

Time Series Analysis I: Models & Forecasting

Prepared by Dr. Sumaya Alkubaisi

FNT633
Machine Learning in Finance

Lecture 9



Learning Goals

By the end of this lecture, you will be able to:

- Understand time series components
- Learn AR, MA, ARMA, ARIMA, SARIMA models
- Apply preprocessing techniques
- Use forecasting metrics (MAE, RMSE, MAPE)
- Analyze a synthetic stock price example

Time Series

- A time series is a sequence of data points (observations) collected and recorded (measured) over time in order.

Financial Examples:

- S&P 500 daily returns: Measures how the stock market index changes each day.
- EUR/USD exchange rate: Shows how the value of the euro changes against the dollar over time.
- Bitcoin price: Tracks Bitcoin's price movements daily or hourly.
- Monthly CPI: Measures inflation by tracking changes in consumer prices every month.
- Quarterly GDP: Shows the economic growth of a country every three months.

Components of a Time Series

- **Trend:** The long-term direction of the data (upward, downward, or flat).

Example

A company's revenue increases gradually every year; long-term upward trend.

- **Seasonality:** Patterns that repeat at regular intervals (daily, monthly, yearly).

Example

Ice-cream sales go up every summer and drop every winter; repeating seasonal pattern.

- **Cyclic:** Longer, irregular cycles caused by economic or market conditions.

Example

Housing prices rise and fall over several years due to economic cycles; irregular ups/downs.

- **Noise:** Random fluctuations that cannot be explained by any pattern.

Example

Sudden spikes in stock prices caused by unexpected news; random movements with no pattern.

Chapter
9

Stationarity (Concept)

- A time series is stationary when its statistical properties do not change over time.

This means:

- The mean stays constant
- The variance stays constant
- No long-term trend
- No seasonal patterns

Stationary data is easier to model and is required for many forecasting methods.

Stationarity importance:

- Models like AR, MA, ARMA, and ARIMA work only when the data is stationary.
- When the series is stationary, the model's patterns stay stable, which makes forecasting more accurate and reliable.

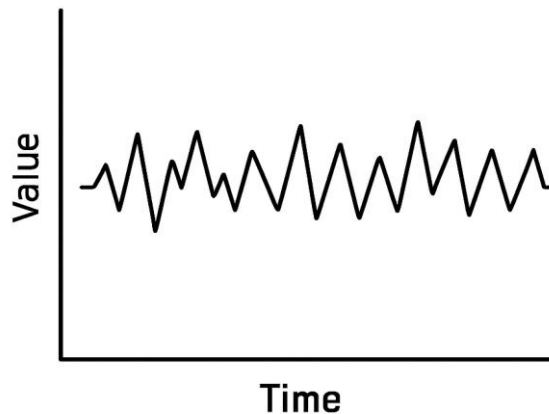
Chapter
9

Visual Intuition

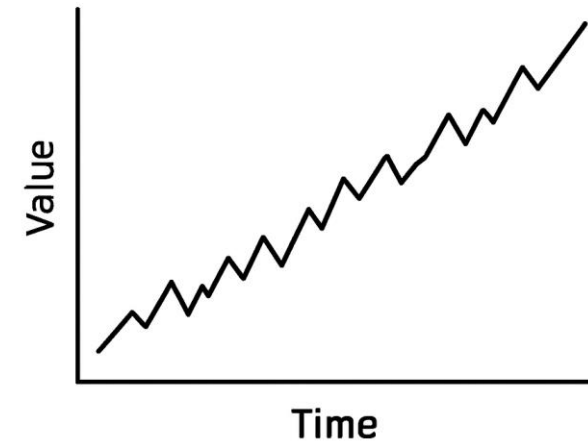
Non-stationary: The line keeps going up or down over time: has a trend.

Stationary: The line moves up and down around a flat average: no trend, stable behavior.

Stationary



Non-stationary



Chapter
9

Forecasting Foundations

- Forecasting means using past/historical data to predict what will happen in the future.
Example: using last year's stock prices to predict tomorrow's price.

One-Step & Multi-Step Forecasting**One-step ahead:**

- You predict only the next point ($t+1$).
- Most accurate because it's close to the known data.

Multi-step ahead:

- You predict several future points ($t+1, t+2, \dots t+n$).
- The further you try to predict, the more uncertainty and error you get.

Chapter
9

Rolling & Expanding Window

Rolling window:

- We use a fixed-size window of recent data (e.g., last 30 days).
- As time moves forward, the window slides forward.
- Always same size.

Expanding window:

- We start with a small window, and it keeps growing as more data arrives.
- Uses all past data up to the current point.

1. Collect data

Get historical financial data (prices, returns, rates).

2. Clean & preprocess

Handle missing values, remove outliers, format correctly.

3. Test stationarity

Check if the series is stable (ADF test, visualization).

4. Fit model

Apply models like AR, MA, ARIMA, SARIMA.

5. Forecast

Predict future values (one-step or multi-step).

6. Evaluate errors

Compare predictions to actual values using metrics (MSE, RMSE, MAE).

Handling Missing Values:

- Fill gaps using previous values (forward fill),
 - interpolate between points,
 - or remove them if there are very few.
- ❖ Interpolation in time series means estimating missing values by looking at surrounding data points.
- Linear interpolation: Connects the points with a straight line and fills missing values along it.
 - Other types: Can use polynomial, spline, or more advanced methods depending on the pattern.

Smoothing & Moving Averages:

- Reduce short-term fluctuations in the data to make trends easier to see
- Use simple or exponential moving averages to reduce noise and highlight trends.

Differencing: Subtract previous value from current ($\Delta y = y(t) - y(t-1)$) to remove trends and stabilize the series.

Scaling & Normalization: Apply z-score or min-max scaling to adjust value ranges, which some models require.

ADF Test: Checks stationarity. If p-value < 0.05 , the series is stationary; if > 0.05 , it's non-stationary.

Classical Time Series Models - Autoregressive (AR) Model

- Predicts future value using past values.
- Formula: $y(t) = c + \phi_1 y(t-1) + \dots + \phi_p y(t-p) + \varepsilon(t)$
- p = number of past observations considered.
- So AR compute Next value = starting point + effect of past values + tiny unpredictable changes

Example: AR(2) uses the previous two values, $y(t-1)$ and $y(t-2)$, to predict $y(t)$.

Visual Understanding/Intuition: Past values (Lagged) strongly influence the next value.

Classical Time Series Models - Moving Average (MA) Model

- Predicts the next value based on past errors (residuals), not the values themselves.
- Formula: $y(t) = \mu + \theta_1 \varepsilon(t-1) + \dots + \theta_q \varepsilon(t-q)$
- q = number of past error terms used.
- So MA prediction depends on the baseline level + the last two mistakes the model made.

Example: MA(2) uses the previous two error terms, $\varepsilon(t-1)$ and $\varepsilon(t-2)$, to predict $y(t)$.

Visual Understanding/Intuition: Large errors today affect future predictions.

In conclusion: AR looks at past values, MA looks at past errors.

Classical Time Series Models – ARMA Model

ARMA combines both AR and MA:

- AR part: uses past values.
- MA part: uses past errors.
- It can model more complex patterns, but it requires the data to be stationary.

ARMA(p, q) Use Cases

ARMA(p, q) is commonly used for:

- Financial returns
- Volatility patterns
- Short-term fluctuations in stationary time series

It works well when the series has no clear trend but has some autocorrelation structure.

Where: p = number of past values used (from the AR part) and q = number of past errors used (from the MA part)

Strengths

- Simple to use: The formulas and concepts are straightforward.
- Easy to interpret: You can clearly see how past values (AR) or past errors (MA) affect predictions.
- Good for short-term forecasting: They work well when the data has short memory and stable patterns.

Limitations

- Require stationarity: AR, MA, and ARMA only work well if the data has no trend and stable variance.
- Not suitable for long-term trends: They cannot capture long-term growth or downward movement.
- Limited complexity: They cannot model seasonal patterns or non-linear behavior unless extended.

Advanced Time Series Models - ARIMA Model

- ARIMA(p, d, q) combines three parts:
 - p: number of AR terms (past values)
 - d: number of differencing steps to make the data stationary
 - q: number of MA terms (past errors)
- It extends ARMA by adding differencing.
- What “I” Means (Integrated)
 - The “I” in ARIMA stands for Integrated, which simply means: you apply differencing to remove trends and make the series stationary.

Advanced Time Series Models - ARIMA Model

ARIMA Workflow

1. Check stationarity of the series
2. Apply differencing until it becomes stationary (this gives you d)
3. Select p , d , q using tools like ACF/PACF
4. Fit the model to the data
5. Evaluate the forecast accuracy

Advanced Time Series Models - SARIMA (Seasonal ARIMA)

- SARIMA adds seasonal terms to ARIMA.
- **SARIMA(p, d, q)(P, D, Q)_s**: adds AR, MA, and differencing for both regular patterns and repeating seasonal patterns.

SARIMA(p, d, q) (P, D, Q)_s has two parts:

1.The first part (p, d, q)

These are the normal ARIMA terms

Advanced Time Series Models - SARIMA (Seasonal ARIMA)

SARIMA(p, d, q) (P, D, Q)_s has two parts:

2. The seasonal part (P, D, Q)

These do the same thing, but for seasonal patterns:

P: seasonal AR: uses past values from one season ago
(e.g., last year's same month)

D: seasonal differencing: removes seasonal trends
(e.g., $y(t) - y(t-12)$)

Q: seasonal MA: uses past seasonal errors
(e.g., error from one year ago)

s: the length of the season

12: monthly data with yearly seasonality

7: daily data with weekly seasonality

4: quarterly data

When to use ARIMA vs SARIMA

ARIMA

- Use ARIMA when the data does NOT have seasonality (non-seasonal data).
- It handles trends and short-term patterns, but not repeating cycles.

SARIMA

- Use SARIMA when the data HAS clear seasonal patterns, like
 - monthly patterns ($s = 12$)
 - weekly patterns ($s = 7$)
 - quarterly patterns ($s = 4$)

Evaluation Metrics

MAE – Mean Absolute Error

- MAE measures the average size of the errors without considering direction.
- It tells you, on average, how far your predictions are from the actual values.

RMSE – Root Mean Square Error

- RMSE also measures error size, but it penalizes large errors more because it squares the errors first.
- Useful when big mistakes are especially bad.

MAPE – Mean Absolute Percentage Error

- MAPE measures the error as a percentage.
- It is good for data where the scale matters, like stock prices or sales, because it shows how big the error is relative to the value.

Evaluation Metrics

Which Metric for Finance?

- Stock prices: RMSE (big errors matter more)
- Returns: MAE (simple, stable, no percentage issues)
- Percentage-based forecasts: MAPE (shows error relative to value)

Case Study: Synthetic Daily Stock Price Dataset

Dataset Description

- Synthetic daily stock prices for ~2 years (500 days)
- Generated with a linear upward trend + added random noise
- Creates realistic movement similar to real stocks:
- gradual growth
- small daily fluctuations

Time Series Forecasting using (ARIMA)

- Predict future values of a time series by modeling trends and past patterns using ARIMA

Case Study: Synthetic Daily Stock Price Dataset

Libraries used:

- pandas
- numpy
- matplotlib
- seaborn
- statsmodels: A Python library for statistical modeling, including time series analysis (ARIMA, SARIMA, ADF test, etc.). It allows fitting classical econometric models.

Python Lab: Time Series Forecasting (ARIMA)

#Create a Synthetic Daily Stock Price Dataset

```
import numpy as np
```

```
import pandas as pd
```

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

```
import seaborn as sns
```

```
np.random.seed(42)
```

```
# Generate 500 days of synthetic stock prices
```

```
days = 500
```

```
time_index = pd.date_range(start="2020-01-01", periods=days)
```

```
# Create trend + noise
```

```
trend = np.linspace(100, 150, days)
```

```
noise = np.random.normal(0, 2, days)
```

```
prices = trend + noise
```

```
df = pd.DataFrame({"Date": time_index, "Price": prices})
```

```
df.set_index("Date", inplace=True)
```

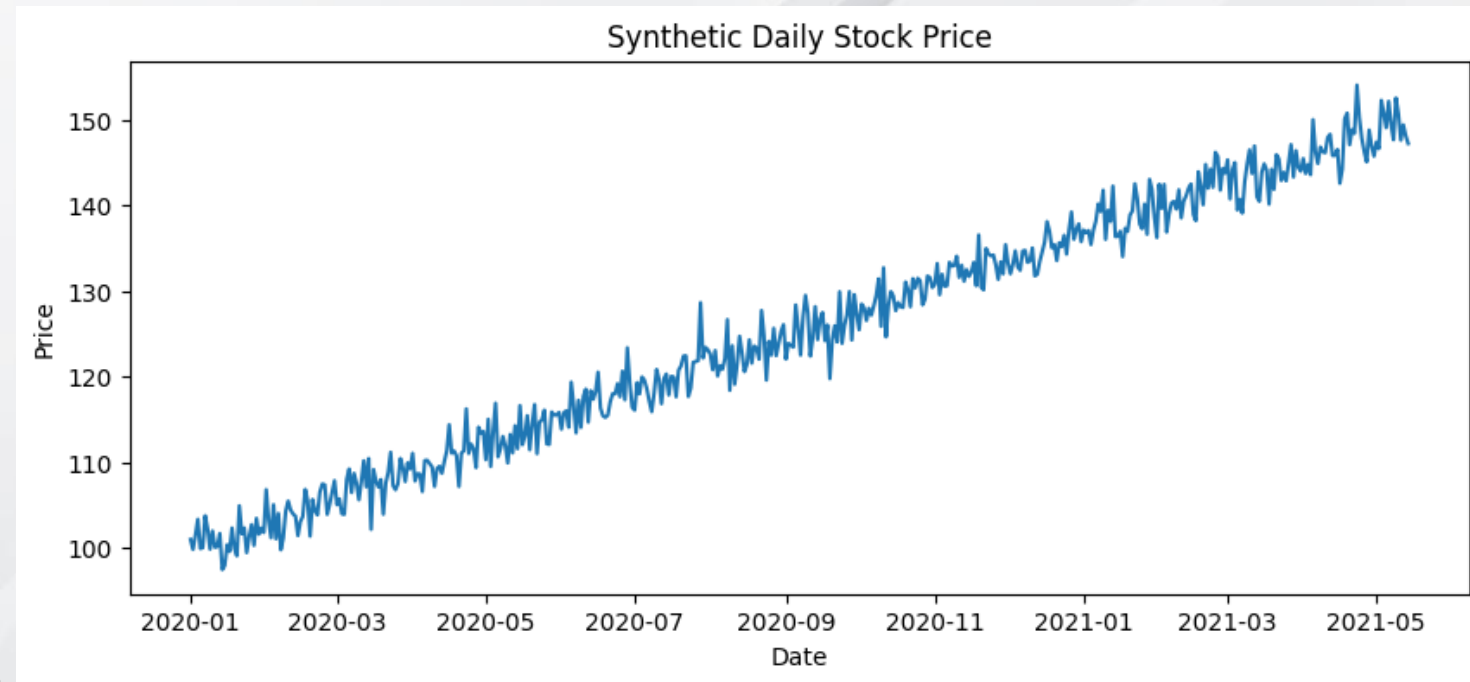
```
df.head()
```

Python Lab: Time Series Forecasting (ARIMA)

#Plot the Time Series

```
plt.figure(figsize=(10,4))  
plt.plot(df["Price"])  
plt.title("Synthetic Daily Stock Price")  
plt.xlabel("Date")  
plt.ylabel("Price")  
plt.show()
```

❖ The series shows a clear upward trend overtime, so its mean is not constant. Because a stationary series must have a constant mean, this strong trend indicates that the data is non-stationary.



Python Lab: Time Series Forecasting (ARIMA)

#ADF Test (Stationarity Check)

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

```
result = adfuller(df["Price"])  
print("ADF Statistic:", result[0])  
print("p-value:", result[1])
```

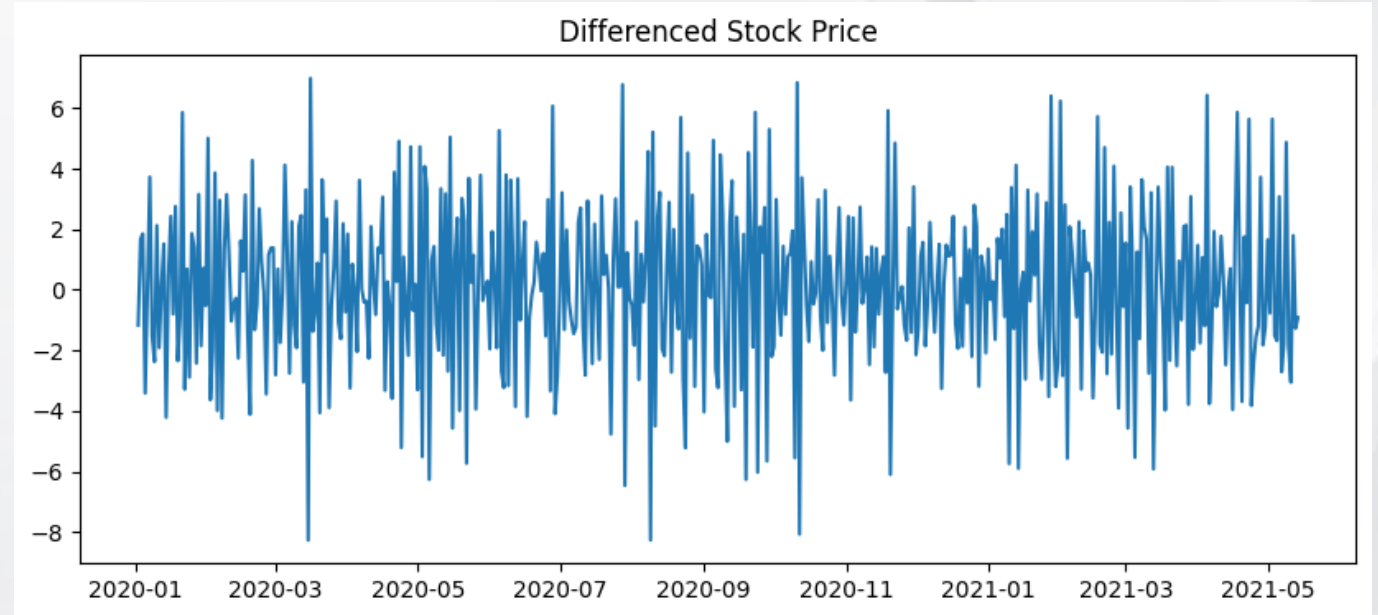
- **p-value = 0.903:** much greater than 0.05. (expected because of trend)
- ❖ The series is non-stationary. It has a trend or changing mean, so you would need to difference it before using ARIMA.

Python Lab: Time Series Forecasting (ARIMA)

#Differencing (to Make it Stationary)

```
df["Price_diff"] = df["Price"].diff()
```

```
plt.figure(figsize=(10,4))  
plt.plot(df["Price_diff"])  
plt.title("Differenced Stock Price")  
plt.show()
```



- After differencing the original series, the values fluctuate around a constant mean of zero with stable variance.

This transformation makes the series visually stationary and suitable for time series modeling.

Python Lab: Time Series Forecasting (ARIMA)

```
#Fit ARIMA Model - We use ARIMA(1,1,1)  
#Understand and fit ARIMA on full data (teaching purposes).
```

```
from statsmodels.tsa.arima.model import ARIMA
```

```
# Remove NaN after differencing  
ts_data = df["Price"].dropna()
```

```
model = ARIMA(ts_data, order=(1,1,1))  
model_fit = model.fit()  
print(model_fit.summary())
```

❖ In **ARIMA(1,1,1)**:

p = 1: Uses 1 past value (AR term)

d = 1: 1 differencing step to make the series stationary

q = 1: Uses 1 past error (MA term)

Python Lab: Time Series Forecasting (ARIMA)

```
#Forecast Next 30 Days
```

```
forecast_steps = 30  
forecast = model_fit.forecast(steps=forecast_steps)
```

```
plt.figure(figsize=(10,4))  
plt.plot(df["Price"], label="Historical")
```

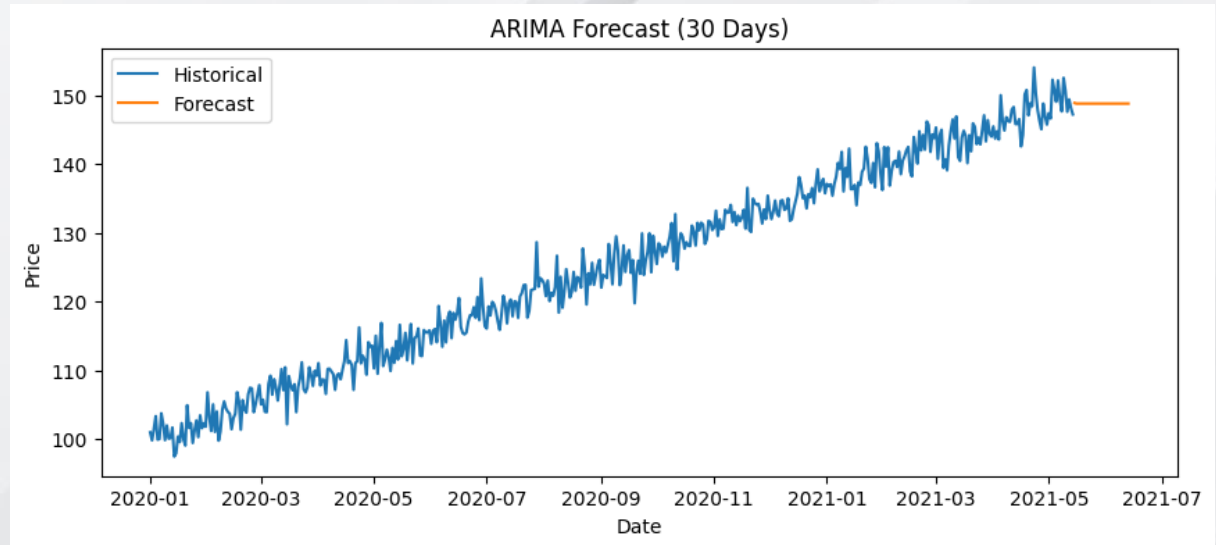
```
# Generate forecast dates  
forecast_dates = pd.date_range(start=df.index[-1] + pd.Timedelta(days=1), periods=forecast_steps)
```

```
plt.plot(forecast_dates, forecast, label="Forecast")  
plt.title("ARIMA Forecast (30 Days)")  
plt.xlabel("Date")  
plt.ylabel("Price")  
plt.legend()  
plt.show()
```

Python Lab: Time Series Forecasting (ARIMA)

#Forecast Next 30 Days - Output

- Historical data shows a strong upward trend ($\sim 100 \rightarrow \sim 150$).
- 30-day forecast is flat (~ 149.5), not continuing the trend.
- Flat forecast occurs because ARIMA models the stationary series, removing trends.
- Implication: Model captures short-term fluctuations but doesn't extrapolate strong trends.
- Visual check is helpful, but evaluation metrics are still needed to quantify accuracy.



Python Lab: Time Series Forecasting (ARIMA)

#Train/Test Split + Forecasting Accuracy

#Split data: forecast on unseen data, evaluate (validation & tuning)

80% training, 20% testing

```
train_size = int(len(df) * 0.8)
```

```
train, test = df["Price"][:train_size], df["Price"][train_size:]
```

Fit on training only

```
model = ARIMA(train, order=(1,1,1))
```

```
model_fit = model.fit()
```

Forecast length = test set length (Purpose: Compare predicted values with actual test data to measure

#accuracy using metrics like MAE, RMSE, or MAPE.)

```
predictions = model_fit.forecast(steps=len(test))
```

Python Lab: Time Series Forecasting (ARIMA)

#Evaluation Metrics

```
from sklearn.metrics import mean_absolute_error, mean_squared_error  
import numpy as np
```

```
mae = mean_absolute_error(test, predictions)  
rmse = np.sqrt(mean_squared_error(test, predictions))
```

```
print("MAE:", mae)  
print("RMSE:", rmse)
```

Python Lab: Time Series Forecasting (ARIMA)

#Evaluation Metrics - Output

Forecast Accuracy

MAE $\approx$ 4.9: Predictions are on average $\sim$ 5 units off

RMSE $\approx$ 5.85: Larger deviations slightly increase the error

Interpretation: Errors are small relative to stock prices ($\sim$ 100–150) $\rightarrow$ short-term forecasts are reasonably accurate

Key Idea:

Smaller values = better predictions

Compare errors to the scale of the data to see if they are acceptable

Example: Stock prices $\sim$ 100–150 $\rightarrow$ MAE $\sim$ 5 is reasonably accurate

Python Lab: Time Series Forecasting (ARIMA)

#Plot Forecast vs Actual

```
plt.figure(figsize=(10,4))  
plt.plot(test.index, test, label="Actual")  
plt.plot(test.index, predictions, label="Predicted")  
plt.title("Forecast vs Actual (Test Data)")  
plt.xlabel("Date")  
plt.ylabel("Price")  
plt.legend()  
plt.show()
```

Python Lab: Time Series Forecasting (ARIMA)

#Plot Forecast vs Actual - Output

- Actual Data (Blue): Feb → May 2021, prices rise ~137.5 → 152.5 with daily fluctuations.
- Predicted Data (Orange): Flat line near 140, failing to capture trend or volatility.
- Model Performance:
 - Does not capture upward trend
 - Ignores daily fluctuations → high errors
 - Forecast is inadequate for this strongly trending, volatile series

In conclusion: Visual comparison shows the model underperforms when trends and volatility are strong.

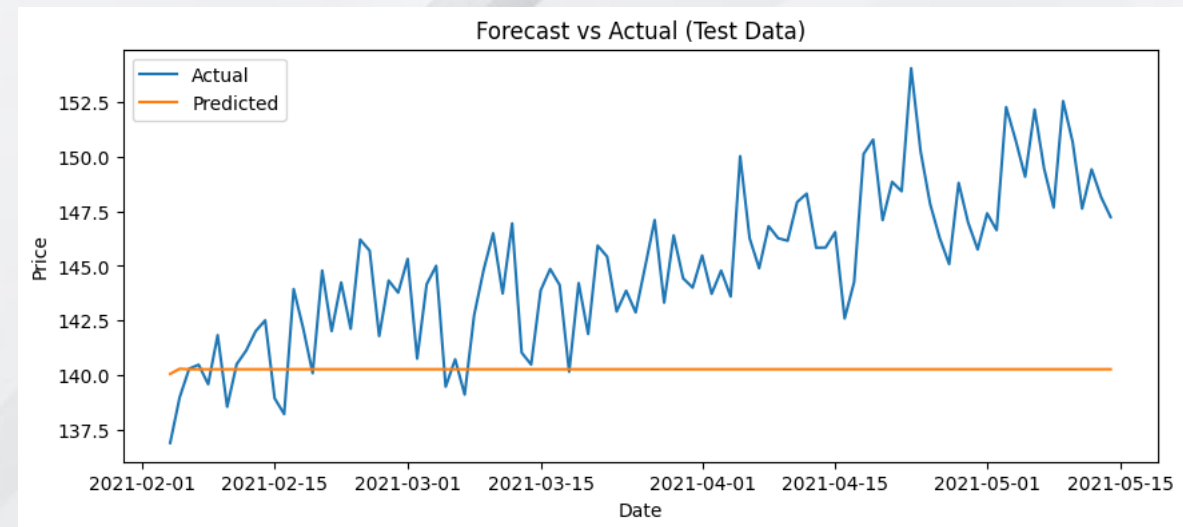
Forecast vs Actual – Model Suitability

Observation: ARIMA(1,1,1) produces a flat forecast: fails to capture trend and volatility.

Conclusion: Model is not suitable for strongly trending, volatile stock prices.

Recommendation:

Use trend-aware or more advanced models for better predictions.



Lecture

9

Recap

- Introduced time series data and its components: trend, seasonality, cycles, noise.
- Explained why forecasting is important in finance (stocks, volatility, demand, returns).
- Covered stationarity, how to check it, and why models need it.
- Applied transformations like log scaling and differencing to achieve stationarity.
- Reviewed AR, MA, and ARIMA models and when each is useful in financial forecasting.
- Evaluated forecasts using a proper train/test split and metrics (MAE, RMSE, MAPE).
- Lab demo: created synthetic stock data, made it stationary, fitted ARIMA, forecasted, and compared predictions to actual values.

- López de Prado, M. Advances in Financial Machine Learning. Wiley.
- Klaas, J. Machine Learning for Finance: Principles and Practice for Financial Insiders. Packt Publishing.
- Coursera – Machine Learning & Time Series Courses
(Used for structure, pedagogical flow, and conceptual clarity.)
- Hyndman, R.J. & Athanasopoulos, G. Forecasting: Principles and Practice. OTexts.
- Box, G.E.P, Jenkins, G.M., & Reinsel, G.C. Time Series Analysis: Forecasting and Control.
- Brownlee, J. Introduction to Time Series Forecasting. Machine Learning Mastery.