In [2]:

```python
#!/usr/bin/env python
# coding: utf-8

# Import necessary libraries
import yfinance
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from statsmodels.tsa.arima.model import ARIMA
import math
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import LSTM, Dense
from statsmodels.tsa.api import SimpleExpSmoothing

#Suppress/ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Download data using Yahoo Finance API
api_key = '9BTZQJA8HHVIH64EMVXJ2M4C9XH16KT5W5'
symbol_data = ['ETH-USD', 'Ethereum']
df = yfinance.download(symbol_data[0], '2021-06-01', '2023-07-30')

# Initial data exploration
df.info()
df.describe()

# Plot the closing prices
plt.figure(figsize=(10, 8))
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.plot(df['Close'])
plt.title(symbol_data[1] + ' Price in the Last 2 Years')
plt.show()

# Log transformation of the closing prices
dfclose = df['Close']
dflog = np.log(dfclose)

# Split data into training and testing sets
training_data, testing_data = dflog[3:int(len(dflog) * 0.6)], dflog[int(len(dflog) * 0.6):]
```

```python

# Plot the training and testing data
plt.figure(figsize=(10, 8))
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.plot(dflog, 'green', label='Train data')
plt.plot(testing_data, 'blue', label='Test data')
plt.title('Train vs Test Data - ETH Prices')
plt.legend()

# Find the best ARIMA model parameters
best_rmse = float('inf')
best_order = None

for p in range(4):
    for d in range(4):
        for q in range(4):
            model = ARIMA(training_data, order=(p, d, q))
            fitted_model = model.fit()
            fcast = fitted_model.forecast(steps=len(testing_data), alpha=0.05)
            rmse = math.sqrt(mean_squared_error(testing_data, fcast))
            if rmse < best_rmse:
                best_rmse = rmse
                best_order = (p, d, q)

# Fit the best ARIMA model
model = ARIMA(training_data, order=best_order)
fitted_model = model.fit()
print(fitted_model.summary())

# Plot residuals and density
residuals = pd.DataFrame(fitted_model.resid)
fig, ax = plt.subplots(1, 2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()

# Make ARIMA predictions
fcast = fitted_model.forecast(steps=len(testing_data), alpha=0.05)

# Plot ARIMA predictions with confidence intervals
if isinstance(fcast, tuple):
    fc_series = fcast[0]
```

```
    conf = fcast[2]
else:
    fc_series = fcast
    conf = None

if conf is not None:
    lower_series = pd.Series(conf[:, 0], index=testing_data.index)
    upper_series = pd.Series(conf[:, 1], index=testing_data.index)

fc_series = pd.Series(fc_series, index=testing_data.index)

plt.figure(figsize=(10, 8))
if conf is not None:
    plt.fill_between(lower_series.index, lower_series, upper_series, color='k', alpha=0.09)
plt.plot(testing_data.index, fc_series, label='Forecast', color='blue')
plt.plot(testing_data.index, testing_data, label='Actual', color='green')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Prediction of ' + symbol_data[1] + ' Price - ARIMA Model')
plt.legend()
plt.show()

#Create RMSE and MAPE result archive
rmse_mape_results = []

# Calculate RMSE and MAPE for ARIMA Model
rmse = mean_squared_error(np.exp(testing_data), np.exp(fcast), squared=False) # Back-transform to o
mape = mean_absolute_percentage_error(np.exp(testing_data), np.exp(fcast)) # Back-transform to orig
print("RMSE: ", rmse)
print("MAPE: ", mape)

rmse_mape_results.append(['ARIMA', rmse, mape])

# LSTM modeling
def lstm_split(data, n_steps):
    X, y = [], []
    for i in range(len(data) - n_steps + 1):
        X.append(data[i:i + n_steps, :-1])
        y.append(data[i + n_steps - 1, -1])
    return np.array(X), np.array(y)

X_feat = df.iloc[:,0:4]

```

```
X1, y1 = lstm_split(X_feat.values, n_steps=2)
train_split = 0.8
split_idx = int(np.ceil(len(X1) * train_split))
date_index = X_feat.index

X_train, X_test = X1[:split_idx], X1[split_idx:]
y_train, y_test = y1[:split_idx], y1[split_idx:]
X_train_date, X_test_date = date_index[:split_idx], date_index[split_idx:]

# Create and compile the LSTM model
lstm = Sequential()
lstm.add(LSTM(32, input_shape=(X_train.shape[1], X_train.shape[2]), activation='relu', return_seque
lstm.add(LSTM(32, activation='relu'))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')

# Train the LSTM model
history = lstm.fit(X_train, y_train, epochs=100, batch_size=4, verbose=2, shuffle=False)

# Make predictions using the LSTM model
y_pred = lstm.predict(X_test)

# Copy the X_test_date set for graphing y values
y_date_index = X_test_date

# Plot LSTM predictions vs. actual values
plt.figure(figsize=(10, 8))
plt.plot(y_date_index[1:len(y_date_index)], y_pred, label='Forecast', color='blue')
plt.plot(y_date_index[1:len(y_date_index)], y_test, label='Actual', color='green')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Prediction of ' + symbol_data[1] + ' Price - LSTM Model #1')
plt.legend()
plt.show()

# Calculate RMSE and MAPE for LSTM Model #1
rmse = mean_squared_error(y_pred, y_test, squared=False)
mape = mean_absolute_percentage_error(y_pred, y_test)
print("RSME: ", rmse)
print("MAPE: ", mape)

rmse_mape_results.append(['LSTM #1', rmse, mape])

```

```
# Create and train another LSTM model with different parameters
lstm = Sequential()
lstm.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2]), activation='relu', return_seque
lstm.add(LSTM(50, activation='relu'))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')
history = lstm.fit(X_train, y_train, epochs=100, batch_size=4, verbose=2, shuffle=False)

# Make predictions using the second LSTM model
y_pred = lstm.predict(X_test)

# Plot LSTM predictions vs. actual values for the second model
plt.figure(figsize=(10, 8))
plt.plot(y_date_index[1:len(y_date_index)], y_pred, label='Forecast', color='blue')
plt.plot(y_date_index[1:len(y_date_index)], y_test, label='Actual', color='green')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Prediction of ' + symbol_data[1] + ' Price - LSTM Model #2')
plt.legend()
plt.show()

# Calculate RMSE and MAPE for LSTM Model #2
rmse = mean_squared_error(y_pred, y_test, squared=False)
mape = mean_absolute_percentage_error(y_pred, y_test)
print("RSME: ", rmse)
print("MAPE: ", mape)

rmse_mape_results.append(['LSTM #2', rmse, mape])

# Create and train a third LSTM model with different parameters
X1, y1 = lstm_split(X_feat.values, n_steps=10)
train_split = 0.8
split_idx = int(np.ceil(len(X1) * train_split))
date_index = X_feat.index
X_train, X_test = X1[:split_idx], X1[split_idx:]
y_train, y_test = y1[:split_idx], y1[split_idx:]
X_train_date, X_test_date = date_index[:split_idx], date_index[split_idx:]

lstm = Sequential()
lstm.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2]), activation='relu', return_seque
lstm.add(LSTM(50, activation='relu'))
lstm.add(Dense(1))
lstm.compile(loss='mean_squared_error', optimizer='adam')
```

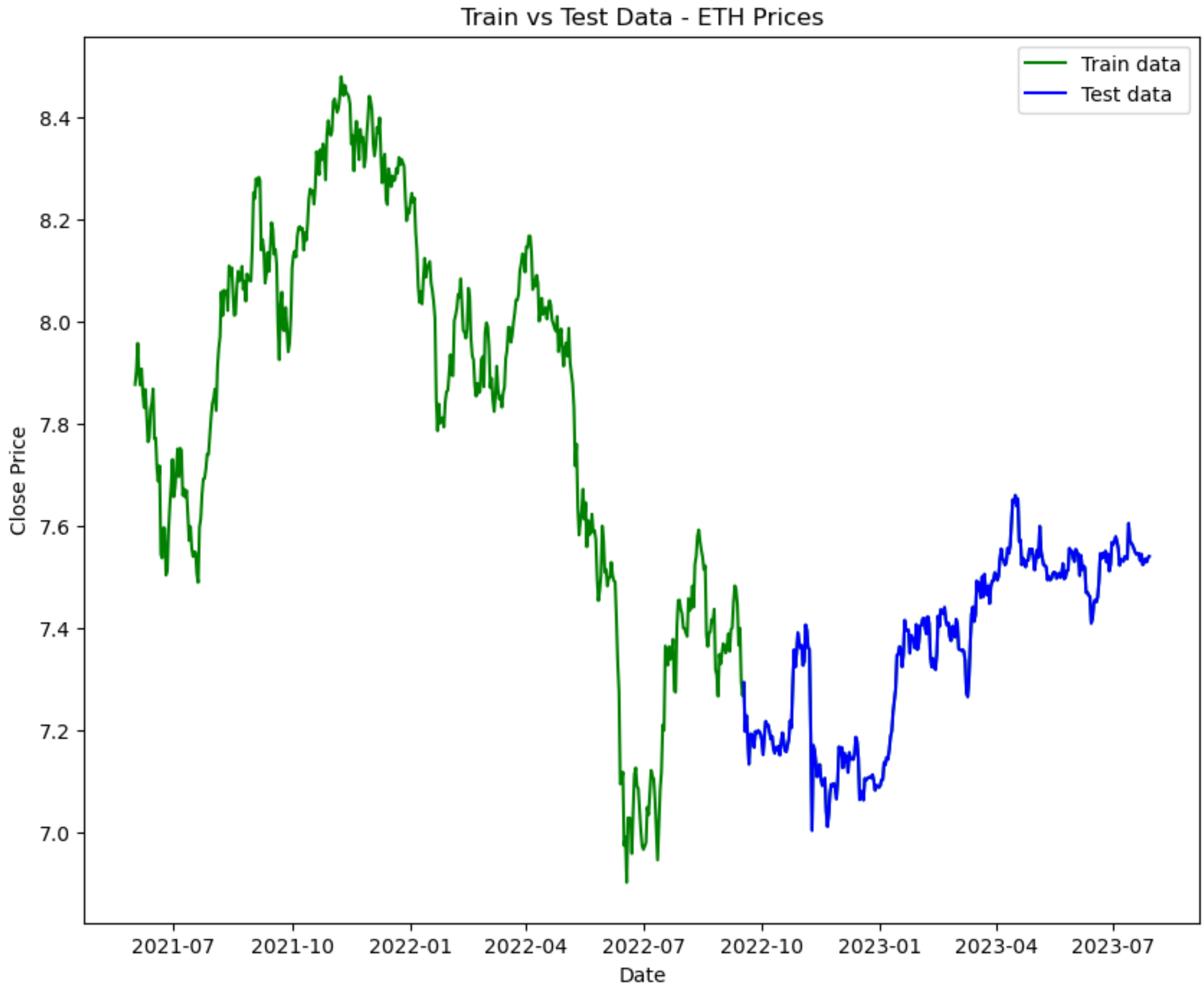
```
history = lstm.fit(X_train, y_train, epochs=100, batch_size=4, verbose=2, shuffle=False)

# Make predictions using the third LSTM model
y_pred = lstm.predict(X_test)

# Plot LSTM predictions vs. actual values for the third model
plt.figure(figsize=(10, 8))
plt.plot(y_date_index[2:len(y_date_index)], y_pred, label='Forecast', color='blue')
plt.plot(y_date_index[2:len(y_date_index)], y_test, label='Actual', color='green')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Prediction of ' + symbol_data[1] + ' Price - LSTM Model #3')
plt.legend()
plt.show()

# Calculate RMSE and MAPE for LSTM Model #3
rmse = mean_squared_error(y_test, y_pred, squared=False)
mape = mean_absolute_percentage_error(y_test, y_pred)
print("RSME: ", rmse)
print("MAPE: ", mape)

rmse_mape_results.append(['LSTM #3', rmse, mape])

# Simple Moving Average
train_split = 0.8
split_idx = int(np.ceil(len(X_feat) * train_split))
train = X_feat[['Close']].iloc[:split_idx]
test = X_feat[['Close']].iloc[split_idx:]

# Calculate the Simple Moving Average
test_pred = np.array([train.rolling(10).mean().iloc[-1]] * len(test)).reshape((-1, 1))

# Plot Simple Moving Average vs. actual values
plt.figure(figsize=(10, 8))
plt.plot(test.index, test)
plt.plot(test.index, test_pred)
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.title('Simple Moving Average - ETH')
plt.show()

print('RMSE: %.3f' % mean_squared_error(test, test_pred, squared=False))
print('MAPE: %.3f' % mean_absolute_percentage_error(test, test_pred))
```

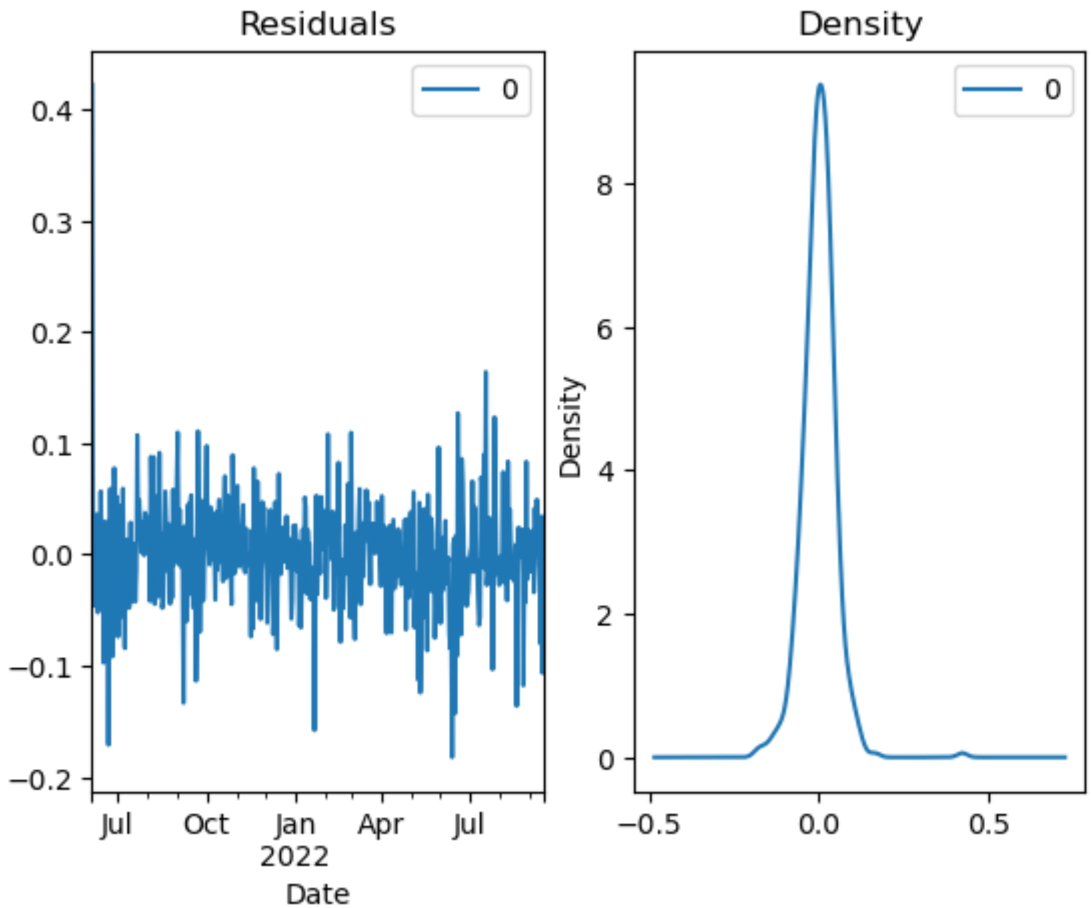
```
rmse = mean_squared_error(test, test_pred, squared=False)
mape = mean_absolute_percentage_error(test, test_pred)

rmse_mape_results.append(['SMA', rmse, mape])

# Exponential Moving Average
X = X_feat[['Close']].values
train_split = 0.8
split_idx = int(np.ceil(len(X) * train_split))
train = X[:split_idx]
test = X[split_idx:]
test_concat = np.array([]).reshape((0, 1))

# Calculate Exponential Moving Average
for i in range(len(test)):
    train_fit = np.concatenate((train, np.asarray(test_concat)))
    fit = SimpleExpSmoothing(np.asarray(train_fit)).fit(smoothing_level=0.2)
    test_pred = fit.forecast(1)
    test_concat = np.concatenate((np.asarray(test_concat), test_pred.reshape((-1, 1))))

# Plot Exponential Moving Average vs. actual values
plt.figure(figsize=(10, 8))
plt.plot(test)
plt.plot(test_concat)
plt.xlabel('Epoch')
plt.ylabel('Close Price')
plt.title('Exponential Moving Average - ETH')
plt.show()

print('RMSE: %.3f' % mean_squared_error(test, test_concat, squared=False))
print('MAPE: %.3f' % mean_absolute_percentage_error(test, test_concat))

rmse = mean_squared_error(test, test_concat, squared=False)
mape = mean_absolute_percentage_error(test, test_concat)

rmse_mape_results.append(['EMA', rmse, mape])

#Create dataframe with various model RMSE and MAPE values
df = pd.DataFrame(rmse_mape_results, columns = ['Model', 'RMSE', 'MAPE'])
df.sort_values(by='RMSE', ascending=False, inplace=True)
print('\n')
print(df.to_string(index=False))
```

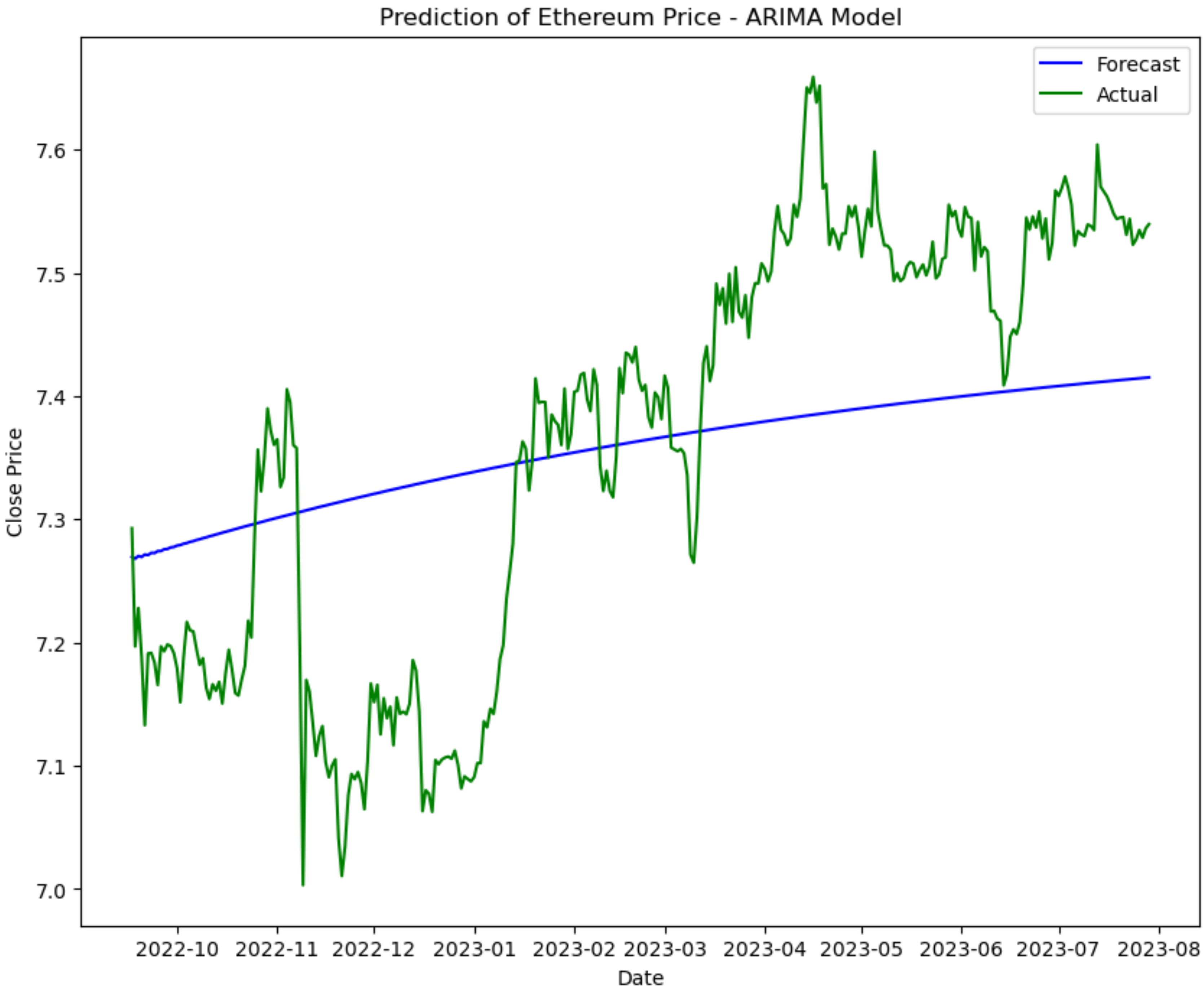
```
[*********************100%***********************]  1 of 1 completed
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 789 entries, 2021-06-01 to 2023-07-29
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       789 non-null    float64
 1   High       789 non-null    float64
 2   Low        789 non-null    float64
 3   Close      789 non-null    float64
 4   Adj Close  789 non-null    float64
 5   Volume     789 non-null    int64  
dtypes: float64(5), int64(1)
memory usage: 43.1 KB
```


Ethereum Price in the Last 2 Years
Close Price
Date
5000
4500
4000
3500
3000
2500
2000
1500
1000
2021-07
2021-10
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  470
Model:                 ARIMA(3, 0, 2)   Log Likelihood                 779.410
Date:                Sun, 20 Aug 2023   AIC                          -1544.820
Time:                        17:05:43   BIC                          -1515.751
Sample:                    06-04-2021   HQIC                         -1533.384
                         - 09-16-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.4743      0.464     16.110      0.000       6.565       8.384
ar.L1          0.7861      3.784      0.208      0.835      -6.630       8.202
ar.L2          0.8111      0.762      1.064      0.287      -0.683       2.305
ar.L3         -0.5996      3.208     -0.187      0.852      -6.887       5.688
ma.L1          0.2305      3.807      0.061      0.952      -7.231       7.692
ma.L2         -0.6079      3.310     -0.184      0.854      -7.095       5.879
sigma2         0.0021      0.000     18.598      0.000       0.002       0.002
===================================================================================
Ljung-Box (L1) (Q):                   0.11   Jarque-Bera (JB):                49.59
Prob(Q):                              0.74   Prob(JB):                         0.00
Heteroskedasticity (H):               1.23   Skew:                            -0.36
Prob(H) (two-sided):                  0.20   Kurtosis:                         4.42
===================================================================================

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


Train vs Test Data - ETH Prices
Train data
Test data
Close Price
8.4
8.2
8.0
7.8
7.6
7.4
7.2
7.0
2021-07
2021-10
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
Date


Residuals
0
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
Jul
Oct
Jan
2022
Apr
Jul
Date
Density
0
8
6
4
2
0
Density
−0.5
0.0
0.5


Prediction of Ethereum Price - ARIMA Model
Forecast
Actual
Close Price
7.0
7.1
7.2
7.3
7.4
7.5
7.6
2022-10
2022-11
2022-12
2023-01
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
Date

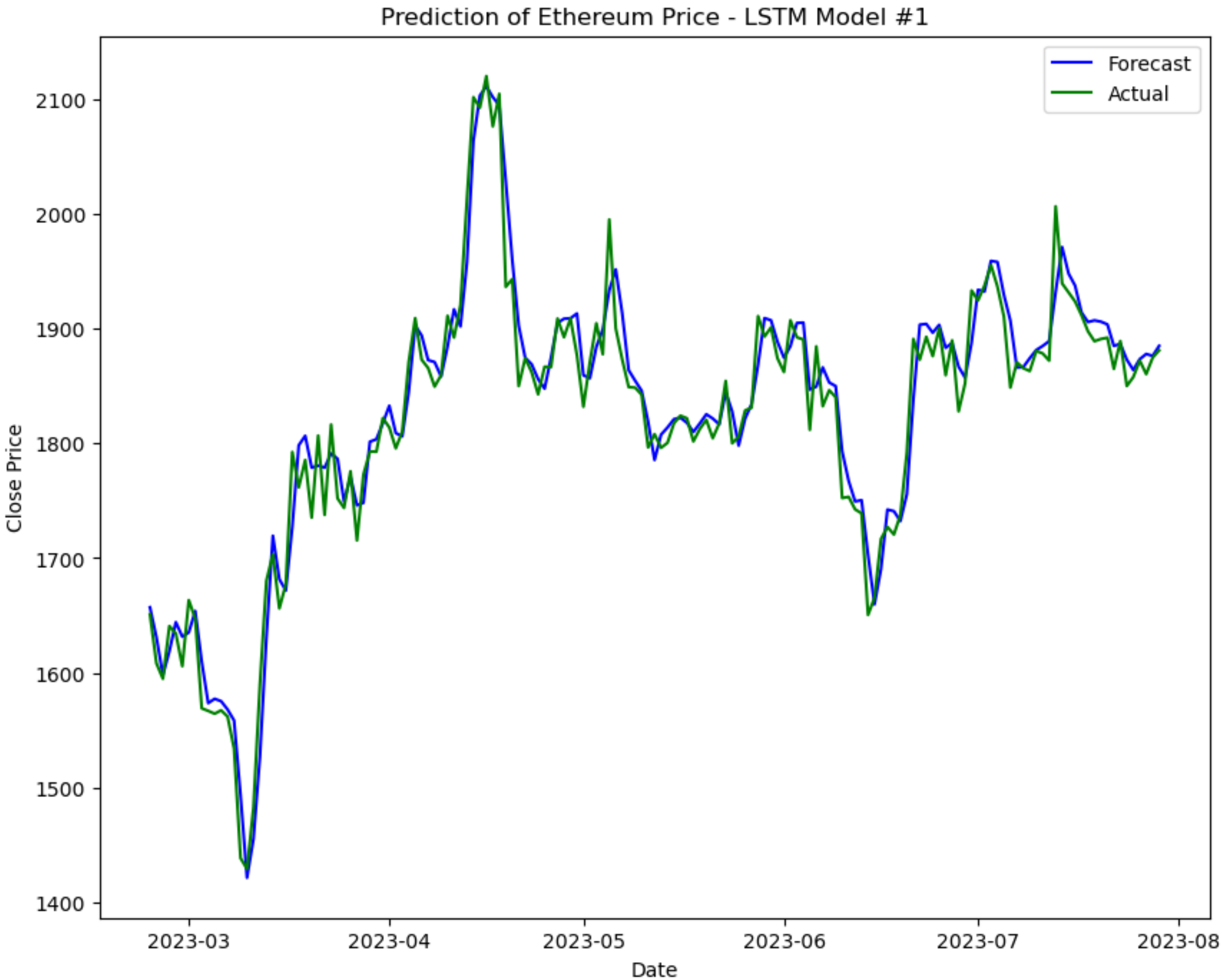
```
RMSE:  214.46334411707326
MAPE:  0.1221243596224059
Epoch 1/100
158/158 - 3s - loss: 3520655.2500
Epoch 2/100
158/158 - 0s - loss: 85558.3672
Epoch 3/100
158/158 - 0s - loss: 11080.6279
Epoch 4/100
158/158 - 0s - loss: 11085.5654
Epoch 5/100
158/158 - 0s - loss: 11097.4561
Epoch 6/100
158/158 - 0s - loss: 11106.8799
Epoch 7/100
158/158 - 0s - loss: 11113.9219
Epoch 8/100
158/158 - 0s - loss: 11119.4043
Epoch 9/100
158/158 - 0s - loss: 11124.2080
Epoch 10/100
158/158 - 0s - loss: 11128.9414
Epoch 11/100
158/158 - 0s - loss: 11133.8555
Epoch 12/100
158/158 - 1s - loss: 11138.9531
Epoch 13/100
158/158 - 0s - loss: 11144.0391
Epoch 14/100
158/158 - 0s - loss: 11148.9297
Epoch 15/100
158/158 - 0s - loss: 11153.3428
Epoch 16/100
158/158 - 0s - loss: 11157.1074
Epoch 17/100
158/158 - 0s - loss: 11160.0186
Epoch 18/100
158/158 - 0s - loss: 11161.9355
Epoch 19/100
158/158 - 0s - loss: 11162.6924
Epoch 20/100
158/158 - 0s - loss: 11162.1748
Epoch 21/100
```

```
158/158 - 0s - loss: 11160.2578
Epoch 22/100
158/158 - 0s - loss: 11156.9268
Epoch 23/100
158/158 - 0s - loss: 11152.2246
Epoch 24/100
158/158 - 0s - loss: 11146.2275
Epoch 25/100
158/158 - 0s - loss: 11139.0273
Epoch 26/100
158/158 - 0s - loss: 11130.6748
Epoch 27/100
158/158 - 0s - loss: 11121.0586
Epoch 28/100
158/158 - 0s - loss: 11109.9980
Epoch 29/100
158/158 - 0s - loss: 11097.1064
Epoch 30/100
158/158 - 0s - loss: 11081.9219
Epoch 31/100
158/158 - 0s - loss: 11063.9189
Epoch 32/100
158/158 - 0s - loss: 11042.5293
Epoch 33/100
158/158 - 0s - loss: 11017.2676
Epoch 34/100
158/158 - 0s - loss: 10987.8086
Epoch 35/100
158/158 - 0s - loss: 10953.4580
Epoch 36/100
158/158 - 0s - loss: 10914.1396
Epoch 37/100
158/158 - 0s - loss: 10869.6816
Epoch 38/100
158/158 - 0s - loss: 10819.9980
Epoch 39/100
158/158 - 0s - loss: 10765.0918
Epoch 40/100
158/158 - 0s - loss: 10705.0488
Epoch 41/100
158/158 - 0s - loss: 10640.0264
Epoch 42/100
158/158 - 1s - loss: 10570.2227
```

```
Epoch 43/100
158/158 - 0s - loss: 10495.9180
Epoch 44/100
158/158 - 1s - loss: 10417.4180
Epoch 45/100
158/158 - 0s - loss: 10335.0732
Epoch 46/100
158/158 - 0s - loss: 10249.2461
Epoch 47/100
158/158 - 0s - loss: 10160.3574
Epoch 48/100
158/158 - 0s - loss: 10068.6533
Epoch 49/100
158/158 - 0s - loss: 9980.0117
Epoch 50/100
158/158 - 0s - loss: 9889.0430
Epoch 51/100
158/158 - 0s - loss: 9795.8750
Epoch 52/100
158/158 - 0s - loss: 9703.2646
Epoch 53/100
158/158 - 0s - loss: 9606.9268
Epoch 54/100
158/158 - 0s - loss: 9509.2354
Epoch 55/100
158/158 - 0s - loss: 9412.5479
Epoch 56/100
158/158 - 0s - loss: 9310.5693
Epoch 57/100
158/158 - 0s - loss: 9213.6826
Epoch 58/100
158/158 - 0s - loss: 9113.7461
Epoch 59/100
158/158 - 0s - loss: 9014.3213
Epoch 60/100
158/158 - 0s - loss: 8916.1582
Epoch 61/100
158/158 - 0s - loss: 8819.1719
Epoch 62/100
158/158 - 0s - loss: 8718.7334
Epoch 63/100
158/158 - 0s - loss: 8617.5488
Epoch 64/100
```

```
158/158 - 0s - loss: 8534.3145
Epoch 65/100
158/158 - 0s - loss: 8414.8467
Epoch 66/100
158/158 - 0s - loss: 8317.2734
Epoch 67/100
158/158 - 0s - loss: 8219.6152
Epoch 68/100
158/158 - 0s - loss: 8123.0137
Epoch 69/100
158/158 - 1s - loss: 8013.2490
Epoch 70/100
158/158 - 0s - loss: 7905.6729
Epoch 71/100
158/158 - 0s - loss: 7798.9399
Epoch 72/100
158/158 - 0s - loss: 7741.1909
Epoch 73/100
158/158 - 0s - loss: 7718.7622
Epoch 74/100
158/158 - 0s - loss: 7547.1362
Epoch 75/100
158/158 - 0s - loss: 7408.3955
Epoch 76/100
158/158 - 0s - loss: 7284.4043
Epoch 77/100
158/158 - 0s - loss: 7177.1602
Epoch 78/100
158/158 - 0s - loss: 7074.4463
Epoch 79/100
158/158 - 0s - loss: 6974.7314
Epoch 80/100
158/158 - 0s - loss: 6877.0918
Epoch 81/100
158/158 - 0s - loss: 6780.1270
Epoch 82/100
158/158 - 1s - loss: 6680.9604
Epoch 83/100
158/158 - 0s - loss: 6592.0576
Epoch 84/100
158/158 - 0s - loss: 6489.5581
Epoch 85/100
158/158 - 0s - loss: 6391.7798
```

```
Epoch 86/100
158/158 - 1s - loss: 6310.5200
Epoch 87/100
158/158 - 0s - loss: 6218.2734
Epoch 88/100
158/158 - 0s - loss: 6109.2441
Epoch 89/100
158/158 - 0s - loss: 6008.6523
Epoch 90/100
158/158 - 0s - loss: 5917.4780
Epoch 91/100
158/158 - 0s - loss: 5824.5977
Epoch 92/100
158/158 - 0s - loss: 5710.4258
Epoch 93/100
158/158 - 0s - loss: 5619.0464
Epoch 94/100
158/158 - 0s - loss: 5528.8667
Epoch 95/100
158/158 - 0s - loss: 5441.2798
Epoch 96/100
158/158 - 0s - loss: 5353.3413
Epoch 97/100
158/158 - 0s - loss: 5268.4297
Epoch 98/100
158/158 - 0s - loss: 5182.5537
Epoch 99/100
158/158 - 1s - loss: 5100.0439
Epoch 100/100
158/158 - 0s - loss: 5016.7827
```


Prediction of Ethereum Price - LSTM Model #1
Forecast
Actual
Close Price
2100
2000
1900
1800
1700
1600
1500
1400
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
Date

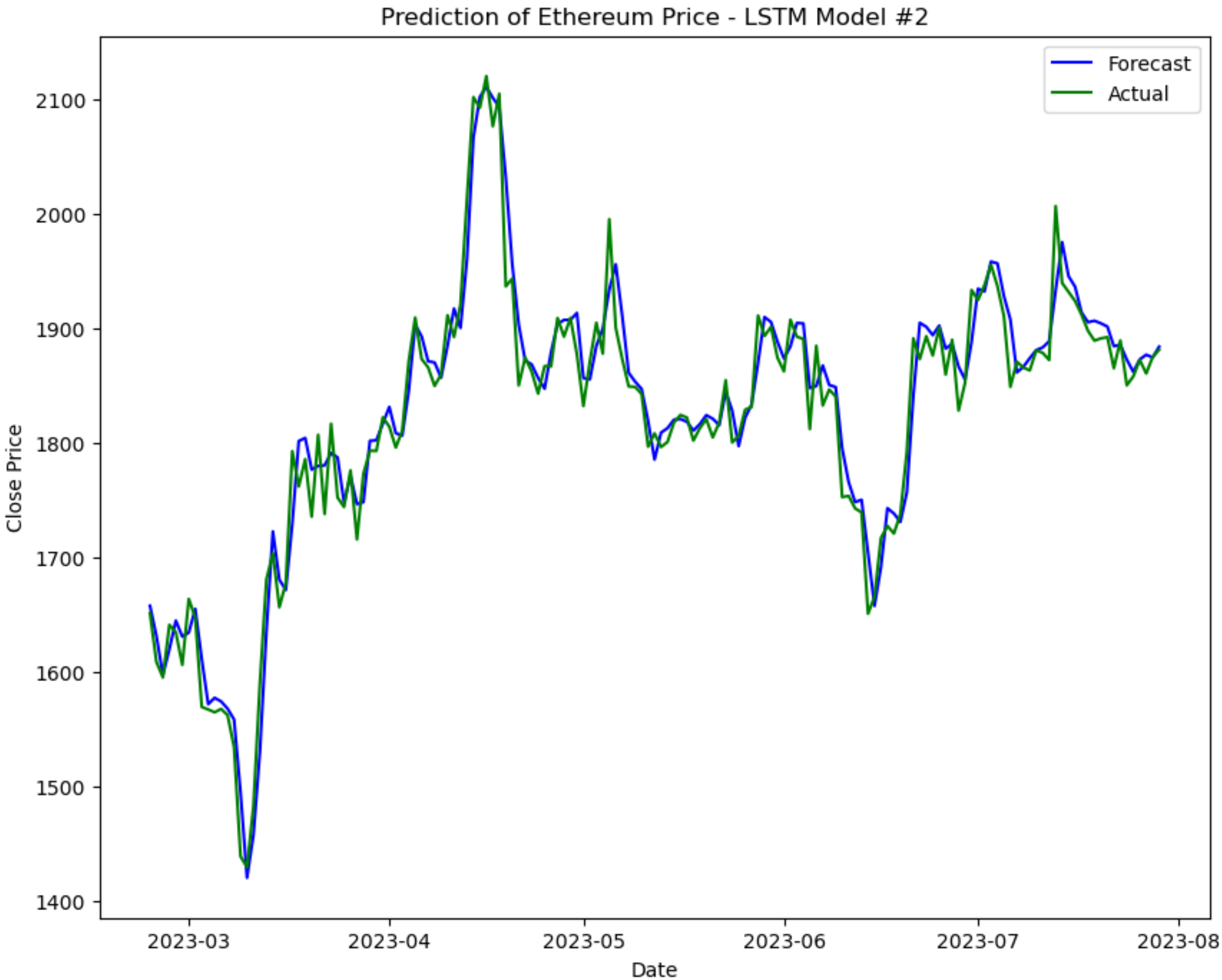
```
RSME:  25.649953744067773
MAPE:  0.0106846644100832
Epoch 1/100
158/158 - 3s - loss: 4885883.0000
Epoch 2/100
158/158 - 0s - loss: 19880.2812
Epoch 3/100
158/158 - 0s - loss: 12526.6045
Epoch 4/100
158/158 - 0s - loss: 12583.5840
Epoch 5/100
158/158 - 0s - loss: 12643.7500
Epoch 6/100
158/158 - 0s - loss: 12699.5693
Epoch 7/100
158/158 - 0s - loss: 12750.4941
Epoch 8/100
158/158 - 0s - loss: 12797.5156
Epoch 9/100
158/158 - 0s - loss: 12842.4492
Epoch 10/100
158/158 - 0s - loss: 12887.6084
Epoch 11/100
158/158 - 0s - loss: 12934.5020
Epoch 12/100
158/158 - 0s - loss: 12982.6602
Epoch 13/100
158/158 - 0s - loss: 13029.3887
Epoch 14/100
158/158 - 0s - loss: 13070.6826
Epoch 15/100
158/158 - 0s - loss: 13102.4092
Epoch 16/100
158/158 - 0s - loss: 13121.3672
Epoch 17/100
158/158 - 0s - loss: 13125.6816
Epoch 18/100
158/158 - 0s - loss: 13113.9004
Epoch 19/100
158/158 - 0s - loss: 13094.3369
Epoch 20/100
158/158 - 0s - loss: 13055.5654
Epoch 21/100
```

```
158/158 - 0s - loss: 13005.3486
Epoch 22/100
158/158 - 0s - loss: 12944.5615
Epoch 23/100
158/158 - 0s - loss: 12871.1006
Epoch 24/100
158/158 - 0s - loss: 12789.8584
Epoch 25/100
158/158 - 0s - loss: 12705.4336
Epoch 26/100
158/158 - 0s - loss: 12626.7969
Epoch 27/100
158/158 - 0s - loss: 12525.3008
Epoch 28/100
158/158 - 0s - loss: 12420.5713
Epoch 29/100
158/158 - 0s - loss: 12312.6230
Epoch 30/100
158/158 - 0s - loss: 12202.0684
Epoch 31/100
158/158 - 0s - loss: 12089.4355
Epoch 32/100
158/158 - 0s - loss: 11975.2500
Epoch 33/100
158/158 - 0s - loss: 11860.2441
Epoch 34/100
158/158 - 0s - loss: 11745.1289
Epoch 35/100
158/158 - 1s - loss: 11623.8379
Epoch 36/100
158/158 - 0s - loss: 11486.5098
Epoch 37/100
158/158 - 0s - loss: 11378.5205
Epoch 38/100
158/158 - 0s - loss: 11261.5264
Epoch 39/100
158/158 - 0s - loss: 11153.8604
Epoch 40/100
158/158 - 1s - loss: 11047.0947
Epoch 41/100
158/158 - 0s - loss: 10940.5752
Epoch 42/100
158/158 - 0s - loss: 10808.7822
```

```
Epoch 43/100
158/158 - 0s - loss: 10651.1807
Epoch 44/100
158/158 - 0s - loss: 10549.2285
Epoch 45/100
158/158 - 0s - loss: 10446.0645
Epoch 46/100
158/158 - 1s - loss: 10365.8896
Epoch 47/100
158/158 - 0s - loss: 10807.0957
Epoch 48/100
158/158 - 0s - loss: 9914.2539
Epoch 49/100
158/158 - 0s - loss: 9669.5479
Epoch 50/100
158/158 - 0s - loss: 9559.6191
Epoch 51/100
158/158 - 0s - loss: 9454.1416
Epoch 52/100
158/158 - 0s - loss: 9338.3252
Epoch 53/100
158/158 - 0s - loss: 9239.2295
Epoch 54/100
158/158 - 0s - loss: 9119.4941
Epoch 55/100
158/158 - 0s - loss: 9042.6865
Epoch 56/100
158/158 - 0s - loss: 8973.7188
Epoch 57/100
158/158 - 0s - loss: 8892.1826
Epoch 58/100
158/158 - 0s - loss: 8845.2510
Epoch 59/100
158/158 - 0s - loss: 8660.0723
Epoch 60/100
158/158 - 0s - loss: 8529.8926
Epoch 61/100
158/158 - 0s - loss: 8402.6729
Epoch 62/100
158/158 - 0s - loss: 8303.5986
Epoch 63/100
158/158 - 0s - loss: 8198.8906
Epoch 64/100
```

```
158/158 - 0s - loss: 8097.7544
Epoch 65/100
158/158 - 1s - loss: 8007.2402
Epoch 66/100
158/158 - 1s - loss: 7906.7466
Epoch 67/100
158/158 - 0s - loss: 7797.0903
Epoch 68/100
158/158 - 1s - loss: 7700.3613
Epoch 69/100
158/158 - 1s - loss: 7611.6831
Epoch 70/100
158/158 - 0s - loss: 7504.0454
Epoch 71/100
158/158 - 0s - loss: 7415.2266
Epoch 72/100
158/158 - 0s - loss: 7326.7739
Epoch 73/100
158/158 - 0s - loss: 7219.1816
Epoch 74/100
158/158 - 0s - loss: 7147.1416
Epoch 75/100
158/158 - 0s - loss: 7041.3940
Epoch 76/100
158/158 - 0s - loss: 6966.9531
Epoch 77/100
158/158 - 0s - loss: 6866.8687
Epoch 78/100
158/158 - 0s - loss: 6775.5244
Epoch 79/100
158/158 - 0s - loss: 6699.2427
Epoch 80/100
158/158 - 0s - loss: 6613.0347
Epoch 81/100
158/158 - 0s - loss: 6551.2090
Epoch 82/100
158/158 - 0s - loss: 6442.3442
Epoch 83/100
158/158 - 1s - loss: 6364.8711
Epoch 84/100
158/158 - 0s - loss: 6282.0752
Epoch 85/100
158/158 - 0s - loss: 6203.8599
```

```
Epoch 86/100
158/158 - 0s - loss: 6104.8560
Epoch 87/100
158/158 - 0s - loss: 6031.2407
Epoch 88/100
158/158 - 1s - loss: 5946.9131
Epoch 89/100
158/158 - 0s - loss: 5894.9390
Epoch 90/100
158/158 - 1s - loss: 5809.9272
Epoch 91/100
158/158 - 1s - loss: 5704.2529
Epoch 92/100
158/158 - 0s - loss: 5626.6675
Epoch 93/100
158/158 - 0s - loss: 5568.4541
Epoch 94/100
158/158 - 0s - loss: 5470.4692
Epoch 95/100
158/158 - 0s - loss: 5411.0913
Epoch 96/100
158/158 - 1s - loss: 5327.7354
Epoch 97/100
158/158 - 0s - loss: 5238.6265
Epoch 98/100
158/158 - 0s - loss: 5157.5435
Epoch 99/100
158/158 - 0s - loss: 5084.8750
Epoch 100/100
158/158 - 0s - loss: 5009.1875
```


Prediction of Ethereum Price - LSTM Model #2
Forecast
Actual
2100
2000
1900
1800
1700
1600
1500
1400
Close Price
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
Date

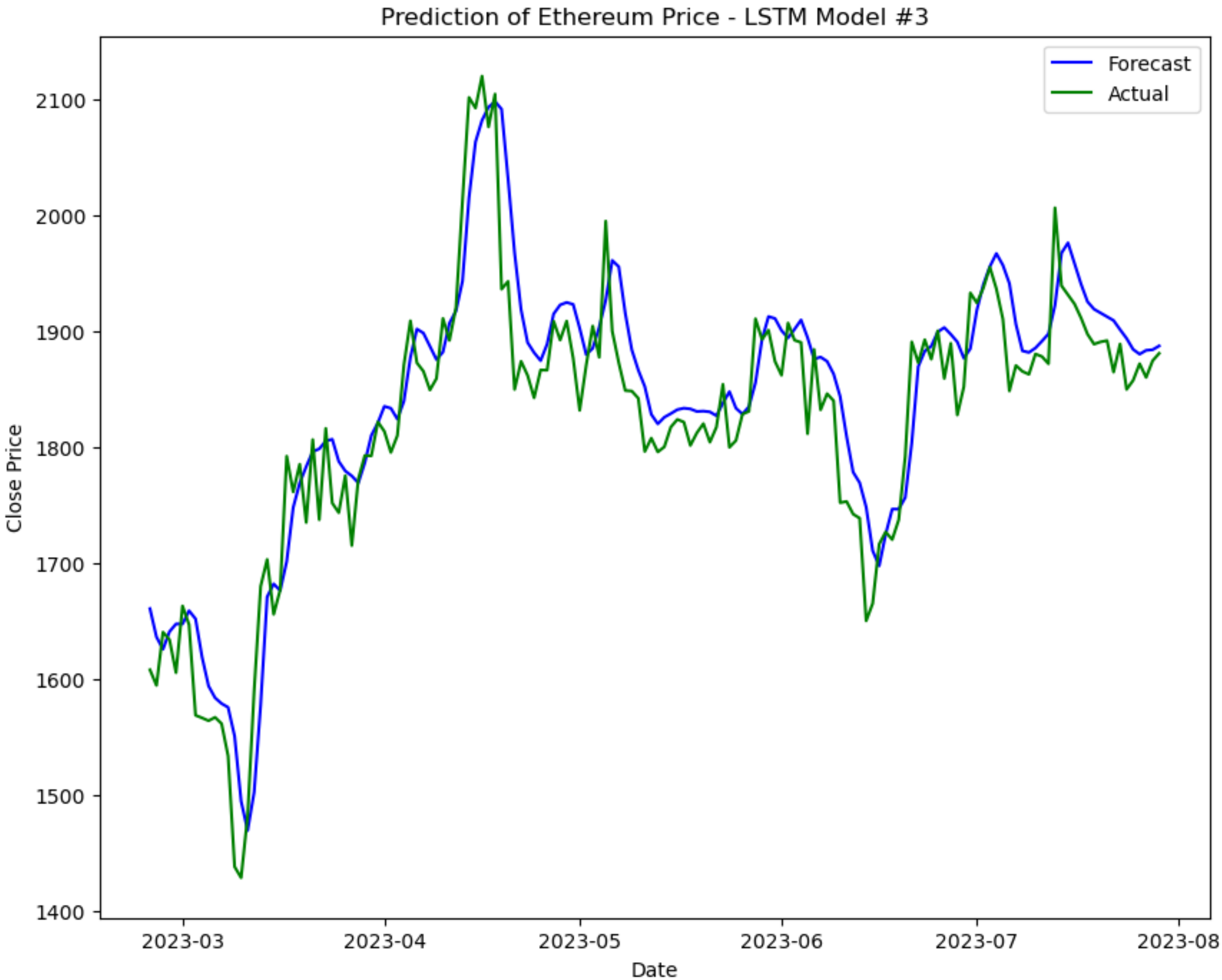
```
RSME:  25.45402250441576
MAPE:  0.01050813366010295
Epoch 1/100
156/156 - 3s - loss: 993242.4375
Epoch 2/100
156/156 - 1s - loss: 95655.6094
Epoch 3/100
156/156 - 1s - loss: 91966.5859
Epoch 4/100
156/156 - 1s - loss: 76655.4922
Epoch 5/100
156/156 - 1s - loss: 72186.7500
Epoch 6/100
156/156 - 1s - loss: 62809.3516
Epoch 7/100
156/156 - 1s - loss: 84122.5469
Epoch 8/100
156/156 - 1s - loss: 74256.9141
Epoch 9/100
156/156 - 1s - loss: 68096.8516
Epoch 10/100
156/156 - 1s - loss: 63784.3789
Epoch 11/100
156/156 - 1s - loss: 64502.8320
Epoch 12/100
156/156 - 1s - loss: 54430.6172
Epoch 13/100
156/156 - 1s - loss: 55158.8984
Epoch 14/100
156/156 - 1s - loss: 53446.7227
Epoch 15/100
156/156 - 1s - loss: 50834.7578
Epoch 16/100
156/156 - 1s - loss: 146411.5781
Epoch 17/100
156/156 - 1s - loss: 66100.1250
Epoch 18/100
156/156 - 1s - loss: 68441.3359
Epoch 19/100
156/156 - 1s - loss: 66080.1953
Epoch 20/100
156/156 - 1s - loss: 59813.8203
Epoch 21/100
```

```
156/156 - 1s - loss: 64812.4375
Epoch 22/100
156/156 - 1s - loss: 59041.5508
Epoch 23/100
156/156 - 1s - loss: 58150.1719
Epoch 24/100
156/156 - 1s - loss: 63898.1406
Epoch 25/100
156/156 - 1s - loss: 62557.2930
Epoch 26/100
156/156 - 1s - loss: 63513.7383
Epoch 27/100
156/156 - 1s - loss: 63577.0430
Epoch 28/100
156/156 - 1s - loss: 65465.2695
Epoch 29/100
156/156 - 1s - loss: 65971.7109
Epoch 30/100
156/156 - 1s - loss: 65384.6094
Epoch 31/100
156/156 - 1s - loss: 72738.7812
Epoch 32/100
156/156 - 1s - loss: 88447.8672
Epoch 33/100
156/156 - 1s - loss: 84851.5625
Epoch 34/100
156/156 - 1s - loss: 83687.6641
Epoch 35/100
156/156 - 1s - loss: 82688.4844
Epoch 36/100
156/156 - 1s - loss: 81864.4219
Epoch 37/100
156/156 - 1s - loss: 80943.2031
Epoch 38/100
156/156 - 1s - loss: 86002.4922
Epoch 39/100
156/156 - 1s - loss: 96863.8438
Epoch 40/100
156/156 - 1s - loss: 91257.4219
Epoch 41/100
156/156 - 1s - loss: 84444.5234
Epoch 42/100
156/156 - 2s - loss: 85008.6094
```

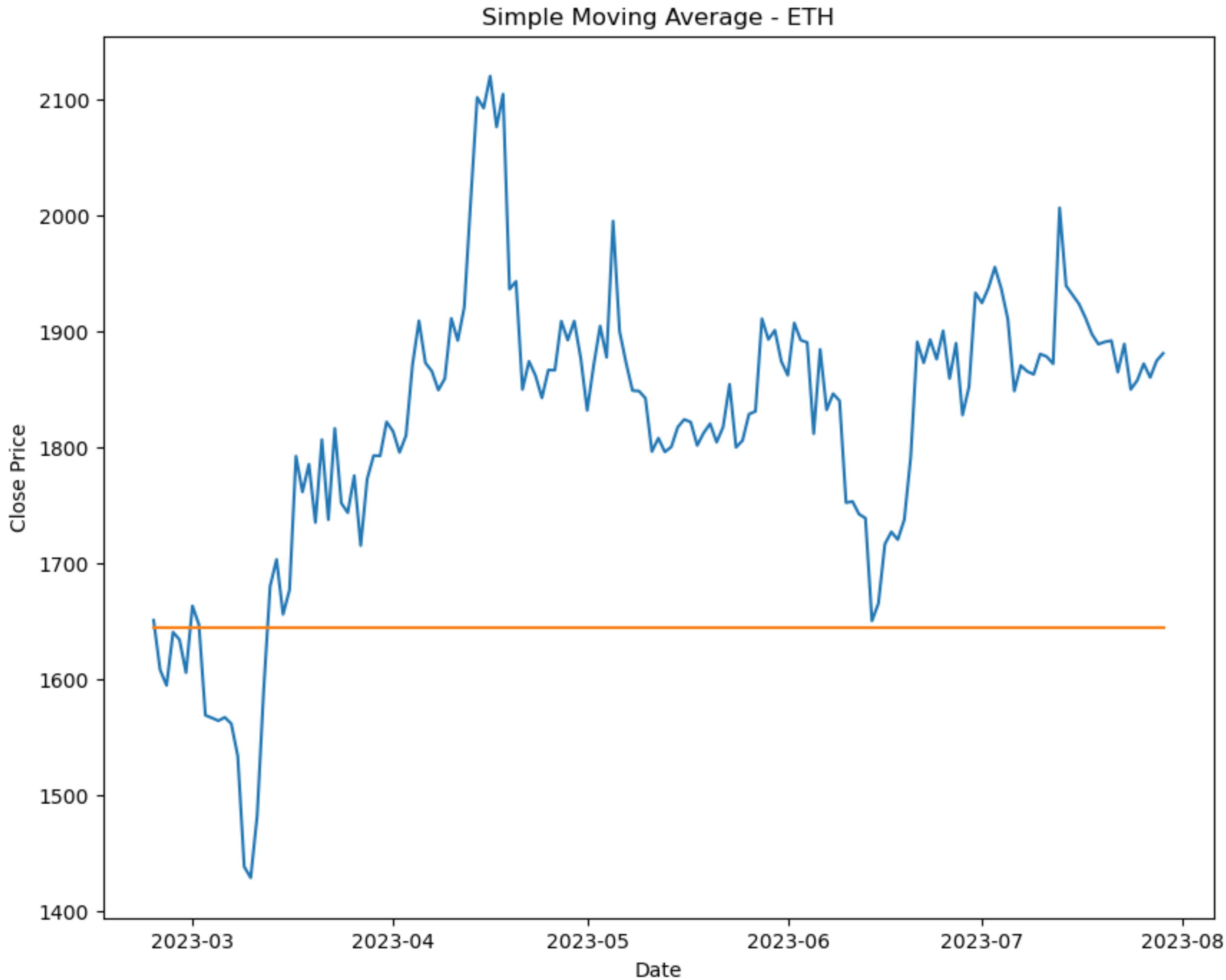
```
Epoch 43/100
156/156 - 1s - loss: 78094.4141
Epoch 44/100
156/156 - 1s - loss: 76030.0859
Epoch 45/100
156/156 - 1s - loss: 73651.9922
Epoch 46/100
156/156 - 1s - loss: 77510.1641
Epoch 47/100
156/156 - 1s - loss: 76022.5391
Epoch 48/100
156/156 - 1s - loss: 77069.3359
Epoch 49/100
156/156 - 1s - loss: 88953.9531
Epoch 50/100
156/156 - 1s - loss: 78170.1797
Epoch 51/100
156/156 - 1s - loss: 73419.7031
Epoch 52/100
156/156 - 1s - loss: 76308.0547
Epoch 53/100
156/156 - 1s - loss: 73882.2734
Epoch 54/100
156/156 - 1s - loss: 73699.8359
Epoch 55/100
156/156 - 1s - loss: 72491.1875
Epoch 56/100
156/156 - 1s - loss: 69760.7734
Epoch 57/100
156/156 - 1s - loss: 67462.0859
Epoch 58/100
156/156 - 1s - loss: 67629.6641
Epoch 59/100
156/156 - 1s - loss: 64420.6406
Epoch 60/100
156/156 - 1s - loss: 66313.8906
Epoch 61/100
156/156 - 1s - loss: 65715.0391
Epoch 62/100
156/156 - 1s - loss: 63143.2305
Epoch 63/100
156/156 - 1s - loss: 64341.6680
Epoch 64/100
```

```
156/156 - 1s - loss: 67663.1641
Epoch 65/100
156/156 - 1s - loss: 65469.0898
Epoch 66/100
156/156 - 1s - loss: 63982.1602
Epoch 67/100
156/156 - 1s - loss: 62814.5625
Epoch 68/100
156/156 - 1s - loss: 61390.0820
Epoch 69/100
156/156 - 1s - loss: 60205.4922
Epoch 70/100
156/156 - 1s - loss: 59083.9219
Epoch 71/100
156/156 - 1s - loss: 57531.9805
Epoch 72/100
156/156 - 1s - loss: 56151.9570
Epoch 73/100
156/156 - 1s - loss: 55284.8320
Epoch 74/100
156/156 - 1s - loss: 52256.5664
Epoch 75/100
156/156 - 1s - loss: 50352.7812
Epoch 76/100
156/156 - 1s - loss: 48601.6289
Epoch 77/100
156/156 - 1s - loss: 46698.5820
Epoch 78/100
156/156 - 1s - loss: 44996.9336
Epoch 79/100
156/156 - 1s - loss: 43063.1992
Epoch 80/100
156/156 - 1s - loss: 41246.5703
Epoch 81/100
156/156 - 1s - loss: 39065.2070
Epoch 82/100
156/156 - 1s - loss: 37090.7578
Epoch 83/100
156/156 - 1s - loss: 34701.2148
Epoch 84/100
156/156 - 1s - loss: 32187.2949
Epoch 85/100
156/156 - 1s - loss: 30192.2148
```

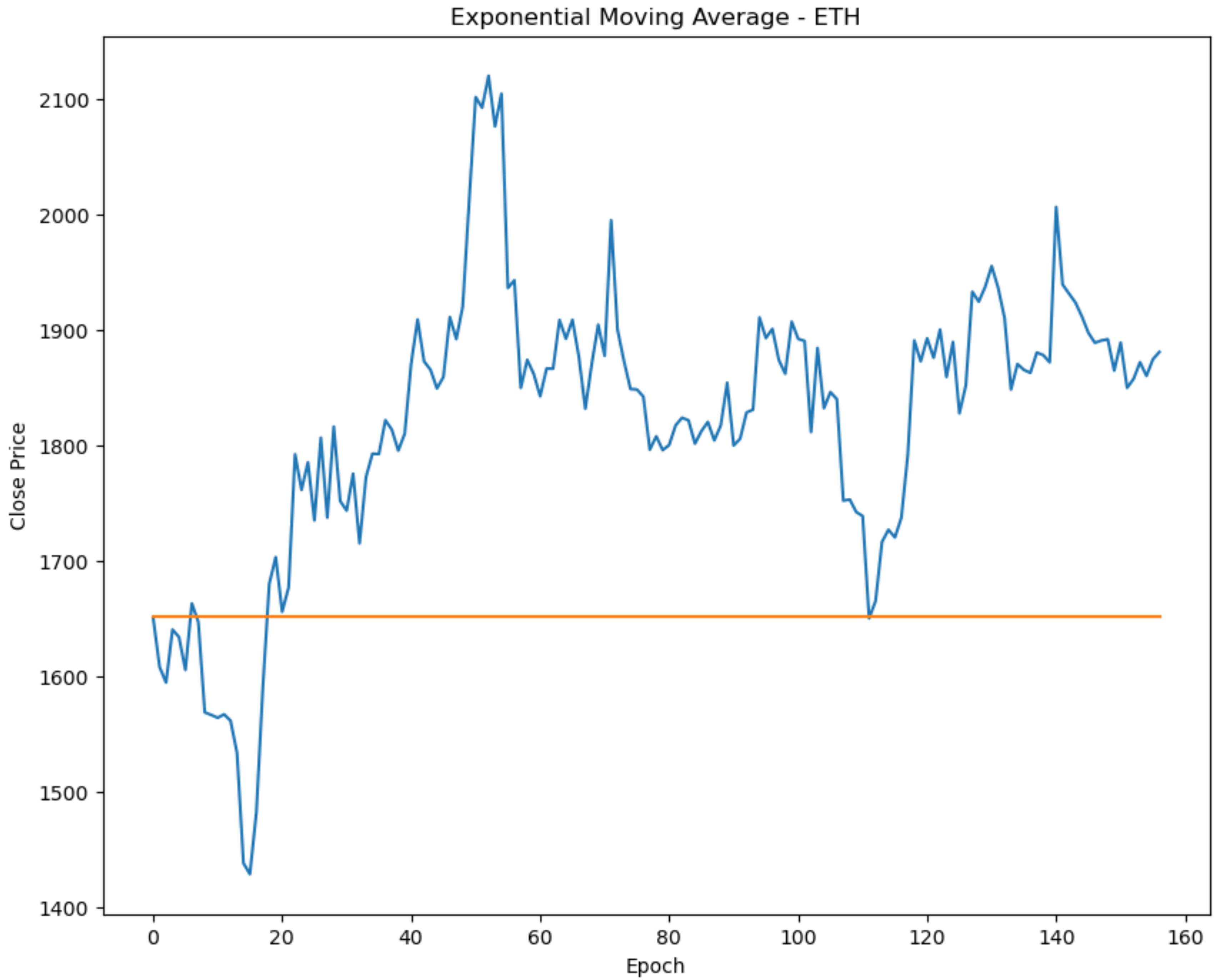
```
Epoch 86/100
156/156 - 1s - loss: 28130.7461
Epoch 87/100
156/156 - 1s - loss: 26264.4180
Epoch 88/100
156/156 - 1s - loss: 24561.7246
Epoch 89/100
156/156 - 1s - loss: 23204.9805
Epoch 90/100
156/156 - 1s - loss: 22096.7324
Epoch 91/100
156/156 - 1s - loss: 21096.3711
Epoch 92/100
156/156 - 1s - loss: 20334.2500
Epoch 93/100
156/156 - 1s - loss: 19753.7344
Epoch 94/100
156/156 - 1s - loss: 19192.0801
Epoch 95/100
156/156 - 1s - loss: 18869.4922
Epoch 96/100
156/156 - 1s - loss: 18109.7715
Epoch 97/100
156/156 - 1s - loss: 17843.4688
Epoch 98/100
156/156 - 1s - loss: 17648.2363
Epoch 99/100
156/156 - 1s - loss: 17053.6328
Epoch 100/100
156/156 - 1s - loss: 16745.6992
```


Prediction of Ethereum Price - LSTM Model #3
Forecast
Actual
Close Price
2100
2000
1900
1800
1700
1600
1500
1400
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
Date

```
RSME:  43.336154226630356
MAPE:  0.018489559901510955
```


Simple Moving Average - ETH
Close Price
Date
2100
2000
1900
1800
1700
1600
1500
1400
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08

```
RMSE: 213.570
MAPE: 0.103
```


Exponential Moving Average - ETH
Close Price
Epoch
2100
2000
1900
1800
1700
1600
1500
1400
0
20
40
60
80
100
120
140
160

```
RMSE: 208.221
MAPE: 0.101


  Model       RMSE     MAPE
  ARIMA 214.463344 0.122124
    SMA 213.570456 0.103361
    EMA 208.220523 0.100607
LSTM #3  43.336154 0.018490
LSTM #1  25.649954 0.010685
LSTM #2  25.454023 0.010508
```

In [ ]:

```
1
```