```
import yfinance as yf
```

```
/usr/local/lib/python3.10/dist-packages/yfinance/base.py:48: FutureWarning: The default dtype for empty Series will be 'object' inst
  _empty_series = pd.Series()
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
bitcoin = yf.Ticker("BTC-INR")
```

```
bitcoin.info
```


```
{'name': 'Bitcoin',
 'startDate': 1278979200,
 'description': 'Bitcoin (BTC) is a cryptocurrency launched in 2010. Users are able to generate BTC through the process of mining.
Bitcoin has a current supply of 19,637,118. The last known price of Bitcoin is 51,632.01233526 USD and is up 1.10 over the last 24
hours. It is currently trading on 10824 active market(s) with $16,219,218,573.17 traded over the last 24 hours. More information
can be found at https://bitcoin.org/.',
 'maxAge': 86400,
 'priceHint': 2,
 'previousClose': 4287594.0,
 'open': 4287594.0,
 'dayLow': 4252803.0,
 'dayHigh': 4289538.0,
 'regularMarketPreviousClose': 4287594.0,
 'regularMarketOpen': 4287594.0,
 'regularMarketDayLow': 4252803.0,
 'regularMarketDayHigh': 4289538.0,
 'volume': 1190407700480,
 'regularMarketVolume': 1190407700480,
 'averageVolume': 2031444542872,
 'averageVolume10days': 2071351748260,
 'averageDailyVolume10Day': 2071351748260,
 'marketCap': 83516805087232,
 'fiftyTwoWeekLow': 1611163.8,
 'fiftyTwoWeekHigh': 4391519.0,
 'fiftyDayAverage': 3768850.8,
 'twoHundredDayAverage': 3012378.8,
 'currency': 'INR',
 'fromCurrency': 'BTC',
 'toCurrency': 'INR=X',
 'lastMarket': 'CoinMarketCap',
 'coinMarketCapLink': 'https://coinmarketcap.com/currencies/bitcoin',
 'volume24Hr': 1190407700480,
 'volumeAllCurrencies': 1190407700480,
 'circulatingSupply': 19638062,
 'exchange': 'CCC',
 'quoteType': 'CRYPTOCURRENCY',
 'symbol': 'BTC-INR',
 'underlyingSymbol': 'BTC-INR',
 'shortName': 'Bitcoin INR',
 'longName': 'Bitcoin INR',
 'firstTradeDateEpochUtc': 1410912000,
 'timeZoneFullName': 'UTC',
 'timeZoneShortName': 'UTC',
 'uuid': '892be90d-0089-36ad-b305-df577f7792e2',
 'messageBoardId': 'finmb_BTC_CCC',
 'trailingPegRatio': None}
```


```
bitcoin.history(period='5d')
```

Date	Open	High	Low	Close	Volume	Dividends	S Sp
2024-02-22 00:00:00+00:00	4331742.5	4340472.0	4201856.5	4299430.5	2374088359158	0.0	
2024-02-23 00:00:00+00:00	4300713.0	4309904.0	4220758.0	4251773.5	2106114469501	0.0	
2024-02-24	4250027.5	4267456.0	4190658.5	4204763.0	1775918029779	0.0	

```
df=bitcoin.history(start='2001-01-19', end='2022-05-13', actions=False)
```

```
df
```

Date	Open	High	Low	Close	Volume
2014-09-17 00:00:00+00:00	2.844333e+04	2.854223e+04	2.755250e+04	2.785164e+04	1282359120
2014-09-18 00:00:00+00:00	2.782277e+04	2.782277e+04	2.508574e+04	2.577412e+04	2093992320
2014-09-19 00:00:00+00:00	2.575365e+04	2.598884e+04	2.336609e+04	2.402334e+04	2307413745
2014-09-20 00:00:00+00:00	2.401585e+04	2.575756e+04	2.372438e+04	2.488181e+04	2243150060
2014-09-21 00:00:00+00:00	2.483197e+04	2.509612e+04	2.392506e+04	2.426826e+04	1617399085
...	...	...	...	...	...
2022-05-08 00:00:00+00:00	2.732061e+06	2.732061e+06	2.607091e+06	2.621124e+06	2829200767989
2022-05-09 00:00:00+00:00	2.621182e+06	2.635168e+06	2.343956e+06	2.343956e+06	4901564644382

Next steps: Generate code with df View recommended plots

```
df.index
```

```
DatetimeIndex(['2014-09-17 00:00:00+00:00', '2014-09-18 00:00:00+00:00',
               '2014-09-19 00:00:00+00:00', '2014-09-20 00:00:00+00:00',
               '2014-09-21 00:00:00+00:00', '2014-09-22 00:00:00+00:00',
               '2014-09-23 00:00:00+00:00', '2014-09-24 00:00:00+00:00',
               '2014-09-25 00:00:00+00:00', '2014-09-26 00:00:00+00:00',
               ...
               '2022-05-03 00:00:00+00:00', '2022-05-04 00:00:00+00:00',
               '2022-05-05 00:00:00+00:00', '2022-05-06 00:00:00+00:00',
               '2022-05-07 00:00:00+00:00', '2022-05-08 00:00:00+00:00',
               '2022-05-09 00:00:00+00:00', '2022-05-10 00:00:00+00:00',
               '2022-05-11 00:00:00+00:00', '2022-05-12 00:00:00+00:00'],
              dtype='datetime64[ns, UTC]', name='Date', length=2795, freq=None)
```

```
df.shape
```

```
(2795, 5)
```

```
df.isnull().sum()
```

```
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

```
df=df.drop(['Open','High','Volume','Low'],axis=1)
```

```
df.head()
```

Date	Close
2014-09-17 00:00:00+00:00	27851.640625
2014-09-18 00:00:00+00:00	25774.119141
2014-09-19 00:00:00+00:00	24023.335938
2014-09-20 00:00:00+00:00	24881.808594
2014-09-21 00:00:00+00:00	24268.257812

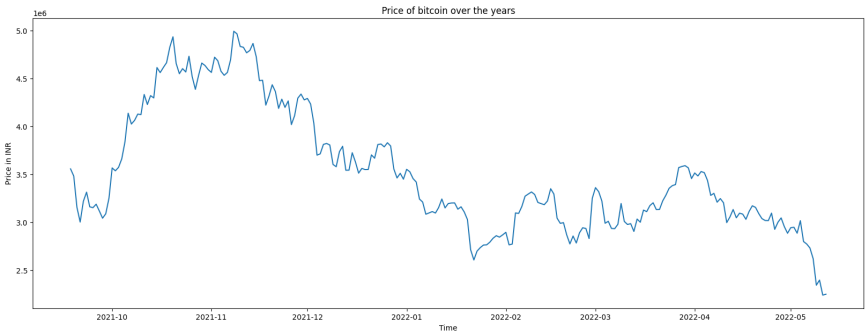
```
import matplotlib.pyplot as plt
```

```
plt.rcParams["figure.figsize"] = (20,7)
```

```
plt.figure(figsize=(20,7))
plt.title("Price of bitcoin over the years")
plt.plot(df['2021-09-18':'2024-01-01'])
plt.ylabel("Price in INR")
```

```
plt.xlabel("Time")
```

```
Text(0.5, 0, 'Time')
```




```
data=df.values
```

```
data
```

```
array([[  27851.640625  ],
       [  25774.11914062],
       [  24023.3359375 ],
       ...,
       [2397189.5       ],
       [2240213.5       ],
       [2249575.25      ]])
```

```
len(data)
```

```
2795
```

```
import math
```

```
train_len=math.ceil(len(data)*0.92)
```

```
train_len
```

```
2572
```

```
min_max_scalar=MinMaxScaler(feature_range=(0,1))
```

```
scaled_data=min_max_scalar.fit_transform(data)
```

```
len(scaled_data)
```

```
2795
```

```
scaled_data
```

```
array([[0.00336983],
       [0.00295295],
       [0.00260162],
       ...,
       [0.47881612],
       [0.44731632],
       [0.44919491]])
```

```
train_data=scaled_data[0:train_len,:]
```

```
len(train_data)
```

```
2572
```

```
interval=60


x_train=[]
y_train=[]

for i in range(interval,len(train_data)):
    x_train.append(train_data[i-interval:i,0])
    y_train.append(train_data[i,0])


x_train
```

```
              0.03329724, 0.03153508, 0.03128523, 0.02982466, 0.03080188,
              0.0311064 , 0.03072629, 0.02994765, 0.02934746, 0.03027997,
              0.03114246, 0.03159748, 0.03161247, 0.03165864, 0.03043045,
              0.03111334, 0.0304276 , 0.02846513, 0.02806237, 0.02880501,
              0.0282779 , 0.02663127, 0.02360225, 0.02271039, 0.02654272,
              0.02768513, 0.02709185, 0.03417558, 0.03224547, 0.0340846 ,
              0.0330547 , 0.03335412, 0.03106623, 0.03042469, 0.03215264,
              0.03391807, 0.03285596, 0.03325129, 0.03480601, 0.03273968,
              0.03239186, 0.03362662, 0.03475322, 0.03931135, 0.03881781]),
       array([0.03418213, 0.0357793 , 0.03591337, 0.03218847, 0.03285459,
              0.03008505, 0.02969973, 0.03034324, 0.03211864, 0.03072761,
              0.03128249, 0.03302813, 0.0326021 , 0.03284587, 0.03329724,
              0.03153508, 0.03128523, 0.02982466, 0.03080188, 0.0311064 ,
              0.03072629, 0.02994765, 0.02934746, 0.03027997, 0.03114246,
              0.03159748, 0.03161247, 0.03165864, 0.03043045, 0.03111334,
              0.0304276 , 0.02846513, 0.02806237, 0.02880501, 0.0282779 ,
              0.02663127, 0.02360225, 0.02271039, 0.02654272, 0.02768513,
              0.02709185, 0.03417558, 0.03224547, 0.0340846 , 0.0330547 ,
              0.03335412, 0.03106623, 0.03042469, 0.03215264, 0.03391807,
              0.03285596, 0.03325129, 0.03480601, 0.03273968, 0.03239186,
              0.03362662, 0.03475322, 0.03931135, 0.03881781, 0.04105342]),
       array([0.0357793 , 0.03591337, 0.03218847, 0.03285459, 0.03008505,
              0.02969973, 0.03034324, 0.03211864, 0.03072761, 0.03128249,
              0.03302813, 0.0326021 , 0.03284587, 0.03329724, 0.03153508,
              0.03128523, 0.02982466, 0.03080188, 0.0311064 , 0.03072629,
              0.02994765, 0.02934746, 0.03027997, 0.03114246, 0.03159748,
              0.03161247, 0.03165864, 0.03043045, 0.03111334, 0.0304276 ,
              0.02846513, 0.02806237, 0.02880501, 0.0282779 , 0.02663127,
              0.02360225, 0.02271039, 0.02654272, 0.02768513, 0.02709185,
              0.03417558, 0.03224547, 0.0340846 , 0.0330547 , 0.03335412,
              0.03106623, 0.03042469, 0.03215264, 0.03391807, 0.03285596,
              0.03325129, 0.03480601, 0.03273968, 0.03239186, 0.03362662,
              0.03475322, 0.03931135, 0.03881781, 0.04105342, 0.04148722]),
       array([0.03591337, 0.03218847, 0.03285459, 0.03008505, 0.02969973,
              0.03034324, 0.03211864, 0.03072761, 0.03128249, 0.03302813,
              0.0326021 , 0.03284587, 0.03329724, 0.03153508, 0.03128523,
              0.02982466, 0.03080188, 0.0311064 , 0.03072629, 0.02994765,
              0.02934746, 0.03027997, 0.03114246, 0.03159748, 0.03161247,
              0.03165864, 0.03043045, 0.03111334, 0.0304276 , 0.02846513,
              0.02806237, 0.02880501, 0.0282779 , 0.02663127, 0.02360225,
              0.02271039, 0.02654272, 0.02768513, 0.02709185, 0.03417558,
              0.03224547, 0.0340846 , 0.0330547 , 0.03335412, 0.03106623,
              0.03042469, 0.03215264, 0.03391807, 0.03285596, 0.03325129,
              0.03480601, 0.03273968, 0.03239186, 0.03362662, 0.03475322,
              0.03931135, 0.03881781, 0.04105342, 0.04148722, 0.04060985]),
       array([0.03218847, 0.03285459, 0.03008505, 0.02969973, 0.03034324,
              0.03211864, 0.03072761, 0.03128249, 0.03302813, 0.0326021 ,
              0.03284587, 0.03329724, 0.03153508, 0.03128523, 0.02982466,
              0.03080188, 0.0311064 , 0.03072629, 0.02994765, 0.02934746,
              0.03027997, 0.03114246, 0.03159748, 0.03161247, 0.03165864,
              0.03043045, 0.03111334, 0.0304276 , 0.02846513, 0.02806237,
              0.02880501, 0.0282779 , 0.02663127, 0.02360225, 0.02271039,
              0.02654272, 0.02768513, 0.02709185, 0.03417558, 0.03224547,
              0.0340846 , 0.0330547 , 0.03335412, 0.03106623, 0.03042469,
              0.03215264, 0.03391807, 0.03285596, 0.03325129, 0.03480601,
              0.03273968, 0.03239186, 0.03362662, 0.03475322, 0.03931135,
              0.03881781, 0.04105342, 0.04148722, 0.04060985, 0.04126659]),
       ...]
```

```
y_train
```

```
        0.02934745534397972,
        0.03027997363010327,
        0.031142456238950476,
        0.0315974796364099,
        0.03161247317219069,
        0.03165863896492099,
        0.030430448278616767,
        0.031113340811668923,
        0.030427595054534684,
        0.028465134989234343,
        0.028062366352821052,
        0.028805013550220818,
        0.028277900781228178,
        0.026631267538525683,
        0.023602250363465523,
        0.022710390520511645,
        0.026542717258826606,
        0.027685132504236855,
        0.027091850019828546,
        0.034175580802524405,
        0.032245465637917206,
        0.03408460057923894,
        0.033054702695817256,
        0.033354118776826515,
        0.0310662281247927,
        0.03042469479928641,
        0.032152641792917165,
        0.033918066355177724,
        0.032855955258328226,
        0.03325128983507271,
        0.03480600536657705,
        0.032739684809277844,
        0.03239185798120557,
        0.0336266236245429,
        0.03475321758564745,
        0.039311346525339116,
        0.03881781087359375,
        0.041053424816703876,
        0.04148721834574596,
        0.04060984566968357,
        0.04126658887433511,
        0.04476345641357975,
        ...]
```

```
import numpy as np
```

```
x_train,y_train=np.array(x_train),np.array(y_train)
```

```
x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
```

```
x_train.shape
```

```
(2512, 60, 1)
```

```
import tensorflow as tf
import keras
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten,Dense,Dropout,BatchNormalization,LSTM
from keras import regularizers
from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adamax
```

```
model=Sequential()
```

```
model.add(LSTM(50,return_sequences=True,input_shape=(x_train.shape[1],1)))
```

```
model.add(LSTM(units=50))
```

```
model.add(Dense(50))
```

```
model.add(Dense(1))
```

```
model.compile(optimizer="adam",loss="mean_squared_error")
```

```
history=model.fit(x_train,y_train,batch_size=64,epochs=100)
```

```
40/40 [==============================] - 3s 73ms/step - loss: 1.1778e-04
Epoch 74/100
40/40 [==============================] - 3s 84ms/step - loss: 1.2982e-04
Epoch 75/100
40/40 [==============================] - 3s 65ms/step - loss: 1.3430e-04
Epoch 76/100
40/40 [==============================] - 3s 65ms/step - loss: 1.2418e-04
Epoch 77/100
40/40 [==============================] - 3s 65ms/step - loss: 1.5015e-04
Epoch 78/100
40/40 [==============================] - 4s 94ms/step - loss: 1.3265e-04
Epoch 79/100
40/40 [==============================] - 3s 65ms/step - loss: 1.2127e-04
Epoch 80/100
40/40 [==============================] - 3s 64ms/step - loss: 1.2165e-04
Epoch 81/100
40/40 [==============================] - 3s 64ms/step - loss: 1.2341e-04
Epoch 82/100
40/40 [==============================] - 3s 71ms/step - loss: 1.2927e-04
Epoch 83/100
40/40 [==============================] - 3s 86ms/step - loss: 1.2656e-04
Epoch 84/100
40/40 [==============================] - 3s 64ms/step - loss: 1.2630e-04
Epoch 85/100
40/40 [==============================] - 3s 64ms/step - loss: 1.3073e-04
Epoch 86/100
40/40 [==============================] - 3s 64ms/step - loss: 1.2338e-04
Epoch 87/100
40/40 [==============================] - 4s 91ms/step - loss: 1.4644e-04
Epoch 88/100
40/40 [==============================] - 3s 68ms/step - loss: 1.2915e-04
Epoch 89/100
40/40 [==============================] - 3s 65ms/step - loss: 1.3122e-04
Epoch 90/100
40/40 [==============================] - 3s 65ms/step - loss: 1.4108e-04
Epoch 91/100
40/40 [==============================] - 3s 67ms/step - loss: 1.3016e-04
Epoch 92/100
40/40 [==============================] - 4s 90ms/step - loss: 1.5471e-04
Epoch 93/100
40/40 [==============================] - 3s 65ms/step - loss: 1.1425e-04
Epoch 94/100
40/40 [==============================] - 3s 64ms/step - loss: 1.2775e-04
Epoch 95/100
40/40 [==============================] - 3s 64ms/step - loss: 1.4535e-04
Epoch 96/100
40/40 [==============================] - 3s 85ms/step - loss: 1.2684e-04
Epoch 97/100
40/40 [==============================] - 3s 72ms/step - loss: 1.4084e-04
Epoch 98/100
40/40 [==============================] - 3s 65ms/step - loss: 1.4210e-04
Epoch 99/100
40/40 [==============================] - 3s 65ms/step - loss: 1.4079e-04
Epoch 100/100
40/40 [==============================] - 3s 64ms/step - loss: 1.4167e-04
```

```
test_data=scaled_data[train_len-interval:,:]
```

```
x_test=[]
y_test=data[train_len:,:]
for i in range(interval,len(test_data)):
    x_test.append(test_data[i-interval:i,0])
```

```
y_test
```

```
       [3210373.75],
       [3247912.75],
       [3204303.  ],
       [2997283.75],
       [3055683.  ],
       [3133692.  ],
       [3047441.5 ],
       [3095510.75],
       [3085665.5 ],
       [3031660.75],
       [3114242.  ],
       [3172704.5 ],
       [3155890.75],
       [3091907.5 ],
       [3038692.  ],
       [3019340.75],
       [3017970.  ],
       [3096191.5 ],
       [2927203.75],
       [3000809.75],
       [3046247.5 ],
       [2954609.  ],
       [2886123.25],
       [2943784.25],
       [2948474.25],
       [2886684.5 ],
       [3017147.5 ],
       [2799536.25],
       [2773460.75],
       [2731985.5 ],
       [2621124.25],
       [2343955.75],
       [2397189.5 ],
       [2240213.5 ],
       [2249575.25]])
```

```
x_test
```

```
               0.61095358, 0.62660736, 0.60929979, 0.61894567, 0.61697006,
               0.60613312, 0.6227044 , 0.63443585, 0.6310619 , 0.61822261,
               0.60754406, 0.60366091, 0.60338585, 0.61908227, 0.58517212,
               0.59994237, 0.60906019, 0.59067143, 0.57692865, 0.58849927,
               0.58944039, 0.57704127, 0.6032208 , 0.55955356, 0.55432108,
               0.5459984 , 0.52375228, 0.4681339 , 0.47881612, 0.44731632])]
```

```
len(y_test)
```

```
223
```

```
len(x_test)
```

```
223
```

```
x_test=np.array(x_test)
x_test=np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

```
predictions=model.predict(x_test)
predictions=min_max_scalar.inverse_transform(predictions)
```

```
7/7 [==============================] - 2s 16ms/step
```

```
predictions[0:5]
```

```
array([[3399685.5],
       [3409987.8],
       [3418403.2],
       [3489297.8],
       [3653614. ]], dtype=float32)
```

```
rmse_error=np.sqrt(np.mean(predictions-y_test)**2)
rmse_error
```

```
169477.68497757846
```

```
train_data=df[0:train_len]
valid_data=df[train_len:]
valid_data['predictions']=predictions
```

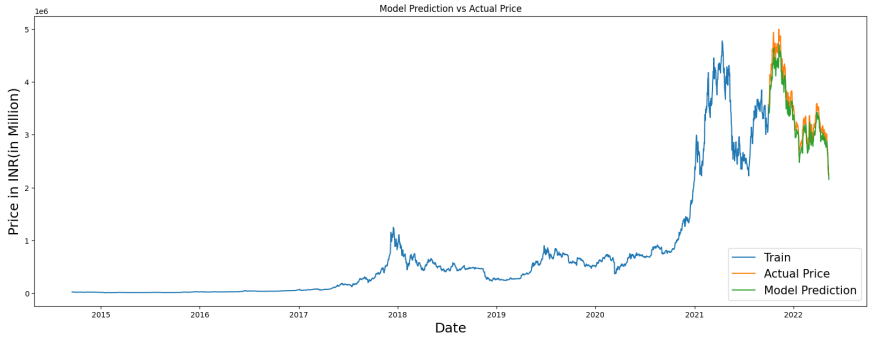
```
<ipython-input-60-373018d9c8b9>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  valid_data['predictions']=predictions
```

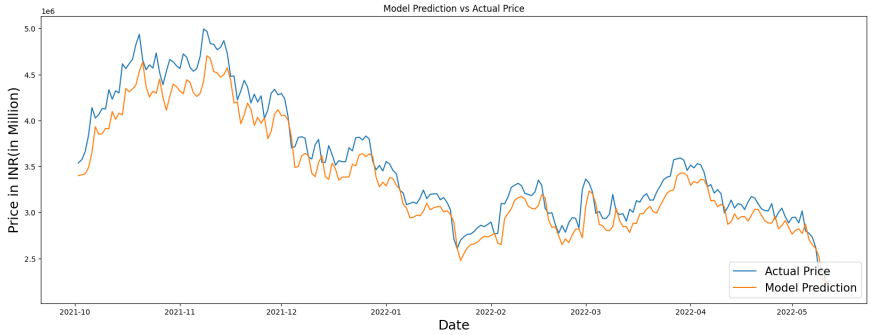
```
train_data.head()
```

Date	Close
2014-09-17 00:00:00+00:00	27851.640625
2014-09-18 00:00:00+00:00	25774.119141
2014-09-19 00:00:00+00:00	24023.335938
2014-09-20 00:00:00+00:00	24881.808594
2014-09-21 00:00:00+00:00	24268.257812

```
plt.figure(figsize=(20,7))
plt.title("Model Prediction vs Actual Price")
plt.xlabel("Date",fontsize=18)
plt.ylabel("Price in INR(in Million)",fontsize=18)
plt.plot(train_data['Close'])
plt.plot(valid_data['Close'])
plt.plot(valid_data['predictions'])
plt.legend(['Train','Actual Price','Model Prediction'],loc='lower right',fontsize=15)
plt.show()
```




```
plt.figure(figsize=(20,7))
plt.title("Model Prediction vs Actual Price")
plt.xlabel("Date",fontsize=18)
plt.ylabel("Price in INR(in Million)",fontsize=18)
# plt.plot(train_data['Close'])
plt.plot(valid_data['Close'])
plt.plot(valid_data['predictions'])
plt.legend(['Actual Price','Model Prediction'],loc='lower right',fontsize=15)
plt.show()
```




```
valid_data.head(30)
```

Date	Close	predictions
2021-10-02 00:00:00+00:00	3537914.00	3399685.50
2021-10-03 00:00:00+00:00	3574140.25	3409987.75
2021-10-04 00:00:00+00:00	3662498.25	3418403.25
2021-10-05 00:00:00+00:00	3841122.50	3489297.75
2021-10-06 00:00:00+00:00	4139246.00	3653614.00
2021-10-07 00:00:00+00:00	4026320.25	3931873.50
2021-10-08 00:00:00+00:00	4064102.50	3850283.25
2021-10-09 00:00:00+00:00	4129844.50	3854173.00
2021-10-10 00:00:00+00:00	4124631.50	3912062.25
2021-10-11 00:00:00+00:00	4334517.50	3910221.50
2021-10-12 00:00:00+00:00	4230211.50	4097603.75
2021-10-13 00:00:00+00:00	4322679.00	4012020.00
2021-10-14 00:00:00+00:00	4298842.50	4079265.00
2021-10-15 00:00:00+00:00	4615743.00	4060669.50
2021-10-16 00:00:00+00:00	4563153.50	4347150.00
2021-10-17 00:00:00+00:00	4616900.00	4311572.00
2021-10-18 00:00:00+00:00	4666578.00	4342549.50
2021-10-19 00:00:00+00:00	4827383.00	4386404.00
2021-10-20 00:00:00+00:00	4936755.50	4534532.50
2021-10-21 00:00:00+00:00	4657473.50	4636790.00
2021-10-22 00:00:00+00:00	4551546.50	4366046.00
2021-10-23 00:00:00+00:00	4604490.00	4254775.50
2021-10-24 00:00:00+00:00	4569445.00	4318236.00
2021-10-25 00:00:00+00:00	4733682.50	4295657.50
2021-10-26 00:00:00+00:00	4520599.00	4451377.00
2021-10-27 00:00:00+00:00	4388398.50	4249597.00
2021-10-28 00:00:00+00:00	4534387.00	4111793.75
2021-10-29 00:00:00+00:00	4662651.00	4259331.00
2021-10-30 00:00:00+00:00	4637240.50	4394331.00
2021-10-31 00:00:00+00:00	4594486.50	4368485.50

Next steps: Generate code with valid_data View recommended plots

```
df_test=bitcoin.history(start='2001-01-19', end='2021-05-13', actions=False)
```

```
df_test.head(1)
```

Date	Open	High	Low	Close	Volume
2014-09-17 00:00:00+00:00	28443.328125	28542.228516	27552.5	27851.640625	1282359120

Next steps: Generate code with df_test View recommended plots

```
df_test=df_test.drop(['Open','High','Volume','Low'],axis=1)
```

```
test_value=df_test[-60:].values
```

```
test_value
```

```
array([[4310706.5 ],
       [4054141.25],
       [4119249.75],
       [4272263.5 ],
```

```
[4204622.  ],
[4226751.  ],
[4224360.  ],
[4165414.75],
[3946860.5 ],
[3974729.  ],
[3833831.5 ],
[3754396.75],
[3994554.75],
[4055135.  ],
[4053488.75],
[4200873.5 ],
[4327242.  ],
[4313079.  ],
[4330986.5 ],
[4357392.5 ],
[4226751.5 ],
[4311483.  ],
[4327746.5 ],
[4276035.5 ],
[4170133.5 ],
[4347744.  ],
[4352631.5 ],
[4468330.5 ],
[4499101.5 ],
[4491485.5 ],
[4774301.  ],
[4734890.5 ],
[4731307.5 ],
[4591076.5 ],
[4524215.5 ],
[4191673.  ],
[4172547.  ],
[4260859.5 ],
[4066144.75],
[3886924.75],
[3825294.75],
[3747273.5 ],
[3668868.  ],
[4041646.  ],
[4104108.5 ],
[4074136.75],
[3968757.5 ],
[4279028.5 ],
[4284798.5 ],
[4196083.  ],
[4223535.  ],
[3934903.  ],
[4237814.  ],
[4151654.75],
[4203679.5 ],
[4310081.5 ],
[4267878.5 ],
[4105709.25]
```

```
test_value=min_max_scalar.transform(test_value)
```

```
test=[]
test.append(test_value)
```

```
test
```

```
[0.82437559],
[0.8550803 ],
[0.84150693],
[0.84594747],
[0.84546768],
[0.83363935],
[0.78978288],
[0.79537515],
[0.76710177],
[0.75116189],
[0.79935351],
[0.81150992],
[0.81117957],
[0.84075473],
[0.86611263],
[0.86327059],
[0.86686402],
[0.87216282],
[0.84594757],
[0.86295033],
[0.86621387],
[0.85583721],
[0.83458625],
[0.87022669],
```

```
[0.90059904],
[0.89907077],
[0.95582231],
[0.94791395],
[0.94719496],
[0.91905532],
[0.90563857],
[0.8389085 ],
[0.83507055],
[0.8527919 ],
[0.81371921],
[0.77775579],
[0.76538873],
[0.74973249],
[0.73399915],
[0.80880313],
[0.82133725],
[0.81532293],
[0.79417687],
[0.8564378 ],
[0.85759565],
[0.83979344],
[0.84530213],
[0.78738341],
[0.84816744],
[0.83087818],
[0.8413178 ],
[0.8626691 ],
[0.85420037],
[0.82165847],
[0.83269687],
[0.72449439]])]
```

```
test=np.array(test)
```

```
test
```

```
[0.82437559],
[0.8550803 ],
[0.84150693],
[0.84594747],
[0.84546768],
[0.83363935],
[0.78978288],
[0.79537515],
[0.76710177],
[0.75116189],
[0.79935351],
[0.81150992],
[0.81117957],
[0.84075473],
[0.86611263],
[0.86327059],
[0.86686402],
[0.87216282],
[0.84594757],
[0.86295033],
[0.86621387],
[0.85583721],
[0.83458625],
[0.87022669],
[0.87120745],
[0.89442434],
[0.90059904],
[0.89907077],
[0.95582231],
[0.94791395],
[0.94719496],
[0.91905532],
[0.90563857],
[0.8389085 ],
[0.83507055],
[0.8527919 ],
[0.81371921],
[0.77775579],
[0.76538873],
[0.74973249],
[0.73399915],
[0.80880313],
[0.82133725],
[0.81532293],
[0.79417687],
[0.8564378 ],
[0.85759565],
[0.83979344],
[0.84530213],
[0.78738341],
```

```
                [0.8413178 ],
                [0.8626691 ],
                [0.85420037],
                [0.82165847],
                [0.83269687],
                [0.72449439]]])
```

```
test=np.reshape(test,(test.shape[0],test.shape[1],1))
```

```
tomorrow_prediction=model.predict(test)
```

```
     1/1 [==============================] - 0s 28ms/step
```

```
tomorrow_prediction=min_max_scalar.inverse_transform(tomorrow_prediction)
```

```
tomorrow_prediction
```