NVIDIA STOCK PREDICTING-ARIMA

Members : Daniel, Brinta

Content *overview*

Data Intro (Why Nvidia) ?

NVIDIA :

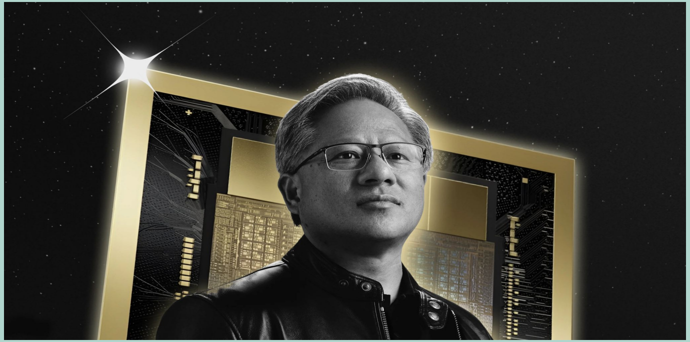
- **Founded:** 1993
- **Founder & CEO:** Jensen Huang
- **Headquarters:** Santa Clara, California
- **Core Businesses:** GPUs, AI computing, data centers, autonomous vehicles, cloud infrastructure

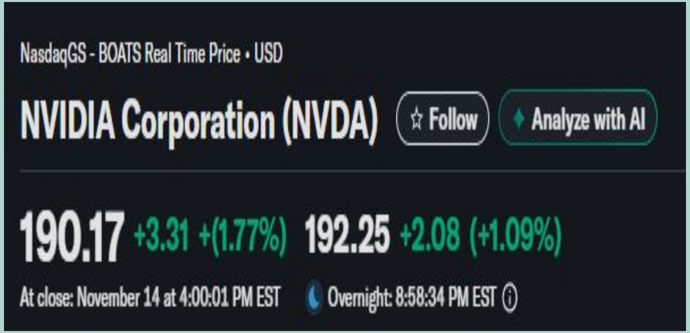
Market Significance

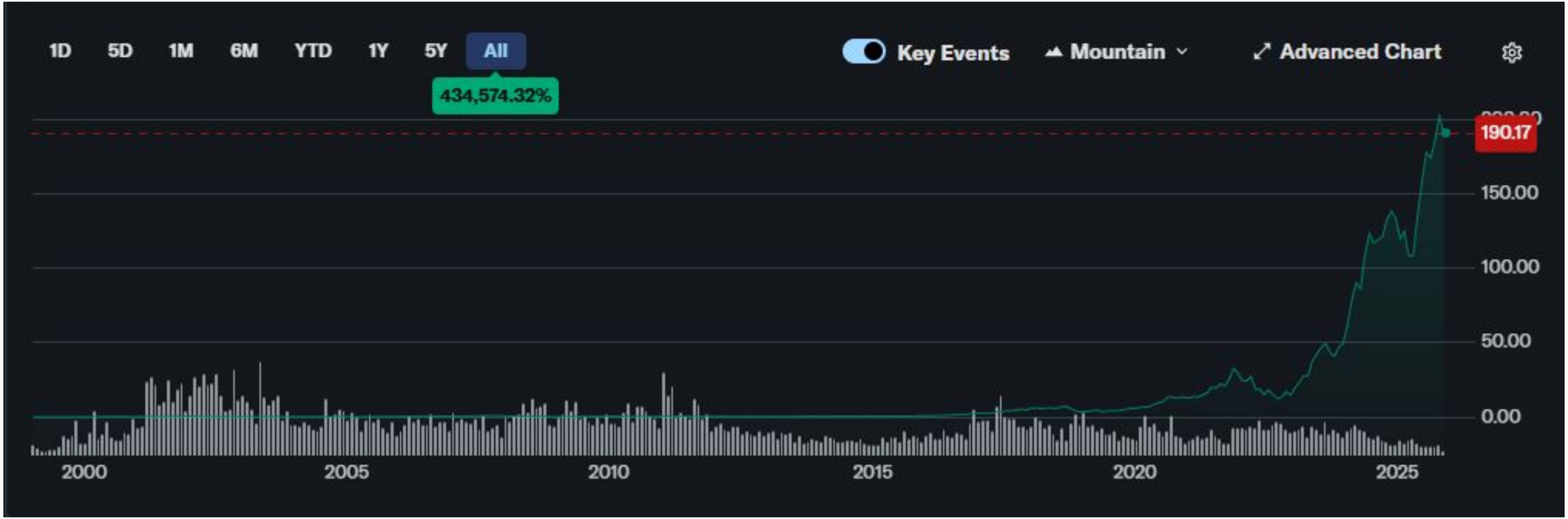
- **Ticker Symbol:** NVDA (NASDAQ)
- **Current Stock Price :** ~$190/share - Start from $5 (2017)
- Became one of the **top 5 most valuable U.S. companies**

Why NVIDIA Was Selected

- High volatility makes it ideal for short-term forecasting
- Stock behavior reflects real-world macro trends (tech cycles, AI investments, etc.)






1D
5D
1M
6M
YTD
1Y
5Y
All
434,574.32%
Key Events
Mountain
Advanced Chart
190.17
150.00
100.00
50.00
0.00
2000
2005
2010
2015
2020
2025

NVIDIA Stock Dataset & Variable Breakdown

Dataset Source

- Provider: NVIDIA Historical Stock Price Dataset (Yfinance)
- Time Range: 2 years of data
- Total Observations: ~500 rows (daily frequency)
- Target Variable: **Close price**

Data Overview

The dataset provides the following columns for analysis:

- **Date**: The date of the stock data entry.
- **Price**: The price of the stock on that date.
- **Adj Close**: The adjusted closing price, accounting for stock splits and dividends.
- **Close**: The closing price of the stock on that date.
- **High**: The highest price during the trading day.
- **Low**: The lowest price during the trading day.
- **Open**: The opening price on the day.
- **Volume**: The number of shares traded on that day.

Price	Close	High	Low	Open	Volume
Ticker	NVDA	NVDA	NVDA	NVDA	NVDA
count	1255.000000	1255.000000	1255.000000	1255.000000	1.255000e+03
mean	62.886648	63.968154	61.690980	62.891741	3.969987e+08
std	54.963195	55.806759	54.060165	55.027769	1.819776e+08
min	11.213526	11.720918	10.800024	10.957835	9.788400e+07
25%	18.584849	18.916072	18.067006	18.413221	2.460908e+08
50%	31.892086	32.694478	31.287299	32.217442	3.824624e+08
75%	114.261547	116.566815	111.538470	114.100646	5.072100e+08
max	207.039993	212.190002	205.559998	208.080002	1.543911e+09

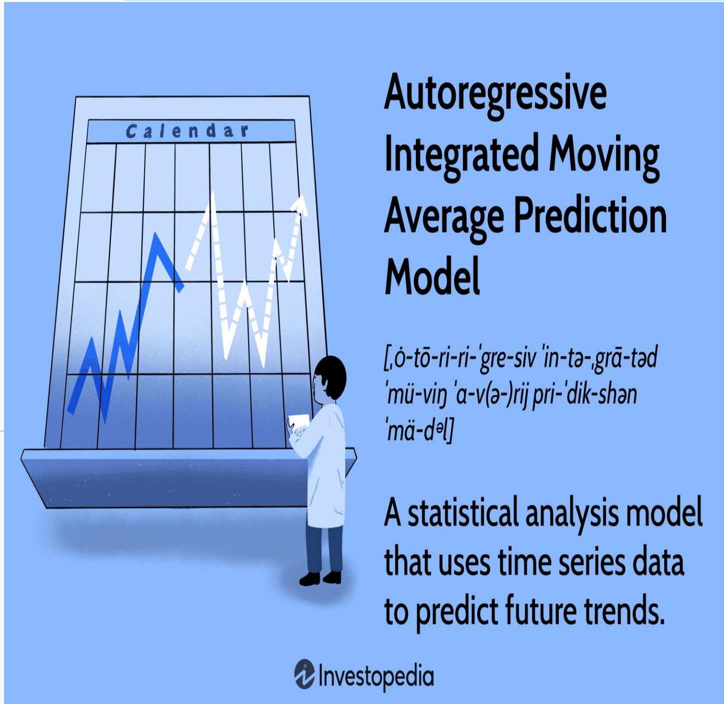
```
df = pd.DataFrame(data)
display(df.head(5))
```

Price	Close	High	Low	Open	Volume
Ticker	NVDA	NVDA	NVDA	NVDA	NVDA
Date					
2020-11-16	13.474119	13.607960	13.115962	13.132412	413776000
2020-11-17	13.381402	13.554872	13.263263	13.511256	312028000
2020-11-18	13.387879	13.564341	13.144872	13.424268	510924000
2020-11-19	13.399345	13.446451	13.060131	13.172538	565936000
2020-11-20	13.047923	13.453435	13.025241	13.413059	341088000

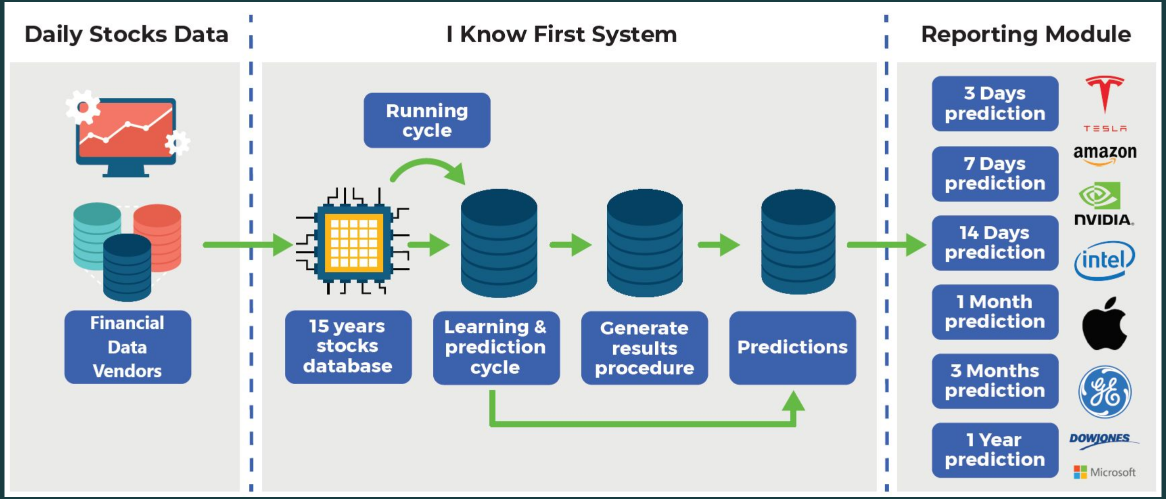
What Is an ARIMA Time Series Model?

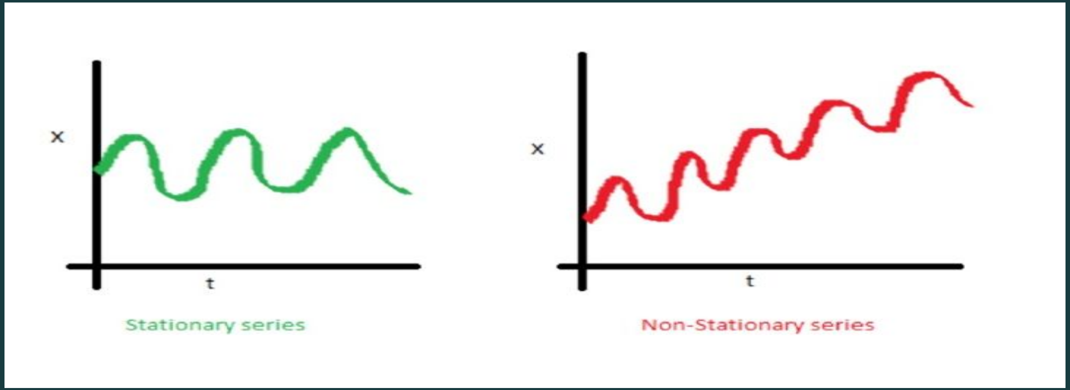
- **ARIMA** = AutoRegressive Integrated Moving Average
  - **AR:** Past values to predict future values.
  - **I:** Remove trends and make the series stationary.
  - **MA:** Uses past forecasting errors -> improve predictions.
- Why It's Useful:
  - Captures trends + patterns over time
  - Works well for finance, sales, demand forecasting
  - **short-term forecasting**

$$\left(1-\sum_{i=1}^{p'}\alpha_i L^i\right)X_t=\left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t$$

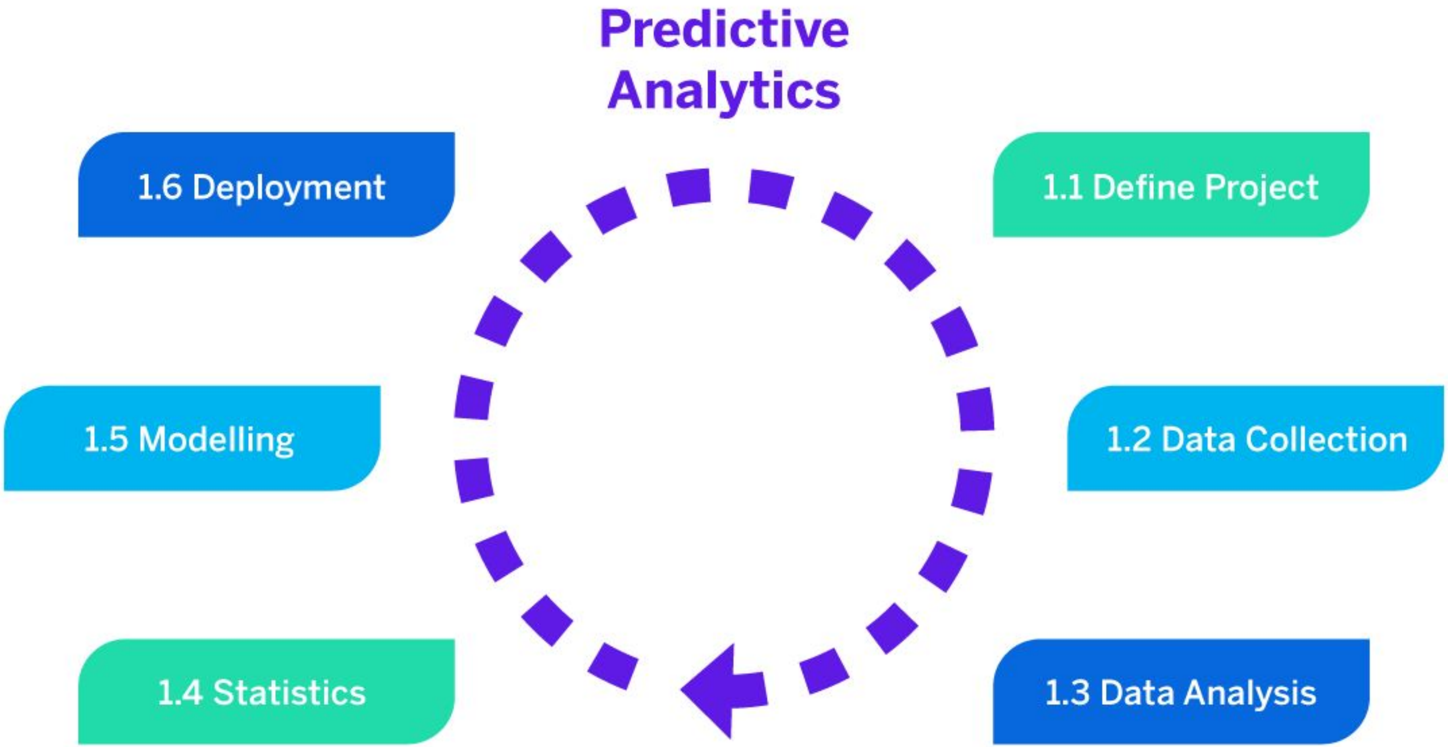


Stationarity Test- ADF test





Predictive Analytics Process



DATA CLEANING

```
data.isna().sum()
```

Price	Ticker	0
Close	NVDA	0
High	NVDA	0
Low	NVDA	0
Open	NVDA	0
Volume	NVDA	0

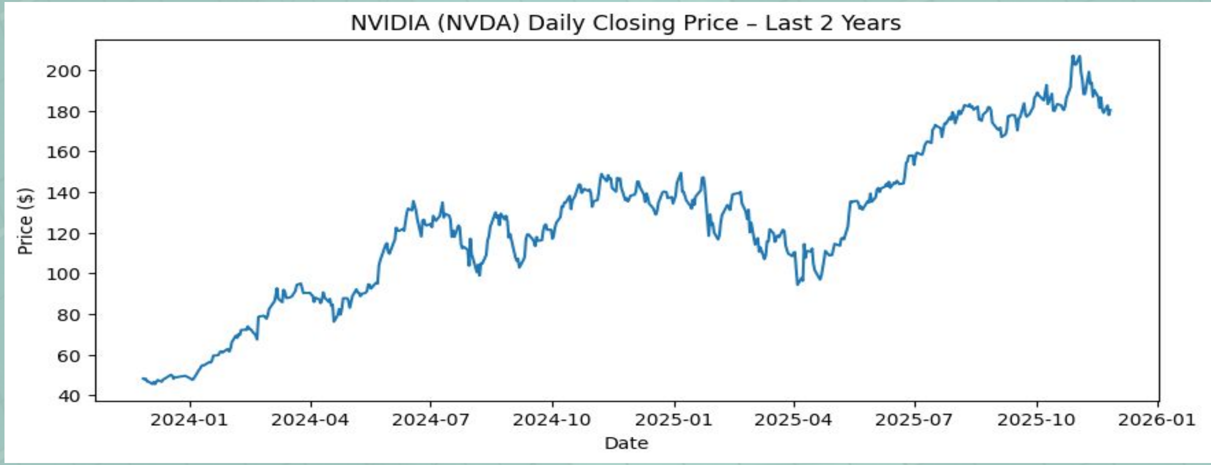
dtype: int64

```
# Extract the 'Close' price as the target variable
target = data['Close']

# Drop the 'Close' column from the original DataFrame
data = data.drop('Close', axis=1)

# Display the first few values of the target variable
print("\nTarget variable (Close Price):")
print(target.head())
```

```
Target variable (Close Price):
Ticker            NVDA
Date
2023-11-27   48.213585
2023-11-28   47.792831
2023-11-29   48.111649
2023-11-30   46.742455
2023-12-01   46.737453
```

```
print("\nModified DataFrame (data) after dropping 'Close':")
print(data.tail())
```

```
Modified DataFrame (data) after dropping 'Close':
Price             High         Low        Open     Volume
Ticker            NVDA        NVDA        NVDA       NVDA
Date
2025-11-20  196.000000  179.850006  195.949997  343504800
2025-11-21  184.559998  172.929993  181.240005  346926200
2025-11-24  183.500000  176.479996  179.490005  256618300
2025-11-25  178.160004  169.550003  174.910004  320600300
2025-11-26  182.910004  178.240005  181.630005  183852000
```



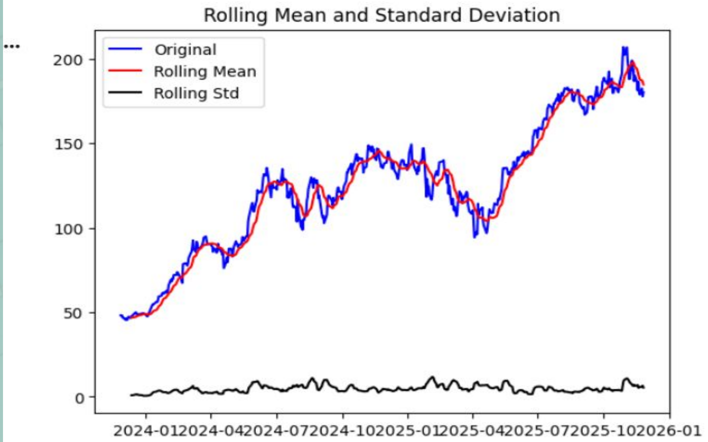
Test if NVIDIA stock price series is stationary or not?

Stock Price Over time :

- Rolling mean and rolling standard deviation change over time
- Strong upward trend → non-stationary behavior
- The p-value is 0.99 >> 0.05
- fail to reject the null hypothesis
- Conclusion : the series is non-stationary

Action :

- need to transform to stationary form before modeling



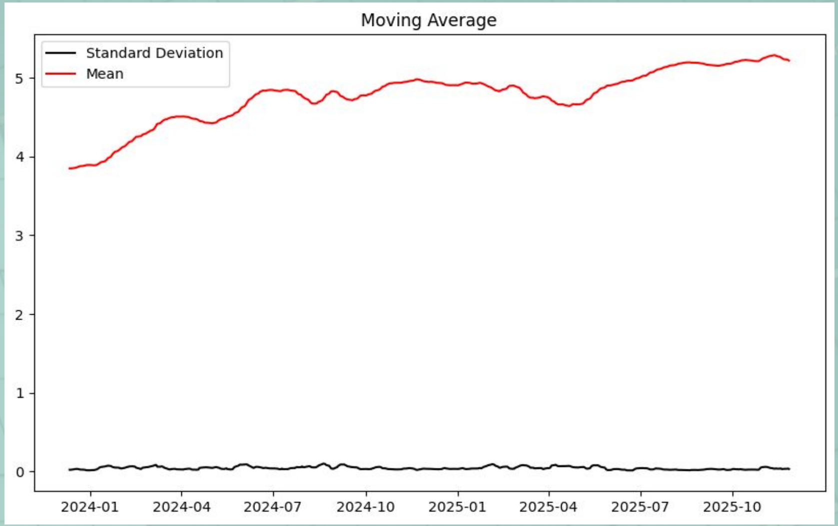

```
Results of dickey fuller test
Test Statistics                  -1.514419
p-value                           0.526367
No. of lags used                  4.000000
Number of observations used     498.000000
critical value (1%)              -3.443549
critical value (5%)              -2.867361
critical value (10%)             -2.569870
dtype: float64
```

```python
from statsmodels.tsa.stattools import adfuller

result = adfuller(target)
print('ADF Statistic: ', result[0])
print('p-value:', result[1])
```

```
ADF Statistic:  -1.5144190504473505
p-value: 0.5263667651818924
```

Why Log Transform the Data

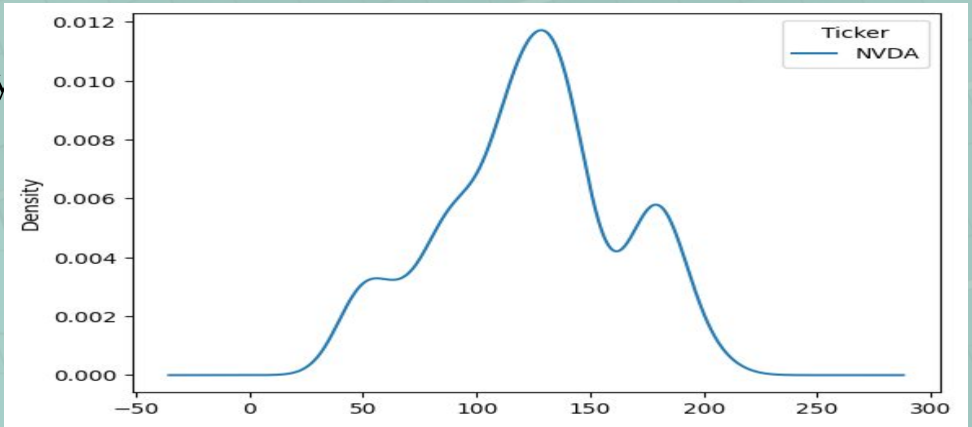
- **Stabilize Volatility:** It ensures that price swings (volatility) are consistent over time, regardless of the stock's absolute price.
- **Normalize Scale:** It converts exponential price growth into a more linear trend, which is better suited for linear models like ARIMA.
- **Focus on Returns:** It allows the model to analyze percentage changes (returns), which are more relevant in finance than absolute dollar changes.
- **Meet ARIMA Assumptions:** It helps satisfy the statistical assumption that the data's variance remains constant Homoscedasticity.



Understanding the Price Distribution

1. Price Density Plot

- Shows what price has occurred more frequently
- Bimodal curve - not normally distributed
- 1st big peak around 120-150
- 2nd smaller peak at 170-200- showing the recent growth and AI boom

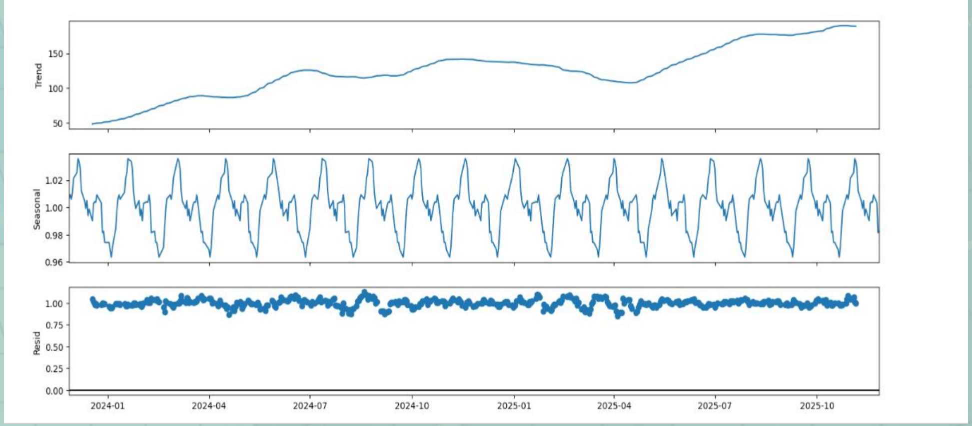


2. Trend & Seasonal Pattern

The graph is broken down into:

- Trend: long-term direction (upward)
- Seasonal: Spike and trough pattern
- Fluctuates between 1.00 to 1.02- typically add or subtract about 2% from the price
- Noise: random fluctuations around 1.00 mark

```
decomposition = seasonal_decompose(df['Close'], model='additive', period=30)
result = seasonal_decompose(df['Close'], model='multiplicative', period = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
```



Preparing the Time Series for ARIMA

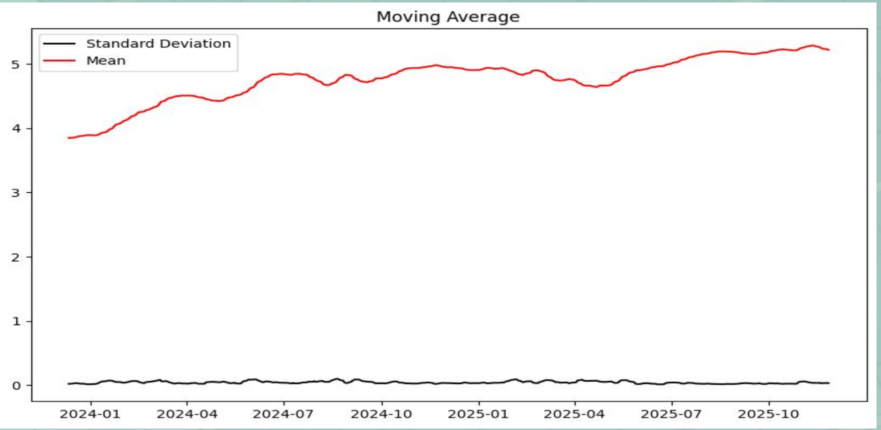
Why is it important :

- ARIMA only works on data that is stationary
- Stationary = no trend, no changing variance, no seasonality
- The raw NVIDIA price was not stationary (p-value = 0.99)

```python
#taking log transformation of closing price
from pylab import rcParams
rcParams['figure.figsize'] = 10, 6
df_log = np.log(df_close)
moving_avg = df_log.rolling(12).mean()
std_dev = df_log.rolling(12).std()
plt.legend(loc='best')
plt.title('Moving Average')
plt.plot(std_dev, color ="black", label = "Standard Deviation")
plt.plot(moving_avg, color="red", label = "Mean")
plt.legend()
plt.show()
```

What We Did

- Applied log transform to reduce volatility
- Used rolling mean visualization to inspect trend
- Stabilized variance and reduced trend effect



DEVELOP ARIMA MODEL

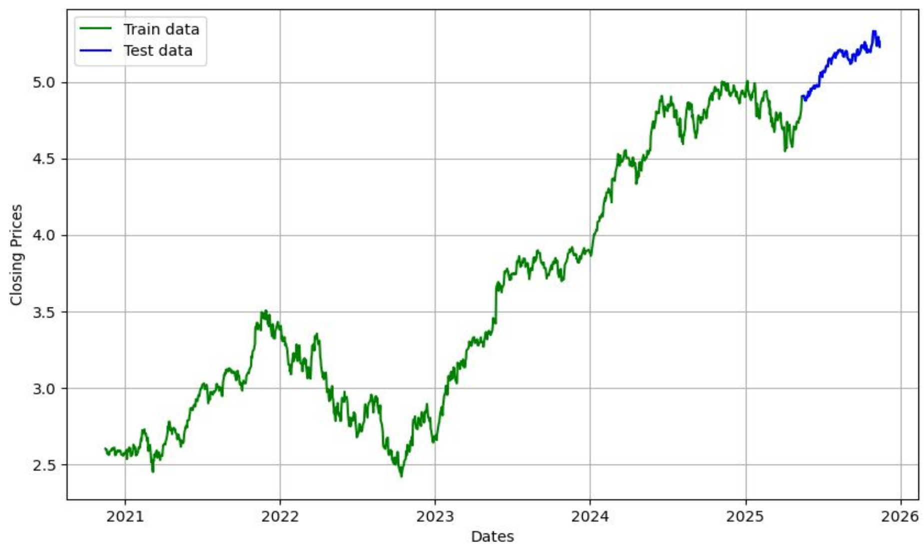
ARIMA :

- Split to test the model on unseen future data
- Prevents overfitting
- train on history → predict future

After log transformation, the data was divided into:

- 90% Training Data (fit the ARIMA model)
- 10% Testing Data (used to evaluate predictions)

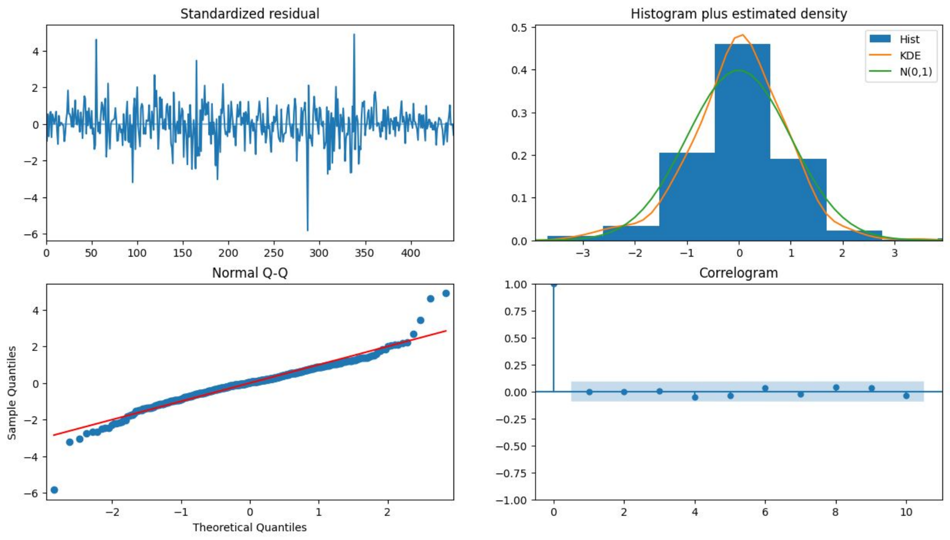
```
#split data into train and training set
train_data, test_data = df_log[3:int(len(df_log)*0.9)], df_log[int(len(df_log)*0.9):]
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Closing Prices')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test_data, 'blue', label='Test data')
plt.legend()
```



ARIMA Model Selection & Results

ARIMA :

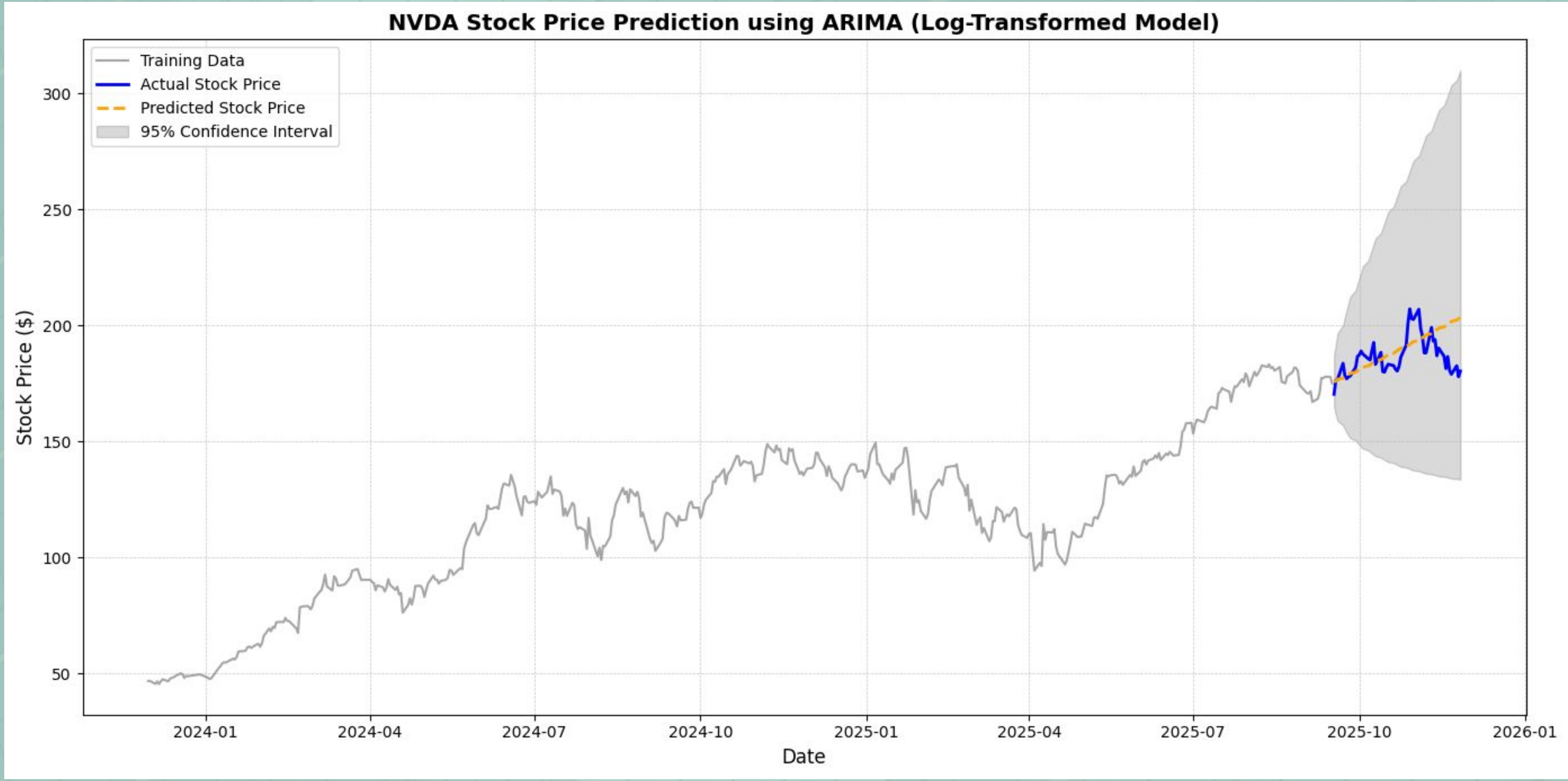
```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-1785.697, Time=0.15 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-1788.641, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-1787.935, Time=0.18 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-1784.107, Time=0.07 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-1789.952, Time=0.16 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-1790.270, Time=0.16 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-1789.451, Time=1.07 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-1788.811, Time=0.40 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=-1788.084, Time=0.12 sec

Best model:  ARIMA(3,1,0)(0,0,0)[0] intercept
Total fit time: 2.411 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  449
Model:               SARIMAX(3, 1, 0)   Log Likelihood                 900.135
Date:                Fri, 28 Nov 2025   AIC                          -1790.270
Time:                        02:56:26   BIC                          -1769.746
Sample:                             0   HQIC                         -1782.179
                                - 449
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0032      0.002      2.041      0.041      0.000       0.006
ar.L1         -0.0888      0.039     -2.267      0.023     -0.166      -0.012
ar.L2          0.0793      0.045      1.781      0.075     -0.008       0.166
ar.L3         -0.0720      0.046     -1.550      0.121     -0.163       0.019
sigma2         0.0011   4.16e-05     25.306      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               405.46
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):               0.93   Skew:                            -0.20
Prob(H) (two-sided):                  0.65   Kurtosis:                         7.64
===================================================================================
```



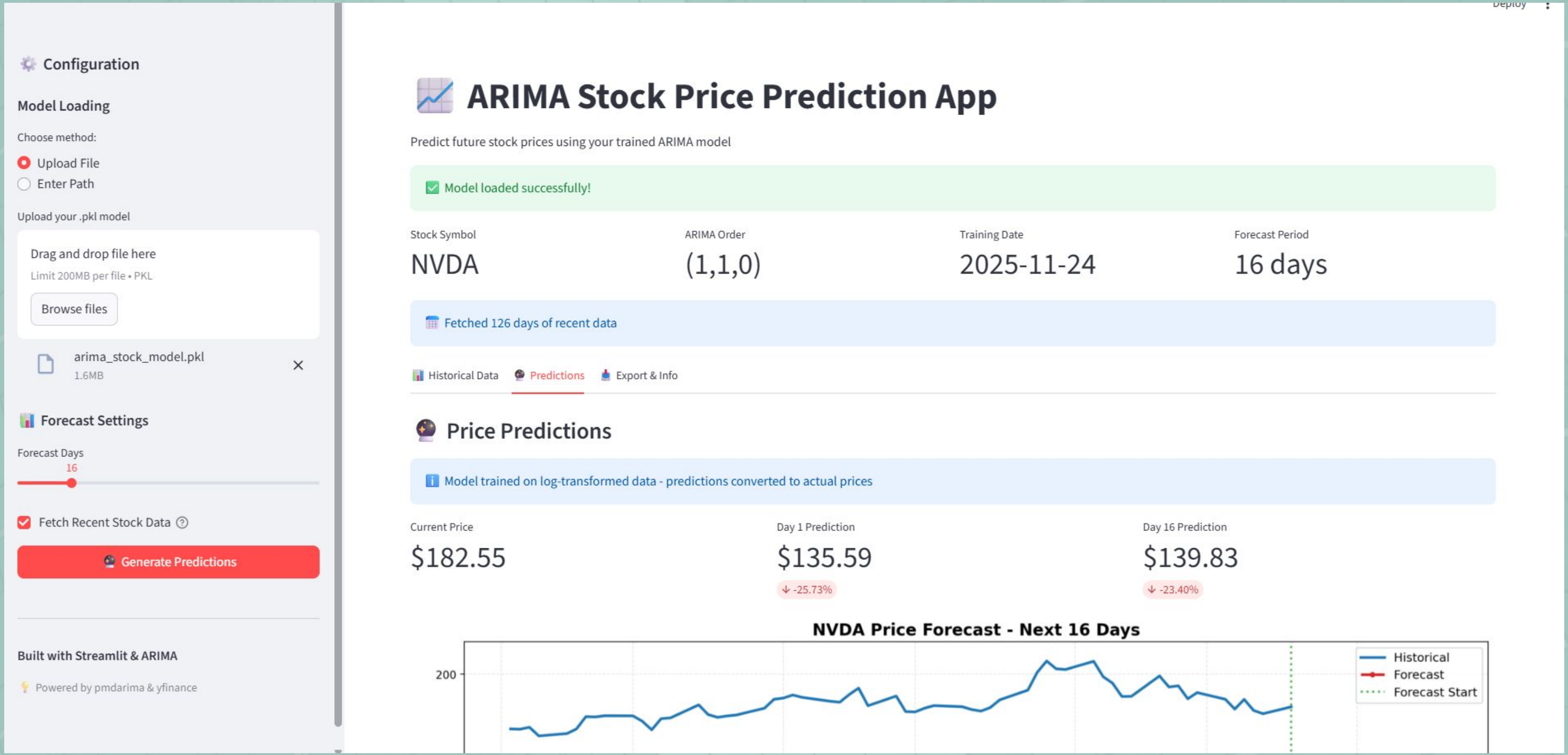

```python
model_autoARIMA = auto_arima(train_data, test='adf', max_p=3, max_q=3, seasonal=False)
```

PREDICTING STOCK PRICES

ARIMA :



forecasted about 60 days


Deploy
Configuration
Model Loading
Choose method:
Upload File
Enter Path
Upload your .pkl model
Drag and drop file here
Limit 200MB per file • PKL
Browse files
arima_stock_model.pkl
1.6MB
Forecast Settings
Forecast Days
16
Fetch Recent Stock Data
Generate Predictions
Built with Streamlit & ARIMA
Powered by pmdarima & yfinance
ARIMA Stock Price Prediction App
Predict future stock prices using your trained ARIMA model
Model loaded successfully!
Stock Symbol
NVDA
ARIMA Order
(1,1,0)
Training Date
2025-11-24
Forecast Period
16 days
Fetched 126 days of recent data
Historical Data
Predictions
Export & Info
Price Predictions
Model trained on log-transformed data - predictions converted to actual prices
Current Price
$182.55
Day 1 Prediction
$135.59
-25.73%
Day 16 Prediction
$139.83
-23.40%
NVDA Price Forecast - Next 16 Days
200
Historical
Forecast
Forecast Start

THE OPPORTUNITY

"No model can perfectly predict stock prices because markets are influenced by external events like news, earnings reports, and macroeconomic factors. However, for a statistical baseline model, this ARIMA approach performs well and is appropriate for short-term forecasting."

Key learning 1

Data Preparation is Critical because Log transformation helped stabilize variance and Raw stock prices contain trend and volatility. BE CAREFUL WHILE CLEANING

Key learning 2

ARIMA is a Strong Baseline for Short-Term Forecasting

Key learning 3

Model Logic Must Match Business Context

Thank you