1: Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
re
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import math
```

2: Load and Explore Dataset

In [2]:
```
multimodal = pd.read_csv("B:/Dublin City University/Practicum/Proj/Dataset/
main/processed/multimodal_dataset_final2.csv")
multimodal.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	2016-07-01	17924.240234	18002.380859	17916.910156	17949.369141	82160000	17949.369141
1	2016-06-30	17712.759766	17930.609375	17711.800781	17929.990234	133030000	17929.990234
2	2016-06-29	17456.019531	17704.509766	17456.019531	17694.679688	106380000	17694.679688
3	2016-06-28	17190.509766	17409.720703	17190.509766	17409.720703	112190000	17409.720703
4	2016-06-27	17355.210938	17355.210938	17063.080078	17140.240234	138740000	17140.240234

In [3]:
```
missing_values = multimodal.isnull().sum()
print("\nMissing values per column:")
missing_values[missing_values > 0]
```

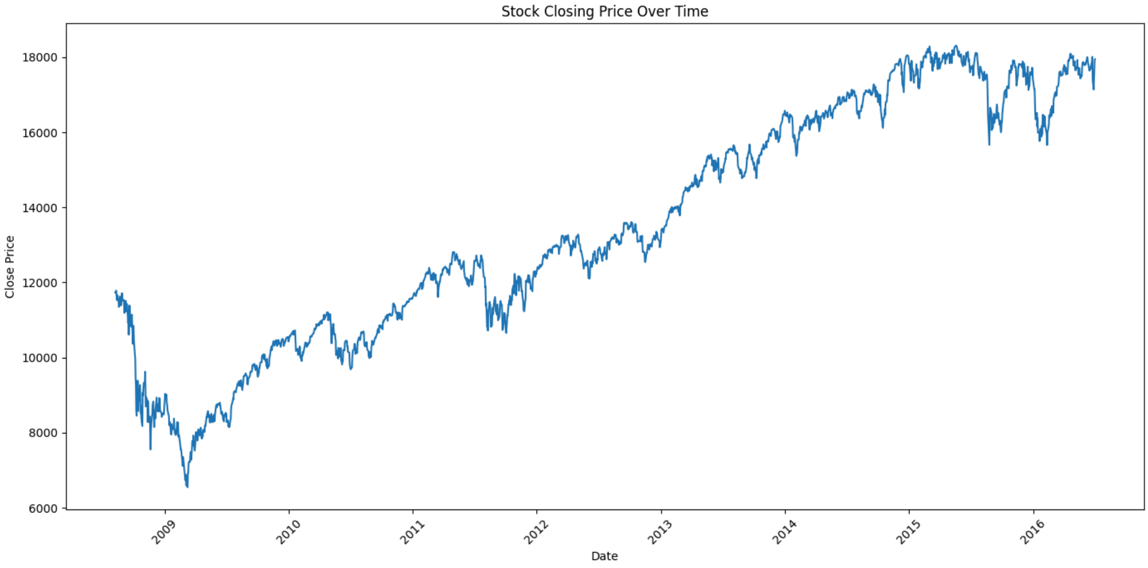
```
Missing values per column:
```

Out[3]:
```
Volatility    9
pct_change    1
dtype: int64
```

In [4]:
```
multimodal.describe()
multimodal.dtypes
```

Out[4]:
```
Date                  object
Open                 float64
High                 float64
Low                  float64
Close                float64
Volume                 int64
Adj Close            float64
Volatility           float64
news_sentiment       float64
Label                  int64
reddit_sentiment     float64
Target                 int64
pct_change           float64
final_sentiment      float64
dtype: object
```

In [5]:
```
plt.figure(figsize=(14, 7))
plt.plot(pd.to_datetime(multimodal['Date']), multimodal['Close'])
plt.title('Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3: Data Preprocessing

In [6]:
```
# Force to string to ensure str methods work
multimodal['Date'] = multimodal['Date'].astype(str)
multimodal['Date'] = multimodal['Date'].str.strip()
```

In [7]:
```
multimodal['Date'] = pd.to_datetime(multimodal['Date'], format='%Y-%m-%d')
multimodal['Date'].dtype
```

Out[7]:
```
dtype('<M8[ns]')
```

In [8]:
```
multimodal_processed = multimodal.copy()

# Fill missing volatility values with rolling standard deviation
multimodal_processed['Volatility'] = multimodal_processed['Volatility'].fillna(
    multimodal_processed['Close'].rolling(window=10).std()
    )

multimodal_processed['pct_change'] = multimodal_processed['pct_change'].fillna(
        multimodal_processed['Close'].pct_change()
    )

multimodal_processed['Next_Close'] = multimodal_processed['Close'].shift(-1)

# Drop the last row which will have NaN in Next_Close
multimodal_processed = multimodal_processed.dropna()

print(f"Shape after preprocessing: {multimodal_processed.shape}")
print("Missing values after preprocessing:")
print(multimodal_processed.isnull().sum())
```

```
Shape after preprocessing: (1979, 15)
Missing values after preprocessing:
Date                0
Open                0
High                0
Low                 0
Close               0
Volume              0
Adj Close           0
Volatility          0
news_sentiment      0
Label               0
reddit_sentiment    0
Target              0
pct_change          0
final_sentiment     0
Next_Close          0
dtype: int64
```

In [9]:
```
multimodal_processed.dtypes
```

Out[9]:
```
Date                datetime64[ns]
Open                       float64
High                       float64
Low                        float64
Close                      float64
Volume                       int64
Adj Close                  float64
Volatility                 float64
news_sentiment             float64
Label                        int64
reddit_sentiment           float64
Target                       int64
pct_change                 float64
final_sentiment            float64
Next_Close                 float64
dtype: object
```

In [10]:
```
multimodal_processed.set_index('Date', inplace=True)
```

In [11]:
```
multimodal_processed.sort_index(inplace=True)
```

In [12]:
```
print(pd.date_range(start=multimodal_processed.index.min(), end=multimodal_
processed.index.max()).difference(multimodal_processed.index))
```

```
DatetimeIndex(['2008-08-16', '2008-08-17', '2008-08-23', '2008-08-24',
               '2008-08-30', '2008-08-31', '2008-09-01', '2008-09-06',
               '2008-09-07', '2008-09-13',
               ...
               '2016-05-22', '2016-05-28', '2016-05-29', '2016-05-30',
               '2016-06-04', '2016-06-05', '2016-06-11', '2016-06-12',
               '2016-06-18', '2016-06-19'],
              dtype='datetime64[ns]', length=892, freq=None)
```

In [13]:
```
# total 2864 days and approximately 409.14 weeks
# missing values(predictable) are weekly holidays (saturday and sunday) ->
409.14 * 2 = 818.28 - classic case of MAR - Missing at Random
# other missing values can be the public holidays which are random (again M
AR)
# need to verify this firmly if 'ALL OF THE' missing values are because of
public holidays and not some other causes - will need other dataset contain
ing record of public dataset
# public holidays are periods of inactivity - hence should not be filled/im
puted or edited in any way
# could aggregate the data to periods (weekly or monthly) but only have ~29
00 rows on a daily basis, aggregating this would shrink the data and neural
networks like LSTM wont
# perform well on this - hence will need to stick to daily data and use a d
ifferent approach to handle missing
```

In [13]:
```
# only uses business days and excludes weekends (pandas library), but publi
c holidays still needs addressing
multimodal_processed = multimodal_processed.asfreq('B')
```

In [14]:
```
print(pd.date_range(start=multimodal_processed.index.min(), end=multimodal_
processed.index.max()).difference(multimodal_processed.index))
```

```
DatetimeIndex(['2008-08-16', '2008-08-17', '2008-08-23', '2008-08-24',
               '2008-08-30', '2008-08-31', '2008-09-06', '2008-09-07',
               '2008-09-13', '2008-09-14',
               ...
               '2016-05-21', '2016-05-22', '2016-05-28', '2016-05-29',
               '2016-06-04', '2016-06-05', '2016-06-11', '2016-06-12',
               '2016-06-18', '2016-06-19'],
              dtype='datetime64[ns]', length=820, freq=None)
```

In [15]:

```
start_date = multimodal_processed.index.min()
end_date = multimodal_processed.index.max()

# Generate the full date range
full_date_range = pd.date_range(start=start_date, end=end_date)

# Count the number of Saturdays and Sundays
weekend_days = np.sum(full_date_range.weekday >= 5)  # 5 = Saturday, 6 = Su
nday
print("Total Weekend Days (Expected):", weekend_days)
```

```
Total Weekend Days (Expected): 820
```

In [16]:

```
duplicate_dates = multimodal_processed.index.duplicated().sum()
print("Total Duplicate Dates:", duplicate_dates)
```

```
Total Duplicate Dates: 0
```

In [17]:

```
print("Date Range (Index):", multimodal_processed.index.min(), "to", multim
odal_processed.index.max())
print("DataFrame Shape:", multimodal_processed.shape)
```

```
Date Range (Index): 2008-08-11 00:00:00 to 2016-06-20 00:00:00
DataFrame Shape: (2051, 14)
```

In [19]:

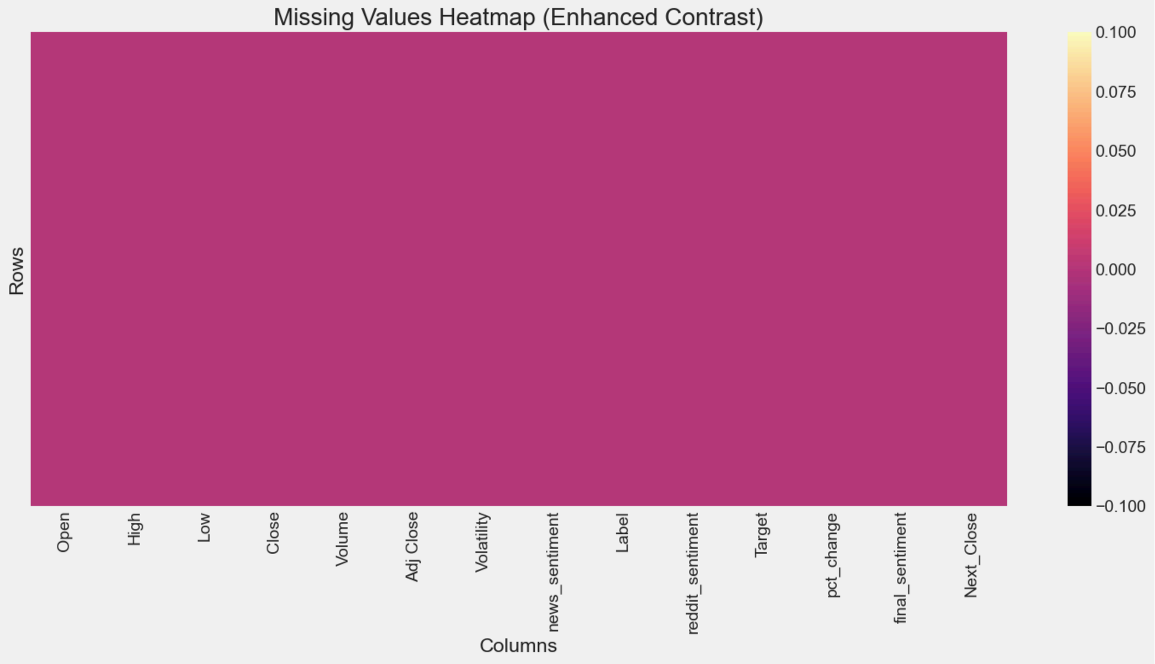
```
import pandas_market_calendars as mcal

# Define the NYSE calendar
nyse = mcal.get_calendar('XNYS')

# Get the valid trading days for the full date range
start_date = multimodal_processed.index.min()
end_date = multimodal_processed.index.max()
trading_days = nyse.schedule(start_date=start_date, end_date=end_date)

# Convert to a flat date index
valid_trading_days = pd.date_range(start=trading_days.index.min(), end=trad
ing_days.index.max(), freq='B')

# Check for remaining gaps
remaining_gaps = valid_trading_days.difference(multimodal_processed.index)
print("\nRemaining Gaps After Applying NYSE Calendar:")
print(remaining_gaps)
print("\nTotal Remaining Gaps:", len(remaining_gaps))
```

```
---------------------------------------------------------------------------
-
ModuleNotFoundError                       Traceback (most recent call las
t)
Cell In[19], line 1
----> 1 import pandas_market_calendars as mcal
      3 # Define the NYSE calendar
      4 nyse = mcal.get_calendar('XNYS')

ModuleNotFoundError: No module named 'pandas_market_calendars'
```

In [19]:
```
import sys
print(sys.executable)
```

```
C:\Users\abhis\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundatio
n.Python.3.11_qbz5n2kfra8p0\python.exe
```

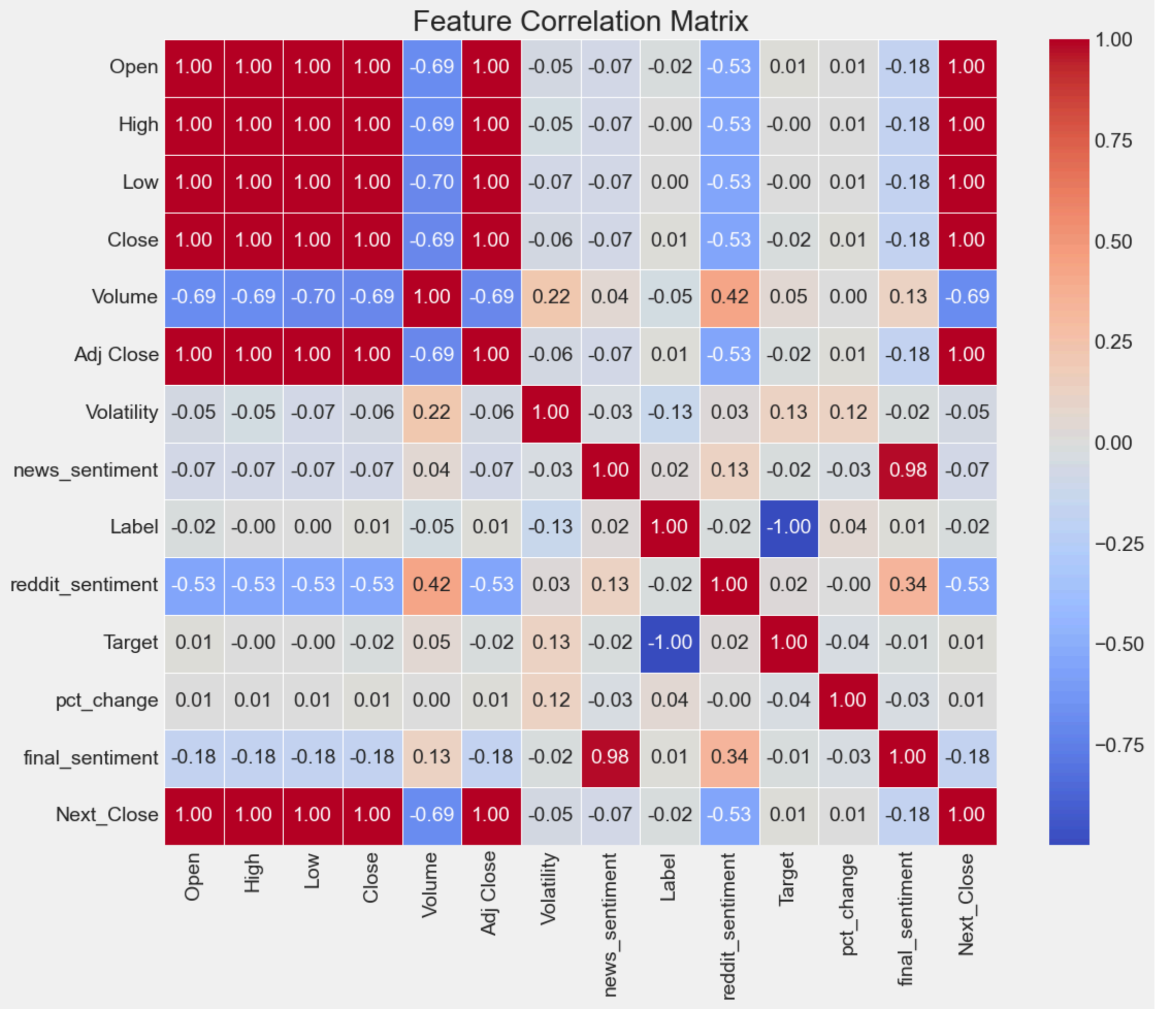
In [ ]:
```
plt.figure(figsize=(15, 8))
sns.heatmap(multimodal_processed.isna(), cmap='magma', cbar=True, yticklabe
ls=False)
plt.title("Missing Values Heatmap (Enhanced Contrast)")
plt.xlabel("Columns")
plt.ylabel("Rows")
plt.tight_layout()
plt.show()
```



In [101]:
```
multimodal_processed.isna().sum()
```

Out[101]:
```
Open                0
High                0
Low                 0
Close               0
Volume              0
Adj Close           0
Volatility          0
news_sentiment      0
Label               0
reddit_sentiment    0
Target              0
pct_change          0
final_sentiment     0
Next_Close          0
dtype: int64
```

```
In [96]: # Plot correlation matrix
         plt.figure(figsize=(12, 10))
         numeric_columns = multimodal_processed.select_dtypes(include=[np.number]).c
         olumns
         correlation_matrix = multimodal_processed[numeric_columns].corr()
         sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', lin
         ewidths=0.5)
         plt.title('Feature Correlation Matrix')
         plt.tight_layout()
         plt.savefig('correlation_matrix.png')
         plt.show()
         plt.close()
```



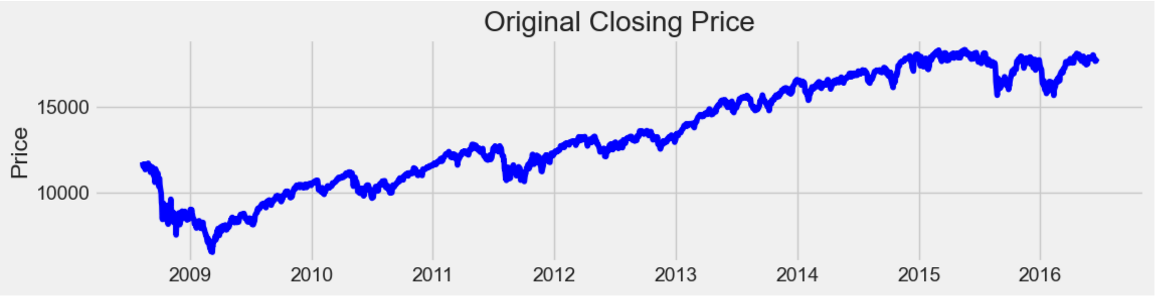
4: Time Series Stationarity Analysis for ARIMA

In [77]:

```python
# Plot the original closing price
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
# plt.plot(multimodal_processed['Date'], multimodal_processed['Close'], color='blue')
plt.plot(multimodal_processed.index, multimodal_processed['Close'], color='blue')
plt.title('Original Closing Price')
plt.ylabel('Price')
```

Out[77]:

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



In [63]:

```python
multimodal_processed.index
```

Out[63]:

```
DatetimeIndex(['2016-06-20', '2016-06-17', '2016-06-16', '2016-06-15',
               '2016-06-14', '2016-06-13', '2016-06-10', '2016-06-09',
               '2016-06-08', '2016-06-07',
               ...
               '2008-08-22', '2008-08-21', '2008-08-20', '2008-08-19',
               '2008-08-18', '2008-08-15', '2008-08-14', '2008-08-13',
               '2008-08-12', '2008-08-11'],
              dtype='datetime64[ns]', name='Date', length=1979, freq=None)
```

In [34]:

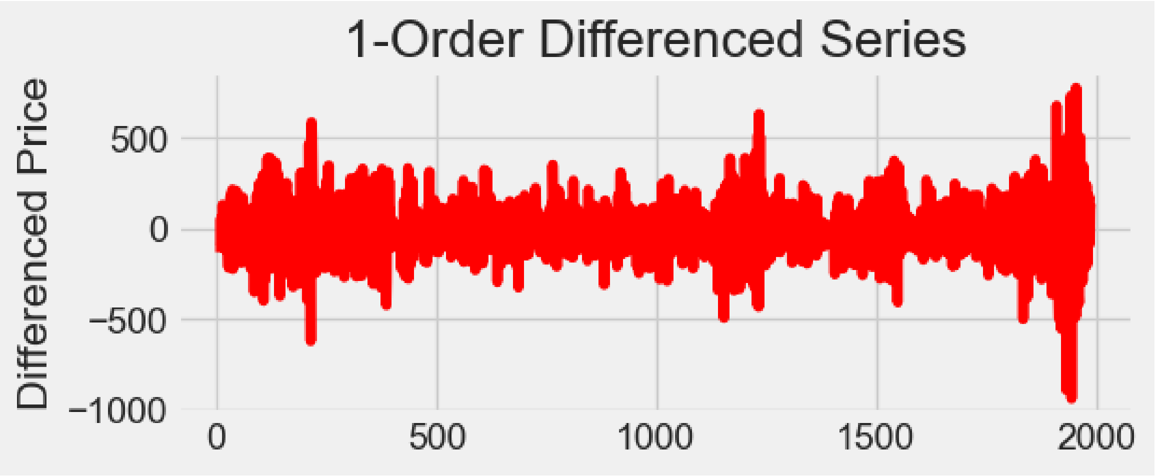
```python
# Conduct Augmented Dickey-Fuller test on the original series
result = adfuller(multimodal_processed['Close'].dropna())
print(f"ADF Statistic (original): {result[0]:.4f}")
print(f"p-value: {result[1]:.4f}")
print(f"Critical Values:")
for key, value in result[4].items():
    print(f"\t{key}: {value:.4f}")

is_stationary = result[1] < 0.05
print(f"Original series is {'stationary' if is_stationary else 'non-stationary'}")
```

```
ADF Statistic (original): -1.3159
p-value: 0.6218
Critical Values:
	1%: -3.4337
	5%: -2.8630
	10%: -2.5676
Original series is non-stationary
```

In [35]:

```
diff_order = 0
diff_series = multimodal_processed['Close']

while not is_stationary and diff_order < 2:
    diff_order += 1
    diff_series = diff_series.diff().dropna()

    # Plot the differenced series
    plt.subplot(2, 1, 2)
    plt.plot(diff_series.index, diff_series, color='red')
    plt.title(f'{diff_order}-Order Differenced Series')
    plt.ylabel('Differenced Price')

    # Check stationarity again
    result = adfuller(diff_series.dropna())
    print(f"\nADF Statistic (diff order {diff_order}): {result[0]:.4f}")
    print(f"p-value: {result[1]:.4f}")
    is_stationary = result[1] < 0.05
    print(f"Differenced series (order {diff_order}) is {'stationary' if is_stationary else 'non-stationary'}")
plt.tight_layout()
plt.savefig('stationarity_analysis.png')
plt.show()
plt.close()
```

```
ADF Statistic (diff order 1): -21.6368
p-value: 0.0000
Differenced series (order 1) is stationary
```



5: Data Splitting and Scalling

In [36]:

```
# Determine the split point
test_size=0.2
split_idx = int(len(multimodal_processed) * (1 - test_size))
print(f"Training set size: {split_idx}, Test set size: {len(multimodal_processed) - split_idx}")

# For ARIMA model - only need the 'Close' price
arima_train = multimodal_processed.iloc[:split_idx]['Close']
arima_test = multimodal_processed.iloc[split_idx:]['Close']

# Select relevant features for LSTM
features = ['Open', 'High', 'Low', 'Close', 'Volume', 'Volatility', 'pct_change']

features.extend(['news_sentiment', 'reddit_sentiment', 'final_sentiment'])
```

```
Training set size: 1583, Test set size: 396
```

In [37]:

```
X = multimodal_processed[features].values
y = multimodal_processed['Next_Close'].values

# Split into train and test sets
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]

# Scale the data for LSTM
feature_scaler = MinMaxScaler()
X_train_scaled = feature_scaler.fit_transform(X_train)
X_test_scaled = feature_scaler.transform(X_test)

target_scaler = MinMaxScaler()
y_train_scaled = target_scaler.fit_transform(y_train.reshape(-1, 1))
y_test_scaled = target_scaler.transform(y_test.reshape(-1, 1))

print("Data scaled successfully")
```

```
Data scaled successfully
```

In [38]:

```python
seq_length = 60  # Using 60 days of data to predict

# Create sequences for LSTM
X_train_seq, y_train_seq = [], []
X_test_seq, y_test_seq = [], []

# Generate training sequences
for i in range(len(X_train_scaled) - seq_length):
    X_train_seq.append(X_train_scaled[i:i+seq_length])
    y_train_seq.append(y_train_scaled[i+seq_length])

# Generate testing sequences
for i in range(len(X_test_scaled) - seq_length):
    X_test_seq.append(X_test_scaled[i:i+seq_length])
    y_test_seq.append(y_test_scaled[i+seq_length])

# Convert to NumPy arrays
X_train_seq = np.array(X_train_seq)
y_train_seq = np.array(y_train_seq)
X_test_seq = np.array(X_test_seq)
y_test_seq = np.array(y_test_seq)

# Print the shapes for verification
print(f"LSTM sequence shape - X_train: {X_train_seq.shape}, y_train: {y_train_seq.shape}")
print(f"LSTM sequence shape - X_test: {X_test_seq.shape}, y_test: {y_test_seq.shape}")

# Extract corresponding test dates (aligning with sequence lengths)
test_dates = multimodal_processed.index[split_idx + seq_length:]

# Print a sample to verify
print("\nSample Training Sequence:")
print(X_train_seq[0])
print("\nSample Test Sequence:")
print(X_test_seq[0])
```

```
LSTM sequence shape - X_train: (1523, 60, 10), y_train: (1523, 1)
LSTM sequence shape - X_test: (336, 60, 10), y_test: (336, 1)

Sample Training Sequence:
[[0.93299113 0.95279989 0.9381293  0.94116314 0.14059409 0.37780364
  0.39524121 0.21898866 0.38411377 0.24627422]
 [0.93259365 0.92798547 0.92264233 0.926126   0.37133717 0.35749881
  0.33598101 0.31194747 0.32357792 0.29685431]
 [0.91738733 0.93048774 0.90745562 0.93284291 0.12911103 0.33709318
  0.44224563 0.1875     0.46612683 0.24945345]
 [0.92914128 0.931426   0.92567187 0.92206959 0.13248022 0.33635225
  0.3565494  0.21419937 0.32612024 0.22348612]
 [0.92996634 0.92804146 0.92183498 0.9260866  0.13187747 0.31735162
  0.42902949 0.1875     0.41413112 0.23228201]
 [0.94384238 0.94661376 0.93749181 0.93277113 0.14416419 0.19472176
  0.4420873  0.1875     0.49006572 0.25735919]
 [0.95639604 0.95192127 0.94684593 0.94817349 0.12693187 0.1296872
  0.4846428  0.28654001 0.25493235 0.25489563]
 [0.96000732 0.9596598  0.9588046  0.96206764 0.09470821 0.1211561
  0.47675642 0.1875     0.47469413 0.25228277]
 [0.9555952  0.96091605 0.96065593 0.96437015 0.09713464 0.15448225
  0.42037813 0.72292862 0.33661554 0.61316597]
 [0.95609476 0.95942785 0.96115379 0.95662935 0.10871044 0.16538429
  0.37196116 1.         0.60930814 0.91356719]
 [0.94328486 0.95318686 0.94805526 0.9545485  0.0980774  0.17215778
  0.3992025  0.6875     0.55551378 0.65856002]
 [0.94028452 0.9396084  0.93267906 0.94141719 0.11415059 0.15954859
  0.34568191 0.1875     0.43738751 0.23996237]
 [0.93903866 0.94023664 0.93428113 0.94506898 0.10378029 0.15316791
  0.42706449 0.         0.47720455 0.11076678]
 [0.93504032 0.9368125  0.92980427 0.9394011  0.10837043 0.12646018
  0.38169686 0.05695236 0.20651169 0.06460797]
 [0.95091191 0.94730847 0.93664634 0.93911467 0.21479352 0.10338535
  0.40787149 0.83608651 0.36870287 0.70966919]
 [0.94341932 0.94427606 0.94827701 0.94908712 0.10011746 0.08490207
  0.45792048 0.27765023 0.46536762 0.31764239]
 [0.94720556 0.94607543 0.94586195 0.9438782  0.09354909 0.08727213
  0.38397866 0.         0.63452737 0.1627222 ]
 [0.93278492 0.94643098 0.93792379 0.94657018 0.109375   0.07399825
  0.42237151 0.1875     0.51338958 0.26506183]
 [0.90845874 0.92905189 0.91368095 0.92970715 0.12065715 0.06953573
  0.32729674 0.         0.14179883 0.        ]
 [0.90635522 0.90668829 0.90846757 0.90500006 0.12268175 0.14174938
  0.28817978 0.1875     0.33581343 0.20641781]
 [0.89827524 0.90914162 0.90353233 0.90592863 0.16008284 0.17131536
  0.41387502 0.25525002 0.43417497 0.29033546]
 [0.90718051 0.9024299  0.89126079 0.8983307  0.13464392 0.21221775
  0.37159496 0.05996428 0.3516788  0.11483556]
 [0.9056877  0.91665526 0.90132526 0.90890567 0.10928227 0.21574376
  0.46190091 0.05180304 0.29811346 0.09094996]
 [0.92888753 0.92425858 0.90729749 0.90929535 0.14659063 0.21726251
  0.41119803 1.         0.58963336 0.90706965]
 [0.9092202  0.93059153 0.9144398  0.9302474  0.12368633 0.21019336
  0.51298558 0.6875     0.39530368 0.60565109]
 [0.93000686 0.92813707 0.9122131  0.9099144  0.12090443 0.18476183
  0.3096433  0.1875     0.295331   0.19304861]
 [0.93000686 0.93553166 0.92525275 0.93138224 0.12387179 0.16469667
  0.51550652 0.71710017 0.38611591 0.62508852]
 [0.95410238 0.94961475 0.93514736 0.93029472 0.12206355 0.13107474
  0.4039431  0.87802175 0.50272353 0.78576526]
 [0.93180797 0.95143051 0.93695019 0.9554782  0.10413576 0.18925442
 0.53265707 0.1875     0.50272496 0.26153987]
[0.93380012 0.93378009 0.93021911 0.92969084 0.11928165 0.18471298
 0.28445243 0.1875     0.70158823 0.3272139 ]
[0.92295838 0.92927905 0.92005538 0.93371601 0.11067322 0.17915301
 0.42899607 0.         0.30567325 0.05411906]
[0.92460172 0.92829663 0.92414853 0.92445091 0.11300692 0.14806233
 0.36394912 0.3711495  0.45182808 0.38415317]
[0.93277677 0.92852403 0.92336193 0.92335523 0.13385571 0.13234898
 0.40388502 0.22980391 0.25777524 0.21276193]
[0.94850711 0.94398812 0.93050785 0.93490768 0.13700853 0.11249202
 0.46606273 0.6875     0.33351192 0.58524453]
[0.93842773 0.94883628 0.94238441 0.95116684 0.11079686 0.13407401
 0.48875306 0.03247945 0.2964854  0.07574234]
[0.94182465 0.93747101 0.92832492 0.93754282 0.19822577 0.10210418
 0.34318851 0.07479824 0.4885317  0.17129239]
[0.96625404 0.96321551 0.94502234 0.94416463 0.14297416 0.10525597
 0.44159861 0.1875     0.49874515 0.26022555]
[0.96303912 0.96891795 0.95931035 0.96860159 0.15560089 0.15072347
 0.52819605 0.71771341 0.41348209 0.63459167]
[0.96202392 0.96415241 0.96091693 0.96266245 0.13006924 0.16269469
 0.38070237 0.38230618 0.26879599 0.3321772 ]
[0.96243633 0.95799541 0.95183666 0.96114608 0.11646884 0.17031829
 0.4019544  0.06204365 0.15361221 0.05100323]
[0.96175387 0.96218051 0.95811274 0.96421936 0.19428474 0.17855616
 0.4241035  0.         0.45279338 0.10270507]
[0.97424597 0.9715552  0.96434958 0.96175811 0.14575606 0.15962573
 0.39740563 0.1875     0.60087565 0.29395386]
[0.97038095 0.97858764 0.97212487 0.97494513 0.14187685 0.13825238
 0.47290432 0.22697198 0.61161998 0.32746823]
[0.96488867 0.97110906 0.96672178 0.96999841 0.12581911 0.11787015
 0.38554771 0.1875     0.26175108 0.18195893]
[0.95076115 0.96016194 0.95099006 0.9642669  0.12515455 0.11396708
 0.38171945 0.22049729 0.30096943 0.21996135]
[0.95490436 0.95178492 0.95320638 0.95189729 0.16961857 0.08154085
 0.34964922 0.27132009 0.38476954 0.28621943]
[0.95331671 0.9546391  0.95528871 0.95525563 0.11761251 0.05605961
 0.42555181 0.1875     0.45581454 0.24604784]
[0.93354638 0.94953554 0.93868264 0.95315146 0.12874011 0.06154031
 0.39908295 0.04420276 0.25696123 0.07158962]
[0.91380729 0.92926631 0.91695885 0.93146919 0.11221872 0.12017176
 0.30420432 0.6875     0.55790612 0.65935008]
[0.915562   0.92777469 0.91722797 0.91235934 0.15252535 0.20429808
 0.31569204 0.         0.5619352  0.13874884]
[0.91195887 0.92344847 0.91402405 0.91474179 0.11062685 0.24356266
 0.42104452 0.25162652 0.38887692 0.27262504]
[0.92724397 0.9226058  0.90895031 0.91068424 0.12628276 0.27389809
 0.3892367  0.1875     0.60188932 0.29428862]
[0.91776036 0.92683301 0.91568477 0.93086644 0.14064045 0.24517439
 0.50910645 0.28238908 0.5335678  0.34376291]
[0.93080771 0.92618951 0.91996064 0.91779762 0.16509026 0.22172015
 0.34525482 0.22923358 0.50936323 0.29541517]
[0.94023699 0.93648631 0.93510337 0.93329508 0.11872527 0.18583633
 0.48566464 0.32359172 0.52731952 0.37297939]
[0.92428415 0.93708064 0.91862769 0.93975817 0.14910979 0.17080989
 0.44088885 0.25744379 0.52125636 0.32075929]
[0.93057841 0.9305797  0.93037386 0.92727715 0.1455706  0.1381736
 0.34839752 0.84858406 0.50196794 0.76316741]
[0.92319696 0.93458998 0.92836868 0.93093709 0.10960682 0.09726365
 0.42722722 0.         0.44004983 0.09849654]
[0.90699739 0.91742326 0.90317998 0.92125106 0.12016259 0.09456145
 0.3618261  0.23361386 0.30853101 0.2324163 ]
 [0.90856196 0.9105472  0.90996655 0.90992255 0.09589824 0.09870701
  0.3535182  0.27357961 0.4783866  0.31885156]]

Sample Test Sequence:
[[0.10167372 0.09520088 0.10718914 0.10039973 0.25248825 0.08832911
  0.39797453 0.1875     0.59921169 0.29340434]
 [0.08792286 0.09335951 0.09594669 0.10198573 0.27179154 0.08398046
  0.42231087 1.         0.73681338 0.95567543]
 [0.08229046 0.07942551 0.08968798 0.08783527 0.24315344 0.08690572
  0.29305531 0.28892687 0.54256553 0.35169771]
 [0.08342042 0.08142542 0.08803865 0.08234253 0.27027695 0.09778933
  0.36363127 0.         0.61890286 0.15756225]
 [0.08317471 0.07995463 0.08952274 0.08341149 0.32265393 0.10057871
  0.41819074 0.1875     0.76098095 0.34682818]
 [0.07412797 0.07495145 0.08220831 0.08315755 0.25553289 0.10170681
  0.40715106 0.86139741 0.63663022 0.81736681]
 [0.07358449 0.0678959  0.07599347 0.07405355 0.42301867 0.11355587
  0.33333313 1.         0.86152803 0.99686209]
 [0.07882168 0.06943772 0.06600761 0.07356323 0.36186325 0.11486607
  0.40514782 1.         0.81266829 0.9807263 ]
 [0.06924888 0.07256459 0.07733781 0.07972257 0.26743323 0.10682115
  0.46105504 0.34609216 0.58567419 0.4093326 ]
 [0.08074094 0.07461696 0.07547836 0.06908601 0.28179092 0.10584113
  0.3202968  0.1875     0.60027278 0.29375476]
 [0.08297416 0.07724146 0.08715742 0.08079141 0.23187129 0.08440176
  0.50805478 0.6875     0.71751068 0.71205903]
 [0.08127514 0.07781371 0.08371789 0.08299056 0.36062685 0.04309163
  0.42764768 0.         0.71409734 0.18899996]
 [0.07219139 0.07408785 0.07855978 0.08189512 0.27339886 0.03487944
  0.4001304  0.6875     0.59830845 0.67269283]
 [0.06663901 0.06401264 0.074745   0.07219641 0.28570104 0.03840403
  0.32828798 0.         0.74628111 0.19962855]
 [0.04801823 0.05928216 0.05618801 0.06750929 0.35004018 0.0454526
  0.36981533 0.6875     0.7471883  0.72185998]
 [0.05223919 0.04271546 0.04267611 0.04783949 0.44525841 0.09993664
  0.24304826 0.48163114 0.59019986 0.5137248 ]
 [0.04072155 0.04553356 0.04185488 0.05306222 0.28755564 0.12116375
  0.45414503 0.         0.50846134 0.12108925]
 [0.04276248 0.03667272 0.0402668  0.04079563 0.26302856 0.16031747
  0.3043359  0.1875     0.66008259 0.31350678]
 [0.02593704 0.04295321 0.03415697 0.04314434 0.35206479 0.17387816
  0.42957215 0.1875     0.6402588  0.30696003]
 [0.03696428 0.03003202 0.03346749 0.02572589 0.32124753 0.22009291
  0.2589978  0.1875     0.59905282 0.29335187]
 [0.03676271 0.03042832 0.0254982  0.03776413 0.46351076 0.21472458
  0.51469878 1.         0.68130042 0.93734244]
 [0.06798797 0.05853396 0.04431732 0.03659895 0.45720512 0.1918746
  0.3991699  0.1875     0.63249961 0.30439758]
 [0.0701448  0.06250577 0.07133198 0.06771104 0.29446402 0.16320557
  0.67919118 0.64253621 0.62613354 0.64774667]
 [0.05790633 0.06339612 0.0645939  0.07075997 0.3535021  0.16038156
  0.43502942 0.26654673 0.54674184 0.33608651]
 [0.04433056 0.04895053 0.05250372 0.05785474 0.29367582 0.1479045
  0.30050952 0.6875     0.74436351 0.7209271 ]
 [0.05047871 0.05459709 0.0495978  0.04415182 0.47677114 0.14858556
  0.29252479 0.         0.54226093 0.13225147]
 [0.06379259 0.05686855 0.05090639 0.05031115 0.35799951 0.14752223
  0.46236007 0.1875     0.56804658 0.28311215]
 [0.05885204 0.05642912 0.05659811 0.06372425 0.39218595 0.1552864
  0.5242781  0.1875     0.95360436 0.41044151]
 [0.05897484 0.05993362 0.06251609 0.05887026 0.32283939 0.15592583
 0.36811897 1.         0.74259407 0.95758449]
[0.05662799 0.05664013 0.06438366 0.05916824 0.31979476 0.12030083
 0.41180514 0.1875     0.77637054 0.35191055]
[0.08148497 0.07210661 0.0627517  0.05639985 0.4871569  0.10105922
 0.38570921 0.37992725 0.36466557 0.36203183]
[0.10632453 0.09837088 0.0878204  0.08154495 0.45728239 0.10262491
 0.62376001 0.77920725 0.48878216 0.70614381]
[0.11974276 0.11060556 0.10429134 0.10626936 0.30115603 0.19119085
 0.61576859 1.         0.49725114 0.87656069]
[0.10684141 0.11176746 0.11291549 0.12044517 0.28397008 0.27994382
 0.5252846  0.         0.60189979 0.15194703]
[0.11827088 0.10944251 0.10934546 0.1070229  0.54791048 0.30576531
 0.30067218 1.         0.51687631 0.88304184]
[0.11516144 0.11105421 0.12156512 0.11872015 0.29814231 0.31645648
 0.50494118 0.1875     0.73926196 0.33965554]
[0.1091268  0.10936319 0.11551654 0.11526773 0.3004451  0.30784281
 0.38129712 0.6875     0.80922939 0.74234887]
[0.1131565  0.10398125 0.11024406 0.10906435 0.3827275  0.29728816
 0.35886873 0.         0.72987706 0.19421116]
[0.10822512 0.10551384 0.11287161 0.11332251 0.26836053 0.26835748
 0.44403868 0.1875     0.74742615 0.34235174]
[0.10661313 0.09889066 0.10856822 0.10801295 0.25392557 0.21527197
 0.36606273 0.6875     0.60922797 0.67629897]
[0.10252323 0.09807133 0.102895   0.10669945 0.32297849 0.10558912
 0.39853417 0.         0.49881027 0.11790201]
[0.10178259 0.09604582 0.10766963 0.10285282 0.27452708 0.04479666
 0.3777986  0.         0.58681505 0.14696534]
[0.10408191 0.09483019 0.1048942  0.10266036 0.27839083 0.04975788
 0.40768916 0.1875     0.65589521 0.31212391]
[0.08624941 0.09720886 0.09428823 0.10404461 0.26485225 0.03961417
 0.42063085 0.34003398 0.53339877 0.38746958]
[0.09990396 0.09138636 0.09341501 0.08596998 0.20018855 0.08394628
 0.26107631 0.3872821  0.61413039 0.45000049]
[0.09942307 0.0908841  0.102947   0.09993488 0.15725457 0.0732648
 0.52547753 0.31243043 0.71001917 0.42484208]
[0.09982519 0.09433727 0.1074075  0.09957554 0.1305638  0.06390107
 0.4063126  0.6875     0.84075327 0.75275954]
[0.09635765 0.09099018 0.1030255  0.09976914 0.14465875 0.06101315
 0.41086173 0.63311649 0.57397433 0.62337004]
[0.09047139 0.08754623 0.09786739 0.09664016 0.06840381 0.04904084
 0.38352479 0.65354104 0.67784077 0.67317741]
[0.09014691 0.08404992 0.09510934 0.0904206  0.16080922 0.05101963
 0.35796539 0.6875     0.67296267 0.69734718]
[0.08439273 0.08262352 0.09242866 0.09024546 0.19576842 0.04946239
 0.40781659 0.1875     0.66617041 0.31551727]
[0.07459155 0.08008631 0.08263567 0.08435738 0.24119065 0.05960314
 0.36044372 1.         0.62431984 0.91852476]
[0.07219139 0.07009851 0.07502449 0.07447436 0.72896575 0.08505927
 0.32691735 0.1875     0.6350605  0.30524331]
[0.0873272  0.07798966 0.08011324 0.07208275 0.29434038 0.09799629
 0.38917577 0.1875     0.76459315 0.3480211 ]
[0.0884652  0.08613723 0.09393012 0.08748522 0.30894535 0.09745119
 0.53892975 0.6875     0.37293307 0.59826326]
[0.09420195 0.08489486 0.09382624 0.08874652 0.27687624 0.08908493
 0.41975146 0.6875     0.56263742 0.66091258]
[0.09095353 0.08668387 0.09897623 0.09443286 0.22556565 0.08233667
 0.45647905 0.6875     0.63209457 0.6838506 ]
[0.08308779 0.08311644 0.09075283 0.09100715 0.2651459  0.07113682
 0.38095987 0.6875     0.83051204 0.74937741]
[0.07527537 0.07851877 0.08332516 0.08339405 0.28978116 0.06245349
 0.34617916 0.24398341 0.51639368 0.30893464]
```

```
[0.06911556 0.06659287 0.07175934 0.07542037 0.27849901 0.06786204
 0.34277402 1.         0.73637143 0.95552948]]
```

6: ARIMA Model

In [42]:

```
multimodal_processed['Date'].describe()
```

Out[42]:

```
count           1979
unique          1979
top       2008-08-11
freq               1
Name: Date, dtype: object
```

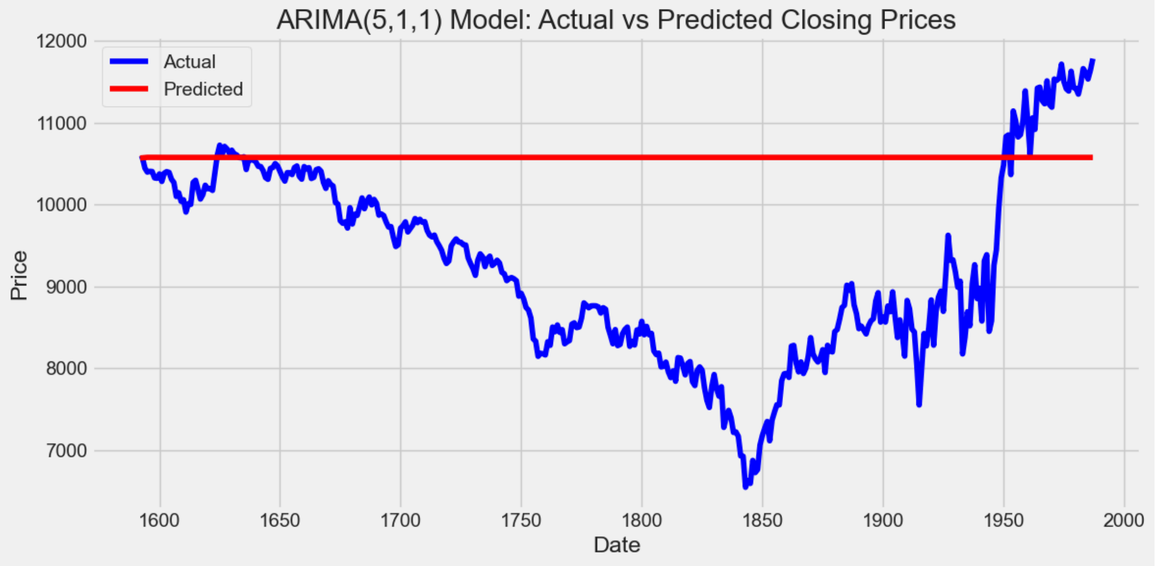
In [39]:

```
# Extract ARIMA training and testing data
train_data = multimodal_processed['Close'][:split_idx]
test_data = multimodal_processed['Close'][split_idx:]
test_dates = multimodal_processed.index[split_idx:]

# Define ARIMA parameters (predefined for simplicity)
p, d, q = 5, diff_order, 1
print(f"Building ARIMA({p},{d},{q}) model...")

# Build and fit ARIMA model
model = ARIMA(train_data, order=(p, d, q))
model_fit = model.fit()
print(model_fit.summary())

# Make predictions
predictions = model_fit.forecast(steps=len(test_data))

# Evaluate the model
mse = mean_squared_error(test_data, predictions)
rmse = math.sqrt(mse)
mae = mean_absolute_error(test_data, predictions)
r2 = r2_score(test_data, predictions)
mape = np.mean(np.abs((test_data - predictions) / test_data)) * 100

print("\nARIMA Model Evaluation:")
print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"MAE: {mae:.4f}")
print(f"R²: {r2:.4f}")
print(f"MAPE: {mape:.4f}%")

# Plot the predictions against actual values
plt.figure(figsize=(12, 6))
plt.plot(test_dates, test_data, color='blue', label='Actual')
plt.plot(test_dates, predictions, color='red', label='Predicted')
plt.title(f'ARIMA({p},{d},{q}) Model: Actual vs Predicted Closing Prices')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.tight_layout()
plt.savefig('arima_predictions.png')
plt.show()
plt.close()

# Print a sample of the predictions for verification
print("\nSample Predictions:")
print(predictions[:5])
```

```
Building ARIMA(5,1,1) model...

C:\Users\abhis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11
_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provi
ded. As a result, forecasts cannot be generated. To use the model for fore
casting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\abhis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11
_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provi
ded. As a result, forecasts cannot be generated. To use the model for fore
casting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\abhis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11
_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmode
ls\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provi
ded. As a result, forecasts cannot be generated. To use the model for fore
casting, use one of the supported classes of index.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results                               
==========================================================================
====
Dep. Variable:                  Close   No. Observations:                 
1583
Model:                 ARIMA(5, 1, 1)   Log Likelihood               -991
5.861
Date:                Wed, 14 May 2025   AIC                          1984
5.721
Time:                        00:32:25   BIC                          1988
3.286
Sample:                             0   HQIC                         1985
9.678
                               - 1583                                    
Covariance Type:                  opg                                    
==========================================================================
====
                 coef    std err          z      P>|z|      [0.025      0.
975]
--------------------------------------------------------------------------
----
ar.L1          0.0933      0.276      0.338      0.735      -0.447       
0.634
ar.L2          0.0219      0.019      1.131      0.258      -0.016       
0.060
ar.L3         -0.0225      0.020     -1.130      0.258      -0.061       
0.016
ar.L4         -0.0223      0.020     -1.128      0.259      -0.061       
0.016
ar.L5         -0.0688      0.022     -3.081      0.002      -0.113      -
0.025
ma.L1         -0.1381      0.276     -0.501      0.617      -0.679       
0.402
sigma2      1.632e+04    434.630     37.547      0.000    1.55e+04    1.72
e+04
==========================================================================
=========
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):            
299.00
Prob(Q):                              0.98   Prob(JB):                    
0.00
Heteroskedasticity (H):               0.73   Skew:                        
0.27
Prob(H) (two-sided):                  0.00   Kurtosis:                    
5.06
==========================================================================
=========

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

ARIMA Model Evaluation:
MSE: 3091101.8061
RMSE: 1758.1530
MAE: 1455.8783
R²: -1.2922
MAPE: 17.5052%
```

```
C:\Users\abhis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11
_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmode
ls\tsa\base\tsa_model.py:837: ValueWarning: No supported index is availabl
e. Prediction results will be given with an integer index beginning at `st
art`.
  return get_prediction_index(
C:\Users\abhis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11
_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\statsmode
ls\tsa\base\tsa_model.py:837: FutureWarning: No supported index is availab
le. In the next version, calling this method in a model without a supporte
d index will result in an exception.
  return get_prediction_index(
```



```
Sample Predictions:
1583    10568.212160
1584    10570.992818
1585    10573.279465
1586    10576.595578
1587    10577.010090
Name: predicted_mean, dtype: float64
```

7: LSTM Regression model