Complete Notes on Time Series Forecasting in Machine Learning

Prepared by Grok

September 07, 2025

Contents

1 Introduction to Time Series Forecasting

A time series is a sequence of data points collected at regular intervals over time, such as daily stock prices, monthly sales, or hourly temperature readings. The goal of time series forecasting is to analyze historical data to predict future values, which is critical for planning and decision-making in various fields.

- **Business**: Forecasting sales to manage inventory or predict revenue.
- **Finance**: Predicting stock prices or market trends.
- **Weather**: Estimating future temperatures or rainfall.
- **Example**: A retailer uses past sales data to predict next quarters demand, ensuring optimal stock levels.

Visualization: A line plot with time on the x-axis (e.g., days 1 to 30) and values (e.g., sales) on the y-axis shows trends and patterns. The plot below demonstrates a time series with a slight upward trend and periodic fluctuations.

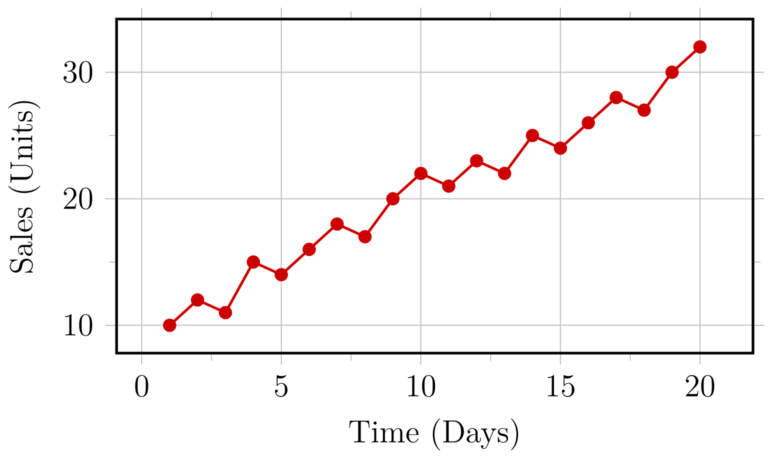

Figure 1: Example Time Series: Daily Sales

2 Components of Time Series

Time series data typically consists of four components that create the observed patterns:

- **Trend**: The long-term direction of the data, such as a steady increase in sales over years.
- **Seasonality**: Repeating patterns at fixed intervals, like higher sales every December.
- **Cyclical**: Fluctuations over longer periods without fixed intervals, such as economic booms and busts.
- **Irregular (Noise)**: Random variations that cannot be modeled, like a sudden sales drop due to an unexpected event.

Additive Decomposition Formula:

$$Y_t = T_t + S_t + C_t + I_t$$

where:

- Y_t: Observed value at time t.
- T_t: Trend component.
- S_t: Seasonal component.
- C_t: Cyclical component.
- I_t: Irregular (noise) component.

Example Calculation: Suppose we have monthly sales data: [100, 120, 140, 160, 180]. Assume a linear trend $T_t = 90 + 10t$, seasonal component repeating every 4 months [10, 5, -5, -10], and no cyclical component. Calculate components for $t = 3$:

- Trend: $T_3 = 90 + 10 \cdot 3 = 120$.
- Seasonal: $S_3 = -5$ (third position in [10, 5, -5, -10]).
- Cyclical: $C_3 = 0$.
- Irregular: $I_3 = Y_3 - (T_3 + S_3) = 140 - (120 + (-5)) = 140 - 115 = 25$.

Thus, $Y_3 = 120 + (-5) + 0 + 25 = 140$.

3 Preprocessing Time Series Data

Preprocessing ensures data is clean and suitable for modeling. Key steps include handling missing values, outliers, and ensuring stationarity.

3.1 Handling Missing Values

Fill missing data to avoid model errors:

- **Mean Imputation**: Replace with the average of the series.
- **Forward Fill**: Use the previous value.
- **Example**: Data = [10, NaN, 14, 16]. Mean = (10+14+16)/3 = 13.33. Imputed: [10, 13.33, 14, 16].

3.2 Detecting and Handling Outliers

Outliers are extreme values that can skew models.

- **Method**: Identify values beyond 3 standard deviations from the mean.

- **Example**: Data = [10, 12, 14, 100, 18]. Mean = 30.8, Std = 37.3. Threshold = 30.8 ś 3 $\cdot 37.3 = [-81.1, 142.7].100 is within bounds, but if it were 200, cap it at 142.7.$

3.3 Ensuring Stationarity

Stationary data has constant mean and variance. Non-stationary data (with trends or seasonality) needs transformation.

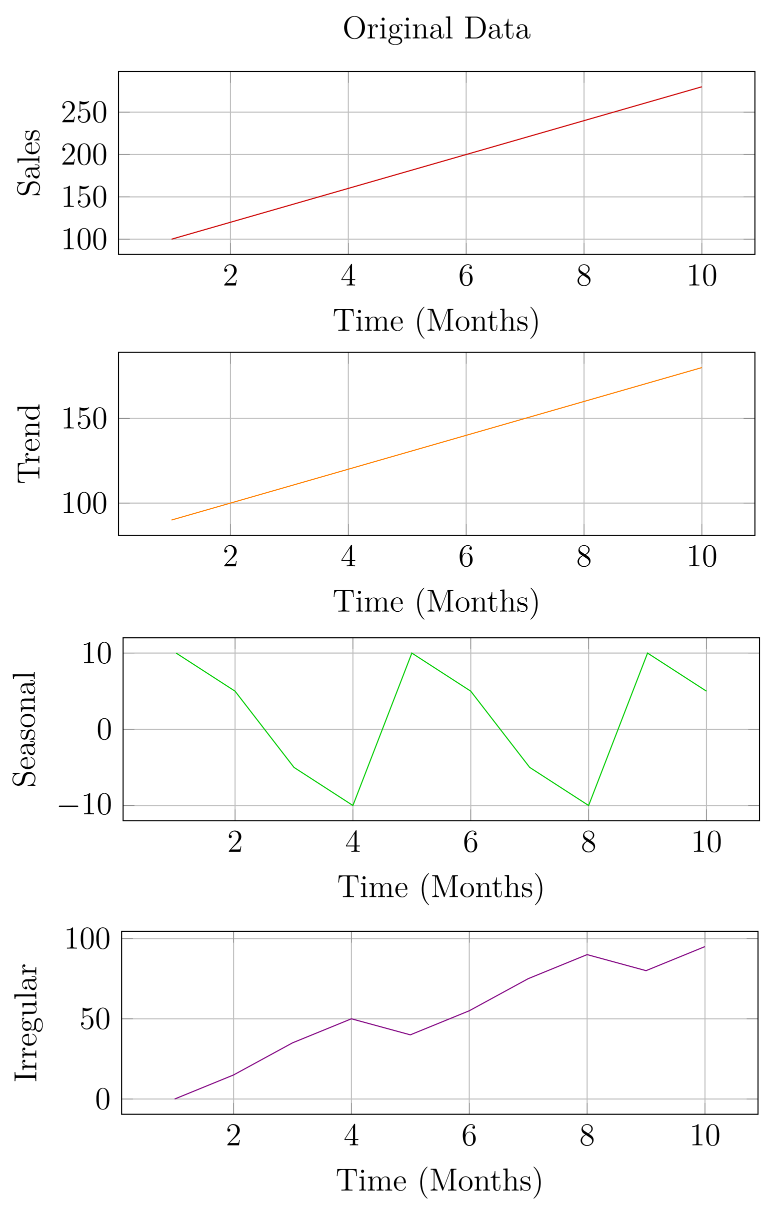

Figure 2: Time Series Decomposition

- **Test**: Augmented Dickey-Fuller (ADF). If p-value > 0.05, data is non-stationary.
- **Differencing**: Subtract previous value to remove trend.
- **Formula**: $\Delta Y_t = Y_t - Y_{t-1}$.
- **Example Calculation**: Data = [10, 12, 15, 19, 24]. Difference = [NaN, 12-10=2, 15-12=3, 19-15=4, 24-19=5] = [NaN, 2, 3, 4, 5].

Moving Average Smoothing:

$$MA_t = \frac{1}{k} \sum_{i=0}^{k-1} Y_{t-i}$$

Example (k=3): Data = [10, 12, 14, 16, 18]. MA = [NaN, NaN, (10+12+14)/3=12, (12+14+16)/3=14, (14+16+18)/3=16].

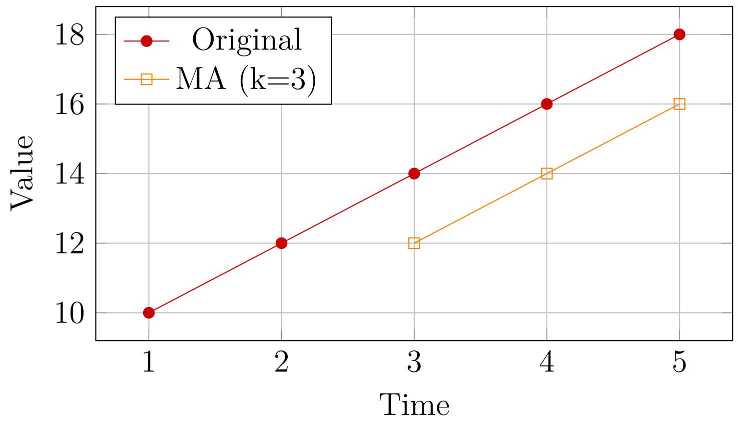

Figure 3: Moving Average Smoothing

4 Classical Forecasting Methods

Classical methods are simple and effective for linear patterns.

4.1 Moving Average

Predicts the next value as the average of the last k values.

- **When to Use**: Smooth out noise for short-term forecasts.
- **Example**: For data [10, 12, 14, 16], predict next with k=3: (12+14+16)/3 = 14.

```python
import pandas as pd
data = pd.Series([10, 12, 14, 16, 18])
ma = data.rolling(window=3).mean()
```

4.2 Autoregression (AR)

Predicts based on past values.

- **Formula**: AR(p): $Y_t = c + \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \epsilon_t$.
- **Example (AR(1))**: $Y_t = 0.5Y_{t-1} + \epsilon_t$. If $Y_{t-1} = 10$, predict $Y_t = 0.5 \cdot 10 = 5$ (ignoring constant/error).

Example Calculation: Data = [10, 12, 14, 16], AR(1) with $\phi_1 = 0.6$. Predict Y_5: $Y_5 = 0.6 \cdot 16 = 9.6$.

```python
from statsmodels.tsa.ar_model import AutoReg
model = AutoReg(data, lags=1)
model_fit = model.fit()
yhat = model_fit.predict(start=len(data), end=len(data))
```

4.3 ARIMA

ARIMA(p,d,q) combines autoregression, differencing, and moving average.

- **Components**:
- AR(p): Past values.

- I(d): Differencing to stationarize.
- MA(q): Past errors: $Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$.

Building ARIMA:

1. Test stationarity with ADF.
2. Apply differencing (d) if needed.
3. Use ACF for q, PACF for p.
4. Fit model and forecast.
5. Evaluate with RMSE or AIC (lower is better).

Example Calculation: Data = [10, 12, 14, 16, 18], ARIMA(1,1,0). Step 1: Difference: [NaN, 2, 2, 2, 2]. Step 2: Fit AR(1) on differences with $\phi_1 = 0.5$. Predict next difference: $0.5 \cdot 2 = 1$. Next value: $Y_5 = 18 + 1 = 19$.

```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(data, order=(1,1,0))
model_fit = model.fit()
forecast = model_fit.forecast(steps=3)
```

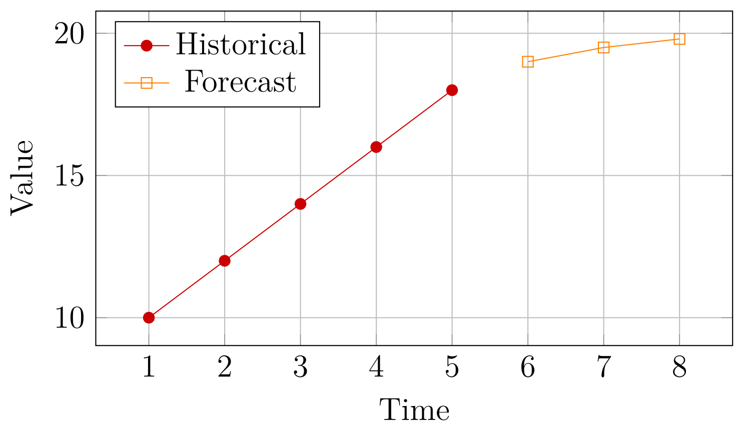

Figure 4: ARIMA(1,1,0) Forecast

4.4 SARIMA

SARIMA(p,d,q)(P,D,Q)s handles seasonality.

- **Example**: Monthly data with yearly seasonality (s=12).
- **Formula**: Extends ARIMA with seasonal AR, differencing, and MA.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
data = pd.Series([...])  % Monthly data
model = SARIMAX(data, order=(1,1,1), seasonal_order=(1,1,1,12))
model_fit = model.fit()
forecast = model_fit.forecast(steps=12)
```

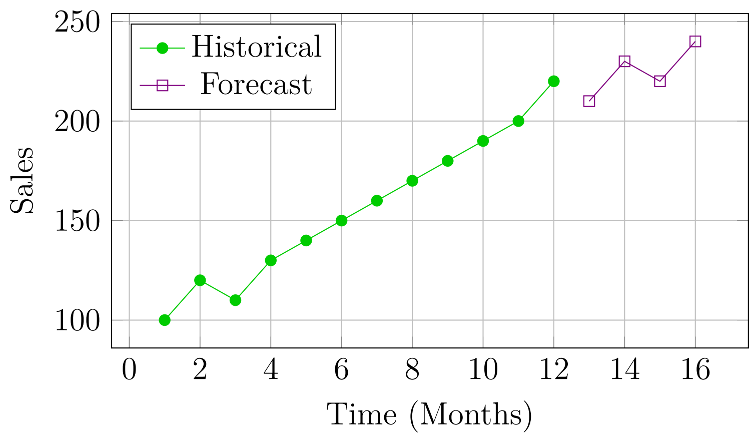

Figure 5: SARIMA Forecast with Seasonality

5 Machine Learning Methods

Machine learning methods excel at capturing complex, non-linear patterns.

5.1 Linear Regression with Lags

Use past values (lags) as features to predict the next value.

- **Example**: Predict Y_t using Y_{t-1}, Y_{t-2}.
- **Calculation**: Data $= [10, 12, 14, 16, 18]$. Features: $[Y_{t-1}, Y_{t-2}] = [12, 10], [14, 12], [16, 14], [18, 16]$. Fit linear model: $Y_t = 2 + 0.8Y_{t-1} + 0.1Y_{t-2}$. Predict: $Y_5 = 2 + 0.8 \cdot 18 + 0.1 \cdot 16 = 18$.

```
from sklearn.linear_model import LinearRegression
X = [[10,8], [12,10], [14,12], [16,14]]  % Lags
y = [12, 14, 16, 18]
model = LinearRegression().fit(X, y)
pred = model.predict([[18,16]])
```

5.2 Long Short-Term Memory (LSTM)

LSTMs handle long-term dependencies using memory cells.

- **When to Use**: Complex sequences with long patterns.
- **Example**: Predicting stock prices based on past weeks.

```
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(32, input_shape=(5,1), return_sequences=True),
    tf.keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mse')
```

5.3 Transformers

Transformers use attention mechanisms to focus on important past data.

- **Pros**: Handle long sequences efficiently.

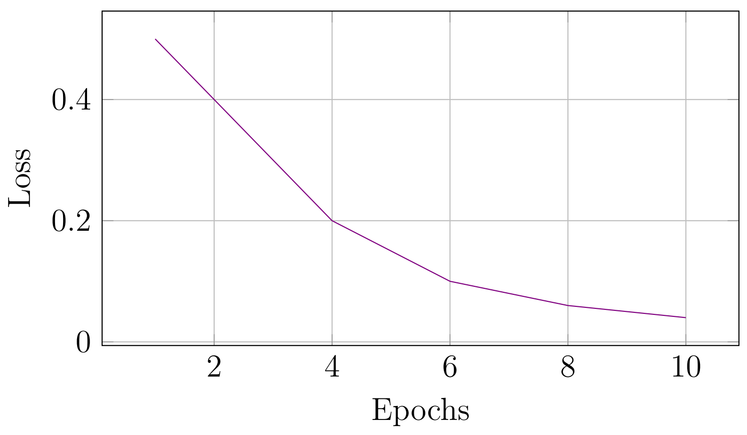

Figure 6: LSTM Training Loss

- **Example**: Forecasting demand over years.

```python
from transformers import TimeSeriesTransformerModel
model = TimeSeriesTransformerModel(...)
model.fit(data)
```

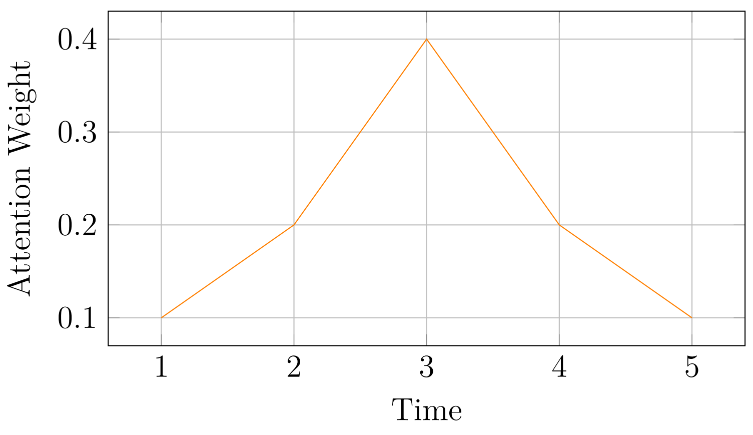

Figure 7: Transformer Attention Weights

6 Business Case Problems

6.1 Case 1: Retail Sales Forecasting

Problem: A retail chain needs to forecast daily sales for the next 30 days to plan inventory. Data shows a yearly seasonal pattern and a slight upward trend.

Data: 2 years of daily sales (730 points), with December peaks and 2% annual growth.

Solution Steps:

1. **Visualize**: Plot data to confirm trend and seasonality.
2. **Preprocess**: Fill missing values with forward fill.
3. **Stationarity**: Apply ADF test. If non-stationary, difference once (d=1).

4. **Model**: Use SARIMA(1,1,1)(1,1,1,365) for daily data with yearly seasonality.
5. **Forecast**: Predict 30 days.
6. **Evaluate**: Compute RMSE on a test set.

Example: Sample data = [100, 102, 104, ..., 200] (730 points). Difference: [NaN, 2, 2, ...]. Fit SARIMA, forecast: [201, 202, ..., 230] (simulated).

```python
from statsmodels.tsa.statespace.sarimax import SARIMAX
import pandas as pd
data = pd.Series([...])  % Daily sales
model = SARIMAX(data, order=(1,1,1), seasonal_order=(1,1,1,365))
model_fit = model.fit()
forecast = model_fit.forecast(steps=30)
```

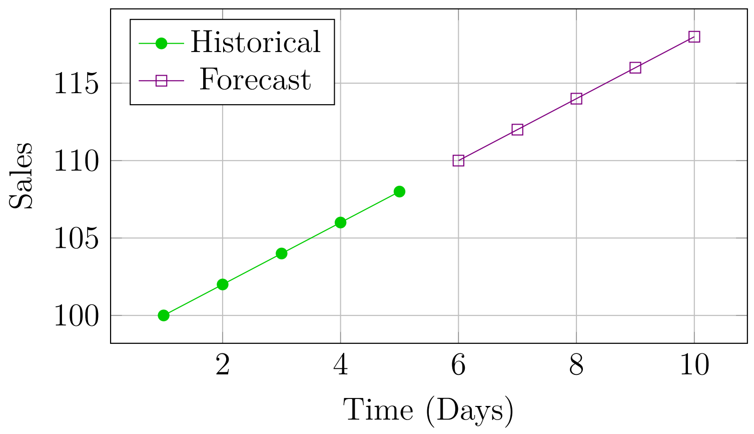

Figure 8: Retail Sales Forecast

Answer: Forecast shows a slight increase with a peak if December is included.

6.2 Case 2: Energy Consumption Prediction

Problem: An energy company wants to predict hourly electricity usage for the next week to optimize grid operations.

Data: 1 year of hourly data (8760 points), with daily and weekly seasonality.

Solution Steps:

1. **Preprocess**: Remove outliers (e.g., cap at 3 std deviations).
2. **Windowing**: Create sequences of 24 hours to predict the next hour.
3. **Model**: Use LSTM with 24-hour input window.
4. **Forecast**: Predict 168 hours (1 week).
5. **Evaluate**: Use MAE to assess accuracy.

Example: Data = [50, 52, 55, ..., 60] (24 hours). Input: [50, 52, ..., 55] Output: 60. LSTM predicts: 59.8.

```python
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(64, input_shape=(24,1)),
    tf.keras.layers.Dense(1)
])
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=50)
```

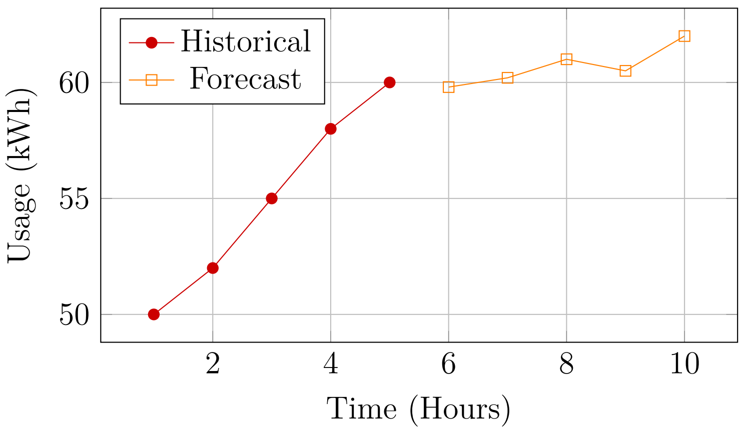

Figure 9: Energy Usage Forecast

Answer: Forecast predicts stable usage with daily peaks.

6.3 Case 3: Stock Price Prediction

Problem: A financial firm wants to predict daily stock prices for the next 10 days to inform trading decisions.

Data: 5 years of daily prices with a trend and irregular fluctuations.

Solution Steps:

1. **Preprocess**: Log-transform to stabilize variance.
2. **Model**: Use ARIMA(2,1,1) after confirming stationarity.
3. **Forecast**: Predict 10 days.
4. **Evaluate**: Compare with actual prices using RMSE.

Example: Data = [100, 102, 105, 108, 110]. Log: [4.605, 4.625, 4.654, 4.682, 4.700]. Difference: [NaN, 0.020, 0.029, 0.028, 0.018]. ARIMA(2,1,1) predicts next difference: 0.019. Next log value: 4.700 + 0.019 = 4.719. Exp(4.719) 112.

```
from statsmodels.tsa.arima.model import ARIMA
import numpy as np
data = np.log(pd.Series([100, 102, 105, 108, 110]))
model = ARIMA(data, order=(2,1,1))
model_fit = model.fit()
forecast = np.exp(model_fit.forecast(steps=10))
```

Answer: Forecast shows a gradual increase in stock price.

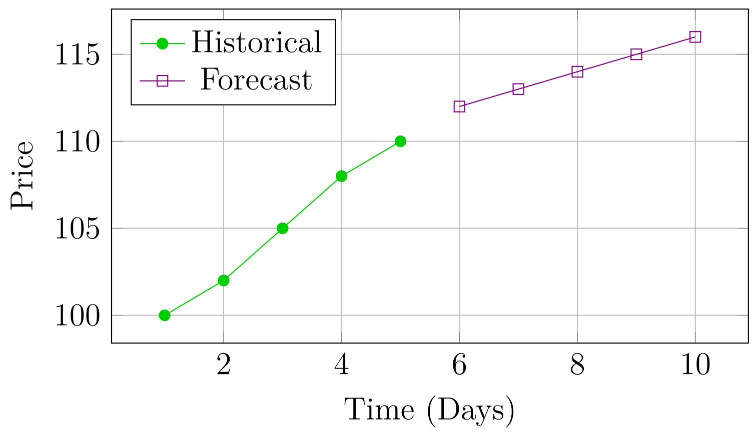

Figure 10: Stock Price Forecast

7 Evaluation and Best Practices

Metrics:

- **MAE**: Mean Absolute Error, average of absolute differences.
- **RMSE**: Root Mean Squared Error, penalizes larger errors.
- **Formula**: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$.

Example: Actual = [10, 12, 14], Predicted = [11, 12, 13]. Errors = [1, 0, -1]. RMSE = $\sqrt{(1^2 + 0^2 + (-1)^2)/3} = \sqrt{2/3} \approx 0.816$.

Best Practices:

- Split data: Train on past, test on future.
- Avoid overfitting: Use validation set.
- Combine models: Try classical (ARIMA) and ML (LSTM) for robustness.

8 Interview Questions and Answers

1. **What is a time series and its applications?**
   Answer: A time series is data collected at regular intervals, like daily sales. Used in sales forecasting, stock prediction, and weather forecasting.
2. **What are the components of a time series?**
   Answer: Trend (long-term direction), seasonality (repeating patterns), cyclical (long-term fluctuations), and irregular (random noise).
3. **How do you test for stationarity?**
   Answer: Use the ADF test. A p-value < 0.05 indicates stationarity.
4. **What is differencing?**
   Answer: Subtracting consecutive values to remove trends. Formula: $\Delta Y_t = Y_t - Y_{t-1}$.

5. **When to use SARIMA?**
   Answer: When data has seasonality, like monthly sales with yearly peaks.
6. **What is an LSTM?**
   Answer: A neural network with memory cells for learning long-term patterns in sequences.
7. **How do you evaluate a forecast?**
   Answer: Use MAE or RMSE to measure prediction errors.
8. **What are ACF and PACF used for?**
   Answer: ACF identifies MA order (q); PACF identifies AR order (p).
9. **Why preprocess data?**
   Answer: To handle missing values, outliers, and non-stationarity for accurate modeling.
10. **Classical vs. ML methods?**
    Answer: Classical methods (e.g., ARIMA) are simpler and good for linear data; ML (e.g., LSTM) handles complex, non-linear patterns but needs more data.

9 Practical Tips for Mastery

- **Practice Datasets**: Use public datasets like Air Passengers, Stock Prices, or Energy Consumption.
- **Tools**: Python (pandas, statsmodels, tensorflow), R for classical models.
- **Automation**: Try AutoML tools like PyCaret for quick model selection.
- **Experiment**: Combine models (e.g., ARIMA + LSTM) for better accuracy.

10 Conclusion

Time series forecasting is a powerful tool for predicting future trends based on past data. From classical methods like ARIMA to advanced ML models like LSTMs and Transformers, mastering these techniques requires understanding data components, preprocessing, modeling, and evaluation. The business cases and interview questions provided offer practical insights for real-world applications.