa.statespace.sarimax import SARIMAX

from pmdarima import auto\_arima

from statsmodels.graphics.tsaplots import plot\_acf , plot\_pacf

from pandas.plotting import autocorrelation\_plot

from statsmodels.tsa.stattools import adfuller

from math import sqrt

from sklearn import preprocessing

from sklearn.metrics import r2\_score , mean\_absolute\_error , mean\_absolute\_percentage\_error , mean\_squared\_error

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.model\_selection import train\_test\_split

import pickle

import warnings

warnings.filterwarnings('ignore')

import os

os.chdir('C:\\Users\\...\\Final\_PJ\\Data')

usdthb = pd.read\_csv('Merged Data.csv')

usdthb.head()

usdthb['Date'] = pd.to\_datetime(usdthb['Date'])

usdthb.set\_index('Date',inplace = True)

model = sm.OLS(usdthb['Value'], sm.add\_constant(usdthb['Policy rate']))

results = model.fit()

print('R\_Square:', format(results.rsquared))

print('MSE:', format(results.mse\_resid))

print(results.summary())

X = usdthb['Policy rate']

y = usdthb['Value']

print(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

X\_train = pd.DataFrame(X\_train)

y\_train = pd.DataFrame(y\_train)

model = SVR()

model.set\_params(C=1.0, epsilon=0.1)

model.fit(X\_train, y\_train)

X\_test = pd.DataFrame(X\_test)

y\_pred = model.predict(X\_test)

print('Evaluation Result for whole data : ','\n')

print('R2 Score for whole data : {0:.2f} %'.format(100\*r2\_score(y\_test, y\_pred)),'\n')

print('Mean Squared Error : ',mean\_squared\_error(y\_test, y\_pred),'\n')

print('Mean Absolute Error : ',mean\_absolute\_error(y\_test, y\_pred),'\n')

print('Root Mean Squared Error : ',sqrt(mean\_squared\_error(y\_test, y\_pred)),'\n')

print('Mean Absolute Percentage Error : {0:.2f} %'.format(100\*mean\_absolute\_percentage\_error(y\_test, y\_pred)))

predictions\_df = pd.Series(y\_pred, index = X\_test.index)

print(predictions\_df)

plt.rcParams['figure.figsize'] = (12,6)

x\_ax = range(len(X\_test))

plt.plot(x\_ax, y\_test, label = 'Actual', color = 'k', linestyle = '--')

plt.plot(x\_ax, y\_pred, label = 'SVR', color = 'b', linestyle = '-')

plt.ylabel('Foreign Exchange Rate')

plt.xlabel('Date')

plt.savefig('Foreign Exchange Rate SVR - THB to USD.png')