Supervised Learning - Time Series Forecast

Machine Learning Workshop – Day 4

Erdener Emin Eker

May 18, 2025

TEDU AI Data Science & AFIT Workshop

What We'll Do Today

- Understand what makes time series data unique.
- Explore core components: trend, seasonality, noise.
- Learn how to make forecasts using simple models.
- Compare traditional methods (like ARIMA) with ML approaches.
- See how lag features and date parts help prediction.
- Get ready for a hands-on Colab session to forecast future values.

Goal: Build strong intuition about time-based prediction.

What is Time Series Data?

- A **time series** is a sequence of observations recorded at regular time intervals.
- The order of data points matters — each value depends (partly) on previous ones.
- Common examples:
    - Daily temperatures
    - Monthly inflation rates
    - Weekly sales or stock prices
- Different from regular datasets where rows are often independent.

Key Idea: Time adds structure — and challenges — to our data.

Why Time Order Matters

- In time series, the position of each value is important — order carries information.
- If you shuffle the data, you lose the pattern.
- Models like linear regression assume data points are independent — this breaks in time series!
- Time series models look for patterns *over time*.

Suggested Visual: Show a simple example:

- Left: ordered line chart of sales over time
- Right: same values, but shuffled $\rightarrow$ messy and meaningless

Visuals make this difference obvious — include side-by-side plots if possible.

Why Time Order Matters

- In time series, the order of the data points is meaningful.
- Each value can depend on previous ones, called **temporal dependence**.
- If you shuffle the values like a normal dataset, you lose the trend and seasonality.
- This is why we must treat time series differently in both analysis and modeling.

Time is not just another variable — it's the backbone of the data structure.

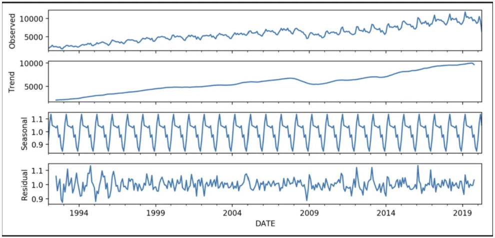
Types of Time Series Data

- Time series data comes in different shapes and forms.
- Key differences include:
  - **Frequency:** yearly, monthly, daily, hourly, even by minute or second.
  - **Length:** short vs. long history.
  - **Patterns:** some show strong trends or seasonality, others are noisy.
- These characteristics affect:
  - How we preprocess the data
  - What models work best
  - How much past data we need for accurate forecasts

Time resolution and structure define your forecasting strategy.

Components of a Time Series

- Most time series are a combination of three main elements:
    - **Trend:** long-term increase or decrease
    - **Seasonality:** regular repeating patterns (e.g., yearly, weekly)
    - **Noise (residual):** random variation not explained by trend/seasonality
- Understanding these helps us build better models and cleaner features.



Decomposing a time series helps us understand what's really driving change.

Additive vs. Multiplicative Models

- **Additive model:**

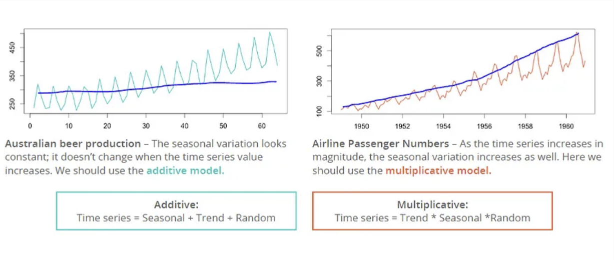
$$\text{Observed} = \text{Trend} + \text{Seasonality} + \text{Noise}$$

  Seasonality effect stays constant over time.

- **Multiplicative model:**

$$\text{Observed} = \text{Trend} \times \text{Seasonality} \times \text{Noise}$$

  Seasonality effect grows or shrinks with the trend.



Australian beer production – The seasonal variation looks constant; it doesn't change when the time series value increases. We should use the additive model.

Airline Passenger Numbers – As the time series increases in magnitude, the seasonal variation increases as well. Here we should use the multiplicative model.

Use additive when variation is stable; multiplicative when it scales with the level.

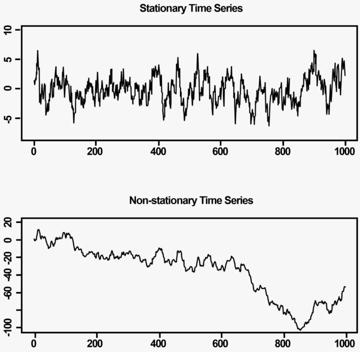
How to Handle Additive vs. Multiplicative Seasonality

- **Additive Seasonality:**
  - Use when seasonal variation stays **constant** over time.
  - No transformation needed — model directly.
  - Methods:
    - STL decomposition (mode = "additive")
    - Linear models with seasonal dummies
    - ARIMA on differenced series
- **Multiplicative Seasonality:**
  - Use when seasonal effect **grows or shrinks** with the trend.
  - Apply **log transformation** to stabilize variance.
  - Methods:
    - STL decomposition (mode = "multiplicative")
    - Holt-Winters exponential smoothing
    - ARIMA on log-transformed series

Tip: Visualize your data first — if the seasonal swings get bigger, go multiplicative.

What is Stationarity?

- A time series is **stationary** if its statistical properties do not change over time.
- That means:
  - The mean stays roughly the same
  - The variance doesn't change much
  - The structure of correlations (autocorrelation) is stable
- Many models (like ARIMA) assume stationarity.
- If your data isn't stationary, you'll need to transform it (e.g., differencing, log).



Left: Non-stationary Right: Stationary

Why Stationarity Matters for Forecasting

- Many forecasting models — like ARIMA — assume that the data is stationary.
- If a series has a trend or changing variance, the model will likely:
  - Make poor predictions
  - Misestimate uncertainty
  - Violate model assumptions
- Stationarity makes the future look "like the past" — this is essential for learning patterns.
- If your data is non-stationary, you'll need to:
  - Remove the trend (differencing)
  - Stabilize variance (log transform)

Models learn from the past — so we must make the past stable first.

How to Make a Series Stationary — Choosing the Right Tool

- **1. Detrending (Remove deterministic trend)**
  - Fit and subtract a trend line (e.g., linear regression).
  - Use when the trend is smooth and predictable.
  - Keeps more of the original structure than differencing.
- **2. Differencing (Remove stochastic trend)**
  - Subtract previous values: $y'_t = y_t - y_{t-1}$
  - Use when trend fluctuates unpredictably or detrending is not enough.
  - Can be applied more than once (first, second difference).
- **3. Log or Box-Cox Transform (Stabilize variance)**
  - Use when variance increases over time (e.g., multiplicative seasonality).
  - Often combined with differencing or detrending.

Look at your data. Use visual plots and tests to decide which method to apply.

Time Series Decomposition Recap

- A time series can be separated into three components:
  - **Trend:** long-term movement
  - **Seasonality:** regular repeating patterns
  - **Residual (Noise):** irregular or unexplained variation
- Why decompose?
  - To better understand underlying patterns
  - To model components separately
  - To clean the data before applying forecasting models
- Tools like `seasonal_decompose()` or STL help automate this process.

Decomposition simplifies complex data and prepares it for accurate forecasting.

Forecasting Goals – What Are We Trying to Do?

- The goal of time series forecasting is to **predict future values based on past observations**.
- Common forecasting tasks:
  - **One-step ahead:** Predict the very next time point
  - **Multi-step ahead:** Predict several future points at once
- Forecasts can be:
  - **Point forecasts:** Single predicted value
  - **Prediction intervals:** Ranges that account for uncertainty
- In practice, we often care about both:
  - Accuracy of prediction
  - Timing of prediction

Forecasting is not just about predicting — it's about predicting at the right time and with the right confidence.

Baseline Forecasting Methods

- Before complex models, we start with simple **baseline methods**:
- **Naive forecast:**
  - Forecast = last observed value
  - Works well for stable series
- **Average forecast:**
  - Forecast = mean of past values
  - Assumes the process is random but centered
- **Seasonal naive:**
  - Forecast = value from the same season last year
  - Example: use last December to predict this December
- These are easy to implement and often surprisingly strong — any good model should beat them!

Always compare against baselines — if a fancy model doesn't beat naive, rethink it.

Evaluating Forecast Accuracy

- Once we make predictions, we need to measure how accurate they are.
- Common evaluation metrics:
  - Average of the absolute differences between predicted and actual values.
  - Easy to interpret — same unit as the data.
- **Root Mean Squared Error (RMSE):**
  - Penalizes larger errors more than MAE.
  - Sensitive to outliers.
- **Mean Absolute Percentage Error (MAPE):**
  - Expresses error as a percentage.
  - Easy to compare across datasets — but undefined when actual = 0.

Use multiple metrics to get a full picture of forecast performance.

Introduction to ARIMA Models

- **ARIMA** stands for:
  - **AR** — Autoregression: uses past values to predict the future
  - **I** — Integration: differencing to make the series stationary
  - **MA** — Moving Average: uses past errors to improve prediction
- Written as ARIMA(p, d, q):
  - $p =$ number of autoregressive terms
  - $d =$ number of differences needed to make the series stationary
  - $q =$ number of moving average terms
- ARIMA is best for:
  - Univariate time series
  - When you want interpretable, statistical forecasts

ARIMA is a workhorse of forecasting — and a strong baseline for any project.

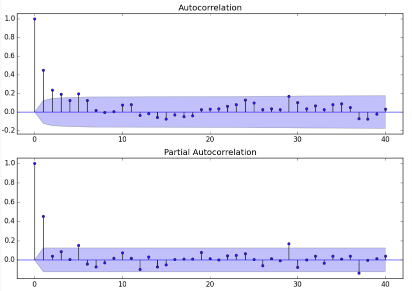
ARIMA Extensions – SARIMA and ARIMAX

- **SARIMA** – Seasonal ARIMA:
  - Adds seasonal terms to capture repeating patterns.
  - Notation: **ARIMA(p, d, q)(P, D, Q)[s]**
  - Example: monthly data with yearly seasonality $\rightarrow s = 12$
- **ARIMAX** – ARIMA with Exogenous Variables:
  - Includes external predictors (e.g., interest rate, temperature, marketing spend).
  - Useful when external factors help explain the target series.
  - Forecast = function of both past values and other variables.
- Many real-world problems require one or both of these extensions.

SARIMA handles seasonality. ARIMAX brings in external signals.

Understanding ACF & PACF

- **ACF (Autocorrelation Function):**
  - Shows how correlated a time series is with its own past values.
  - Helps identify the number of MA (Moving Average) terms (q).
- **PACF (Partial Autocorrelation Function):**
  - Measures the correlation of a series with its lags after removing effects of intermediate lags.
  - Helps identify the number of AR (Autoregressive) terms (p).



Use ACF and PACF plots to identify ARIMA model orders.

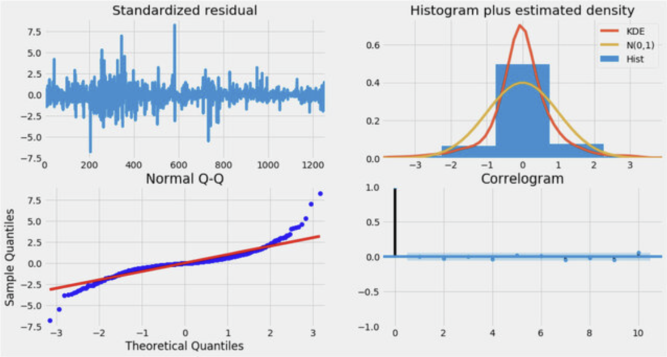
How to Determine ARIMA Orders (p, d, q)

- **Step 1 – Determine differencing order (d):**
  - Use visual inspection (trend?), ADF test, or KPSS test
  - If non-stationary $\rightarrow$ try first or seasonal differencing
- **Step 2 – Estimate AR (p) and MA (q):**
  - Use ACF and PACF plots:
    - PACF cuts off $\rightarrow$ AR (p)
    - ACF cuts off $\rightarrow$ MA (q)
  - If both tail off $\rightarrow$ consider mixed model (ARMA)
- **Optional: Use automated tools (e.g., `auto_arima`)**
  - Fast and useful for testing, but always check residuals manually

Choose model orders by combining visual tools, tests, and judgment.

Model Assumptions and Diagnostic Checks

- After fitting a model, check the **residuals** — they should behave like random noise.
- **Check these assumptions:**
  - **No autocorrelation:** ACF of residuals
  - **Normality:** Histogram or Q-Q plot
  - **Constant variance:** Residuals vs fitted should show no pattern



When to Use Machine Learning Models

- Classical models like ARIMA are powerful but have limitations:
  - Assume linear relationships
  - Typically univariate
  - Require stationary data
- Use **machine learning models** when:
  - You have many input features (external variables, lags, time-based features)
  - The data has complex or nonlinear patterns
  - You want to model interactions that classical models can't capture
- ML models are more flexible but also more sensitive to feature choices and overfitting.

ML models don't assume structure — they learn it. But that power requires more care.

Feature Engineering for Time Series

- ML models don't "see" time — we have to create features that represent it.
- Common types of features:
  - **Lag features:** previous values (e.g., $y_{t-1}, y_{t-2}, \dots$)
  - **Rolling statistics:** moving average, moving std (e.g., last 3 months)
  - **Calendar features:** month, day of week, holiday, etc.
  - **Seasonal indicators:** dummy variables for known cycles
  - **External regressors:** other time series (e.g., weather, interest rate)
- Good feature engineering can improve forecast accuracy significantly.

Feature choices define what the model can learn — this is where the real power lies.

ML Forecasting Workflow

- Applying machine learning to time series requires a specific workflow:
- **1. Create features**
  - Lags, rolling stats, calendar variables, exogenous inputs
- **2. Train/test split**
  - Use the latest data as test set (no shuffling!)
  - Often done with time-based cut (e.g., last 12 months)
- **3. Fit model on training set**
  - Random Forest, XGBoost, MLP, etc.
- **4. Make predictions on test set**
  - Evaluate using MAE, RMSE, MAPE
- **5. (Optional) Rolling/recursive forecasting**
  - Re-train or roll forward one step at a time

Time-aware workflows help ML models make meaningful future predictions.

Summary & What's Next

- **What we learned today:**
  - What makes time series different from other data
  - Core components: trend, seasonality, noise
  - How to make a series stationary
  - ARIMA and its extensions (SARIMA, ARIMAX)
  - ACF/PACF for identifying model orders
  - Residual diagnostics for checking model validity
  - How to use ML models for forecasting
  - Importance of feature engineering and time-aware workflows
- **Next: Colab Demo!**
  - Build lag features
  - Train a forecasting model
  - Compare with baseline and ARIMA
  - Visualize performance

Time to put it into practice — let's go to Colab!