



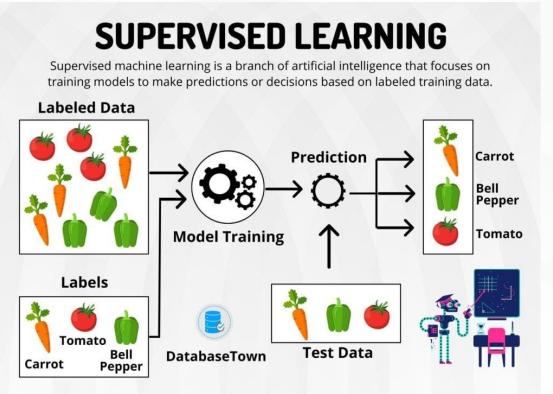
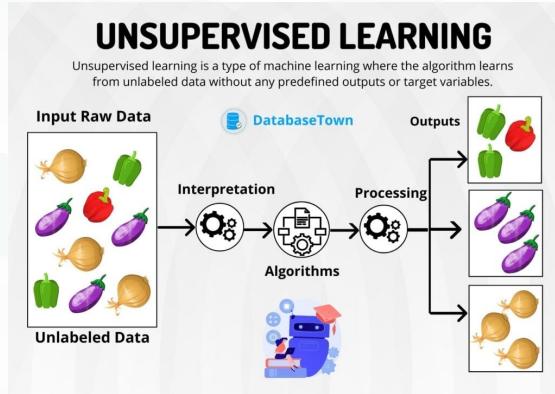
Time Series Analysis

Kalbe Digital University

Outline

- 1 What is Time Series Data
- 2 Framework and Toolkit
- 3 Visualization
- 4 Linear Regression

Kinds of Data We learn



Supervised

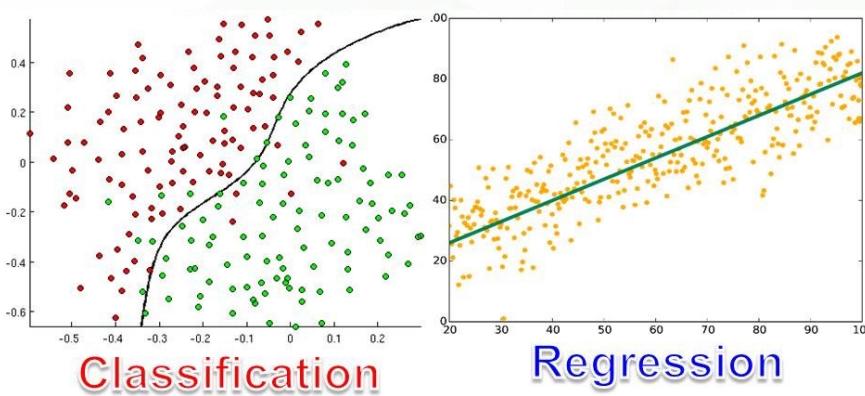
| X₁ | X₂ | X_p | Y |
|----------------------|----------------------|----------------------|----------|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Target

Un-Supervised

| X₁ | X₂ | X_p | Y |
|----------------------|----------------------|----------------------|----------|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

No Target



| No | Tahun | Jumlah Penduduk |
|----|-------|-----------------|
| 1 | 2000 | 174706 |
| 2 | 2001 | 178125 |
| 3 | 2002 | 191990 |
| 4 | 2003 | 201263 |
| 5 | 2004 | 210984 |
| 6 | 2005 | 219351 |
| 7 | 2006 | 225249 |
| 8 | 2007 | 231121 |
| 9 | 2008 | 240553 |
| 10 | 2009 | 250367 |
| 11 | 2010 | 253178 |
| 12 | 2011 | 259913 |
| 13 | 2012 | 268022 |
| 14 | 2013 | 274089 |
| 15 | 2014 | 280109 |
| 16 | 2015 | 285967 |



Time Series Only one feature

What is Time Series Data

Time series is a sequence of data points collected at successive and uniformly spaced points in time.

Format:

- Daily
- Monthly
- Yearly

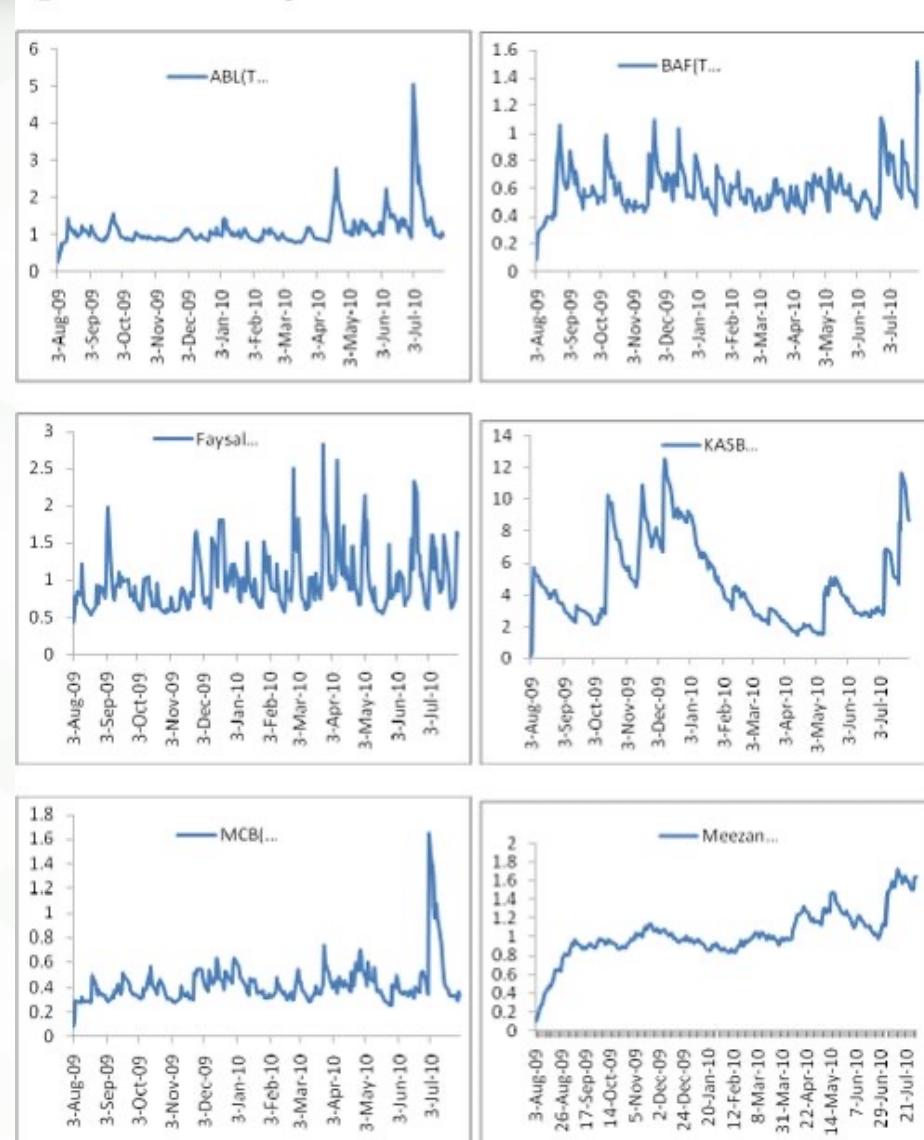
Patterns:

- Seasonality
- Cyclic Patterns
- Noise



Domains:

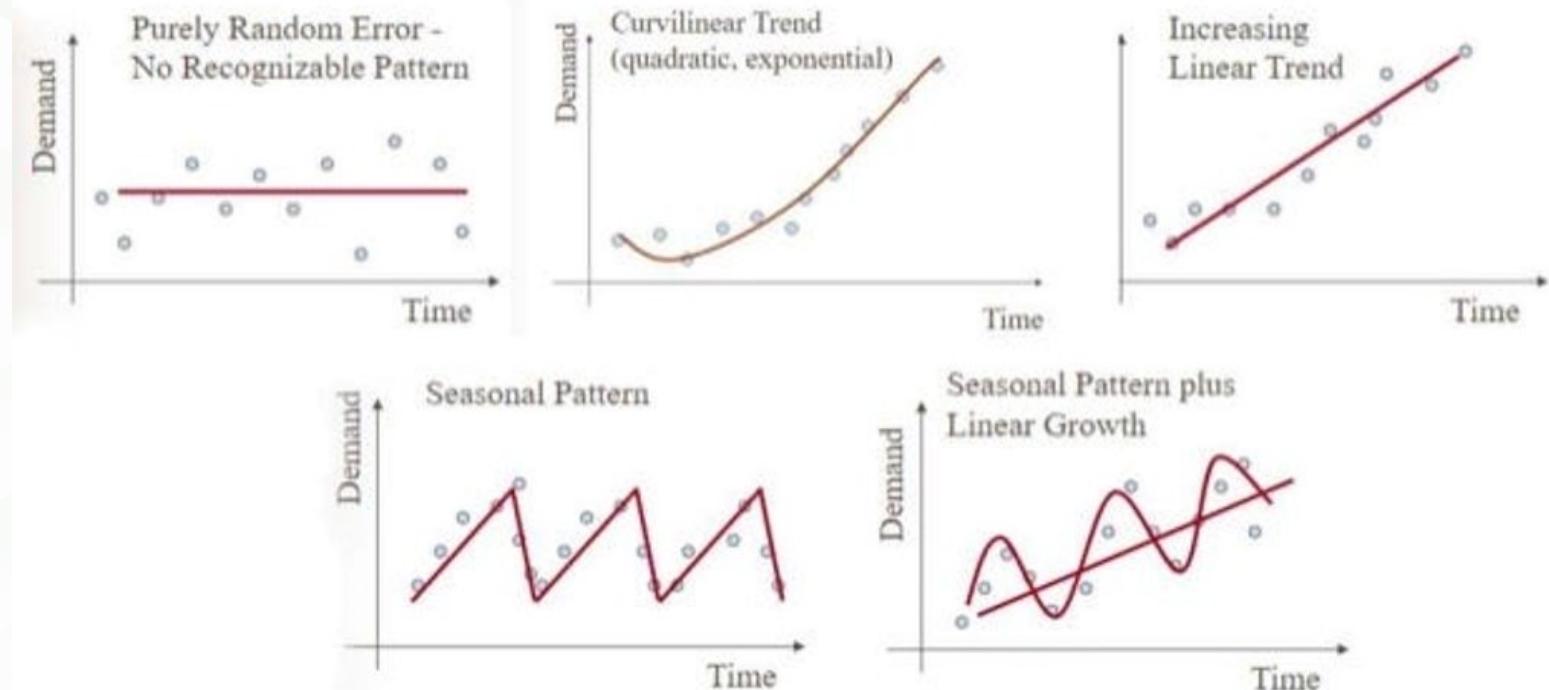
- Economics
- Medicine
- Environment
- Retail and Sales
- Technology



Patterns Inside Time Series Data

Patterns

- Trend
- Seasonality
- Cycles
- Noise
- Level
- Structural Breaks
- Autocorrelation
- Stationarity



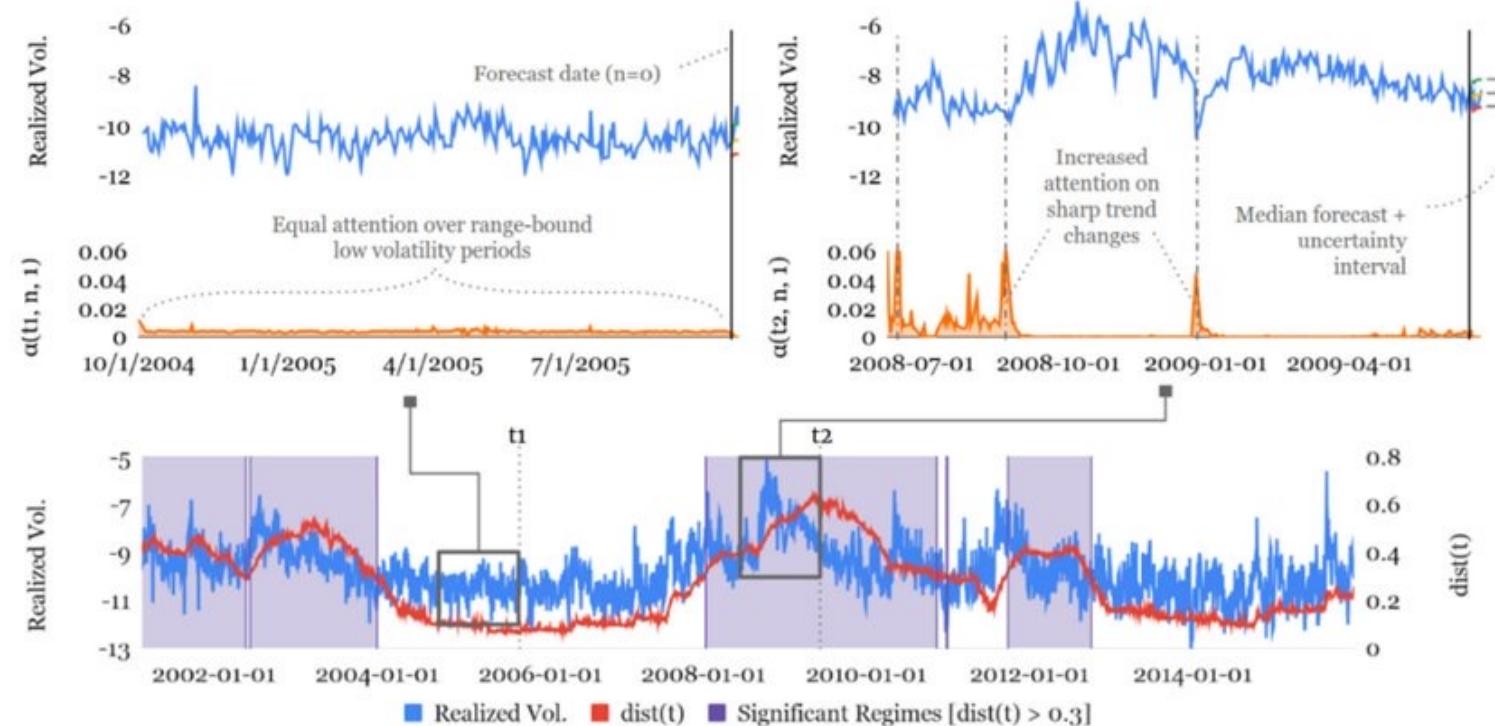
Examples in Manufacturing

Data Sources

- Production Output Data
- Energy Consumption Data
- Employee Productivity Data

The importance:

- Identification of production trends
- Identification of energy consumption patterns that can provide insights into efficiency and potential energy savings.
- Evaluation of employee performance to identify the need for training or skill development.



Time Series Analysis Scopes in EDA

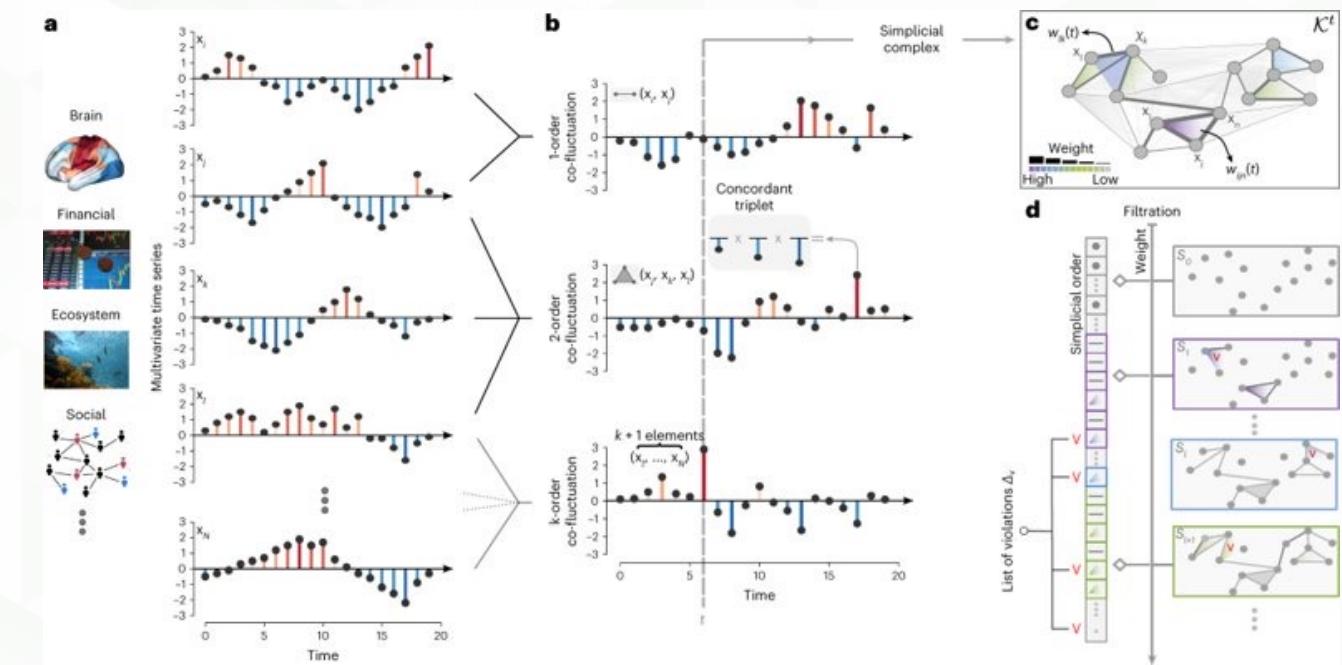
To provide a comprehensive approach to understanding chronological data, ensuring that subsequent analyses or predictions are based on solid foundational knowledge.

Purpose in EDA:

- Understanding Patterns
- Detect Anomalies
- Inform Assumptions
- Data Quality Assessment

Methods and Techniques:

- Visualization: Histogram and Density Plots
- Decomposition
- Autocorrelation
- Partial Autocorrelation
- Rolling Statistics
- Stationarity Testing



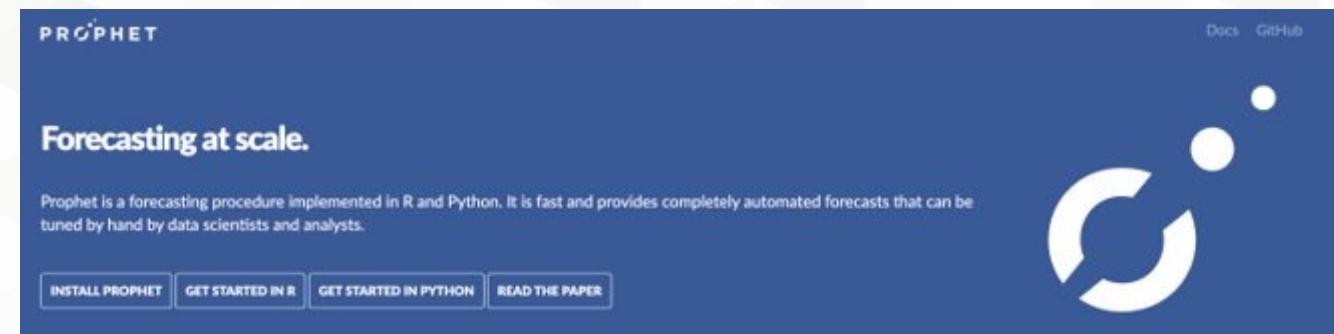
Framework and Toolkit

Methods and Techniques:

- Data Importing (CSV, Database, APIs)
- Handling Missing Values
- Time Zone Handling
- Resampling
- Deduplication
- Outlier Detection

Frameworks and Toolkits:

- Pandas
- StatsModels
- NumPy
- Scikit-Learng
- Prophet
- PyFlux



The screenshot shows the PyFlux library documentation. The top navigation bar includes 'Docs' (highlighted in blue), 'Introduction' (in black), and 'Edit on GitHub' (with a pencil icon). The main content area has a dark background with white text. It features a 'PyFlux' logo and the word 'latest'. Below this, a 'Search docs' bar is followed by a list of time series models: 'Getting Started with Time Series', 'ARIMA models', 'ARIMAX models', 'DAR models', 'Dynamic Linear regression models', 'Beta-t-EGARCH models', 'Beta-t-EGARCH in-mean models', 'Beta-t-EGARCH in-mean regression models', and 'Beta-t-EGARCH long memory models'. To the right, the 'Introduction' section is expanded, with the heading 'What is PyFlux?' and a detailed description of the library's purpose and features.

Docs ➔ Introduction

PyFlux

latest

Search docs

Getting Started with Time Series

ARIMA models

ARIMAX models

DAR models

Dynamic Linear regression models

Beta-t-EGARCH models

Beta-t-EGARCH in-mean models

Beta-t-EGARCH in-mean regression models

Beta-t-EGARCH long memory models

Introduction

What is PyFlux?

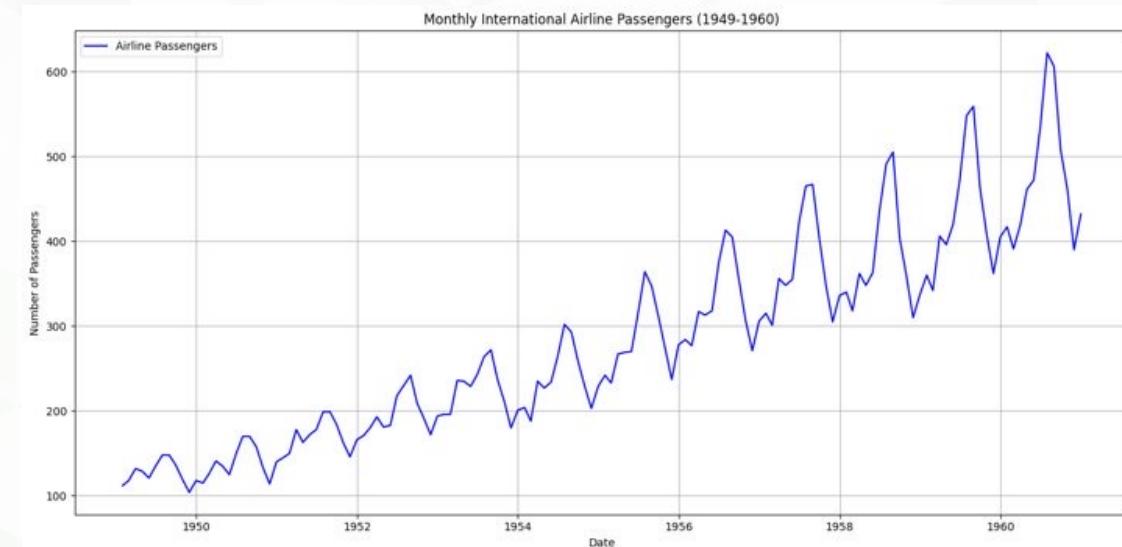
PyFlux is a library for time series analysis and prediction. Users can choose from a flexible range of modelling and inference options, and use the output for forecasting and retrospection. Users can build a full probabilistic model where the data y and latent variables (parameters) z are treated as random variables through a joint probability $p(y, z)$. The advantage of a probabilistic approach is that it gives a more complete picture of uncertainty, which is important for time series tasks such as forecasting. Alternatively, for speed, users can simply use Maximum Likelihood estimation for speed within the same unified API.

Visualization

Common Visualization

- Line Plots
- Seasonal Plots
- Scatter Plots
- Histogram and Density Plots
- Box and Whisker Plots
- Heat Maps
- Lag Plots
- ACF and PACF
- Decomposition Plots
- Rolling Statistics

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to visualize time series data using Line Plot.



Visualization

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to visualize time series data using Scatter, Seasonal, and Lag Plots.

```

# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from pandas.plotting import lag_plot

# Load the AirPassengers dataset from statsmodels
dataset = sm.datasets.get_rdataset("AirPassengers").data

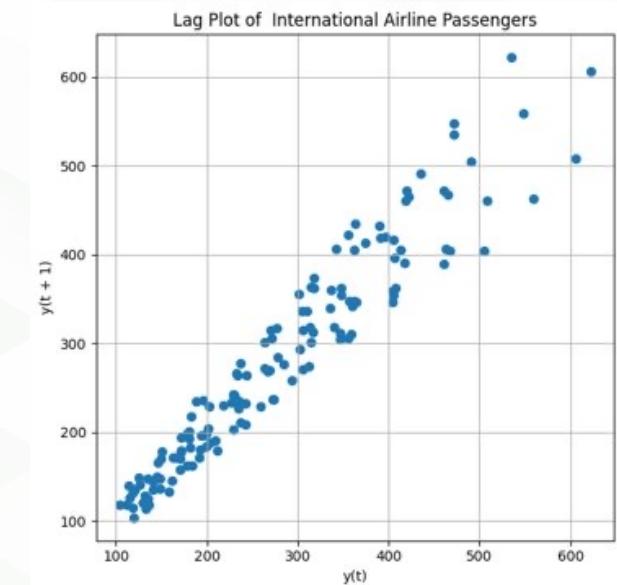
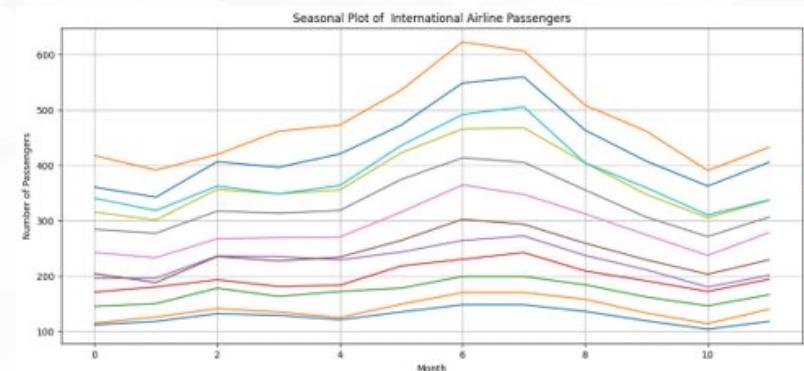
# Convert the 'time' column to a datetime format for better visualization
dataset['Date'] = pd.date_range(start="1949-01", periods=len(dataset), freq='M')
dataset.set_index('Date', inplace=True)

# Scatter Plot
plt.figure(figsize=(14, 6))
plt.scatter(dataset.index, dataset['value'], color='blue', facecolors='none')
plt.title('Scatter Plot of International Airline Passengers (1949-1960)')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.grid(True)
plt.show()

# Seasonal Plot
years = dataset.index.year.unique()
plt.figure(figsize=(14, 6))
for year in years:
    plt.plot(dataset['value'][dataset.index.year == year].values,
             label=str(year))
plt.title('Seasonal Plot of International Airline Passengers')
plt.xlabel('Month')
plt.ylabel('Number of Passengers')
plt.legend(loc="upper left", bbox_to_anchor=(1,1))
plt.grid(True)
plt.show()

# Lag Plot
plt.figure(figsize=(6, 6))
lag_plot(dataset['value'])
plt.title('Lag Plot of International Airline Passengers')
plt.grid(True)
plt.tight_layout()
plt.show()

```



Linear Regression

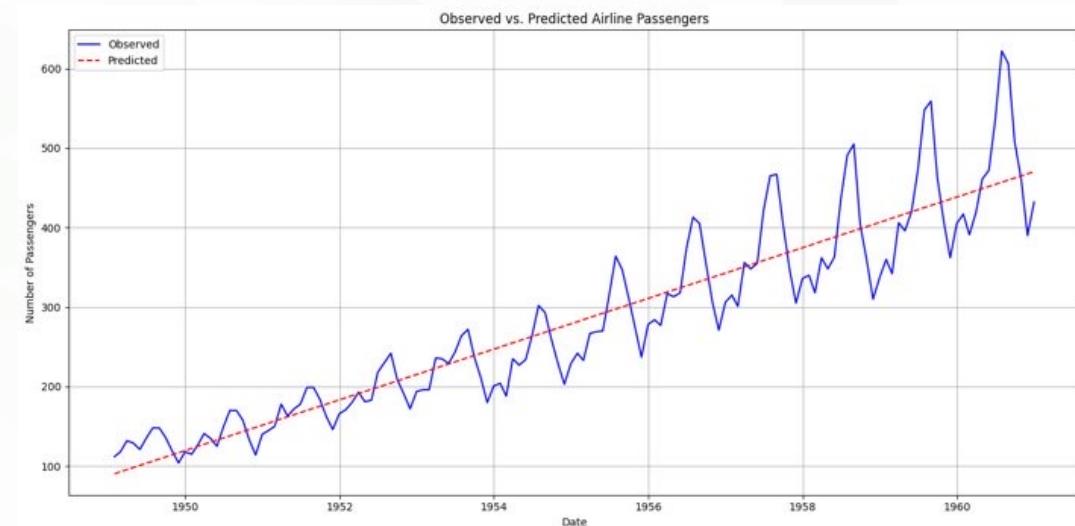
Applications of Linear Regression:

- Trend Estimation
- Seasonal Adjustment
- Forecasting
- Relation to External Factors
- Intervention Analysis
- Error Analysis

Special Concerns:

- Auto Correlation
- Non-Stationarity
- Overfitting

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate linear regression of time series data and visualize it using line plot.



Moving Average

Moving Average (MA): It involves averaging the data points over a consecutive period or 'window' as it moves through the data.

Methods and Techniques:

- Simple Moving Average (SMA)
- Weighted Moving Average (WMA)
- Exponential Moving Average (EMA)
- Centered Moving Average
- Rolling Windows
- Cumulative Moving Average

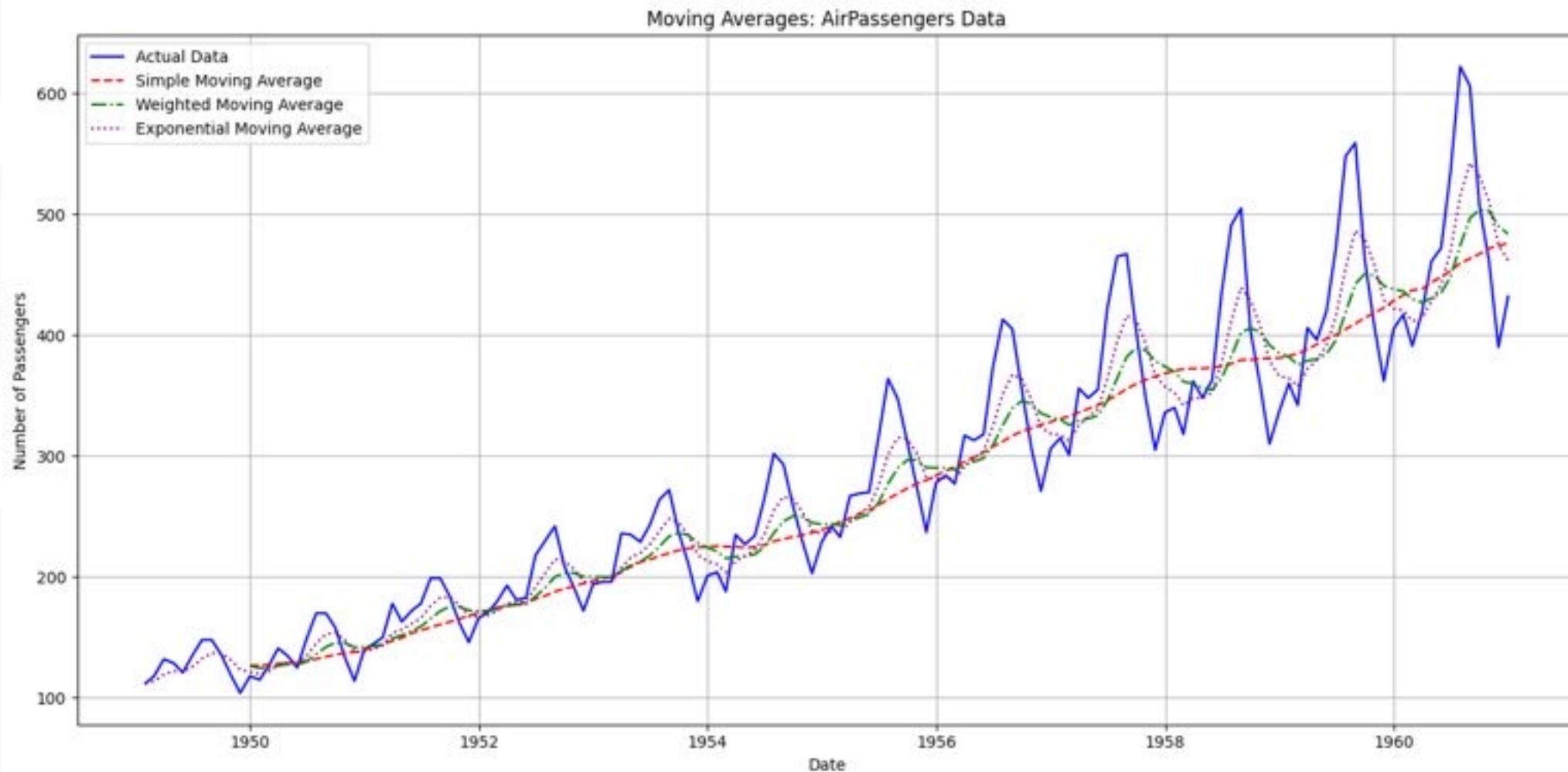
The Use Cases of Moving Average:

- Trend Identification
- Forecasting
- Seasonality Detection
- Trading Strategies

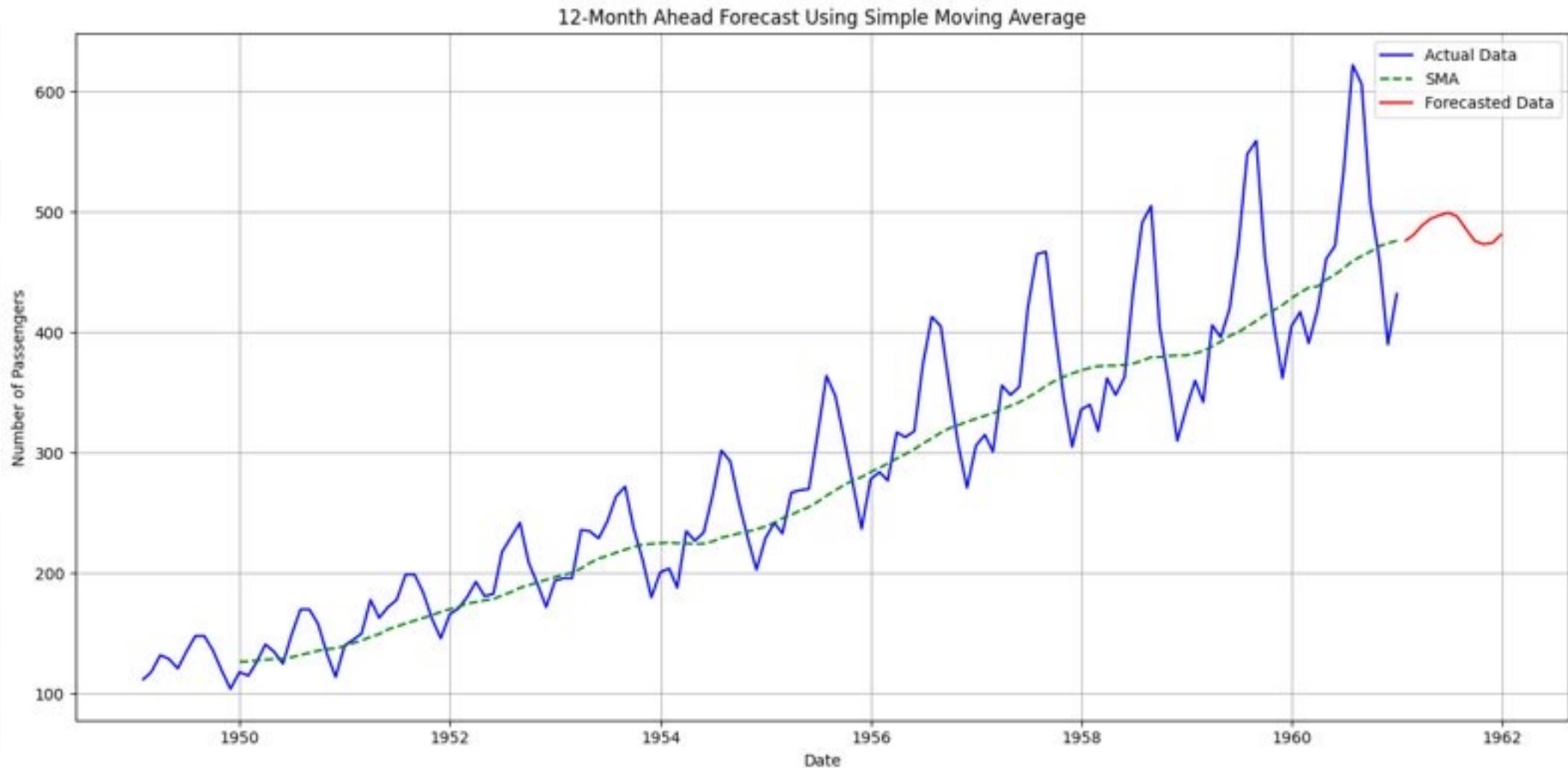
ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate simple moving average, weighted moving average and exponential moving average techniques and visualize it using line plot.

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate forecasting with moving average techniques and visualize it using line plot.

SMA, WMA and EMA



Moving Average Forecast



ARIMA

ARIMA stands for **Auto Regressive Integrated Moving Average**. It's a popular and widely used statistical method for forecasting time series data.

Common Methods:

- Stationarity Testing
- Model Identification
- Estimation
- Model Checking
- Forecasting
- Seasonal ARIMA (SARIMA)

Common Techniques:

- Box-Jenkins Methodology
- Auto ARIMA
- Transformation

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate ARIMA capabilities in forecasting and visualize it using line plot.

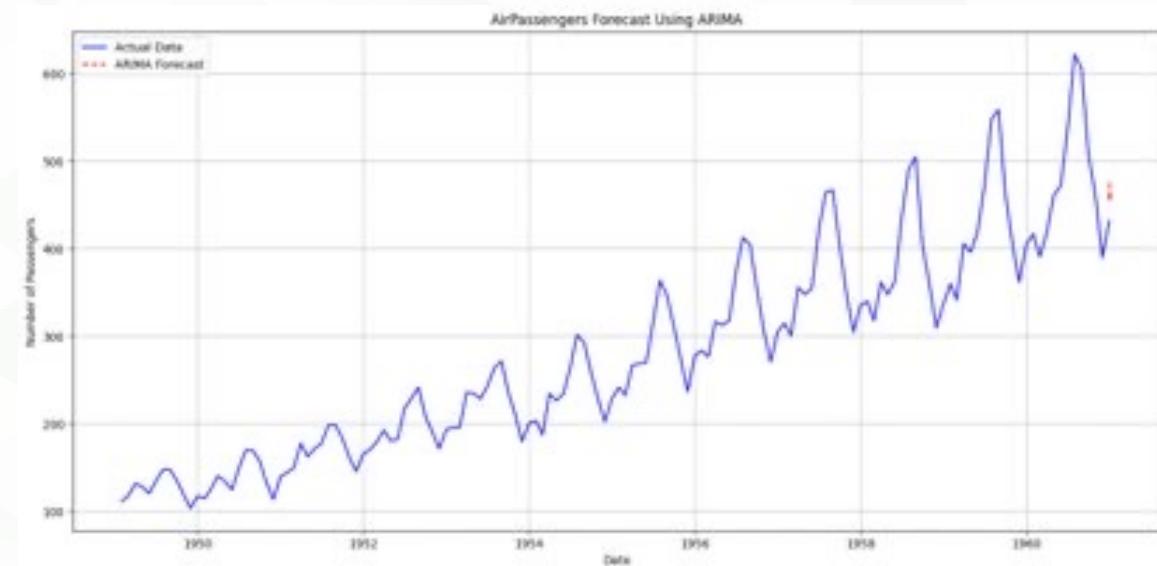
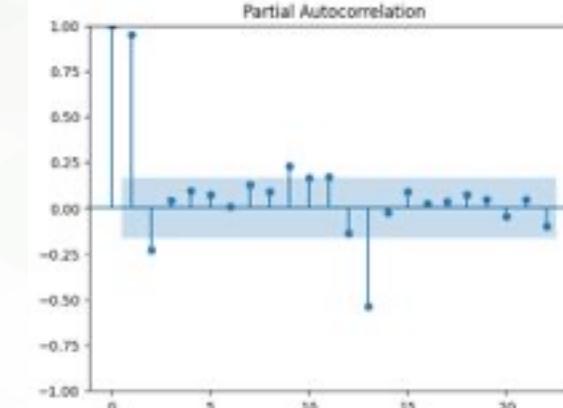
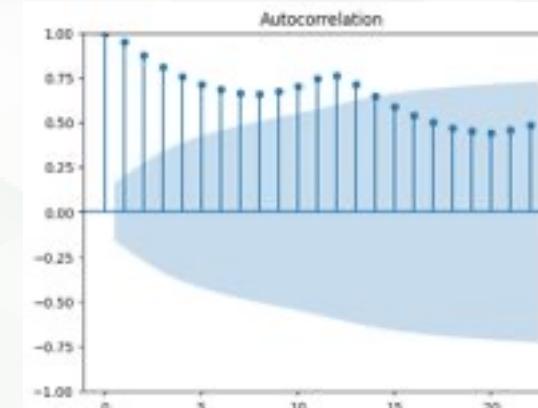
ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate auto ARIMA implementation technique in forecasting and visualize it using line plot.

ARIMA Implementation

```

import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# Load the AirPassengers dataset
dataset = sm.datasets.get_rdataset("AirPassengers").data
dataset['Date'] = pd.date_range(start='1949-01', periods=len(dataset), freq='M')
dataset.set_index('Date', inplace=True)
# Check if the series is stationary using Augmented Dickey-Fuller test
adf_test = adfuller(dataset['value'])
print(f"ADF Statistic: {adf_test[0]}")
print(f"p-value: {adf_test[1]}")
# If p-value > 0.05, the series is non-stationary and needs differencing
# Since AirPassengers dataset typically requires differencing, we'll use d=1
# Model Identification using ACF and PACF plots
plot_acf(dataset['value'])
plot_pacf(dataset['value'])
# Fitting ARIMA model
# Note: The (p,d,q) order is often chosen based on ACF and PACF plots.
# Here, for simplicity, we're using (1,1,1).
# In a real scenario, model selection should be more thorough.
model = ARIMA(dataset['value'], order=(1,1,1))
fit_model = model.fit()
# Forecasting next 12 months
forecast = fit_model.forecast(steps=12)
# Visualization
plt.figure(figsize=(14, 7))
plt.plot(dataset['value'], label='Actual Data', color='blue')
plt.plot(pd.date_range(start=dataset.index[-1], periods=12, closed='right'), forecast, label='ARIMA Forecast', color='red', linestyle='--')
plt.title('AirPassengers Forecast Using ARIMA')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

```



Auto ARIMA Implementation

```

Performing stepwise search to minimize AIC
ARIMA(2,1,2)(1,1,1)[12] : AIC=1020.048, Time=2.43 sec
ARIMA(0,1,0)(0,1,0)[12] : AIC=1031.598, Time=0.04 sec
ARIMA(1,1,0)(1,1,0)[12] : AIC=1020.393, Time=0.15 sec
ARIMA(0,1,1)(0,1,1)[12] : AIC=1021.083, Time=0.24 sec
ARIMA(2,1,2)(0,1,1)[12] : AIC=1019.935, Time=1.01 sec
ARIMA(2,1,2)(0,1,0)[12] : AIC=1019.290, Time=0.34 sec
ARIMA(2,1,2)(1,1,0)[12] : AIC=1019.546, Time=1.02 sec
ARIMA(1,1,2)(0,1,0)[12] : AIC=1024.160, Time=0.19 sec
ARIMA(2,1,2)(0,1,0)[12] : AIC=1017.847, Time=0.34 sec
ARIMA(2,1,1)(1,1,0)[12] : AIC=1017.914, Time=0.87 sec
ARIMA(2,1,1)(0,1,1)[12] : AIC=1018.359, Time=0.99 sec
ARIMA(2,1,1)(1,1,1)[12] : AIC=1018.248, Time=3.08 sec
ARIMA(1,1,1)(0,1,0)[12] : AIC=1022.393, Time=0.12 sec
ARIMA(2,1,0)(0,1,0)[12] : AIC=1022.393, Time=0.09 sec
ARIMA(3,1,1)(0,1,0)[12] : AIC=1019.084, Time=0.37 sec
ARIMA(1,1,0)(0,1,0)[12] : AIC=1020.393, Time=0.11 sec
ARIMA(3,1,0)(0,1,0)[12] : AIC=1023.666, Time=0.31 sec
ARIMA(1,1,2)(0,1,0)[12] : AIC=1021.083, Time=1.58 sec
ARIMA(2,1,1)(0,1,0)[12] intercept : AIC=inf, Time=1.59 sec

Best model: ARIMA(2,1,1)(0,1,0)[12]
Total fit time: 14.993 seconds

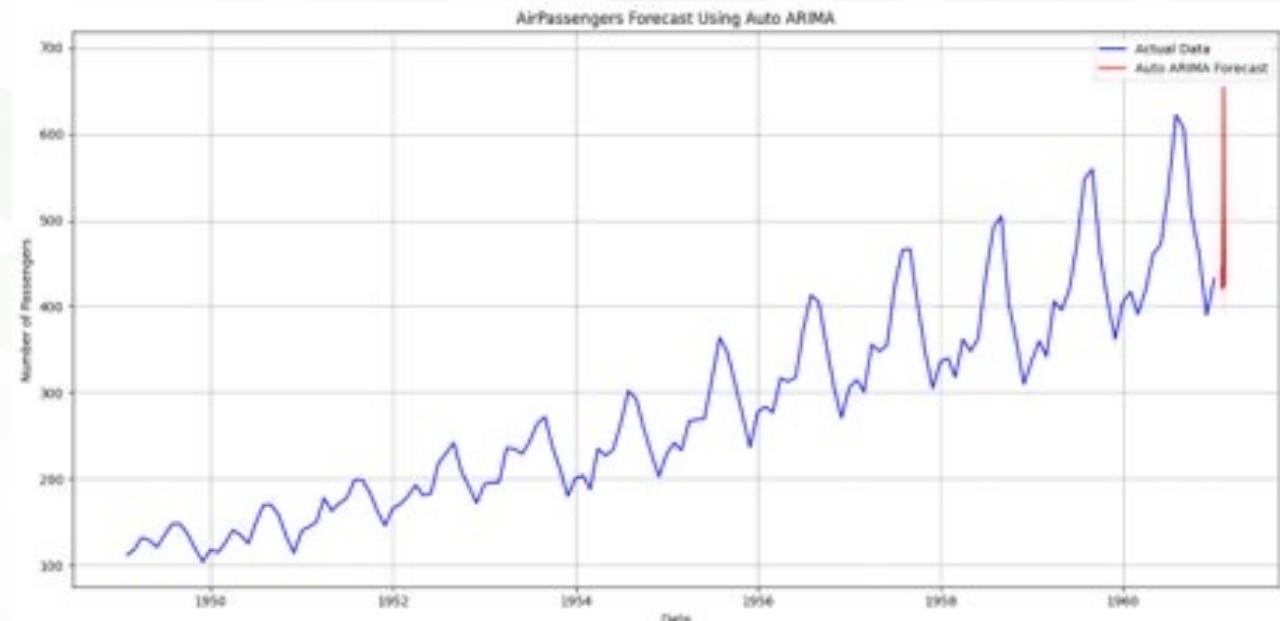
```

SARIMAX Results

| Dep. Variable: | T | No. Observations: | |
|------------------|-------------------------------|-------------------|----------|
| Model: | SARIMAX(2, 1, 1)(0, 1, 0)[12] | log Likelihood | -504.923 |
| Date: | Sat, 26 Aug 2023 | AIC | 1017.847 |
| Time: | 13:15:43 | SIC | 1029.348 |
| Sample: | 01-31-1949 | HQIC | 1022.529 |
| | - 12-31-1960 | | |
| Covariance Type: | opg | | |

| | coef | std err | t | P> t | (0.025 | 0.975) |
|--------|----------|---------|---------|-------|---------|---------|
| ar.L1 | 0.5960 | 0.085 | 6.987 | 0.000 | 0.429 | 0.763 |
| ar.L2 | 0.2143 | 0.091 | 2.343 | 0.019 | 0.035 | 0.394 |
| Re.L1 | -0.5819 | 0.638 | -25.601 | 0.000 | -1.857 | -0.907 |
| sigma2 | 129.3132 | 14.556 | 8.884 | 0.000 | 106.783 | 157.843 |

Ljung-Box (Lj) (Q): 0.00 Jarque-Bera (JB): 7.68
 Prob(Q): 0.98 Prob(JB): 0.02
 Heteroskedasticity (H): 2.33 Skew: -0.01
 Prob(H) (two-sided): 0.01 Kurtosis: 4.39



Missing Values

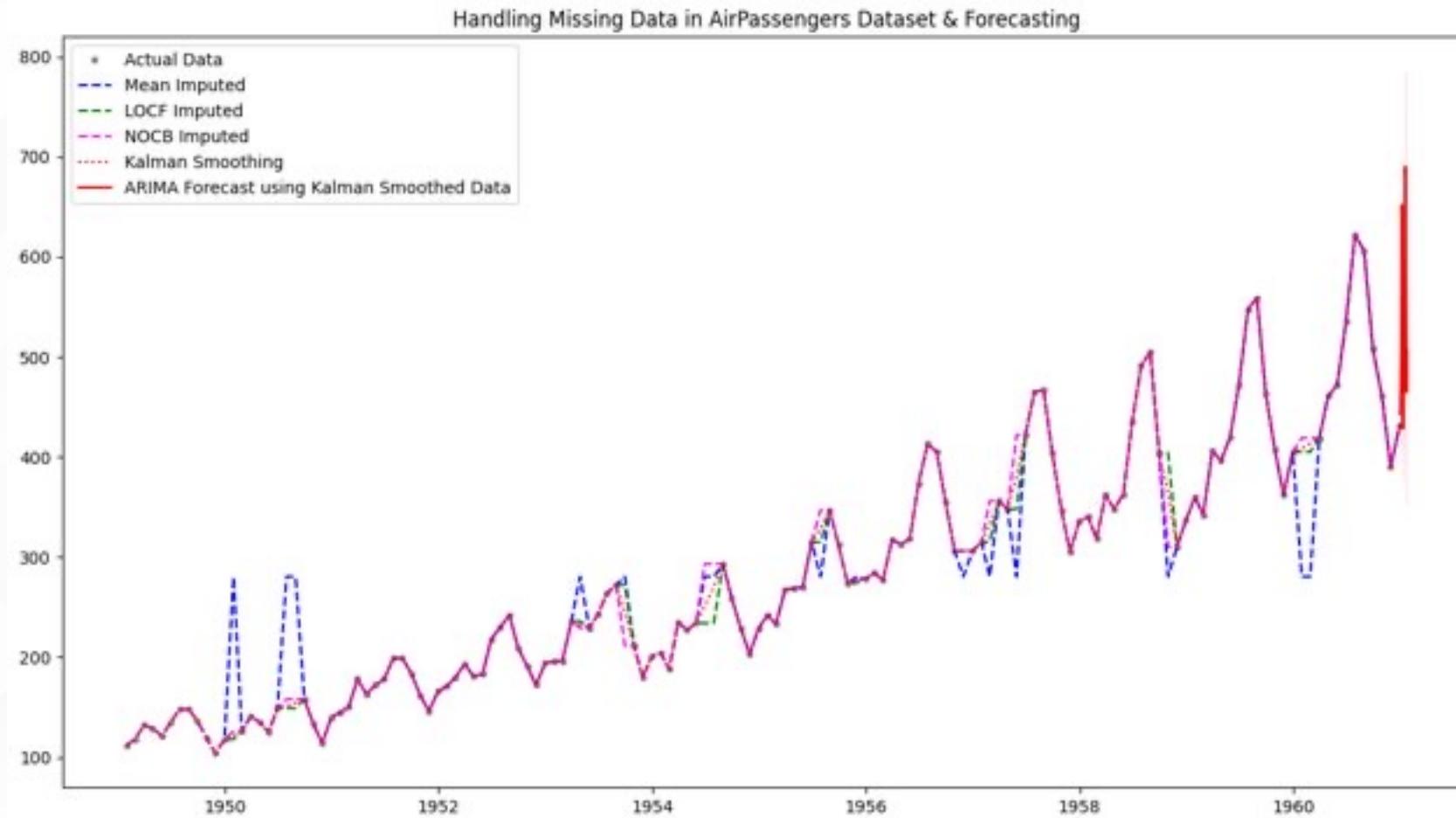
Handling Missing Values:

- Drop Missing Values
- Imputation using Mean/Median
- Last Observation Carried Forward (LOCF)
- Next Observation Carried Backward (NOCB)
- Linear Interpolation
- Time-Based Interpolation
- Seasonal Decomposition
- Use of Statistical Models
- Kalman Smoothing
- Multiple Imputation

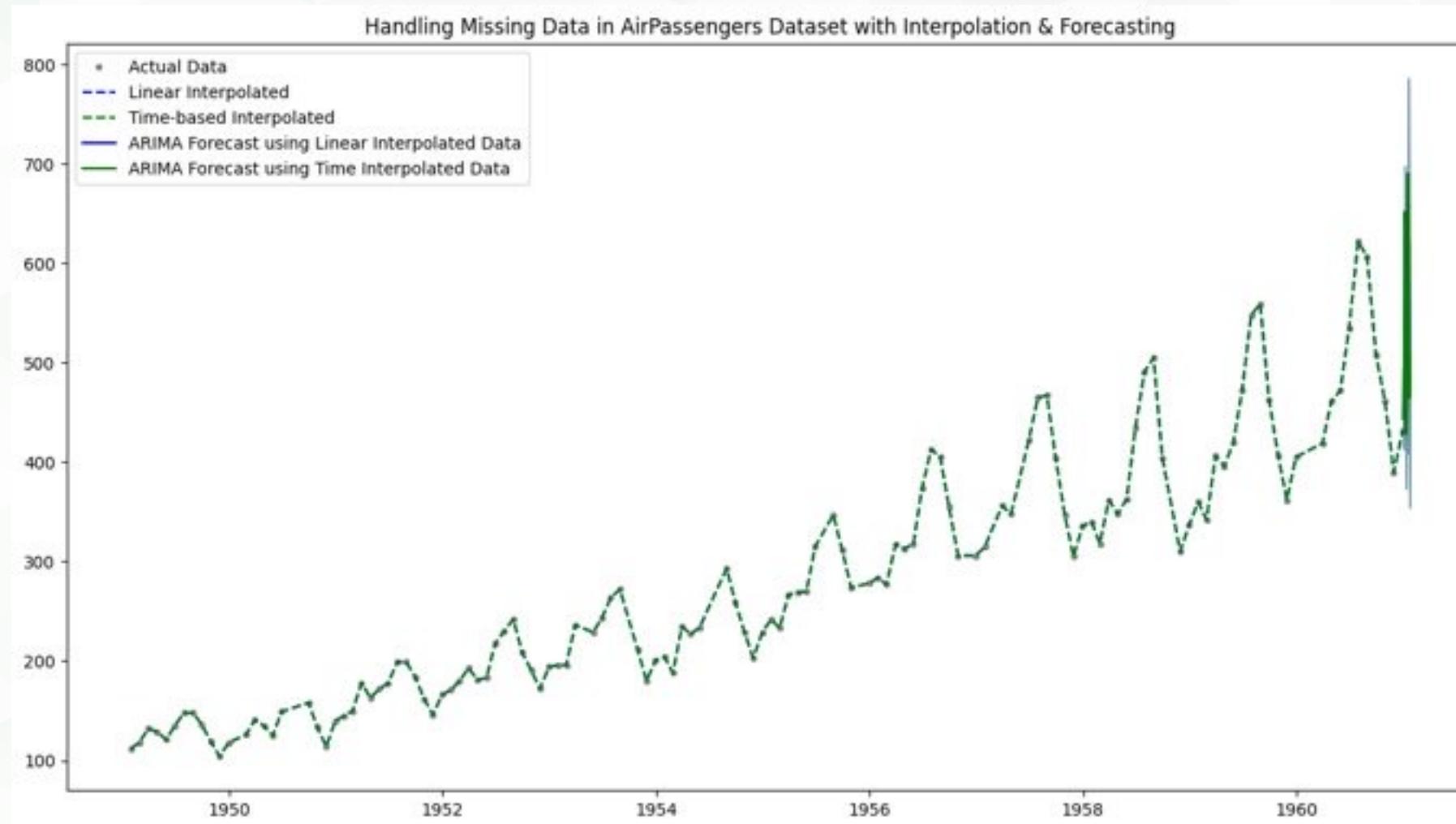
ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate common techniques to handle missing values in ARIMA implementation such as imputation, LOCF, NOCB and Kalman Smoothing then visualize the forecast using line plot.

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate common techniques to handle missing values in ARIMA implementation such as linear and time-based interpolations then visualize the forecast using line plot.

Imputation Techniques



Interpolation Techniques



Serial and Auto Correlations

The correlation of a time series with its own past and future values (positive or negative)

Methods and Techniques:

- Autocorrelation Function (ACF)
- Partial Autocorrelation Function (PACF)
- Liung-Box Test
- Durbin-Watson Statistics
- Autocorrelation Plots

Importance and Usage:

- Model Identification
- Checking Residuals
- Investment and Trading
- Detecting Non-Randomness

ChatGPT Prompt: Use publicly available Airline Passenger dataset, write Python codes to demonstrate auto correlation of time series using ACF and PACF then visualize the forecast using Autocorrelation plot.

ChatGPT Prompt: Use publicly available dataset, write Python codes to demonstrate investment and trading forecast in time series and visualize it using line plot.

ACF and PACF

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

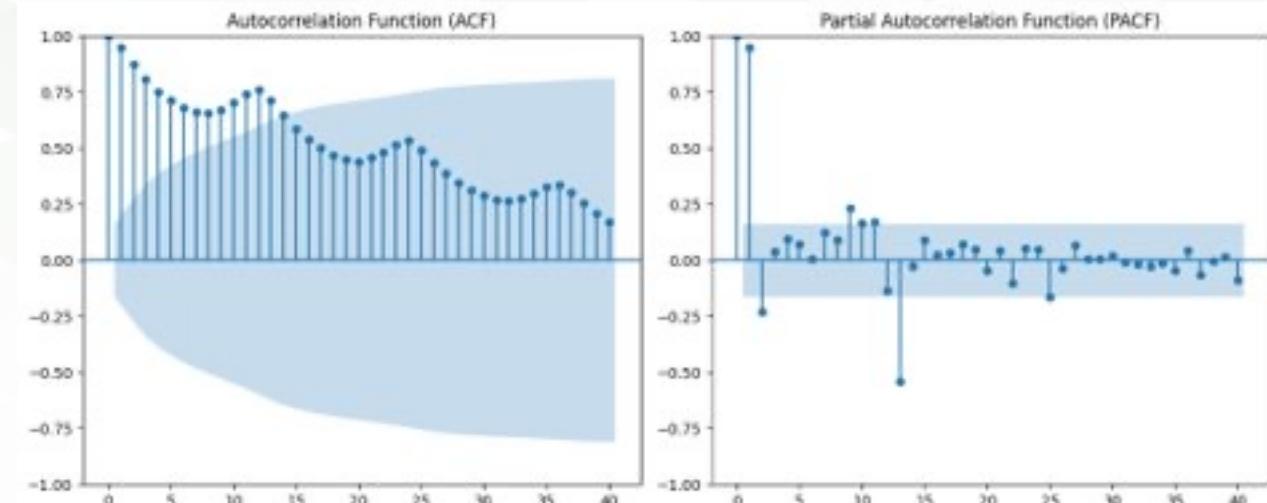
# Load the AirPassengers dataset
dataset = sm.datasets.get_rdataset("AirPassengers").data
dataset['Date'] = pd.date_range(start="1949-01", periods=len(dataset), freq='M')
dataset.set_index('Date', inplace=True)
# Assuming dataset is the name of your DataFrame

# Convert the DataFrame column to a Series
series = dataset['value']

# ACF and PACF plots
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plot_acf(series, lags=40, ax=plt.gca())
plt.title('Autocorrelation Function (ACF)')

plt.subplot(1, 2, 2)
plot_pacf(series, lags=40, ax=plt.gca())
plt.title('Partial Autocorrelation Function (PACF)')

plt.tight_layout()
plt.show()
```



Investment and Trading

```
● import yfinance as yf
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Download historical stock data
data = yf.download('AAPL', start='2010-01-01', end='2022-01-01')
data = data['Close'] # We're going to use the closing prices

# Split into train and test set
train = data[:2020-12-31]
test = data[2021-01-01:]

# Fit an ARIMA model
model = ARIMA(train, order=(5,1,0))
model_fit = model.fit()

# Forecast
forecast = model_fit.forecast(steps=len(test))

# Plot
plt.figure(figsize=(14, 7))
plt.plot(train.index, train, color='blue', label='Training Data')
plt.plot(test.index, test, color='green', marker='o', label='Actual Stock Price')
plt.plot(test.index, forecast, color='red', linestyle='--', label='Forecasted Stock Price')
plt.title('Apple Stock Prices Prediction')
plt.xlabel('Dates')
plt.ylabel('Prices')
plt.legend()
plt.grid(True)
plt.show()
```



Summary

Introduction

- Time Series Data
- Scope of Analysis

Framework and Toolkit

- Visualization
- Regression
- Moving Average
- ARIMA

Advanced Techniques and Evaluation

- Missing Values
- Serial/Autocorrelation

Quizz

Question 1: What is Time Series Data?

- A) Data sorted in alphabetical order.
- B) Data points recorded or indexed in random order.
- C) Data points recorded or indexed in time order.
- D) Data that revolves around time-travel.

Answer: C) Data points recorded or indexed in time order.

Question 2: Which of the following is NOT a common method to handle missing values in time series data?

- A) Linear Interpolation
- B) Backward-fill
- C) Forward-fill
- D) Alphabetical Imputation

Answer: D) Alphabetical Imputation

Question 3: Why is autocorrelation important in time series analysis?

- A) It indicates how data points in a time series are correlated with future points.
- B) It helps in understanding the music composition of a series.
- C) It determines how much cereal to produce in a given month.
- D) It indicates how data points in a time series are correlated with previous points.

Answer: D) It indicates how data points in a time series are correlated with previous points.