Untitled41

June 1, 2023

[31]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from scipy.stats import shapiro
import statsmodels.api as sm
from sklearn.preprocessing import MinMaxScaler
import scipy.stats as stats
from numpy import asarray
from pandas import DataFrame
from pandas import concat
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from xgboost import XGBRegressor
from matplotlib import pyplot
import math
from statistics import mean
import pandas as pd
from sklearn.metrics import mean_squared_error

import statsmodels.api as sm
from pmdarima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse
from sklearn.metrics import mean_squared_log_error
#from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_absolute_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.statespace.tools import diff
```

[2]:
```python
df = pd.read_csv('rate.csv')
df['Date'] = pd.to_datetime(df['Date'],
                        infer_datetime_format = True)
df.set_index('Date', inplace = True)
```

```python
df.head()
```

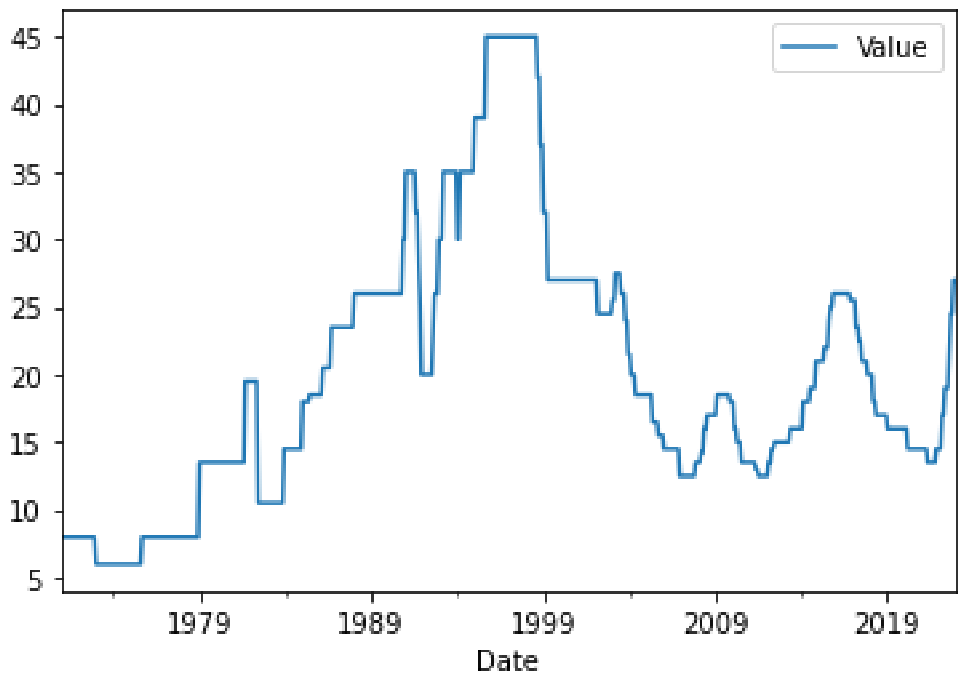
[2]:
```
            Value
Date
1971-01-01    8.0
1971-02-01    8.0
1971-03-01    8.0
1971-04-01    8.0
1971-05-01    8.0
```

[3]:
```python
df.tail()
```

[3]:
```
            Value
Date
2022-08-01   22.0
2022-09-01   24.5
2022-10-01   24.5
2022-11-01   27.0
2022-12-01   27.0
```

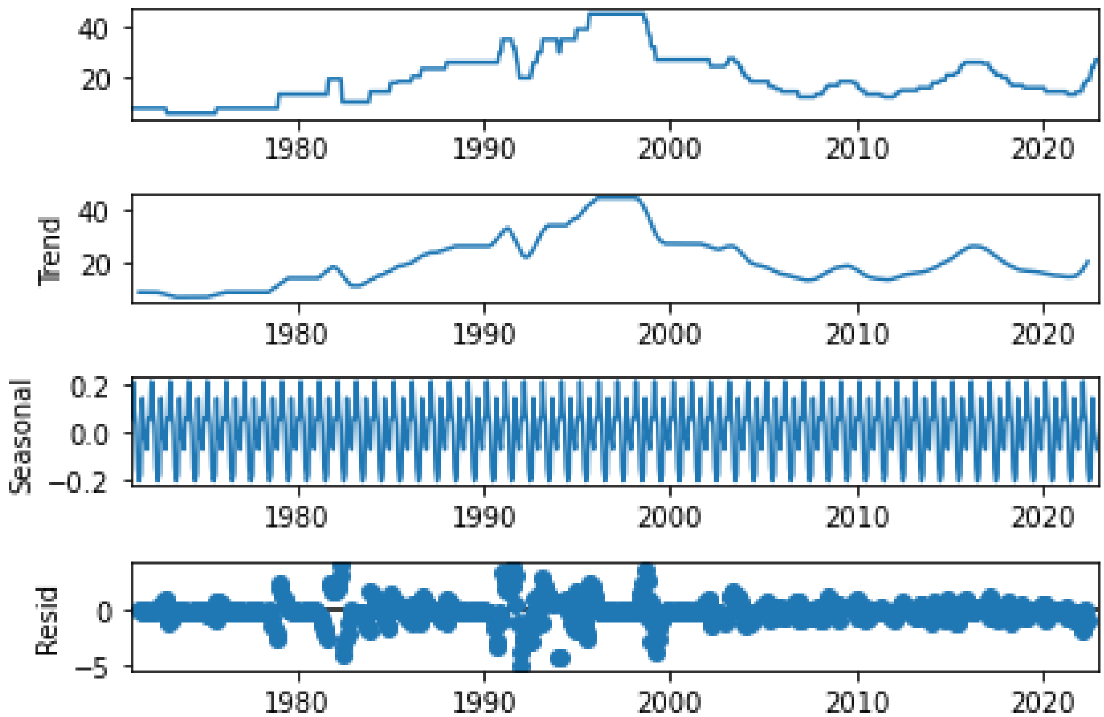
[4]:
```python
df.plot()
```

[4]: <AxesSubplot: xlabel='Date'>



0.0.1 from the graph above, there is a fluctuation pattern in the monetory policy rate, with it increasing and decreasing over the given period. There is more increasing than decreasing and the overall trend is upward.

[5]:
```python
decompose_data = seasonal_decompose(df, model = 'additive')
decompose_data.plot();
```



0.0.2 the trend plot shows an overall downward trend in the monetory policy rate, the seasonal plot shows the fluctuating pattern in the monetory policy rate and the residual plot shows the random variation in the monetory policy rate which is not explained by trend and seasonality.

[6]:
```python
passing_data = adfuller(df['Value'])
def adf_test(answer):
    result = adfuller(answer)
    labels = ['Test parameters','p-value','number of lags used','Dataset␣
 ↪Observations']
    for value, label in zip(result, labels):
        print(label + ':' + str(value) )
        if result[1] <= 0.05:
            print('Dataset is stationary')
        else:
            print('Dataset is non-stationary')
```

0.0.3 the adfuller is used to test for stationarity

[7]:
```
adf_test(df['Value'])
```

```
Test parameters:-1.8737629655182901
Dataset is non-stationary
p-value:0.34450269264617916
Dataset is non-stationary
number of lags used:9
Dataset is non-stationary
Dataset Observations:614
Dataset is non-stationary
```

[8]:
```
dff = df.diff().diff().dropna()
```

[9]:
```
len(dff)
```

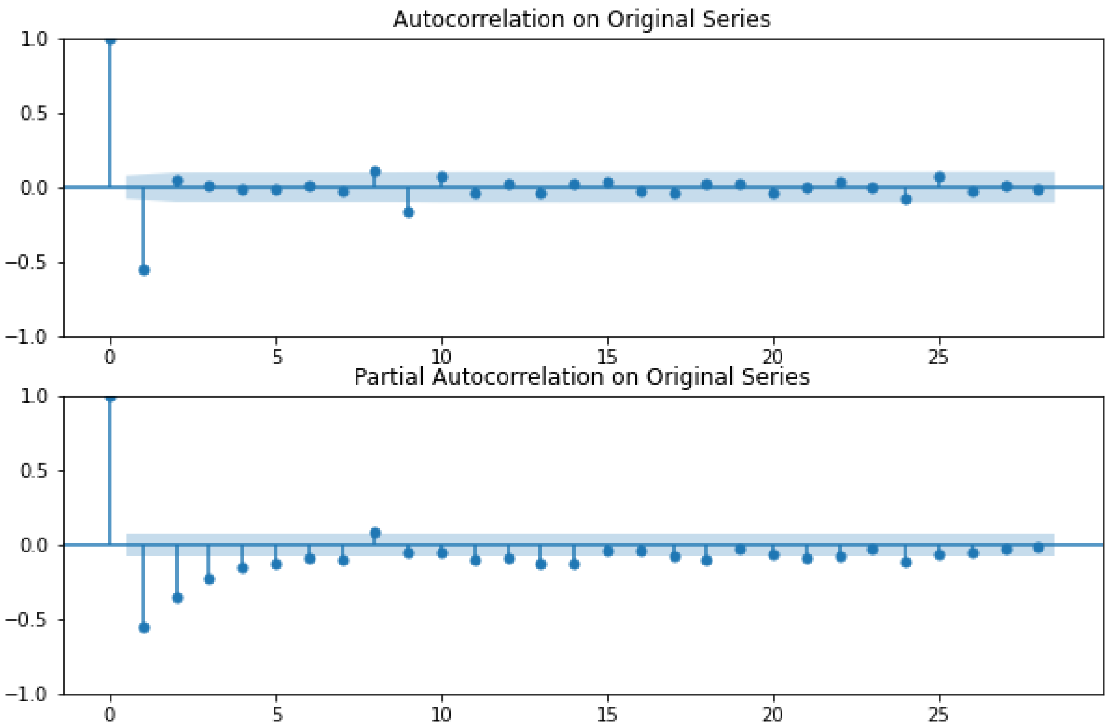
[9]: 622

[10]:
```
adf_test(dff)
```

```
Test parameters:-10.065108059167526
Dataset is stationary
p-value:1.3024915831431611e-17
Dataset is stationary
number of lags used:17
Dataset is stationary
Dataset Observations:604
Dataset is stationary
```

[11]:
```
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(311)
fig = plot_acf(dff, ax = ax1, title = 'Autocorrelation on Original Series')
ax2 = fig.add_subplot(312)
fig1 = plot_pacf(dff, ax2, title = 'Partial Autocorrelation on Original Series')
```

```
C:\Users\ALCHEMY\Anaconda3\lib\site-
packages\statsmodels\graphics\tsaplots.py:348: FutureWarning: The default method
'yw' can produce PACF values outside of the [-1,1] interval. After 0.13, the
default will change tounadjusted Yule-Walker ('ywm'). You can use this method
now by setting method='ywm'.
  warnings.warn(
```



0.0.4 the acf and pacf graph is plot to show the AR and MA processes

[12]:
```
train = dff.iloc[:len(dff)-62]
test = dff.iloc[len(dff)-62:]
#X_train, X_test, y_train, y_test = train_test_split(dff, test_size = 0.1)
```

0.0.5 the dataset is split into training and testing dataset. 90% of the data is used for training and the remaining 10% is used for testing.

[13]:
```
import warnings
warnings.filterwarnings("ignore")

# Fit auto_arima function to Inflation dataset
stepwise_fit = auto_arima(train['Value'], start_p = 1, start_q = 1,
                          test='adf',
                          max_p = 3, max_q = 3, m = 12,
                          start_P = 0, seasonal = True,
                          d = 1, D = 1, trace = True,
                          error_action ='ignore',   # we don't want to know if␣
 ↪an order does not work
                          suppress_warnings = True,  # we don't want␣
 ↪convergence warnings
                          stepwise = True)           # set to stepwise
```

```python
# To print the summary
stepwise_fit.summary()
```

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=3.79 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=2947.469, Time=0.13 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=2491.720, Time=0.44 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=2.57 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=2620.193, Time=0.17 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=2417.289, Time=0.92 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=5.23 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=2.74 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=2768.444, Time=0.80 sec
 ARIMA(2,1,0)(2,1,0)[12]             : AIC=2234.214, Time=1.15 sec
 ARIMA(2,1,0)(1,1,0)[12]             : AIC=2311.149, Time=0.44 sec
 ARIMA(2,1,0)(2,1,1)[12]             : AIC=inf, Time=3.55 sec
 ARIMA(2,1,0)(1,1,1)[12]             : AIC=inf, Time=3.34 sec
 ARIMA(3,1,0)(2,1,0)[12]             : AIC=2153.640, Time=1.08 sec
 ARIMA(3,1,0)(1,1,0)[12]             : AIC=2207.548, Time=0.61 sec
 ARIMA(3,1,0)(2,1,1)[12]             : AIC=inf, Time=9.04 sec
 ARIMA(3,1,0)(1,1,1)[12]             : AIC=inf, Time=7.00 sec
 ARIMA(3,1,1)(2,1,0)[12]             : AIC=inf, Time=10.16 sec
 ARIMA(2,1,1)(2,1,0)[12]             : AIC=inf, Time=9.00 sec
 ARIMA(3,1,0)(2,1,0)[12] intercept   : AIC=2155.639, Time=2.26 sec

Best model:  ARIMA(3,1,0)(2,1,0)[12]
Total fit time: 64.481 seconds
```

[13]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                     SARIMAX Results
================================================================================
==========
Dep. Variable:                                  y   No. Observations:
560
Model:             SARIMAX(3, 1, 0)x(2, 1, 0, 12)   Log Likelihood
-1070.820
Date:                            Thu, 01 Jun 2023   AIC
2153.640
Time:                                    18:01:57   BIC
2179.467
Sample:                                03-01-1971   HQIC
2163.735
                                     - 10-01-2017
Covariance Type:                              opg
==============================================================================
```

```
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.2607      0.021    -59.005      0.000      -1.303      -1.219
ar.L2         -0.9349      0.037    -25.328      0.000      -1.007      -0.863
ar.L3         -0.3777      0.030    -12.604      0.000      -0.436      -0.319
ar.S.L12      -0.5953      0.027    -22.225      0.000      -0.648      -0.543
ar.S.L24      -0.3111      0.026    -11.802      0.000      -0.363      -0.259
sigma2         2.8998      0.099     29.249      0.000       2.705       3.094
================================================================================
===
Ljung-Box (L1) (Q):                  10.72   Jarque-Bera (JB):
470.98
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.30   Skew:
0.04
Prob(H) (two-sided):                  0.00   Kurtosis:
7.55
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

0.0.6 the auto_arima function from the pmdarima package is used to fit the best SARIMA model to the data. and from the result above, SARIMA(3,1,0)(2,1,0)12 is the best fit

[14]:
```python
model = SARIMAX(train['Value'],
               order = (3,1,0),
               seasonal_order = (2,1,0,12))
result = model.fit()
result.summary()
```

[14]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                     SARIMAX Results
================================================================================
==========
Dep. Variable:                               Value   No. Observations:
560
Model:             SARIMAX(3, 1, 0)x(2, 1, 0, 12)   Log Likelihood
-1070.820
Date:                             Thu, 01 Jun 2023   AIC
2153.640
```

```
Time:                                    18:01:58   BIC
2179.467
Sample:                                03-01-1971   HQIC
2163.735
                                     - 10-01-2017
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.2607      0.021    -59.005      0.000      -1.303      -1.219
ar.L2         -0.9349      0.037    -25.328      0.000      -1.007      -0.863
ar.L3         -0.3777      0.030    -12.604      0.000      -0.436      -0.319
ar.S.L12      -0.5953      0.027    -22.225      0.000      -0.648      -0.543
ar.S.L24      -0.3111      0.026    -11.802      0.000      -0.363      -0.259
sigma2         2.8998      0.099     29.249      0.000       2.705       3.094
================================================================================
===
Ljung-Box (L1) (Q):                  10.72   Jarque-Bera (JB):
470.98
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.30   Skew:
0.04
Prob(H) (two-sided):                  0.00   Kurtosis:
7.55
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

0.0.7 the fitted model is then used on the train dataset

[15]:
```python
start = len(train)
end = len(train) + len(test) - 1
```

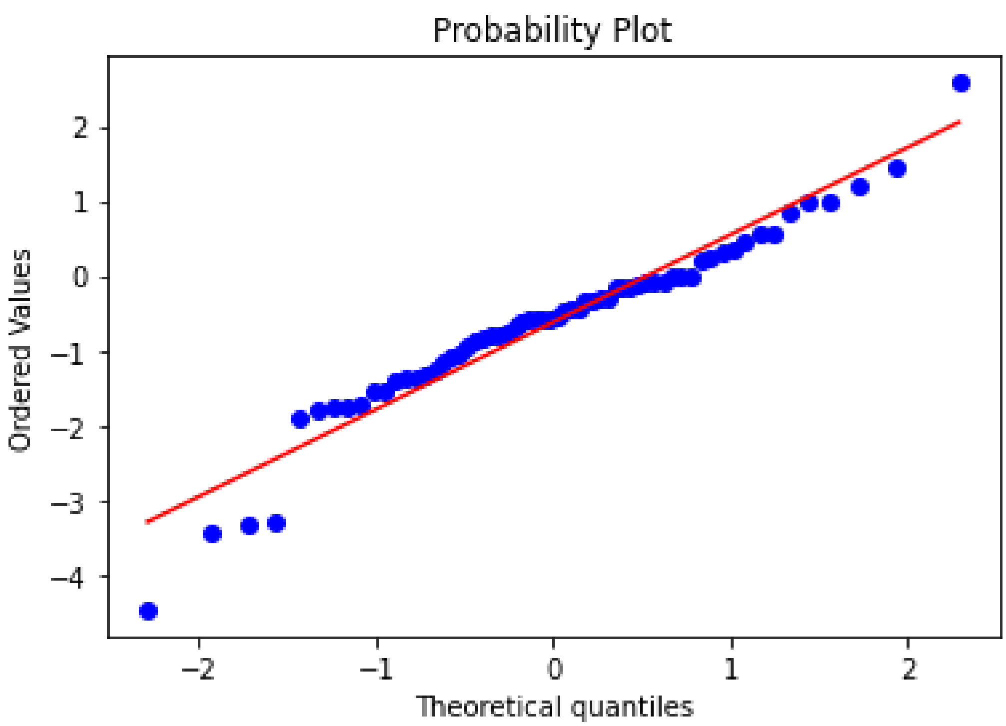
[16]:
```python
# Get the residuals for the test data
predictions = result.predict(start=train.shape[0], end=train.shape[0]+test.
  ↪shape[0]-1)
predictions.head()
residuals = test['Value'] - predictions

# Plot the residuals
#pyplot.plot(residuals)
#pyplot.show()
```

0.0.8 predictions are made on the test dataset and the residual is calculated

[17]:
```python
from scipy.stats import normaltest

_, p_value = normaltest(residuals)

if p_value < 0.05:
    print("The data is not normally distributed.")
else:
    print("The data is normally distributed.")
```

```
The data is not normally distributed.
```

[18]:
```python
from scipy.stats import probplot

# Create a Q-Q plot of the residuals
probplot(residuals, plot=pyplot)
pyplot.show()
```



[19]:
```python
from scipy.stats import kstest

_, p_value = kstest(residuals, "norm")

if p_value < 0.05:
```

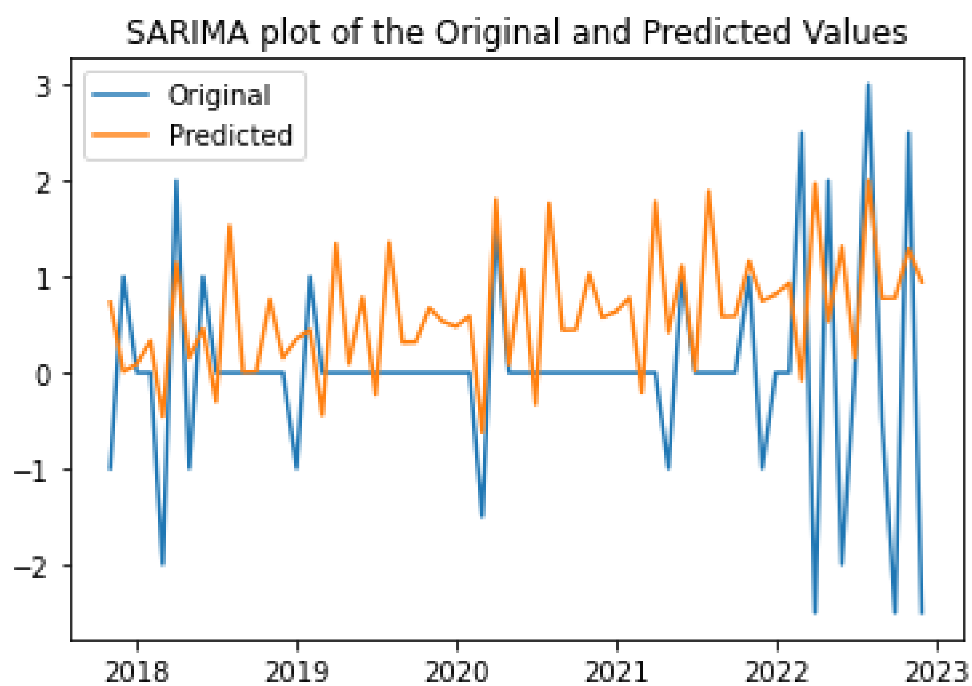
```
    print("The data is not normally distributed.")
else:
    print("The data is normally distributed.")
```

```
The data is not normally distributed.
```

0.0.9 normality is tested and the Q-Q plot is drawn.

[20]:
```
#predictions.plot(legend = True)
#test[' Value'].plot(legend = True)
plt.plot(test['Value'], label='Original')
plt.plot(predictions,  label='Predicted')
plt.title('SARIMA plot of the Original and Predicted Values')
plt.legend()
```

[20]:
```
<matplotlib.legend.Legend at 0x11945e259a0>
```



0.0.10 the predicted values and the original values are then plotted against the date

[21]:
```
rmse_sarima = rmse(test['Value'], predictions)
mae_sarima = mean_absolute_error(test['Value'], predictions)
mape_sarima = mean_absolute_percentage_error(test['Value'], predictions)
```

[22]:
```
print("RMSE_SARIMA: " + str(rmse_sarima))
print('MAE_SARIMA: ' + str(mae_sarima))
print('MAPE_SARIMA: ' + str(mape_sarima))
```

```
RMSE_SARIMA: 1.3242251688865463
MAE_SARIMA: 0.9657809286990032
MAPE_SARIMA: 1779917007701222.5
```

0.0.11 the MAPE, RMSE and MAE evaluation metrics are then calculated to know the accuracy of the fitted model

0.1 LSTM MODEL

[23]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

# Import the plotting library
import matplotlib.pyplot as plt
%matplotlib inline

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

from keras.layers import GRU, Embedding, LSTM

from keras.models import load_model
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
import math
from statistics import mean
import warnings
warnings.filterwarnings('ignore')
```

[24]:
```
data= pd.read_csv('rate.csv')
```

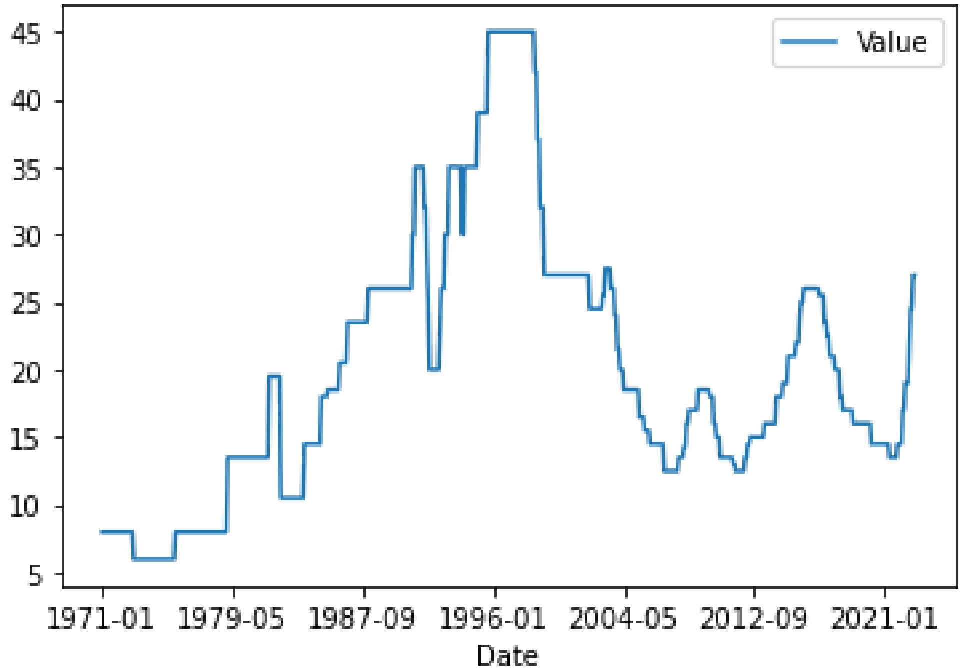
[25]:
```
data = data.set_index('Date')
```

[26]:
```
data.shape
```

[26]: (624, 1)

[27]:
```
data.plot()
```

[27]: <AxesSubplot: xlabel='Date'>



[28]:
```
train_df = data[:561]
print('train shape: ', train_df.shape)
```

```
train shape:  (561, 1)
```

[29]:
```
test_df = data[561:]
print('test_shape: ', test_df.shape)
```

```
test_shape:  (63, 1)
```

[30]:
```
print('Min x: ', np.min(train_df))
print('Max x: ', np.max(train_df))
```

```
Min x:  Value    6.0
dtype: float64
Max x:  Value    45.0
dtype: float64
```

[31]:
```
x_scaler = MinMaxScaler()
train = x_scaler.fit_transform(train_df.values.reshape(-1,1))
test = x_scaler.fit_transform(test_df.values.reshape(-1,1))
```

[32]:
```
print('Min x:', np.min(train))
print('Max x:', np.max(train))
```

```
Min x: 0.0
Max x: 0.9999999999999999
```

[33]:
```
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = [], []
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

[34]:
```
seq = [1,2,3,4,5,6,7,8,9,10]
steps = 3
split_sequence(seq, steps)
```

[34]:
```
(array([[1, 2, 3],
        [2, 3, 4],
        [3, 4, 5],
        [4, 5, 6],
        [5, 6, 7],
        [6, 7, 8],
        [7, 8, 9]]),
 array([ 4,  5,  6,  7,  8,  9, 10]))
```

[35]:
```
n_steps = 5
X_train, y_train = split_sequence(train, n_steps)
```

[36]:
```
X_train.shape
```

[36]:
```
(556, 5, 1)
```

[37]:
```
X_test, y_test = split_sequence(test, n_steps)
```

[38]:
```python
X_test.shape
```

[38]: (58, 5, 1)

[39]:
```python
print(X_train.shape)
print(X_test.shape)
```

```
(556, 5, 1)
(58, 5, 1)
```

[40]:
```python
n_features = 1

# define model
model = Sequential()

# Single layer GRU
#model.add(GRU(32 , input_shape=(n_steps, n_features) ))

# Stacked GRU
#model.add(GRU(8 , input_shape=(n_steps, n_features) , return_sequences=True))
#model.add(GRU(16, return_sequences=True))
#model.add(GRU(32))

# Stacked LSTM
model.add(LSTM(8, activation='relu', input_shape=(n_steps, n_features),␣
 ↪return_sequences=True))
model.add(LSTM(16, activation='relu', return_sequences=True))
model.add(LSTM(32, activation='relu'))

model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')


# simple early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20)
mc = ModelCheckpoint('best_model.h5', monitor='val_loss', mode='min',␣
 ↪verbose=1, save_best_only=True)

history = model.fit(X_train, y_train, validation_split=0.2, epochs=100,␣
 ↪batch_size=32, verbose=1, callbacks=[es, mc])

# fit model
#model.fit(X_train, y_train, epochs=50, verbose=1, callbacks=[es, mc])
```

```
Epoch 1/100
11/14 [======================>…] - ETA: 0s - loss: 0.2278
Epoch 1: val_loss improved from inf to 0.08406, saving model to best_model.h5
```

```
14/14 [==============================] - 16s 155ms/step - loss: 0.2183 -
val_loss: 0.0841
Epoch 2/100
11/14 [======================>…] - ETA: 0s - loss: 0.1784
Epoch 2: val_loss improved from 0.08406 to 0.05241, saving model to
best_model.h5
14/14 [==============================] - 0s 27ms/step - loss: 0.1784 - val_loss:
0.0524
Epoch 3/100
14/14 [==============================] - ETA: 0s - loss: 0.1303
Epoch 3: val_loss improved from 0.05241 to 0.01928, saving model to
best_model.h5
14/14 [==============================] - 0s 30ms/step - loss: 0.1303 - val_loss:
0.0193
Epoch 4/100
12/14 [========================>…] - ETA: 0s - loss: 0.0793
Epoch 4: val_loss improved from 0.01928 to 0.01585, saving model to
best_model.h5
14/14 [==============================] - 0s 32ms/step - loss: 0.0771 - val_loss:
0.0159
Epoch 5/100
14/14 [==============================] - ETA: 0s - loss: 0.0576
Epoch 5: val_loss did not improve from 0.01585
14/14 [==============================] - 0s 24ms/step - loss: 0.0576 - val_loss:
0.0172
Epoch 6/100
14/14 [==============================] - ETA: 0s - loss: 0.0387
Epoch 6: val_loss improved from 0.01585 to 0.00937, saving model to
best_model.h5
14/14 [==============================] - 0s 31ms/step - loss: 0.0387 - val_loss:
0.0094
Epoch 7/100
14/14 [==============================] - ETA: 0s - loss: 0.0170
Epoch 7: val_loss improved from 0.00937 to 0.00215, saving model to
best_model.h5
14/14 [==============================] - 1s 49ms/step - loss: 0.0170 - val_loss:
0.0022
Epoch 8/100
14/14 [==============================] - ETA: 0s - loss: 0.0088
Epoch 8: val_loss did not improve from 0.00215
14/14 [==============================] - 1s 36ms/step - loss: 0.0088 - val_loss:
0.0027
Epoch 9/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0067
Epoch 9: val_loss improved from 0.00215 to 0.00154, saving model to
best_model.h5
14/14 [==============================] - 1s 43ms/step - loss: 0.0068 - val_loss:
0.0015
```

```
Epoch 10/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0057
Epoch 10: val_loss did not improve from 0.00154
14/14 [==============================] - 0s 32ms/step - loss: 0.0057 - val_loss:
0.0026
Epoch 11/100
12/14 [========================>…] - ETA: 0s - loss: 0.0055
Epoch 11: val_loss did not improve from 0.00154
14/14 [==============================] - 0s 25ms/step - loss: 0.0054 - val_loss:
0.0027
Epoch 12/100
12/14 [========================>…] - ETA: 0s - loss: 0.0046
Epoch 12: val_loss improved from 0.00154 to 0.00134, saving model to
best_model.h5
14/14 [==============================] - 0s 32ms/step - loss: 0.0047 - val_loss:
0.0013
Epoch 13/100
14/14 [==============================] - ETA: 0s - loss: 0.0044
Epoch 13: val_loss did not improve from 0.00134
14/14 [==============================] - 1s 46ms/step - loss: 0.0044 - val_loss:
0.0019
Epoch 14/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0041
Epoch 14: val_loss did not improve from 0.00134
14/14 [==============================] - 1s 40ms/step - loss: 0.0042 - val_loss:
0.0014
Epoch 15/100
14/14 [==============================] - ETA: 0s - loss: 0.0040
Epoch 15: val_loss improved from 0.00134 to 0.00127, saving model to
best_model.h5
14/14 [==============================] - 1s 55ms/step - loss: 0.0040 - val_loss:
0.0013
Epoch 16/100
12/14 [========================>…] - ETA: 0s - loss: 0.0035
Epoch 16: val_loss improved from 0.00127 to 0.00113, saving model to
best_model.h5
14/14 [==============================] - 1s 42ms/step - loss: 0.0038 - val_loss:
0.0011
Epoch 17/100
12/14 [========================>…] - ETA: 0s - loss: 0.0034
Epoch 17: val_loss did not improve from 0.00113
14/14 [==============================] - 1s 43ms/step - loss: 0.0039 - val_loss:
0.0013
Epoch 18/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0041
Epoch 18: val_loss did not improve from 0.00113
14/14 [==============================] - 1s 50ms/step - loss: 0.0040 - val_loss:
0.0011
```

```
Epoch 19/100
12/14 [========================>…] - ETA: 0s - loss: 0.0040
Epoch 19: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 34ms/step - loss: 0.0040 - val_loss:
0.0019
Epoch 20/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0044
Epoch 20: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 26ms/step - loss: 0.0042 - val_loss:
0.0012
Epoch 21/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0037
Epoch 21: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 24ms/step - loss: 0.0038 - val_loss:
0.0015
Epoch 22/100
14/14 [==============================] - ETA: 0s - loss: 0.0038
Epoch 22: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 23ms/step - loss: 0.0038 - val_loss:
0.0013
Epoch 23/100
11/14 [======================>…] - ETA: 0s - loss: 0.0040
Epoch 23: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 28ms/step - loss: 0.0037 - val_loss:
0.0013
Epoch 24/100
12/14 [========================>…] - ETA: 0s - loss: 0.0039
Epoch 24: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 30ms/step - loss: 0.0037 - val_loss:
0.0015
Epoch 25/100
14/14 [==============================] - ETA: 0s - loss: 0.0040
Epoch 25: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 23ms/step - loss: 0.0040 - val_loss:
0.0015
Epoch 26/100
14/14 [==============================] - ETA: 0s - loss: 0.0039
Epoch 26: val_loss did not improve from 0.00113
14/14 [==============================] - 0s 25ms/step - loss: 0.0039 - val_loss:
0.0013
Epoch 27/100
12/14 [========================>…] - ETA: 0s - loss: 0.0039
Epoch 27: val_loss improved from 0.00113 to 0.00112, saving model to
best_model.h5
14/14 [==============================] - 0s 26ms/step - loss: 0.0038 - val_loss:
0.0011
Epoch 28/100
14/14 [==============================] - ETA: 0s - loss: 0.0037
```

```
Epoch 28: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 28ms/step - loss: 0.0037 - val_loss:
0.0012
Epoch 29/100
12/14 [========================>…] - ETA: 0s - loss: 0.0040
Epoch 29: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 27ms/step - loss: 0.0037 - val_loss:
0.0011
Epoch 30/100
14/14 [==============================] - ETA: 0s - loss: 0.0036
Epoch 30: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 31ms/step - loss: 0.0036 - val_loss:
0.0012
Epoch 31/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0037
Epoch 31: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 27ms/step - loss: 0.0037 - val_loss:
0.0011
Epoch 32/100
14/14 [==============================] - ETA: 0s - loss: 0.0039
Epoch 32: val_loss improved from 0.00112 to 0.00112, saving model to
best_model.h5
14/14 [==============================] - 0s 30ms/step - loss: 0.0039 - val_loss:
0.0011
Epoch 33/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0043
Epoch 33: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 24ms/step - loss: 0.0042 - val_loss:
0.0024
Epoch 34/100
12/14 [========================>…] - ETA: 0s - loss: 0.0046
Epoch 34: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 26ms/step - loss: 0.0047 - val_loss:
0.0013
Epoch 35/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0039
Epoch 35: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 25ms/step - loss: 0.0039 - val_loss:
0.0011
Epoch 36/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0038
Epoch 36: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 24ms/step - loss: 0.0037 - val_loss:
0.0013
Epoch 37/100
12/14 [========================>…] - ETA: 0s - loss: 0.0036
Epoch 37: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 30ms/step - loss: 0.0036 - val_loss:
```

```
0.0012
Epoch 38/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0037
Epoch 38: val_loss did not improve from 0.00112
14/14 [==============================] - 0s 28ms/step - loss: 0.0036 - val_loss:
0.0011
Epoch 39/100
12/14 [========================>…] - ETA: 0s - loss: 0.0035
Epoch 39: val_loss improved from 0.00112 to 0.00107, saving model to
best_model.h5
14/14 [==============================] - 0s 32ms/step - loss: 0.0036 - val_loss:
0.0011
Epoch 40/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0036
Epoch 40: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 22ms/step - loss: 0.0035 - val_loss:
0.0011
Epoch 41/100
12/14 [========================>…] - ETA: 0s - loss: 0.0031
Epoch 41: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 21ms/step - loss: 0.0035 - val_loss:
0.0011
Epoch 42/100
12/14 [========================>…] - ETA: 0s - loss: 0.0037
Epoch 42: val_loss improved from 0.00107 to 0.00107, saving model to
best_model.h5
14/14 [==============================] - 0s 28ms/step - loss: 0.0037 - val_loss:
0.0011
Epoch 43/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0034
Epoch 43: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 21ms/step - loss: 0.0035 - val_loss:
0.0011
Epoch 44/100
12/14 [========================>…] - ETA: 0s - loss: 0.0037
Epoch 44: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 22ms/step - loss: 0.0036 - val_loss:
0.0011
Epoch 45/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0034
Epoch 45: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 24ms/step - loss: 0.0036 - val_loss:
0.0011
Epoch 46/100
12/14 [========================>…] - ETA: 0s - loss: 0.0036
Epoch 46: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 21ms/step - loss: 0.0036 - val_loss:
0.0013
```

```
Epoch 47/100
12/14 [========================>…] - ETA: 0s - loss: 0.0038
Epoch 47: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 22ms/step - loss: 0.0037 - val_loss:
0.0018
Epoch 48/100
14/14 [==============================] - ETA: 0s - loss: 0.0038
Epoch 48: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 24ms/step - loss: 0.0038 - val_loss:
0.0016
Epoch 49/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0038
Epoch 49: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 21ms/step - loss: 0.0037 - val_loss:
0.0013
Epoch 50/100
12/14 [========================>…] - ETA: 0s - loss: 0.0039
Epoch 50: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 21ms/step - loss: 0.0037 - val_loss:
0.0015
Epoch 51/100
12/14 [========================>…] - ETA: 0s - loss: 0.0033
Epoch 51: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 25ms/step - loss: 0.0036 - val_loss:
0.0012
Epoch 52/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0035
Epoch 52: val_loss did not improve from 0.00107
14/14 [==============================] - 0s 24ms/step - loss: 0.0034 - val_loss:
0.0013
Epoch 53/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0035
Epoch 53: val_loss improved from 0.00107 to 0.00105, saving model to
best_model.h5
14/14 [==============================] - 0s 29ms/step - loss: 0.0034 - val_loss:
0.0010
Epoch 54/100
11/14 [======================>…] - ETA: 0s - loss: 0.0033
Epoch 54: val_loss did not improve from 0.00105
14/14 [==============================] - 0s 22ms/step - loss: 0.0034 - val_loss:
0.0011
Epoch 55/100
12/14 [========================>…] - ETA: 0s - loss: 0.0036
Epoch 55: val_loss improved from 0.00105 to 0.00102, saving model to
best_model.h5
14/14 [==============================] - 0s 27ms/step - loss: 0.0034 - val_loss:
0.0010
Epoch 56/100
```

```
14/14 [==============================] - ETA: 0s - loss: 0.0034
Epoch 56: val_loss did not improve from 0.00102
14/14 [==============================] - 0s 23ms/step - loss: 0.0034 - val_loss:
0.0011
Epoch 57/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0035
Epoch 57: val_loss did not improve from 0.00102
14/14 [==============================] - 0s 24ms/step - loss: 0.0035 - val_loss:
0.0010
Epoch 58/100
11/14 [======================>…] - ETA: 0s - loss: 0.0036
Epoch 58: val_loss improved from 0.00102 to 0.00101, saving model to
best_model.h5
14/14 [==============================] - 0s 27ms/step - loss: 0.0035 - val_loss:
0.0010
Epoch 59/100
14/14 [==============================] - ETA: 0s - loss: 0.0034
Epoch 59: val_loss did not improve from 0.00101
14/14 [==============================] - 0s 25ms/step - loss: 0.0034 - val_loss:
0.0010
Epoch 60/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0036
Epoch 60: val_loss did not improve from 0.00101
14/14 [==============================] - 0s 24ms/step - loss: 0.0035 - val_loss:
0.0010
Epoch 61/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0032
Epoch 61: val_loss did not improve from 0.00101
14/14 [==============================] - 0s 26ms/step - loss: 0.0033 - val_loss:
0.0016
Epoch 62/100
14/14 [==============================] - ETA: 0s - loss: 0.0034
Epoch 62: val_loss did not improve from 0.00101
14/14 [==============================] - 0s 25ms/step - loss: 0.0034 - val_loss:
0.0011
Epoch 63/100
14/14 [==============================] - ETA: 0s - loss: 0.0034
Epoch 63: val_loss did not improve from 0.00101
14/14 [==============================] - 0s 24ms/step - loss: 0.0034 - val_loss:
0.0011
Epoch 64/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0033
Epoch 64: val_loss improved from 0.00101 to 0.00098, saving model to
best_model.h5
14/14 [==============================] - 0s 28ms/step - loss: 0.0033 - val_loss:
9.7734e-04
Epoch 65/100
14/14 [==============================] - ETA: 0s - loss: 0.0032
```

```
Epoch 65: val_loss did not improve from 0.00098
14/14 [==============================] - 0s 27ms/step - loss: 0.0032 - val_loss:
0.0010
Epoch 66/100
11/14 [======================>…] - ETA: 0s - loss: 0.0029
Epoch 66: val_loss did not improve from 0.00098
14/14 [==============================] - 0s 23ms/step - loss: 0.0033 - val_loss:
0.0010
Epoch 67/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0031
Epoch 67: val_loss improved from 0.00098 to 0.00097, saving model to
best_model.h5
14/14 [==============================] - 0s 32ms/step - loss: 0.0031 - val_loss:
9.7203e-04
Epoch 68/100
12/14 [========================>…] - ETA: 0s - loss: 0.0034
Epoch 68: val_loss improved from 0.00097 to 0.00095, saving model to
best_model.h5
14/14 [==============================] - 0s 32ms/step - loss: 0.0031 - val_loss:
9.4823e-04
Epoch 69/100
11/14 [======================>…] - ETA: 0s - loss: 0.0030
Epoch 69: val_loss did not improve from 0.00095
14/14 [==============================] - 0s 22ms/step - loss: 0.0031 - val_loss:
9.8874e-04
Epoch 70/100
14/14 [==============================] - ETA: 0s - loss: 0.0032
Epoch 70: val_loss improved from 0.00095 to 0.00094, saving model to
best_model.h5
14/14 [==============================] - 0s 29ms/step - loss: 0.0032 - val_loss:
9.3585e-04
Epoch 71/100
14/14 [==============================] - ETA: 0s - loss: 0.0031
Epoch 71: val_loss did not improve from 0.00094
14/14 [==============================] - 0s 24ms/step - loss: 0.0031 - val_loss:
0.0011
Epoch 72/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0032
Epoch 72: val_loss improved from 0.00094 to 0.00091, saving model to
best_model.h5
14/14 [==============================] - 0s 29ms/step - loss: 0.0032 - val_loss:
9.1490e-04
Epoch 73/100
14/14 [==============================] - ETA: 0s - loss: 0.0031
Epoch 73: val_loss did not improve from 0.00091
14/14 [==============================] - 0s 30ms/step - loss: 0.0031 - val_loss:
0.0010
Epoch 74/100
```

```
12/14 [=======================>…] - ETA: 0s - loss: 0.0031
Epoch 74: val_loss improved from 0.00091 to 0.00091, saving model to
best_model.h5
14/14 [==============================] - 0s 36ms/step - loss: 0.0030 - val_loss:
9.0806e-04
Epoch 75/100
12/14 [=======================>…] - ETA: 0s - loss: 0.0027
Epoch 75: val_loss improved from 0.00091 to 0.00091, saving model to
best_model.h5
14/14 [==============================] - 0s 33ms/step - loss: 0.0030 - val_loss:
9.0684e-04
Epoch 76/100
14/14 [==============================] - ETA: 0s - loss: 0.0030
Epoch 76: val_loss improved from 0.00091 to 0.00089, saving model to
best_model.h5
14/14 [==============================] - 0s 30ms/step - loss: 0.0030 - val_loss:
8.8766e-04
Epoch 77/100
14/14 [==============================] - ETA: 0s - loss: 0.0030
Epoch 77: val_loss did not improve from 0.00089
14/14 [==============================] - 0s 25ms/step - loss: 0.0030 - val_loss:
9.2109e-04
Epoch 78/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0031
Epoch 78: val_loss improved from 0.00089 to 0.00088, saving model to
best_model.h5
14/14 [==============================] - 0s 30ms/step - loss: 0.0030 - val_loss:
8.8078e-04
Epoch 79/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0029
Epoch 79: val_loss did not improve from 0.00088
14/14 [==============================] - 0s 21ms/step - loss: 0.0029 - val_loss:
9.0453e-04
Epoch 80/100
12/14 [=======================>…] - ETA: 0s - loss: 0.0030
Epoch 80: val_loss did not improve from 0.00088
14/14 [==============================] - 0s 22ms/step - loss: 0.0029 - val_loss:
9.6617e-04
Epoch 81/100
12/14 [=======================>…] - ETA: 0s - loss: 0.0026
Epoch 81: val_loss did not improve from 0.00088
14/14 [==============================] - 0s 21ms/step - loss: 0.0029 - val_loss:
9.2985e-04
Epoch 82/100
12/14 [=======================>…] - ETA: 0s - loss: 0.0027
Epoch 82: val_loss improved from 0.00088 to 0.00087, saving model to
best_model.h5
14/14 [==============================] - 0s 27ms/step - loss: 0.0029 - val_loss:
```

```
8.6618e-04
Epoch 83/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0027
Epoch 83: val_loss did not improve from 0.00087
14/14 [==============================] - 0s 25ms/step - loss: 0.0028 - val_loss:
8.9764e-04
Epoch 84/100
11/14 [======================>…] - ETA: 0s - loss: 0.0030
Epoch 84: val_loss improved from 0.00087 to 0.00085, saving model to
best_model.h5
14/14 [==============================] - 0s 26ms/step - loss: 0.0029 - val_loss:
8.5188e-04
Epoch 85/100
12/14 [========================>…] - ETA: 0s - loss: 0.0028
Epoch 85: val_loss did not improve from 0.00085
14/14 [==============================] - 0s 22ms/step - loss: 0.0029 - val_loss:
8.5643e-04
Epoch 86/100
12/14 [========================>…] - ETA: 0s - loss: 0.0029
Epoch 86: val_loss did not improve from 0.00085
14/14 [==============================] - 0s 22ms/step - loss: 0.0028 - val_loss:
8.7594e-04
Epoch 87/100
12/14 [========================>…] - ETA: 0s - loss: 0.0026
Epoch 87: val_loss improved from 0.00085 to 0.00083, saving model to
best_model.h5
14/14 [==============================] - 0s 27ms/step - loss: 0.0028 - val_loss:
8.3212e-04
Epoch 88/100
14/14 [==============================] - ETA: 0s - loss: 0.0028
Epoch 88: val_loss did not improve from 0.00083
14/14 [==============================] - 0s 23ms/step - loss: 0.0028 - val_loss:
8.3353e-04
Epoch 89/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0028
Epoch 89: val_loss did not improve from 0.00083
14/14 [==============================] - 0s 24ms/step - loss: 0.0027 - val_loss:
8.8805e-04
Epoch 90/100
11/14 [======================>…] - ETA: 0s - loss: 0.0027
Epoch 90: val_loss improved from 0.00083 to 0.00083, saving model to
best_model.h5
14/14 [==============================] - 0s 26ms/step - loss: 0.0027 - val_loss:
8.2545e-04
Epoch 91/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0028
Epoch 91: val_loss improved from 0.00083 to 0.00080, saving model to
best_model.h5
```

```
14/14 [==============================] - 0s 31ms/step - loss: 0.0027 - val_loss:
8.0407e-04
Epoch 92/100
12/14 [========================>…] - ETA: 0s - loss: 0.0023
Epoch 92: val_loss improved from 0.00080 to 0.00080, saving model to
best_model.h5
14/14 [==============================] - 1s 39ms/step - loss: 0.0026 - val_loss:
8.0333e-04
Epoch 93/100
12/14 [========================>…] - ETA: 0s - loss: 0.0025
Epoch 93: val_loss improved from 0.00080 to 0.00080, saving model to
best_model.h5
14/14 [==============================] - 0s 28ms/step - loss: 0.0026 - val_loss:
8.0151e-04
Epoch 94/100
14/14 [==============================] - ETA: 0s - loss: 0.0026
Epoch 94: val_loss improved from 0.00080 to 0.00079, saving model to
best_model.h5
14/14 [==============================] - 0s 29ms/step - loss: 0.0026 - val_loss:
7.8860e-04
Epoch 95/100
14/14 [==============================] - ETA: 0s - loss: 0.0026
Epoch 95: val_loss did not improve from 0.00079
14/14 [==============================] - 0s 32ms/step - loss: 0.0026 - val_loss:
8.4654e-04
Epoch 96/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0026
Epoch 96: val_loss did not improve from 0.00079
14/14 [==============================] - 0s 33ms/step - loss: 0.0025 - val_loss:
7.9369e-04
Epoch 97/100
14/14 [==============================] - ETA: 0s - loss: 0.0025
Epoch 97: val_loss did not improve from 0.00079
14/14 [==============================] - 0s 23ms/step - loss: 0.0025 - val_loss:
8.3156e-04
Epoch 98/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0027
Epoch 98: val_loss did not improve from 0.00079
14/14 [==============================] - 0s 25ms/step - loss: 0.0027 - val_loss:
0.0015
Epoch 99/100
13/14 [==========================>…] - ETA: 0s - loss: 0.0028
Epoch 99: val_loss did not improve from 0.00079
14/14 [==============================] - 0s 25ms/step - loss: 0.0028 - val_loss:
9.2636e-04
Epoch 100/100
14/14 [==============================] - ETA: 0s - loss: 0.0025
Epoch 100: val_loss improved from 0.00079 to 0.00076, saving model to
```

```
best_model.h5
14/14 [==============================] - 0s 29ms/step - loss: 0.0025 - val_loss:
7.6390e-04
```

[41]:
```
history.history
```

[41]:

```
{'loss': [0.21834789216518402,
  0.17835961282253265,
  0.1303446888923645,
  0.07713120430707932,
  0.05761090666055679,
  0.03868783637881279,
  0.016954690217971802,
  0.008835582062602043,
  0.006810174323618412,
  0.005717406049370766,
  0.005416699685156345,
  0.004721294157207012,
  0.004418207332491875,
  0.004211268853396177,
  0.004002233035862446,
  0.0038175771478563547,
  0.0038727386854588985,
  0.004031538497656584,
  0.0040416778065264225,
  0.004163871519267559,
  0.003785597626119852,
  0.0037790967617183924,
  0.003721682820469141,
  0.0036939606070518494,
  0.004043465945869684,
  0.003919016104191542,
  0.0037850206717848778,
  0.0036741981748491526,
  0.0037440224550664425,
  0.0036410707980394363,
  0.0037179940845817327,
  0.0038902717642486095,
  0.004162298981100321,
  0.004736504517495632,
  0.003937981557101011,
  0.0036874180659651756,
  0.003599912393838167,
  0.003582953242585063,
  0.0035558692179620266,
  0.003502075793221593,
  0.0035123489797115326,
```

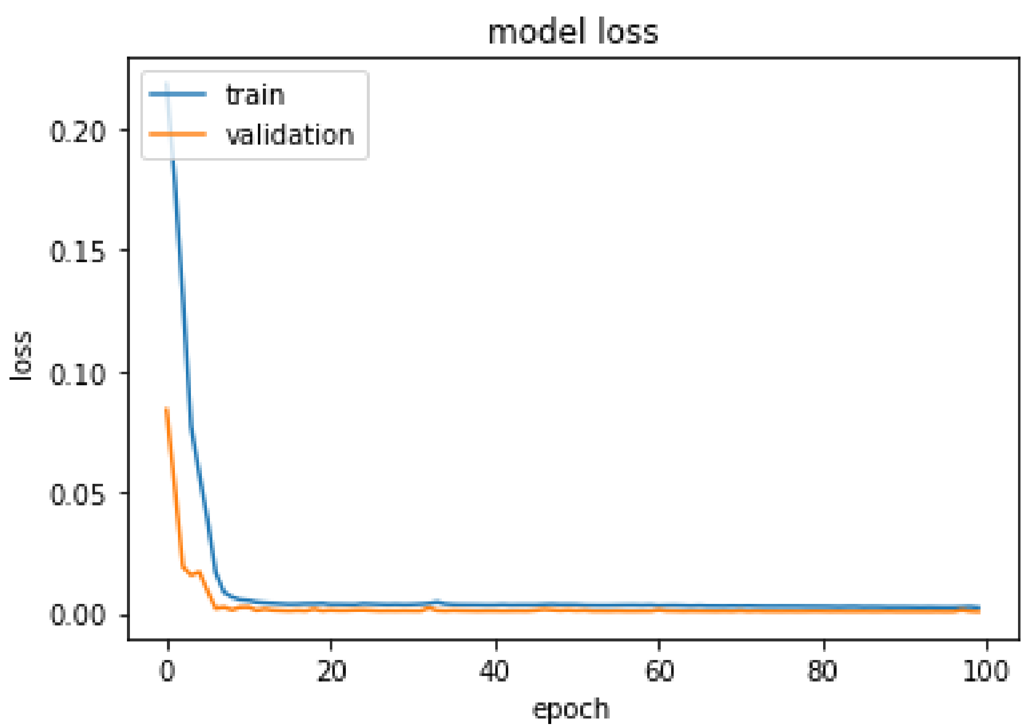
0.0036519980058073997,
0.00351190404035151,
0.003557071555405855,
0.0035569649189710617,
0.0035503075923770666,
0.0037065651267766953,
0.0038089649751782417,
0.00366474618203938,
0.0036774417385458946,
0.003632014850154519,
0.0034297893289476633,
0.0033871151972562075,
0.0033678803592920303,
0.003366706892848015,
0.0034271078184247017,
0.003462321124970913,
0.0034823231399059296,
0.003401447320356965,
0.003514144802466035,
0.003275238210335374,
0.0033510769717395306,
0.0033769693691283464,
0.0032802086789160967,
0.0031675142236053944,
0.0032851085998117924,
0.003134725149720907,
0.003125014016404748,
0.003089701756834984,
0.0031637195497751236,
0.003096040803939104,
0.003169999225065112,
0.003067766549065709,
0.0030231610871851444,
0.003007300430908799,
0.003017701441422105,
0.0029978605452924967,
0.0029956798534840345,
0.0029480380471795797,
0.0029355906881392,
0.002915341407060623,
0.0028790240176022053,
0.0027992823161184788,
0.00286824069917202,
0.0028927954845130444,
0.0027600941248238087,
0.0027901174034923315,
0.002766746561974287,

```
 0.0026698720175772905,
 0.002678210847079754,
 0.0026818052865564823,
 0.0026339066680520773,
 0.0025802338495850563,
 0.00262812664732337,
 0.0025612078607082367,
 0.002518377033993602,
 0.0025252525229007006,
 0.0026510749012231827,
 0.0028051799163222313,
 0.0024687631521373987],
'val_loss': [0.08406383544206619,
 0.05240967497229576,
 0.019281616434454918,
 0.01585421711206436,
 0.017174407839775085,
 0.00936676561832428,
 0.002153054578229785,
 0.0026566735468804836,
 0.0015448599588125944,
 0.002634118078276515,
 0.0027030527126044035,
 0.0013385852798819542,
 0.001873994478955865,
 0.0014162767911329865,
 0.0012676097685471177,
 0.001131392433308065,
 0.0013249424519017339,
 0.001149724586866796,
 0.0019236713415011764,
 0.0011649385560303926,
 0.0014611314982175827,
 0.0012826325837522745,
 0.0013022002531215549,
 0.0014612294035032392,
 0.0014986826572567225,
 0.0013046354288235307,
 0.0011237042490392923,
 0.0011645565973594785,
 0.0011432269820943475,
 0.0011809723218902946,
 0.001142566674388945,
 0.0011204121401533484,
 0.002447539707645774,
 0.001309758983552456,
 0.0011355032911524177,
```

0.0013336176052689552,
0.0011820917716249824,
0.001132530509494245,
0.0010735460091382265,
0.0011188711505383253,
0.0011232373071834445,
0.0010665318695828319,
0.0010806669015437365,
0.0011089937761425972,
0.0010765393963083625,
0.001343654585070908,
0.0017613545060157776,
0.0015577984740957618,
0.001291726715862751,
0.0014821930089965463,
0.001229620655067265,
0.001346708508208394,
0.001048670499585569,
0.001053052139468491,
0.0010151922469958663,
0.0011142342118546367,
0.00101580866612494,
0.001010099076665938,
0.0010393986012786627,
0.0010317793348804116,
0.0015507787466049194,
0.0011288158129900694,
0.0010769616346806288,
0.000977339455857873,
0.00104297767393291,
0.0010124215623363853,
0.0009720308007672429,
0.0009482338791713119,
0.0009887396590784192,
0.0009358463576063514,
0.001146491034887731,
0.0009148980607278645,
0.0010328682838007808,
0.0009080642485059798,
0.0009068434010259807,
0.0008876620559021831,
0.0009210949065163732,
0.0008807781268842518,
0.0009045336628332734,
0.0009661707445047796,
0.000929853820707649,
0.0008661801693961024,

```
  0.0008976422250270844,
  0.0008518838440068066,
  0.0008564285817556083,
  0.000875944911967963,
  0.0008321183850057423,
  0.000833529164083302,
  0.000888047565240413,
  0.0008254541899077594,
  0.0008040661341510713,
  0.0008033254998736084,
  0.0008015077328309417,
  0.0007886015227995813,
  0.00084653653902933,
  0.000793685088865459,
  0.0008315640734508634,
  0.0014946619048714638,
  0.000926357286516577,
  0.0007638997631147504]}
```

[42]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



[43]:
```
model = load_model('best_model.h5')
```

[44]:
```
def rmse(y_true, y_pred):
    return np.sqrt(np.mean(np.square(y_pred - y_true)))
```

[45]:
```
preds = []
# demonstrate predictions
for i in range(X_test.shape[0]):
    x_input = X_test[i]
    x_input = x_input.reshape((1, n_steps, n_features))
    yhat = model.predict(x_input, verbose=0)
    preds.append(yhat[0])
    #print(yhat[0], y_test[i])
```

[46]:
```
print(yhat[0], y_test[i])
```

```
[0.7352169] [1.]
```

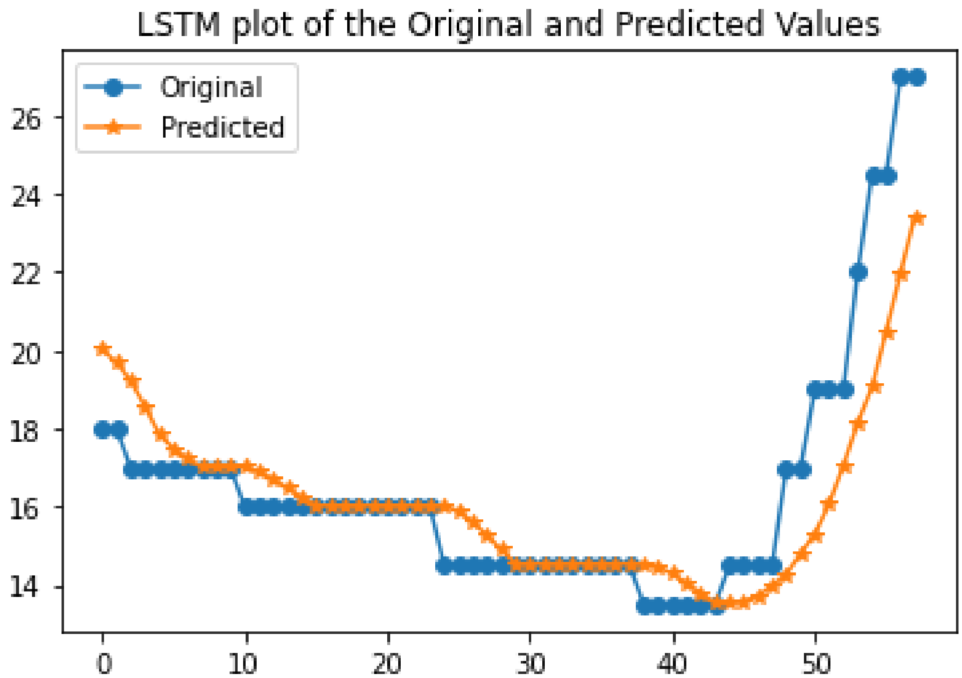
[47]:
```
 #The output of the model is between 0 and 1.
# Do an inverse map to get it back to the scale
# of the original data-set.
preds = x_scaler.inverse_transform(np.array(preds))
# we also rescale the y_test values into their original range (inverse scaling)
actuals = x_scaler.inverse_transform(y_test)
```

[48]:
```
mse_lstm = mean_squared_error(actuals, preds)
rmse_lstm = math.sqrt(mse_lstm)
print('RMSE_LSTM: %.4f' % rmse_lstm)
mae_lstm = mean_squared_error(actuals, preds)
mape_lstm = mean_absolute_percentage_error(actuals, preds)
print('MAPE_LSTM: %.4f' % mape_lstm)
print('MAE_LSTM: %.4f' % mae_lstm)
```

```
RMSE_LSTM: 1.6969
MAPE_LSTM: 0.0560
MAE_LSTM: 2.8796
```

[49]:
```
plt.plot(actuals, marker='o', label='Original')
plt.plot(preds, marker='*', label='Predicted')
plt.title('LSTM plot of the Original and Predicted Values')
plt.legend()
```

[49]:
```
<matplotlib.legend.Legend at 0x1191c89bf10>
```



0.2 SVM MODEL

[50]:
```
from numpy import asarray
from pandas import DataFrame
from pandas import concat
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.svm import SVR
from matplotlib import pyplot
import math
from statistics import mean
import pandas as pd
from sklearn.metrics import mean_squared_error
```

[51]:
```
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
```

```
    # put it all together
    agg = concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

[52]:
```
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]

# fit an svm model and make a one step prediction
def svm_forecast(train, testX):
    # transform list into array
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit model
    model = SVR(kernel = 'linear')
    model.fit(trainX, trainy)
    # make a one-step prediction
    yhat = model.predict(asarray([testX]))
    return yhat[0]
```

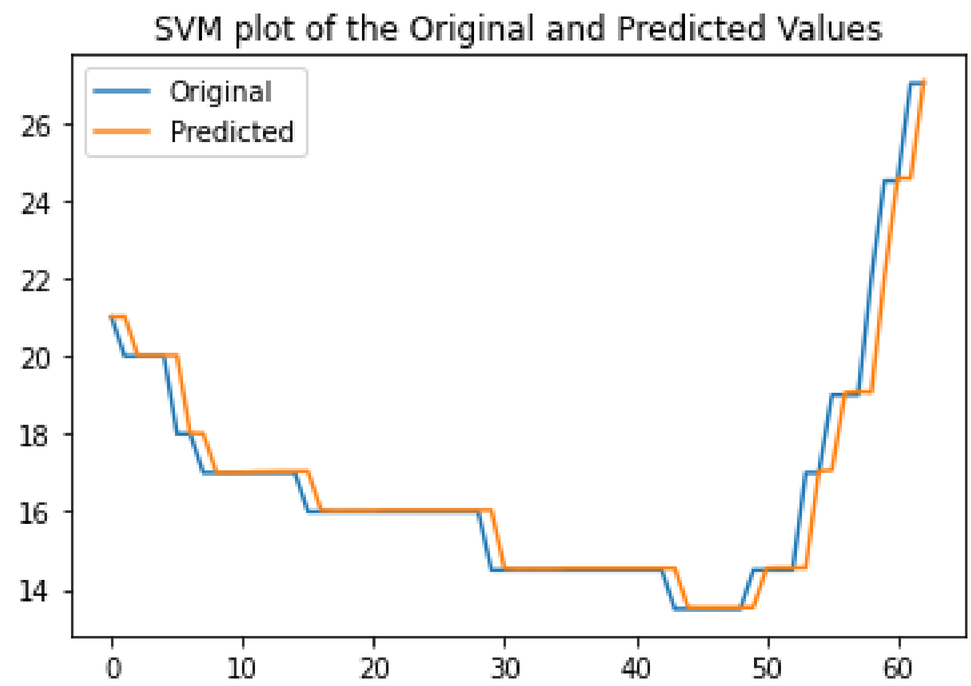
[53]:
```
def walk_forward_validation(data, n_test):
    predictions = list()
    # split dataset
    train, test = train_test_split(data, n_test)
    # seed history with training dataset
    history = [x for x in train]
    # step over each time-step in the test set
    for i in range(len(test)):
        # split test row into input and output columns
        testX, testy = test[i, :-1], test[i, -1]
        # fit model on history and make a prediction
        yhat = svm_forecast(history, testX)
        # store forecast in list of predictions
        predictions.append(yhat)
        # add actual observation to history for the next loop
        history.append(test[i])
        # summarize progress
        print('>expected=%.1f, predicted=%.1f' % (testy, yhat))
    error = mean_absolute_error(test[:, -1], predictions)
    return error, test[:, -1], predictions
```

[54]:
```
series = pd.read_csv('rate.csv', header=0, index_col=0)
values = series.values
# transform the time series data into supervised learning
```

```python
data = series_to_supervised(values, n_in=6)
# evaluate
mae_svm, y, yhat = walk_forward_validation(data, 63)
print('MAE: %.3f' % mae_svm)
mape_svm = mean_absolute_percentage_error(y, yhat)
#mape = mean(abs(yhat - y) / y) * 100
mse_svm = mean_squared_error(y, yhat)
rmse_svm = math.sqrt(mse_svm)
print('RMSE_SVM: %.3f' % rmse_svm)
#maae = mean_absolute_error(y, yhat)
#print('MAEE: %.3f' % maae)
print('MAPE_SVM: %.3f' % mape_svm)
# plot expected vs preducted
pyplot.plot(y, label='Original')
pyplot.plot(yhat, label='Predicted')
pyplot.title('SVM plot of the Original and Predicted Values')
pyplot.legend()
pyplot.show()
```

```
>expected=21.0, predicted=21.0
>expected=20.0, predicted=21.0
>expected=20.0, predicted=20.0
>expected=20.0, predicted=20.0
>expected=20.0, predicted=20.0
>expected=18.0, predicted=20.0
>expected=18.0, predicted=18.0
>expected=17.0, predicted=18.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=17.0, predicted=17.0
>expected=16.0, predicted=17.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.0
```

```
>expected=16.0, predicted=16.0
>expected=14.5, predicted=16.0
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=13.5, predicted=14.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=14.5, predicted=13.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=17.0, predicted=14.6
>expected=17.0, predicted=17.0
>expected=19.0, predicted=17.1
>expected=19.0, predicted=19.1
>expected=19.0, predicted=19.1
>expected=22.0, predicted=19.1
>expected=24.5, predicted=22.0
>expected=24.5, predicted=24.6
>expected=27.0, predicted=24.6
>expected=27.0, predicted=27.1
MAE: 0.359
RMSE_SVM: 0.815
MAPE_SVM: 0.019
```



0.3 MLP MODEL

[55]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

# Import the plotting library
import matplotlib.pyplot as plt
%matplotlib inline

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

from keras.layers import GRU, Embedding, LSTM

from keras.models import load_model
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error
```

```python
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
import math
from statistics import mean
import warnings
warnings.filterwarnings('ignore')
```

[56]:
```python
data= pd.read_csv('rate.csv')
```

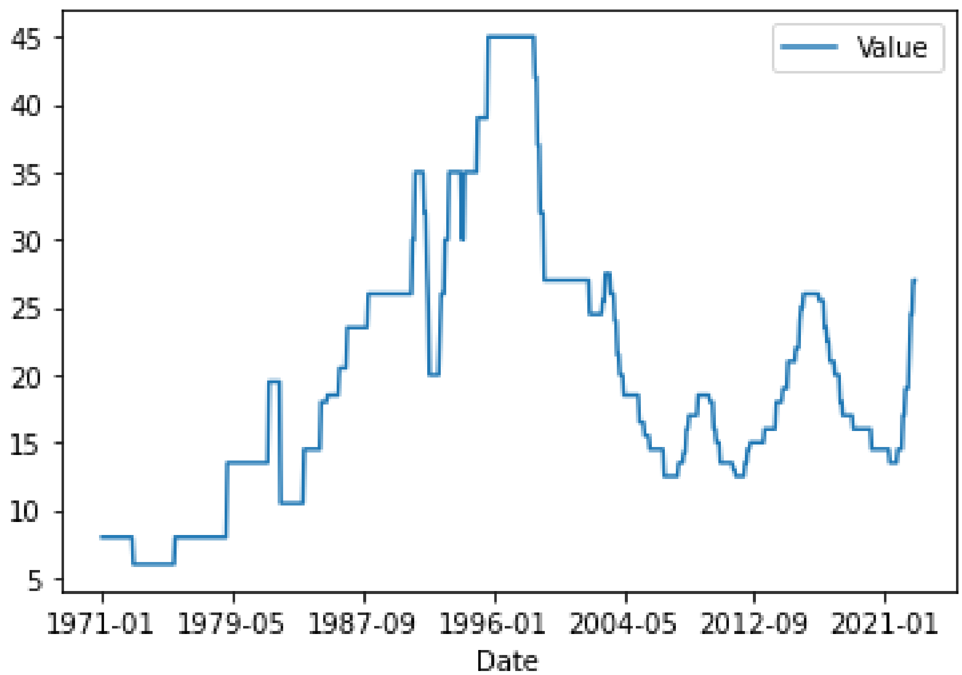
[57]:
```python
data = data.set_index('Date')
```

[58]:
```python
data.shape
```

[58]: (624, 1)

[59]:
```python
data.plot()
```

[59]: <AxesSubplot: xlabel='Date'>



[60]:
```python
train_df = data[:561]
print('train shape: ', train_df.shape)
```

```
train shape:  (561, 1)
```

[61]:
```
test_df = data[561:]
print('test_shape: ', test_df.shape)
```

```
test_shape:  (63, 1)
```

[62]:
```
print('Min x: ', np.min(train_df))
print('Max x: ', np.max(train_df))
```

```
Min x:  Value    6.0
dtype: float64
Max x:  Value    45.0
dtype: float64
```

[63]:
```
x_scaler = MinMaxScaler()
train = x_scaler.fit_transform(train_df.values.reshape(-1,1))
test = x_scaler.fit_transform(test_df.values.reshape(-1,1))
```

[64]:
```
print('Min x:', np.min(train))
print('Max x:', np.max(train))
```

```
Min x: 0.0
Max x: 0.9999999999999999
```

[65]:
```
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = [], []
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return np.array(X), np.array(y)
```

[66]:
```
seq = [1,2,3,4,5,6,7,8,9,10]
steps = 3
split_sequence(seq, steps)
```

[66]:
```
(array([[1, 2, 3],
        [2, 3, 4],
        [3, 4, 5],
        [4, 5, 6],
        [5, 6, 7],
```

```
        [6, 7, 8],
        [7, 8, 9]]),
 array([ 4,  5,  6,  7,  8,  9, 10]))
```

[67]:
```
n_steps = 5
X_train, y_train = split_sequence(train, n_steps)
```

[68]:
```
X_test, y_test = split_sequence(test, n_steps)
```

[69]:
```
print(X_train.shape)
print(X_test.shape)
```

```
(556, 5, 1)
(58, 5, 1)
```

[70]:
```
n_features = 1
model = Sequential()
model.add(Dense(100, activation='relu', input_dim=n_steps))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

[ ]:
```
model.fit(X_train, y_train, epochs = 2000, verbose=0)
```

[ ]:
```
preds = []
# demonstrate predictions
for i in range(X_test.shape[0]):
    x_input = X_test[i]
    x_input = x_input.reshape((1, n_steps, n_features))
    yhat = model.predict(x_input, verbose=0)
    preds.append(yhat[0])
    print(yhat[0], y_test[i])
```

[ ]:
```
 #The output of the model is between 0 and 1.
# Do an inverse map to get it back to the scale
# of the original data-set.
preds = x_scaler.inverse_transform(np.array(preds))
# we also rescale the y_test values into their original range (inverse scaling)
actuals = x_scaler.inverse_transform(y_test)
```

[ ]:
```
mse_mlp = mean_squared_error(actuals, preds)
rmse_mlp = math.sqrt(mse_mlp)
print('RMSE_MLP: %.4f' % rmse_mlp)
mae_mlp = mean_squared_error(actuals, preds)
mape_mlp = mean_absolute_percentage_error(actuals, preds)
print('MAPE_MLP: %.4f' % mape_mlp)
print('MAE_MLP: %.4f' % mae_mlp)
```

[ ]:
```
plt.plot(actuals, marker='o', label='Original')
plt.plot(preds, marker='*', label='Predicted')
plt.title('MLP plot of the Original and Predicted Values')
plt.legend()
```

0.4 Random Forest RF model

[76]:
```
from numpy import asarray
from pandas import DataFrame
from pandas import concat
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.ensemble import RandomForestRegressor
from matplotlib import pyplot
import math
from statistics import mean
import pandas as pd
from sklearn.metrics import mean_squared_error
```

[77]:
```
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

[78]:
```
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]

# fit  randomforest model and make a one step prediction
def randomforest_forecast(train, testX):
    # transform list into array
    train = asarray(train)
    # split into input and output columns
    trainX, trainy = train[:, :-1], train[:, -1]
    # fit model
    model = RandomForestRegressor(n_estimators=1000)
```

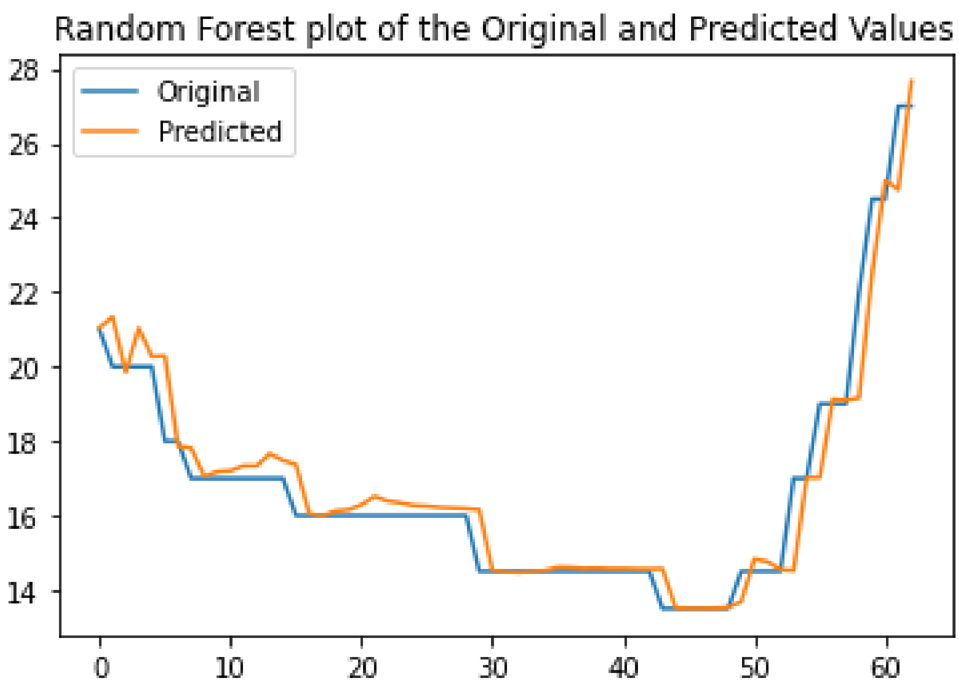
```
    model.fit(trainX, trainy)
    # make a one-step prediction
    yhat = model.predict(asarray([testX]))
    return yhat[0]
```

[79]:
```
def walk_forward_validation(data, n_test):
    predictions = list()
    # split dataset
    train, test = train_test_split(data, n_test)
    # seed history with training dataset
    history = [x for x in train]
    # step over each time-step in the test set
    for i in range(len(test)):
        # split test row into input and output columns
        testX, testy = test[i, :-1], test[i, -1]
        # fit model on history and make a prediction
        yhat = randomforest_forecast(history, testX)
        # store forecast in list of predictions
        predictions.append(yhat)
        # add actual observation to history for the next loop
        history.append(test[i])
        # summarize progress
        print('>expected=%.1f, predicted=%.1f' % (testy, yhat))
    error = mean_absolute_error(test[:, -1], predictions)
    return error, test[:, -1], predictions
```

[80]:
```
series = pd.read_csv('rate.csv', header=0, index_col=0)
values = series.values
# transform the time series data into supervised learning
data = series_to_supervised(values, n_in=6)
# evaluate
mae_randomforest, y, yhat = walk_forward_validation(data, 63)
print('MAE_RANDOMFOREST: %.3f' % mae_randomforest)
mape_randomforest = mean_absolute_percentage_error(y, yhat)
#mape = mean(abs(yhat - y) / y) * 100
mse_randomforest = mean_squared_error(y, yhat)
rmse_randomforest = math.sqrt(mse_randomforest)
print('RMSE_RANDOMFOREST: %.3f' % rmse_randomforest)
#maae = mean_absolute_error(y, yhat)
#print('MAEE: %.3f' % maae)
print('MAPE_RANDOMFOREST: %.3f' % mape_randomforest)
# plot expected vs preducted
pyplot.plot(y, label='Original')
pyplot.plot(yhat, label='Predicted')
pyplot.title('Random Forest plot of the Original and Predicted Values')
pyplot.legend()
pyplot.show()
```

```
>expected=21.0, predicted=21.0
>expected=20.0, predicted=21.3
>expected=20.0, predicted=19.8
>expected=20.0, predicted=21.0
>expected=20.0, predicted=20.3
>expected=18.0, predicted=20.3
>expected=18.0, predicted=17.9
>expected=17.0, predicted=17.8
>expected=17.0, predicted=17.1
>expected=17.0, predicted=17.2
>expected=17.0, predicted=17.2
>expected=17.0, predicted=17.3
>expected=17.0, predicted=17.3
>expected=17.0, predicted=17.7
>expected=17.0, predicted=17.5
>expected=16.0, predicted=17.4
>expected=16.0, predicted=16.1
>expected=16.0, predicted=16.0
>expected=16.0, predicted=16.1
>expected=16.0, predicted=16.2
>expected=16.0, predicted=16.3
>expected=16.0, predicted=16.5
>expected=16.0, predicted=16.4
>expected=16.0, predicted=16.3
>expected=16.0, predicted=16.3
>expected=16.0, predicted=16.3
>expected=16.0, predicted=16.2
>expected=16.0, predicted=16.2
>expected=16.0, predicted=16.2
>expected=14.5, predicted=16.2
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.5
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=14.5, predicted=14.6
>expected=13.5, predicted=14.6
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
>expected=13.5, predicted=13.5
```

```
>expected=13.5, predicted=13.5
>expected=14.5, predicted=13.7
>expected=14.5, predicted=14.8
>expected=14.5, predicted=14.8
>expected=14.5, predicted=14.6
>expected=17.0, predicted=14.5
>expected=17.0, predicted=17.0
>expected=19.0, predicted=17.0
>expected=19.0, predicted=19.1
>expected=19.0, predicted=19.1
>expected=22.0, predicted=19.2
>expected=24.5, predicted=22.5
>expected=24.5, predicted=25.0
>expected=27.0, predicted=24.8
>expected=27.0, predicted=27.7
MAE_RANDOMFOREST: 0.487
RMSE_RANDOMFOREST: 0.848
MAPE_RANDOMFOREST: 0.027
```



0.5 ETS MODEL

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.exponential_smoothing.ets import ETSModel
```

[2]:
```
df = pd.read_csv('rate.csv')
df['Date'] = pd.to_datetime(df['Date'],
                        infer_datetime_format = True)
df.set_index('Date', inplace = True)
df.head()
```

[2]:
```
            Value
Date
1971-01-01    8.0
1971-02-01    8.0
1971-03-01    8.0
1971-04-01    8.0
1971-05-01    8.0
```

[3]:
```
y = df['Value'].values
```

[4]:
```
y = pd.Series(y)
```

[6]:
```
# Create an instance of the ETS model
model = ETSModel(y, error='add', trend='add', seasonal='add',␣
 ↪seasonal_periods=12)

# Fit the model to the time series data
model_fit = model.fit()
```

[8]:
```
# Forecast future values
forecast = model_fit.forecast(steps=12)

# Access the point forecasts
point_forecast = forecast.mean
```

[10]:
```
dff = df.diff().diff().dropna()
```

[17]:
```
train = dff.iloc[:len(dff)-62]
test = dff.iloc[len(dff)-62:]
#X_train, X_test, y_train, y_test = train_test_split(dff, test_size = 0.1)
train_values = train['Value'].values
test_values = test['Value'].values
```

[33]:
```
# Create an instance of the ETS model
model = ETSModel(train_values, error='add', trend='add', seasonal='add',␣
 ↪seasonal_periods=12)

# Fit the model to the training data
model_fit = model.fit()
```

[41]:
```
start = len(train)
end = len(train) + len(test) - 1
```

[46]:
```
# Forecast future values
forecast = model_fit.forecast(steps=len(test_values))

# Access the point forecasts
point_forecast = forecast.mean
#len(point_forecast)
```

[48]:
```
len(forecast)
```

[48]: 62

[49]:
```
# Calculate MAPE using sklearn's mean_absolute_percentage_error function
mape = mean_absolute_percentage_error(test_values, forecast)
```

[50]:
```
mape
```

[50]: 563582285043565.8

[61]:
```
train = dff.iloc[:len(dff)-62]
test = dff.iloc[len(dff)-62:]
#X_train, X_test, y_train, y_test = train_test_split(dff, test_size = 0.1)
train_values = train['Value'].values
test_values = test['Value'].values
```

[62]:
```
# Create an instance of the ETS model
model = ETSModel(train_values, error='add', trend='add', seasonal='add',␣
 ↪seasonal_periods=12)

# Fit the model to the training data
model_fit = model.fit()
```

[63]:
```
# Forecast future values
forecast = model_fit.forecast(steps=len(test_values))

# Access the point forecasts
point_forecast = forecast.mean
#len(point_forecast)
```

[64]:
```
# Calculate MAPE using sklearn's mean_absolute_percentage_error function
mape = mean_absolute_percentage_error(test_values, forecast)
mape
```

[64]: 563582285043565.8

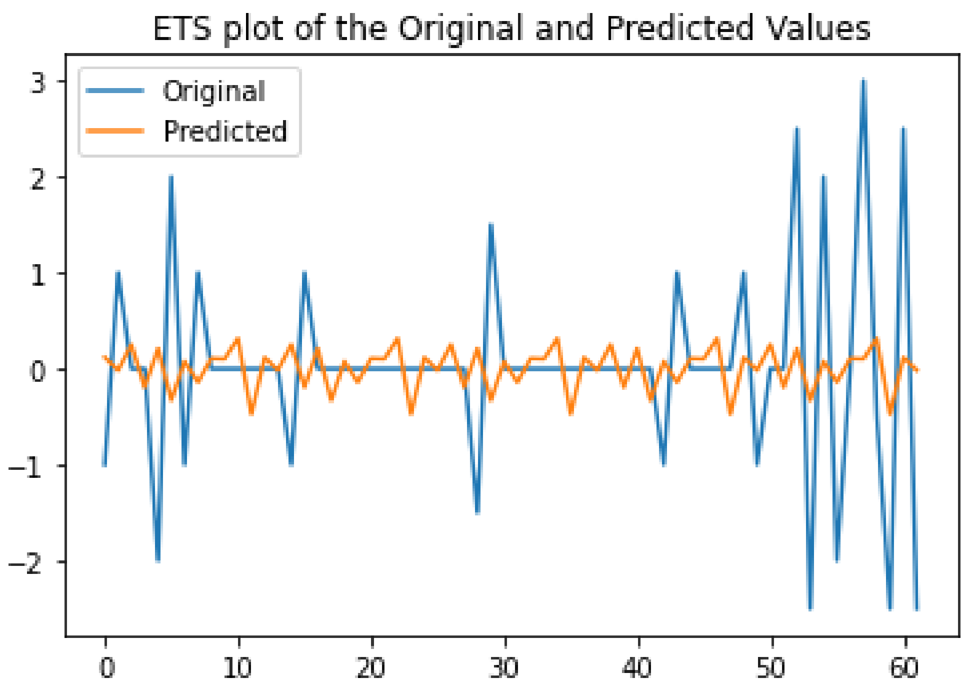
[65]:
```
rmse_ets = rmse(test_values, forecast)
mae_ets = mean_absolute_error(test_values, forecast)
mape_ets = mean_absolute_percentage_error(test_values, forecast)
```

[66]:
```
print("RMSE_ETS: " + str(rmse_ets))
print('MAE_ETS: ' + str(mae_ets))
print('MAPE_ETS: ' + str(mape_ets))
```

```
RMSE_ETS: 1.0836923218701238
MAE_ETS: 0.7348115044603901
MAPE_ETS: 563582285043565.8
```

[67]:
```
#predictions.plot(legend = True)
#test[' Value'].plot(legend = True)
plt.plot(test_values, label='Original')
plt.plot(forecast,  label='Predicted')
plt.title('ETS plot of the Original and Predicted Values')
plt.legend()
```

[67]: <matplotlib.legend.Legend at 0x26233cee8e0>



0.6 TBATS

[89]:
```
train = dff.iloc[:len(dff)-62]
test = dff.iloc[len(dff)-62:]
#X_train, X_test, y_train, y_test = train_test_split(dff, test_size = 0.1)
train_values = train['Value'].values
test_values = test['Value'].values
```

[90]:
```
from tbats import TBATS
```

[91]:
```
model = TBATS(seasonal_periods = [12])
```

[92]:
```
modell = model.fit(train_values)
```

[106]:
```
forecastt = modell.forecast(steps = len(test_values))
forecastt
```

[106]:
```
array([-0.06650583, -0.00425627, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069, -0.00189069, -0.00189069, -0.00189069,
       -0.00189069, -0.00189069])
```

[103]:
```
rmse_tbats = rmse(test_values, forecastt)
mae_tbats = mean_absolute_error(test_values, forecastt)
mape_tbats = mean_absolute_percentage_error(test_values, forecastt)
```

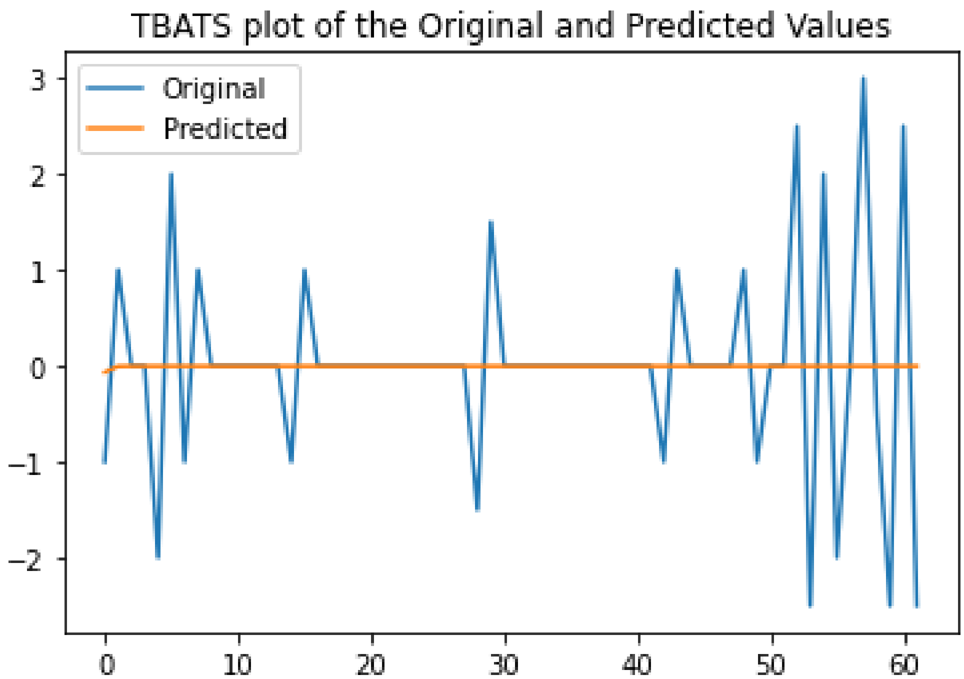
[104]:
```
print("RMSE_TBATS: " + str(rmse_tbats))
print('MAE_TBATS: ' + str(mae_tbats))
print('MAPE_TBATS: ' + str(mape_tbats))
```

```
RMSE_TBATS: 1.0692185608633378
MAE_TBATS: 0.5969289776129075
MAPE_TBATS: 5356149518680.992
```

[105]:
```
#predictions.plot(legend = True)
#test[' Value'].plot(legend = True)
plt.plot(test_values, label='Original')
plt.plot(forecastt,  label='Predicted')
plt.title('TBATS plot of the Original and Predicted Values')
```

```
plt.legend()
```

[105]: <matplotlib.legend.Legend at 0x262339204f0>



0.7 NAIVE

[83]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[84]:
```
train = dff.iloc[:len(dff)-62]
test = dff.iloc[len(dff)-62:]
#X_train, X_test, y_train, y_test = train_test_split(dff, test_size = 0.1)
train_values = train['Value'].values
test_values = test['Value'].values
```

[85]:
```
# Fit the naive model
last_observed_value = train_values[-1]
naive_forecast = np.full(len(test_values), last_observed_value)
```

[86]:
```
mae = mean_absolute_error(test_values, naive_forecast)
```

[87]:
```
rmse_naive = rmse(test_values, naive_forecast)
mae_naive = mean_absolute_error(test_values, naive_forecast)
```

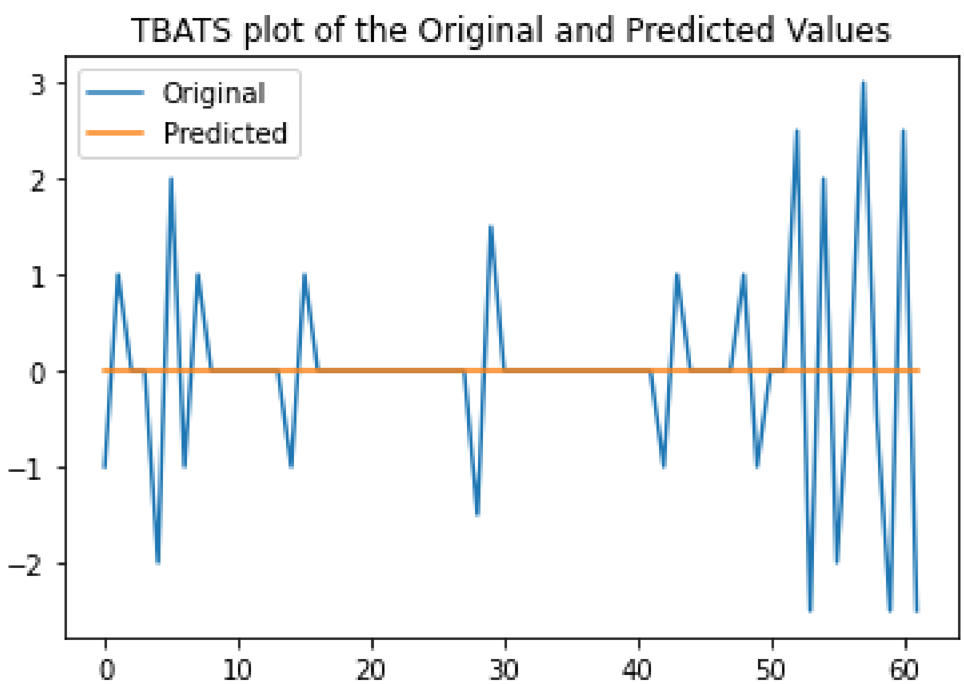
```
mape_naive = mean_absolute_percentage_error(test_values, naive_forecast)
```

[88]:
```
print("RMSE_NAIVE: " + str(rmse_naive))
print('MAE_NAIVE: ' + str(mae_naive))
print('MAPE_NAIVE: ' + str(mape_naive))
```

```
RMSE_NAIVE: 1.0701220913160239
MAE_NAIVE: 0.5967741935483871
MAPE_NAIVE: 0.3709677419354839
```

[97]:
```
#predictions.plot(legend = True)
#test[' Value'].plot(legend = True)
plt.plot(test_values, label='Original')
plt.plot(naive_forecast,  label='Predicted')
plt.title('TBATS plot of the Original and Predicted Values')
plt.legend()
```

[97]: `<matplotlib.legend.Legend at 0x26233847fd0>`



[ ]: