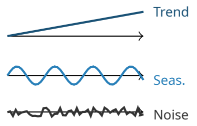
Time Series Models Ultimate Cheat Sheet

End-to-End Guide: From Classical Statistics to Deep Learning

1. INTRODUCTION

Definition: A sequence of data points collected or recorded at specific time intervals.

Components



- **Trend:** Long-term increase or decrease in data.
- **Seasonality:** Repeating patterns at fixed intervals (e.g., daily, monthly).
- **Cyclic:** Fluctuations occurring at irregular intervals (e.g., economic cycles).
- **Noise (Residuals):** Random variation that cannot be explained by the model.

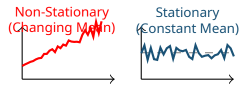
Objectives

- **Forecasting:** Predicting future values based on history.
- **Anomaly Detection:** Identifying unusual data points.
- **Pattern Discovery:** Finding recurring motifs.

2. CHARACTERISTICS

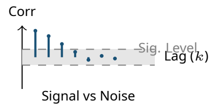
Stationarity

Statistical properties (mean, variance) do not change over time.



- **ADF Test:** Checks for unit root. $p < 0.05$ implies stationary.

Correlation (ACF)



- **Autocorrelation (ACF):** Correlation of a series with its own past values (lags).
- **Partial Autocorrelation (PACF):** Correlation with a lag after removing effects of intermediate lags.

Concepts

- **Lag:** A fixed amount of passing time. Lag 1 is $t-1$.
- **White Noise:** Purely random data with constant mean/variance and zero autocorrelation.

3. DATA PREPARATION

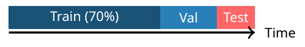
Handling Data

- **Missing timestamps:** Forward fill ($ffill$) or interpolation.
- **Resampling:** Changing frequency (e.g., min to hour).

```
# Resample to daily mean
df_daily = df.resample('D').mean()
# Fill missing
df.fillna(method='ffill', inplace=True)
```

Splitting

Critical: Do NOT shuffle. Use temporal split.



- Train: Past data.
- Test: Future data.

4. CLASSICAL STATISTICAL MODELS

Basic Methods

- **Moving Average (MA):** Average of last k periods. Smooths noise.
- **Exponential Smoothing (ETS):** Weighted average, recent data has more weight.

ARIMA Family

- **AR (AutoRegressive):** Predict using past values. Depends on p (lags).
- **MA (Moving Average):** Predict using past forecast errors. Depends on q (errors).
- **I (Integrated):** Differencing to make data stationary (d).
- **ARIMA(p,d,q):** Combines AR, I, and MA.
- **SARIMA(p,d,q)(P,D,Q)m:** ARIMA + Seasonality (m periods).

5. STATE SPACE / PROBABILISTIC

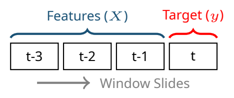
Models

- **Kalman Filter:** Recursive algorithm for estimating state of a dynamic system from noisy measurements.
- **HMM (Hidden Markov):** Systems with unobserved (hidden) states.
- **BSTS (Bayesian Structural):** Decomposes time series into components using Bayesian priors. Good for causal impact analysis.

6. MACHINE LEARNING MODELS

Feature Engineering (Sliding Window)

Convert time series to a supervised learning problem (X, y).



- **Lags:** $t-1, t-2...$ as features.
- **Rolling Window:** Mean/Max of last k days.
- **Date Parts:** Day of week, Month, Hour.

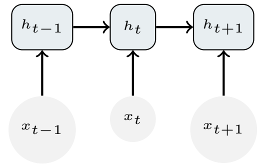
```
df['lag_1'] = df['value'].shift(1)
df['roll_mean'] = df['value'].rolling(7).mean()
```

Algorithms

- **Linear Regression:** Good baseline for trends.
- **XGBoost / LightGBM:** Powerful for non-linear patterns. Handles exogenous variables well.
- **Random Forest:** Robust to outliers, no scaling needed.

7. DEEP LEARNING MODELS

RNN Family



$$\begin{tikzcd} h_{t-1} \arrow[r] & h_t \arrow[r] & h_{t+1} \\ x_{t-1} \arrow[u] & x_t \arrow[u] & x_{t+1} \arrow[u] \end{tikzcd}$$

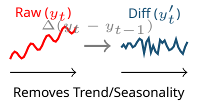
- **RNN:** Process sequences step-by-step. Prone to vanishing gradient.
- **LSTM:** Adds "forget gate" to retain long-term dependencies. Standard for time series.
- **GRU:** Simplified LSTM, faster training.

Modern Architectures

- **TCN (Temporal ConvNet):** Dilated 1D convolutions. Parallelizable, often beats LSTM.
- **Transformers:** Attention mechanisms (Self-Attention). State-of-the-art for long sequences (e.g., Informer, Autoformer).

8. SEASONALITY & TRENDS

Techniques (Differencing)



- **Differencing:** Subtracting t from $t-1$ removes trend.
- **Seasonal Differencing:** Subtracting t from $t-m$ removes seasonality.
- **Decomposition:** Separating Additive ($T + S + R$) or Multiplicative ($T \times S \times R$) components.
- **Fourier Terms:** Using sin/cos features to model cyclic patterns in linear models.

9. IMPLEMENTATION (PYTHON)

Statsmodels (ARIMA)

```python
from statsmodels.tsa.arima.model import ARIMA
# Fit ARIMA(1,1,1)
model = ARIMA(train, order=(1,1,1))
res = model.fit()
preds = res.forecast(steps=10)
```

SARIMA

```python
from statsmodels.tsa.statespace.sarimax import SARIMAX
# Monthly seasonality (12)
model = SARIMAX(train, order=(1,1,1),
                seasonal_order=(1,1,1,12))
```

PyTorch (LSTM Snippet)

```python
class LSTMModel(nn.Module):
    def __init__(self, input_sz, hidden_sz):
        super().__init__()
        self.lstm = nn.LSTM(input_sz, hidden_sz)
        self.linear = nn.Linear(hidden_sz, 1)
    def forward(self, x):
        lstm_out, _ = self.lstm(x)
        return self.linear(lstm_out[:, -1, :])
```

10. PERFORMANCE METRICS

- **MAE (Mean Absolute Error):** Average magnitude of errors. Robust to outliers.
- **MSE (Mean Squared Error):** Penalizes large errors heavily.
- **RMSE (Root MSE):** Same units as target variable.
- **MAPE:** Percentage error. Fails if actuals are 0.
- **MASE:** Compares error to a naive baseline (e.g., persistence). Scale-independent.

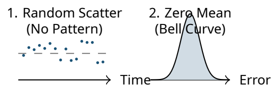
```python
from sklearn.metrics import mean_squared_error,
       mean_absolute_error
import numpy as np

def get_metrics(y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) *
       100
    return {"MAE": mae, "RMSE": rmse, "MAPE": mape}
```

11. DIAGNOSTICS

Residual Analysis

Ideally, residuals (Actual - Predicted) should be white noise (random, centered at 0).



- **Plot Residuals:** Should look random around 0.
- **ACF of Residuals:** Should show no significant correlation.
- **Ljung-Box Test:** Statistical test for white noise.
- **Normality:** Histogram of residuals should be Bell curve.

Backtesting

Simulating real-world forecasting.

- **Expanding Window:** Train grows, test moves forward.
- **Sliding Window:** Train size constant, moves forward.

12. HYPERPARAMETER TUNING

Strategies

- **Grid Search (ARIMA):** Try all (p, d, q) combos based on AIC score.
- **Auto-ARIMA:** Automated stepwise search (library: `pmdarima`).
- **Time Series CV:** Use `TimeSeriesSplit` (scikit-learn), not K-Fold.

13. PRACTICAL TIPS

- **Data Leakage:** Never use future data (e.g., global mean) to impute past missing values.
- **Baselines:** Always beat a "Naive" forecast (predicting today's value for tomorrow) before trying complex models.
- **Concept Drift:** Statistical properties change over time. Retrain frequently.
- **Stationarity:** Deep learning handles non-stationarity better than ARIMA, but scaling (MinMax/Standard) is crucial for DL.

14. REAL WORLD USE CASES

- **Demand:** Retail inventory planning.
- **Energy:** Grid load balancing (hourly).
- **Finance:** Stock volatility, algorithmic trading.
- **IoT:** Predicting machine failure (anomaly) from sensor vibrations.

15. SUMMARY MATRIX

Classical (ARIMA): Best for short-term, univariate, small data, clear seasonality.

ML (XGBoost): Best for tabular data, complex lags, exogenous variables, tabular regression formulation.

Deep Learning (LSTM/TCN): Best for massive datasets, multiple series, complex non-linear dependencies, raw sequence data.