Abstract

In this project, we analyze historical stock price data using time series modeling techniques, specifically Autoregressive (AR) and Moving Average (MA) models. Stock prices change over time based on past price movements and random market fluctuations.

The objective of this study is to understand the temporal behavior of stock prices by visualizing trends, testing stationarity, and analyzing autocorrelation patterns. Based on these observations, AR and MA models are applied to model short-term stock price movements. This project helps in understanding how past stock prices influence future prices and forms a foundation for financial forecasting.

Problem Statement

Stock price data is sequential and time-dependent in nature. Current stock prices are influenced by previous prices, market trends, and random disturbances. Without proper time series analysis, predicting stock prices can result in inaccurate outcomes.

The goal of this project is to statistically analyze stock price data to:

- Identify price trends and fluctuations
- Measure dependency on previous stock prices
- Check stationarity of the data
- Apply AR and MA models for stock price modelling

Objectives

- To study historical stock price movements
- To analyze autocorrelation and partial autocorrelation
- To test stationarity of stock price data
- To implement AR and MA models
- To understand short-term stock price behavior

Dataset Description and Preprocessing

The dataset consists of historical stock market data, including attributes such as Date, Open, High, Low, Close, and Volume. For modeling purposes, the Closing Price is selected.

Preprocessing Steps:

- Date column converted into datetime format
- Date set as the time index
- Missing values removed or handled
- Data sorted chronologically

Time Series Visualization

A time series plot of stock closing prices is generated to observe price behavior over time.

Observations:

- Stock prices show continuous fluctuations
- Presence of upward and downward movements
- Sudden changes indicate market volatility
- This visualization helps in understanding the overall pattern of stock price movements.

Stationarity Testing

Stationarity is essential for applying AR and MA models. A stationary time series has constant mean and variance over time.

The Augmented Dickey-Fuller (ADF) test is used to check stationarity.

Interpretation:

- p-value > 0.05 → Data is non-stationary
- Stock price data is usually non-stationary due to trends
- Hence, differencing may be applied before modeling.

Autocorrelation Function (ACF) Analysis

ACF measures the correlation between current stock prices and their past values.

Results:

- High correlation at initial lags
- Gradual decay of correlation
- Indicates influence of past price movements
- ACF helps in identifying the order of the MA model.

Autoregressive (AR) Model

The AR model predicts the current stock price using its previous price values.

Features:

- Depends on lagged stock prices
- Captures momentum in price movements
- Suitable when PACF shows sharp cutoff

Moving Average (MA) Model

The MA model predicts stock prices using past error terms.

Features:

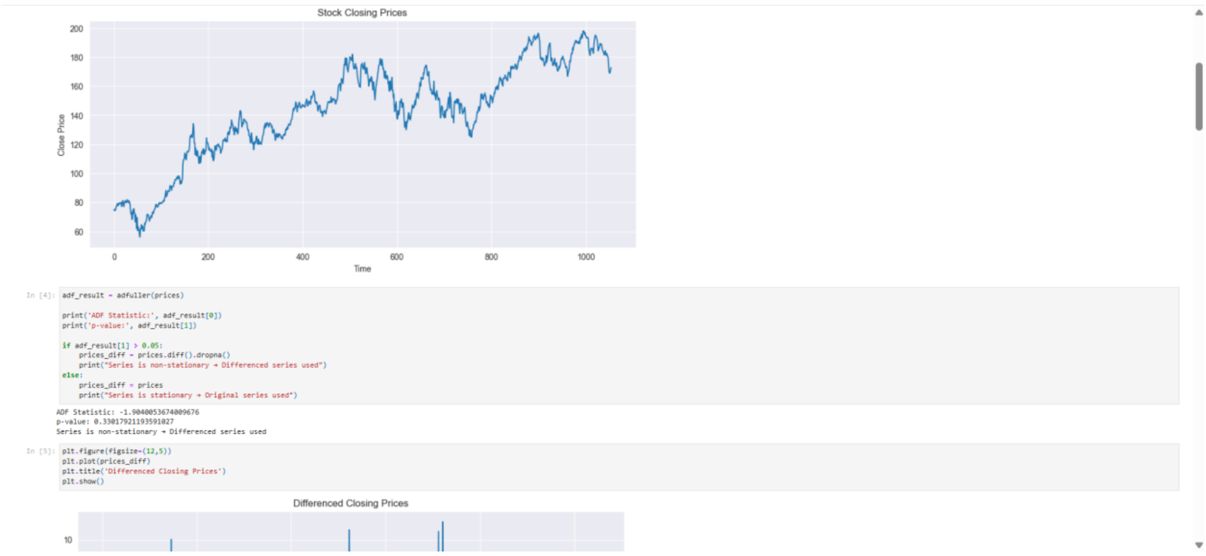
- Models random market shocks
- Smoothens noise in stock price data
- Suitable when ACF shows sharp cutoff

Results and Discussion

- Stock prices show strong dependency on past values
- Non-stationarity is observed in raw data
- ACF and PACF help in selecting AR and MA orders
- AR and MA models effectively model short-term price behavior

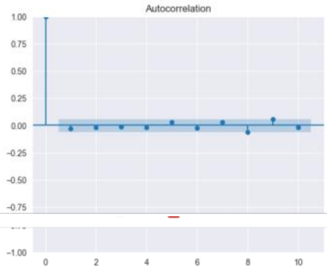
Conclusion

This project demonstrates the application of AR and MA models for modeling stock price time series data. Understanding trends, stationarity, and autocorrelation patterns is essential for building accurate financial models. This study provides a strong foundation for advanced forecasting models such as ARIMA in stock market prediction.



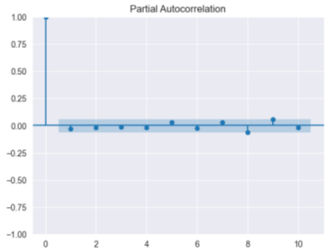
In [6]:
```
plt.figure(figsize=(12,5))
plot_acf(prices_diff, lags=10)
plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```



In [7]:
```
plt.figure(figsize=(12,5))
plot_pacf(prices_diff, lags=10)
plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```



In [8]:
```
ar_lag = 2
ar_model = AutoReg(prices_diff, lags=ar_lag).fit()
print(ar_model.summary())
```


```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                     AutoReg(2)   Log Likelihood               -2508.864
Method:               Conditional MLE   S.D. of innovations              2.633
Date:                Fri, 06 Feb 2026   AIC                           5025.727
Time:                        22:28:49   BIC                           5045.557
Sample:                             2   HQIC                          5033.246
                                 1053
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0973      0.081      1.196      0.232      -0.062       0.257
Close.L1      -0.0285      0.031     -0.922      0.356      -0.089       0.032
```


```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                     AutoReg(2)   Log Likelihood               -2508.864
Method:               Conditional MLE   S.D. of innovations              2.633
Date:                Fri, 06 Feb 2026   AIC                           5025.727
Time:                        22:28:49   BIC                           5045.557
Sample:                             2   HQIC                          5033.246
                                 1053
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0973      0.081      1.196      0.232      -0.062       0.257
Close.L1      -0.0285      0.031     -0.922      0.356      -0.089       0.032
Close.L2      -0.0182      0.031     -0.591      0.554      -0.079       0.042
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.7798           -7.3621j            7.4033           -0.2668
AR.2           -0.7798           +7.3621j            7.4033            0.2668
-----------------------------------------------------------------------------
```

```
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
```

In [9]:
```
ma_order = 2
ma_model = ARIMA(prices_diff, order=(0,0,ma_order)).fit()
print(ma_model.summary())
```

```
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                 ARIMA(0, 0, 2)   Log Likelihood               -2512.684
Date:                Fri, 06 Feb 2026   AIC                           5033.367
Time:                        22:28:57   BIC                           5053.205
Sample:                             0   HQIC                          5040.888
                               - 1053
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0927      0.079      1.178      0.239      -0.062       0.247
ma.L1         -0.0290      0.026     -1.103      0.270      -0.080       0.022
ma.L2         -0.0187      0.027     -0.685      0.494      -0.072       0.035
sigma2         6.9209      0.228     30.394      0.000       6.475       7.367
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               119.26
Prob(Q):                              1.00   Prob(JB):                         0.00
```

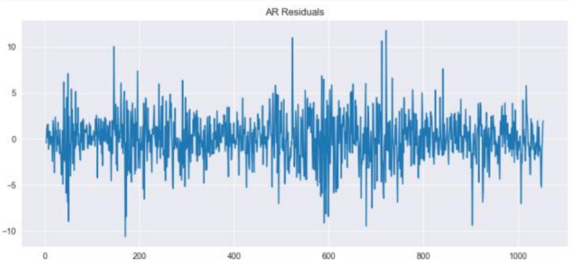
```
Prob(Q):                              1.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.94   Skew:                            -0.08
Prob(H) (two-sided):                  0.53   Kurtosis:                         4.64
===================================================================================

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

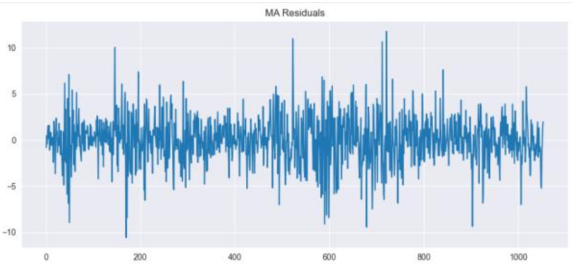
In [10]:
```python
arma_model = ARIMA(prices_diff, order=(ar_lag,0,ma_order)).fit()
print(arma_model.summary())
```

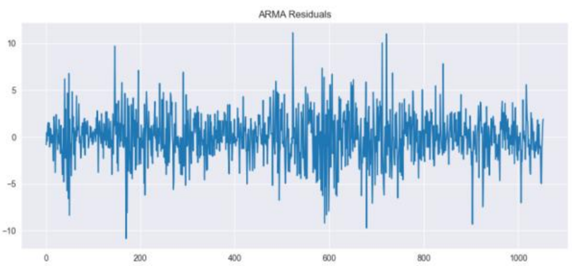
```
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result, forecasts cannot be generated. To use the model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\statespace\sarimax.py:966: UserWarning: Non-stationary starting autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\base\model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                 ARIMA(2, 0, 2)   Log Likelihood               -2503.216
Date:                Fri, 06 Feb 2026   AIC                           5018.431
Time:                        22:29:09   BIC                           5048.187
Sample:                             0   HQIC                          5029.712
                               - 1053
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0927      0.080      1.153      0.249      -0.065       0.250
ar.L1         -1.8631      0.013   -144.794      0.000      -1.888      -1.838
ar.L2         -0.9791      0.013    -77.371      0.000      -1.004      -0.954
ma.L1          1.8560      0.017    106.164      0.000       1.822       1.890
ma.L2          0.9633      0.017     55.829      0.000       0.929       0.997
sigma2         6.8033      0.226     30.126      0.000       6.361       7.246
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               107.96
Prob(Q):                              0.92   Prob(JB):                         0.00
Heteroskedasticity (H):               0.95   Skew:                            -0.07
Prob(H) (two-sided):                  0.64   Kurtosis:                         4.56
===================================================================================

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

In [11]:
```python
plt.figure(figsize=(12,5))
plt.plot(ar_model.resid)
plt.title('AR Residuals')
```

In [11]:
```python
plt.figure(figsize=(12,5))
plt.plot(ar_model.resid)
plt.title('AR Residuals')
plt.show()

plt.figure(figsize=(12,5))
plt.plot(ma_model.resid)
plt.title('MA Residuals')
plt.show()

plt.figure(figsize=(12,5))
plt.plot(arma_model.resid)
plt.title('ARMA Residuals')
plt.show()
```







In [12]:
```
lb_test = acorr_ljungbox(arma_model.resid, lags=[10], return_df=True)
print(lb_test)
```

```
     lb_stat  lb_pvalue
10  2.779979   0.986138
```

In [13]:
```
print("AR  AIC:", ar_model.aic, "BIC:", ar_model.bic)
print("MA  AIC:", ma_model.aic, "BIC:", ma_model.bic)
print("ARMA AIC:", arma_model.aic, "BIC:", arma_model.bic)
```

```
AR  AIC: 5025.727140323777 BIC: 5045.5571298072855
MA  AIC: 5033.367301699888 BIC: 5053.204895748424
ARMA AIC: 5018.43103406367 BIC: 5048.187425136474
```

In [14]:
```
forecast_steps = 5
forecast = arma_model.forecast(steps=forecast_steps)
print(forecast)
```

```
1053   -0.005696
1054    0.224722
1055   -0.057063
1056    0.242319
1057   -0.039553
Name: predicted_mean, dtype: float64
```

```
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:837: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\Users\HP\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\base\tsa_model.py:837: FutureWarning: No supported index is available. In the next version, calling this method in a model without a supported index will result in an exception.
  return get_prediction_index(
```

In [15]:
```
actual = prices_diff[-5:]
predicted = arma_model.predict(start=len(prices_diff)-5, end=len(prices_diff)-1)
rmse = np.sqrt(mean_squared_error(actual, predicted))
print("RMSE:", rmse)
```

```
RMSE: 2.5530175698750086
```