EBirr to USD Exchange Rate Forecasting

Using Linear Regression

Abraham Ararsa and Belete Tekle April , 2021

ETBirr to USD Exchange Rate Forecasting

Agendas

- Project Objectives
- Data
- Employed Algorithm
- Experiments
- Final Results Best Result

Objective

- The Main objective is for this project is to forecast the exchange rate between Ethiopian currency(EBirr) and United States Dollar(USD).
- There are different factors that influence ones country currency but for this project we have considered the following factors only
  - Previous exchange rate
  - GDP
  - Political Stability Index
  - Country Dept
  - Export Import Rate

Data

- We have collected data 10 years of data from different sources
- When , we collect the data it was not compatibility , so We have preprocessed the data before working with it.
- Data-Engineering applied
  - The Date was not in acceptable format with our model,
  - Some data was yearly, so we have propagate that data to yearly
  - Some data was missing(specially current year data), so we have calculated as a mean from the previous data.

Data

Visualization

	Date	Price	Open	High	Low	Change %	et_Import	et_Export	Eth_Imp_Exp_Binc	GDB	Debt	PST
0	Apr 15, 2021	41.6969	41.6838	41.6969	41.3550	0.20%	39.0	32.0	-7.0	92.00	58.70	1.33
1	Apr 14, 2021	41.6150	41.6676	41.6838	41.3500	0.08%	39.0	32.0	-7.0	92.00	58.70	1.33
2	Apr 13, 2021	41.5825	41.6541	41.6905	41.3500	0.06%	39.0	32.0	-7.0	92.00	58.70	1.33
3	Apr 12, 2021	41.5555	41.6377	41.6888	41.3250	0.08%	39.0	32.0	-7.0	92.00	58.70	1.33
4	Apr 09, 2021	41.5227	41.6245	41.6674	41.5962	0.68%	39.0	32.0	-7.0	92.00	58.70	1.33
...	...	...	...	...	...	...	...	...	...	...	...	...
2679	Jan 07, 2011	16.6415	16.5590	16.6415	16.5590	0.00%	137.5	7.6	-129.9	31.95	2.56	1.51
2680	Jan 06, 2011	16.6420	16.5560	16.6990	16.5560	0.02%	137.5	7.6	-129.9	31.95	2.56	1.51
2681	Jan 05, 2011	16.6390	16.5530	16.6960	16.5530	0.02%	137.5	7.6	-129.9	31.95	2.56	1.51
2682	Jan 04, 2011	16.6360	16.5510	16.6360	16.5510	0.02%	137.5	7.6	-129.9	31.95	2.56	1.51
2683	Jan 03, 2011	16.6335	16.5480	16.6335	16.5480	0.02%	137.5	7.6	-129.9	31.95	2.56	1.51

Data

Required Data for Prediction

[201]:
```
1 req_Data = Data[['Date' , 'Price' , 'Eth_Imp_Exp_Blnc' , "GDB" , 'Debt' , 'PST']]
```

[202]:
```
1 req_Data
```

t[202]:

	Date	Price	Eth_Imp_Exp_Blnc	GDB	Debt	PST
0	Apr 15, 2021	41.6969	-7.0	92.00	58.70	1.33
1	Apr 14, 2021	41.6150	-7.0	92.00	58.70	1.33
2	Apr 13, 2021	41.5825	-7.0	92.00	58.70	1.33
3	Apr 12, 2021	41.5555	-7.0	92.00	58.70	1.33
4	Apr 09, 2021	41.5227	-7.0	92.00	58.70	1.33
...	...	...	...	...	...	...
2679	Jan 07, 2011	16.6415	-129.9	31.95	2.56	1.51
2680	Jan 06, 2011	16.6420	-129.9	31.95	2.56	1.51
2681	Jan 05, 2011	16.6390	-129.9	31.95	2.56	1.51
2682	Jan 04, 2011	16.6360	-129.9	31.95	2.56	1.51
2683	Jan 03, 2011	16.6335	-129.9	31.95	2.56	1.51

2684 rows × 6 columns

Data

Required Data Description

In [230]:

```
#explain the data
req_Data.describe()
```

Out[230]:

	Price	Eth_Imp_Exp_Blnc	GDB	Debt	PST
count	2684.000000	2684.000000	2684.000000	2684.000000	2684.00000
mean	23.671885	-61.451602	68.197949	37.526788	-0.78598
std	6.214876	91.861573	21.348068	16.937466	1.23324
min	16.633500	-426.300000	31.950000	2.560000	-1.68000
25%	18.805250	-119.000000	47.650000	28.600000	-1.50000
50%	21.413000	-25.700000	74.300000	34.900000	-1.34000
75%	27.716525	-0.200000	84.270000	57.000000	-1.27000
max	41.696900	37.500000	95.910000	60.000000	1.56000

Data

Divide data to Dependent(Y) and Independent (Y)

In [218]:
```
1 #prepare the data in X (independant variable ) and Y (dependannt variable)
```

In [219]:
```
1 x = req_Data[['Date','Eth_Imp_Exp_Blnc' , 'GDB' , 'Debt' , 'PST']]
```

In [220]:
```
1 x['Date'] = x['Date'].map(datetime.toordinal)
```

```
<ipython-input-220-fcd9041cbd55>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
ng-a-view-versus-a-copy
  x['Date'] = x['Date'].map(datetime.toordinal)
```

In [221]:
```
1 x
```

Out[221]:

	Date	Eth_Imp_Exp_Blnc	GDB	Debt	PST
0	737895	-7.0	92.00	58.70	1.33
1	737894	-7.0	92.00	58.70	1.33
2	737893	-7.0	92.00	58.70	1.33
3	737892	-7.0	92.00	58.70	1.33
4	737889	-7.0	92.00	58.70	1.33
...	...	...	...	...	...
2679	734144	-129.9	31.95	2.56	1.51
2680	734143	-129.9	31.95	2.56	1.51
2681	734142	-129.9	31.95	2.56	1.51
2682	734141	-129.9	31.95	2.56	1.51
2683	734140	-129.9	31.95	2.56	1.51

2684 rows × 5 columns

Data

Divide data to Dependent(Y) and Independent (X)

In [309]:
```
price_Data = req_Data.filter(['Price'])

```

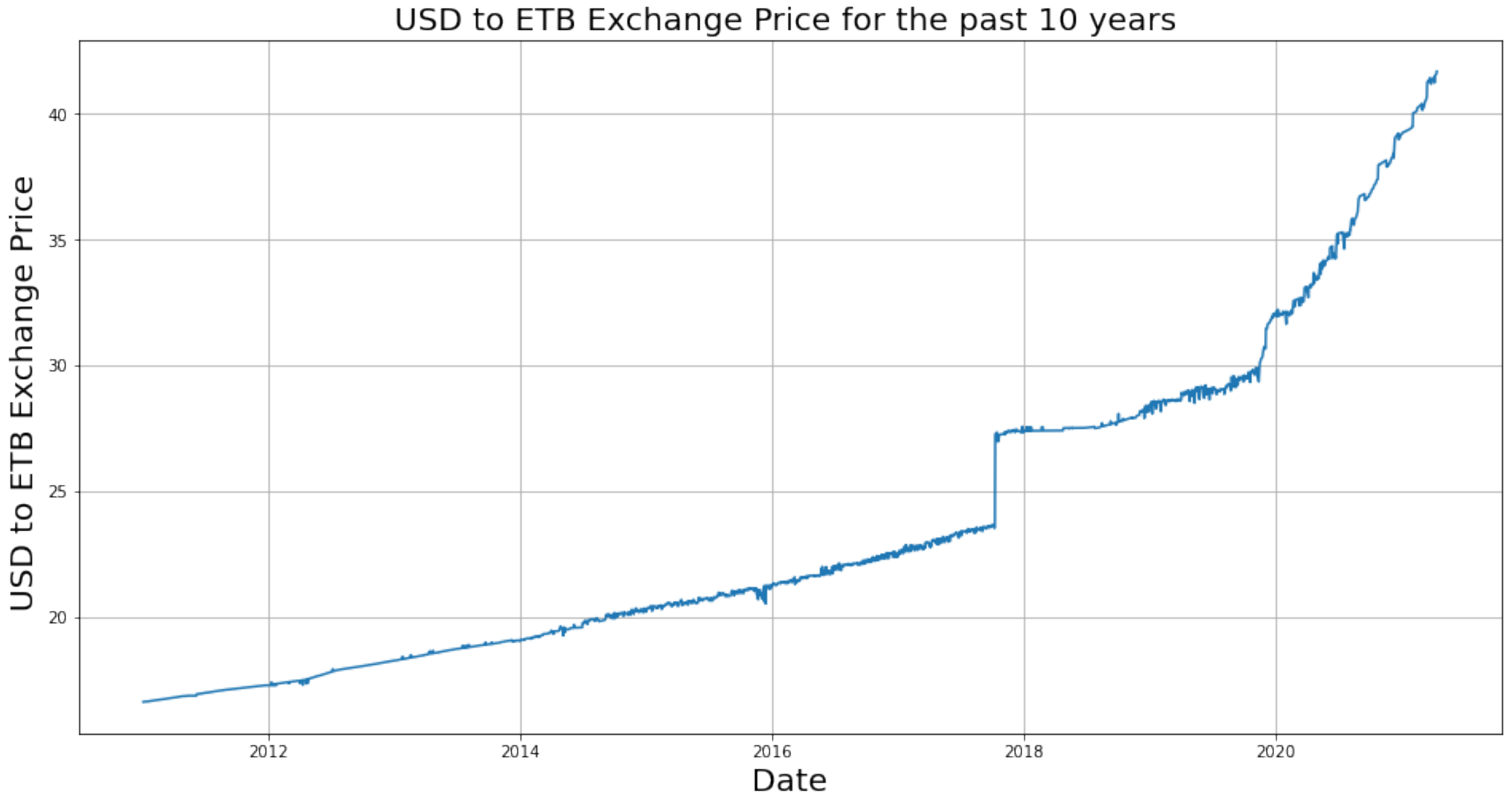
In [ ]:
```

```

In [216]:
```
#change to numpy array
price_Data = price_Data.values
price_Data
```

Out[216]:
```
array([[41.6969],
       [41.615 ],
       [41.5825],
       ...,
       [16.639 ],
       [16.636 ],
       [16.6335]])
```

Data

Visualize the Trend in Exchange



Employed Algorithm

Linear Regression

- We have employed Linear Regression for this Assignment
- What is Linear Regression
  - Is a type of machine learning that try to predict dependent(unknown) variable from (independent)known variable using the previous correlation(parameter/ coefficient).
  - The main mat formula for LR

$$Y_i = f(X_i, \beta) + e_i$$

Y_i = dependent variable

f = function

X_i = independent variable

β = unknown parameters

e_i = error terms

Data Source

- World Bank
- National Bank of Ethiopia
- Census.com
- Theglobaleconomy.COM
- WalleInestor.com
- Investing.com
- Tradingeconomy.com

Experiments

- We have made different experiments to find a best accuracy.
- The only difference for this difference experiment was Data size , Sequence
- Why different experiment?
  - Initiation : first round experiment was bad

Experiments

First Stage

- Data used : All data (with 80/20 percent)
- We used sklearn : train_test_split

In [294]:
```
#lets split the data into train and test data 80/20
# we use the sklearn library

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

```

In [295]:
```
x_train.shape , x.shape , x_test.shape
```

Out[295]:
```
((2147, 5), (2684, 5), (537, 5))
```

Experiments

First Stage | Result (Low Accuracy)

In [301]:
```
pd.DataFrame({'Actual': y_test, 'Predicted': y_pred , "Difference" : y_test - y_pred})
```

	Actual	Predicted	Difference
1327	21.4000	21.223108	0.176892
1282	21.6408	21.870278	-0.229478
2423	17.2940	16.128189	1.165811
1835	19.4095	18.894084	0.515416
418	29.1300	30.140802	-1.010802
...	...	...	...
1225	22.1050	22.959774	-0.854774
712	27.5678	29.083670	-1.515870
2637	16.7530	15.516004	1.236996
1161	22.4401	23.882163	-1.442063
1421	21.1561	22.776006	-1.619906

537 rows × 3 columns

In [302]:
```
Max_Error = max(y_test - y_pred)

Min_Error = min(y_test - y_pred)

Max_Error , Min_Error
```

Out[302]: (3.6014022460709256, -2.4148432464778935)

Experiment II

Without splitter

- Since the last experiment was not good, we used thought the problem was on splitting(because it was random, and the date order was matters for our case)
- Method used : we try to experiment in phases
  - Phase 1 : we used the first 4 years data(2011 - 2014) and try to predict Jan 2015
    - Output Accuracy : very accurate compared to previous
  - Phase 2; we used the first 8 years data(2011 - 2017) and try to predict Jan 2018
    - OUTPUT Accuracy : lower than the previous (4 years)
  - Phase 3 : we used the >99% of the data : but ordered in date
    - OUTPUT Accuracy : better than 8 years data but lower than

Method 1 : Output

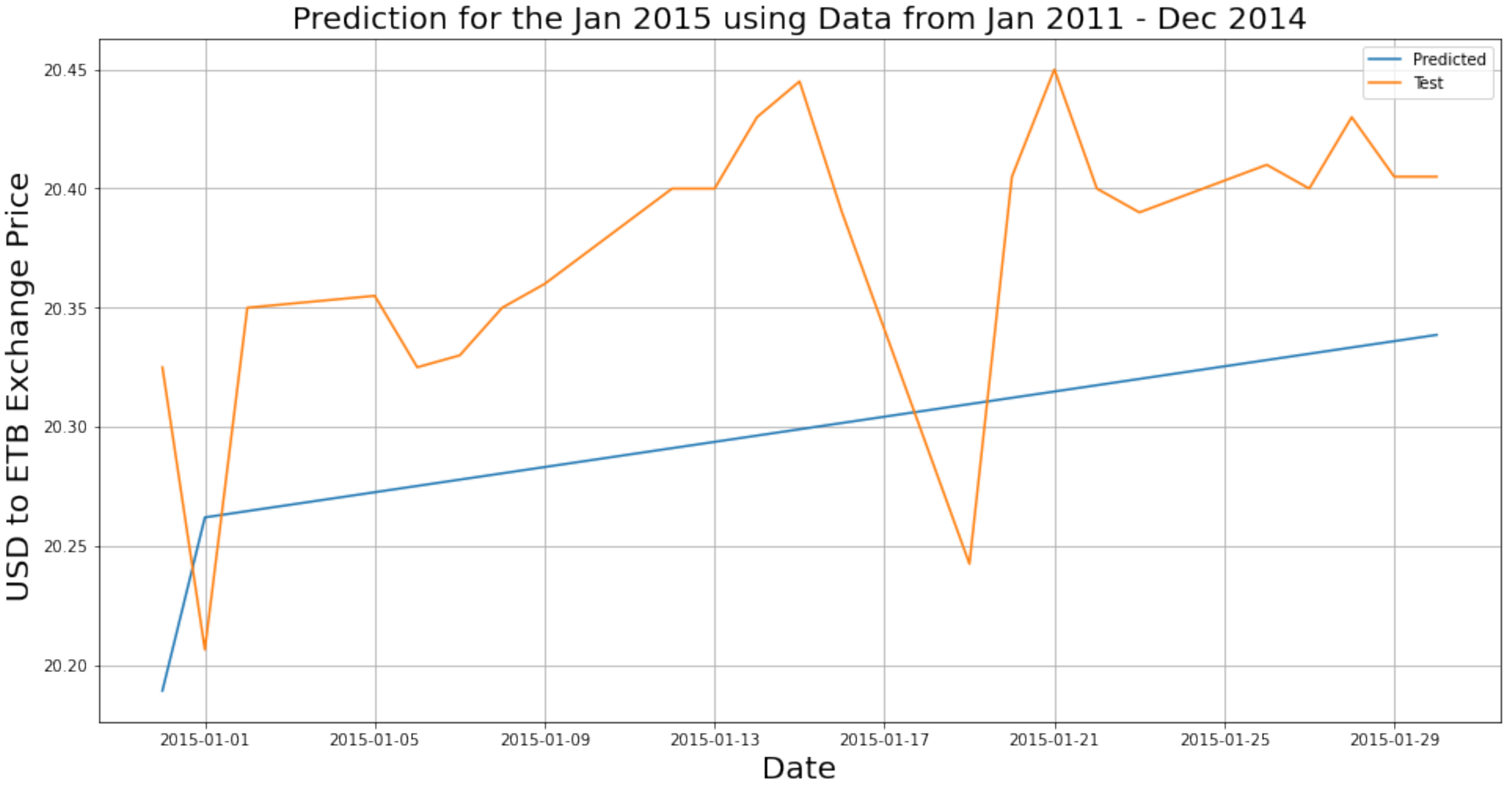
In [53]:
```
pd.DataFrame({'Actual': y_test1, 'Predicted': y_pred1 , "Difference" : y_test1 - y_pred1})
```

Out[53]:

	Actual	Predicted	Difference
1619	20.4050	20.338602	0.066398
1620	20.4050	20.335960	0.069040
1621	20.4300	20.333318	0.096682
1622	20.4000	20.330677	0.069323
1623	20.4100	20.328035	0.081965
1624	20.3900	20.320110	0.069890
1625	20.4000	20.317469	0.082531
1626	20.4500	20.314827	0.135173
1627	20.4050	20.312185	0.092815
1628	20.2425	20.309544	-0.067044
1629	20.3900	20.301619	0.088381

In [59]:
```
Max_Error = max(y_test1 - y_pred1)

Min_Error = min(y_test1 - y_pred1)

Max_Error , Min_Error
```

Out[59]: (0.14602301513998128, -0.06704359530164794)


Prediction for the Jan 2015 using Data from Jan 2011 - Dec 2014
Predicted
Test
USD to ETB Exchange Price
20.45
20.40
20.35
20.30
20.25
20.20
2015-01-01
2015-01-05
2015-01-09
2015-01-13
2015-01-17
2015-01-21
2015-01-25
2015-01-29
Date

Method 2 : Output

In [73]:
```
1 pd.DataFrame({'Actual': y_test2, 'Predicted': y_pred2 , "Difference" : y_test2 - y_pred2})
```

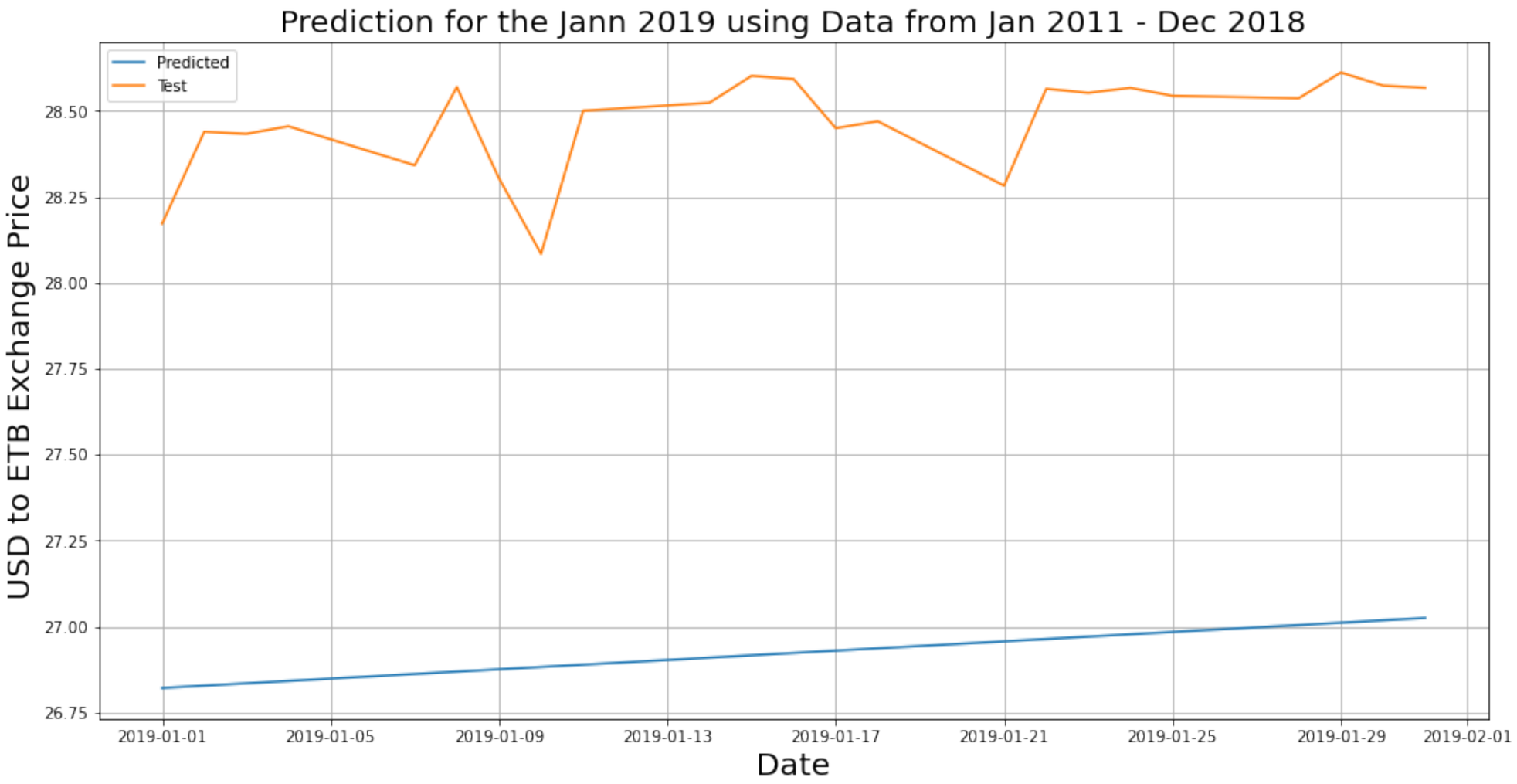
Out[73]:

	Actual	Predicted	Difference
575	28.5676	27.024660	1.542940
576	28.5739	27.017875	1.556025
577	28.6120	27.011090	1.600910
578	28.5373	27.004304	1.532996
579	28.5442	26.983948	1.560252
580	28.5674	26.977163	1.590237
581	28.5528	26.970378	1.582422
582	28.5647	26.963592	1.601108
583	28.2828	26.956807	1.325993
584	28.4700	26.936451	1.533549
585	28.4500	26.929666	1.520334
586	28.5930	26.922881	1.670119
587	28.6022	26.916095	1.686105
588	28.5240	26.909310	1.614690
589	28.5005	26.888954	1.611546
590	28.0845	26.882169	1.202331
591	28.3043	26.875383	1.428917
592	28.5699	26.868598	1.701302
593	28.3422	26.861813	1.480387
			1.614043
595	28.4337	26.834671	1.599029
596	28.4398	26.827886	1.611914
597	28.1724	26.821101	1.351299

click to scroll output; double click to hide

In [75]:
```
1 Max_Error = max(y_test2 - y_pred2)
2
3 Min_Error = min(y_test2 - y_pred2)
4
5 Max_Error , Min_Error
```

Out[75]: (1.7013020741301652, 1.2023314278922683)


Prediction for the Jann 2019 using Data from Jan 2011 - Dec 2018
Predicted
Test
USD to ETB Exchange Price
28.50
28.25
28.00
27.75
27.50
27.25
27.00
26.75
2019-01-01
2019-01-05
2019-01-09
2019-01-13
2019-01-17
2019-01-21
2019-01-25
2019-01-29
2019-02-01
Date

Method 3: Output

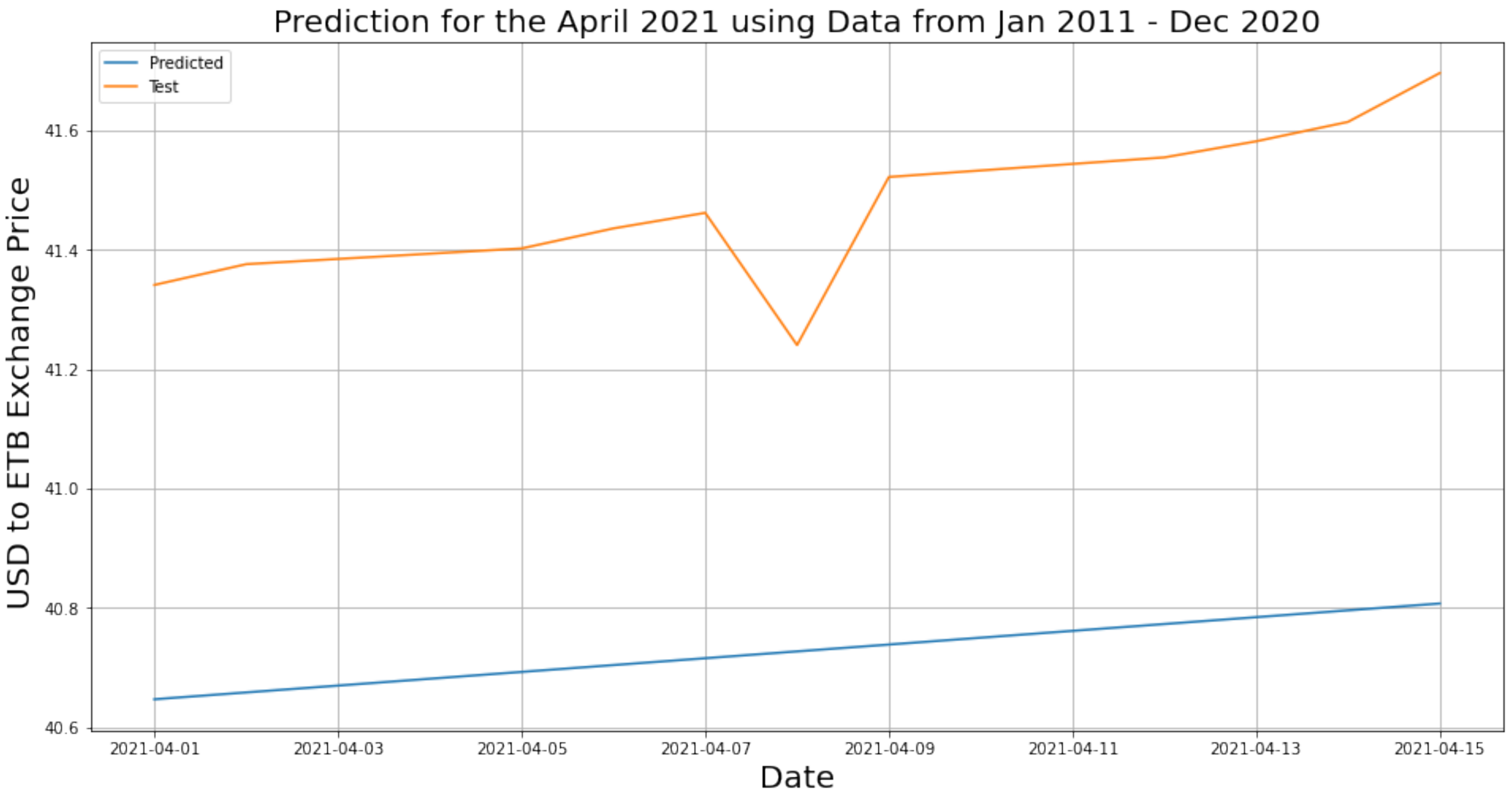
In [105]:
```
1 pd.DataFrame({'Actual': y_test3, 'Predicted': y_pred3 , "Difference" : y_test3 - y_pred3})
```

Out[105]:

	Actual	Predicted	Difference
0	41.6969	40.807666	0.889234
1	41.6150	40.796198	0.818802
2	41.5825	40.784729	0.797771
3	41.5555	40.773261	0.782239
4	41.5227	40.738856	0.783844
5	41.2409	40.727388	0.513512
6	41.4626	40.715919	0.746681
7	41.4365	40.704451	0.732049
8	41.4029	40.692983	0.709917
9	41.3765	40.658578	0.717922
10	41.3417	40.647109	0.694591

In [108]:
```
1 Max_Error3 = max(y_test3 - y_pred3)
2
3 Min_Error3 = min(y_test3 - y_pred3)
4
5 Max_Error3 , Min_Error3
```

Out[108]: (0.8892341311012331, 0.5135124648597795)


Prediction for the April 2021 using Data from Jan 2011 - Dec 2020
Predicted
Test
USD to ETB Exchange Price
41.6
41.4
41.2
41.0
40.8
40.6
2021-04-01
2021-04-03
2021-04-05
2021-04-07
2021-04-09
2021-04-11
2021-04-13
2021-04-15
Date

Final Result | Best result

Experiment 3

- What we have seen was unexplained variation in accuracy so we tried to predict the last month(April) same us experiment method 3 phase 3 but with only the 2021 data(3 month data) and the Accuracy was very good.

Final Result

In [37]:
```
pd.DataFrame({'Actual': y_test1, 'Predicted': y_pred1 , "Difference" : y_test1 - y_pred1})
```

Out[37]:

	Actual	Predicted	Difference
0	41.6969	41.753696	-0.056796
1	41.6150	41.722429	-0.107429
2	41.5825	41.691162	-0.108662
3	41.5555	41.659894	-0.104394
4	41.5227	41.566093	-0.043393
5	41.2409	41.534825	-0.293925
6	41.4626	41.503558	-0.040958
7	41.4365	41.472291	-0.035791
8	41.4029	41.441024	-0.038124

In [38]:
```
Max_Error = max(y_test1 - y_pred1)

Min_Error = min(y_test1 - y_pred1)

Max_Error , Min_Error
```

Out[38]: (-0.03579093912028242, -0.29392541029734076)

Final Result

Graph Representation