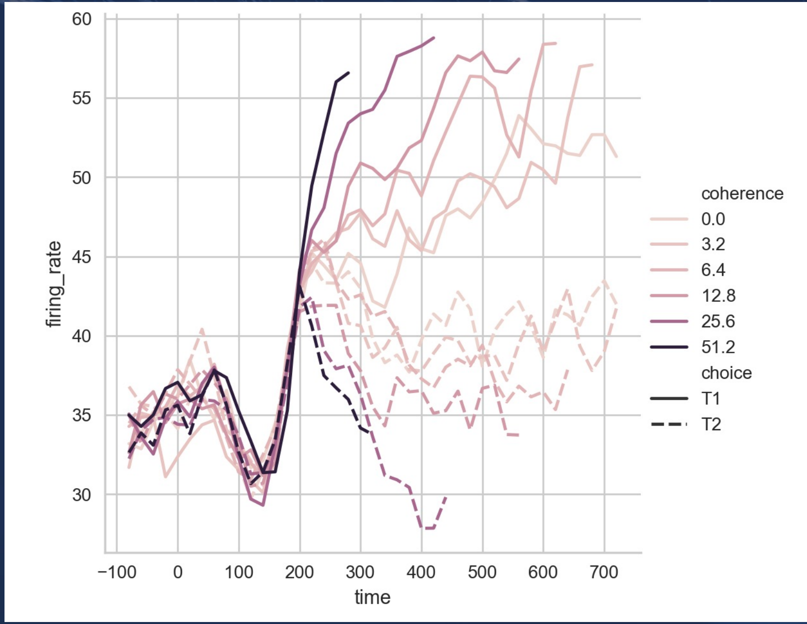
DS5003
Feature Engineering for Time Series Data

An Overview – July 2025

Harold Haugen

Introduction

- Background on Time Series
- Feature Engineering Methods for Time Series
- Example Structure



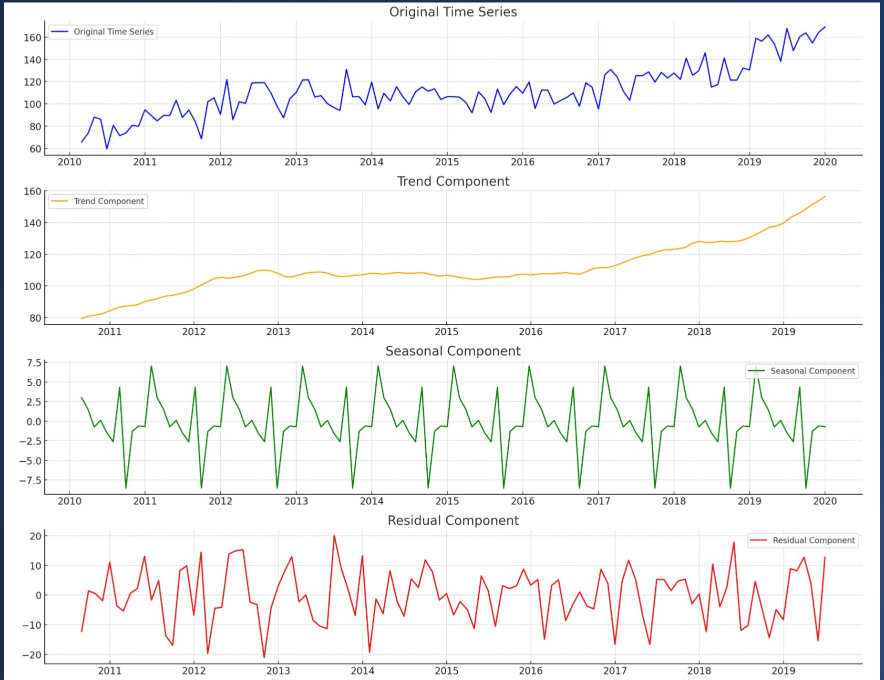
Seaborn Tutorial – Time Series

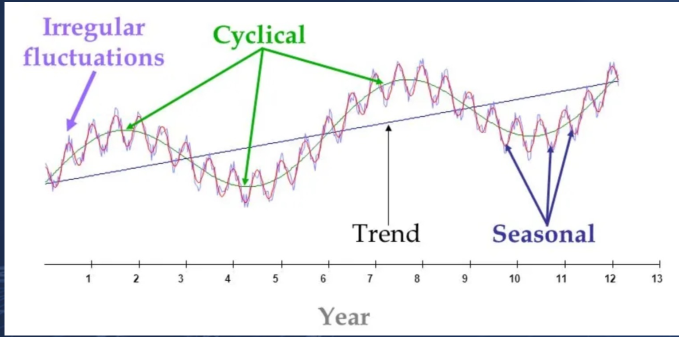
Time Series Background

- Time Series - Data represented in a sequence successively; at regular or irregularly frequencies
- Temporal Characteristics
    - Temporal Dependency – data values at a point in time depends on the values at prior time points
    - Temporal Order – data must be arranged in chronological / sequential order
- Structure – Univariate and Multivariate
    - Multivariate - e.g., vital signs, lab results, chart events…

Time Series Background

- **Four Key Components:**
  - **Trend** – long-term, persistent behavior of a time series.
  - **Seasonality** – repeating patterns at regular fixed time intervals, corresponding to specific seasons, months, weeks , etc.
  - **Cyclical Patterns** – long-term fluctuations that are often linked to non calendar cycles, potentially spanning years at varying lengths.
  - **Noise** – residual / error, random variation present in time series data, or the remaining noise after removing trend and seasonality.





Feature Engineering

Date / Time Features

- Developing slices of date / time data as new features; to provide additional insights
- E.g., Hour, Month, Day of Week, Season, Holiday, and Binary Indicators 0/1 (before noon / after noon)
- Healthcare - heart rate / vital signs, lab results

	load	temp	hour	month	dayofweek
2012-01-01 00:00:00	2,698.00	32.00	0	1	1
2012-01-01 01:00:00	2,558.00	32.67	1	1	1
2012-01-01 02:00:00	2,444.00	30.00	2	1	1
2012-01-01 03:00:00	2,402.00	31.00	3	1	1
2012-01-01 04:00:00	2,403.00	32.00	4	1	1

Feature Engineering

Lag Features

- Capturing past values of the times series through ‘sliding windows’, as new features associated to the target variable
  [Temporal Dependencies]
- Nested Lag – Grouped values by period, e.g., total orders for the previous 4, 8, 12 hours, etc.

```
[2]: # Sample time series data
data = {'value': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}
df = pd.DataFrame(data)

# Create lag features
df['lag1'] = df['value'].shift(1)
df['lag2'] = df['value'].shift(2)

print(df)
```

	value	lag1	lag2
0	1	NaN	NaN
1	2	1.0	NaN
2	3	2.0	1.0
3	4	3.0	2.0
4	5	4.0	3.0
5	6	5.0	4.0
6	7	6.0	5.0
7	8	7.0	6.0
8	9	8.0	7.0
9	10	9.0	8.0

pandas.DataFrame.shift

Feature Engineering

Rolling Windows

- Computation of key statistics (e.g., Mean, Standard Deviation) over a prior set time including T+0 [Temporal Dependencies]

Expanding Windows

- Window continuously expands with time / includes all prior values [Temporal Dependencies]

```
[3]: # Calculate rolling mean and standard deviation
df['rolling_mean'] = df['value'].rolling(window=3).mean()
df['rolling_std'] = df['value'].rolling(window=3).std()

print(df)
```

	value	lag1	lag2	rolling_mean	rolling_std
0	1	NaN	NaN	NaN	NaN
1	2	1.0	NaN	NaN	NaN
2	3	2.0	1.0	2.0	1.0
3	4	3.0	2.0	3.0	1.0
4	5	4.0	3.0	4.0	1.0
5	6	5.0	4.0	5.0	1.0
6	7	6.0	5.0	6.0	1.0
7	8	7.0	6.0	7.0	1.0
8	9	8.0	7.0	8.0	1.0
9	10	9.0	8.0	9.0	1.0

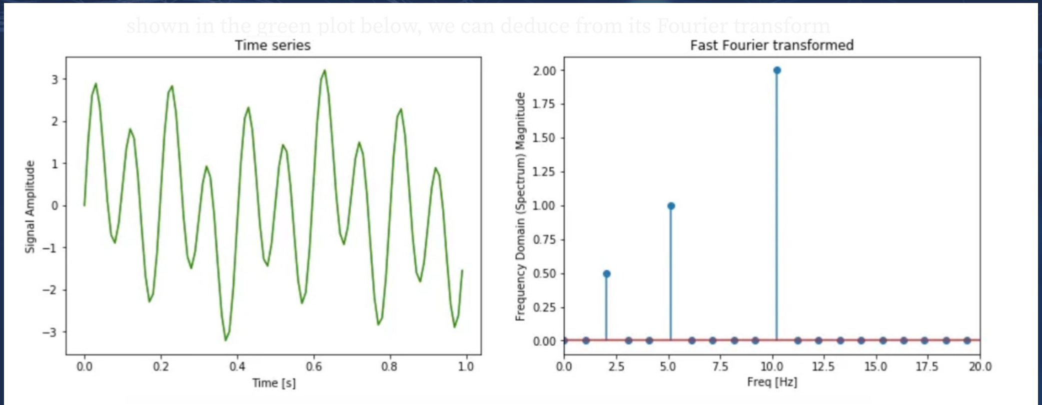
	min	mean	max	load+1
2012-01-01 00:00:00	2,698.00	2,698.00	2,698.00	2,558.00
2012-01-01 01:00:00	2,558.00	2,628.00	2,698.00	2,444.00
2012-01-01 02:00:00	2,444.00	2,566.67	2,698.00	2,402.00
2012-01-01 03:00:00	2,402.00	2,525.50	2,698.00	2,403.00
2012-01-01 04:00:00	2,402.00	2,501.00	2,698.00	2,453.00
2012-01-01 05:00:00	2,402.00	2,493.00	2,698.00	2,560.00
2012-01-01 06:00:00	2,402.00	2,502.57	2,698.00	2,719.00
2012-01-01 07:00:00	2,402.00	2,529.62	2,719.00	2,916.00
2012-01-01 08:00:00	2,402.00	2,572.56	2,916.00	3,105.00
2012-01-01 09:00:00	2,402.00	2,625.80	3,105.00	3,174.00

pandas.DataFrame.rolling

Feature Engineering

Fourier Transforms

- Decomposes a Time Signal into a Frequency Domain
  - Simpler Terms – decomposes a possibly messy signal into pure sine / cosine waves
- Fourier Transforms will assist in revealing:
  - Periodic Behavior not seen in time domain
  - Dominant Cycles/ Frequencies (as features)
  - Isolation of certain frequency bands / Filtering out the Noise
  - Simplify complex signals to a few key frequencies [Temporal Dependencies]



- Time series reveals three components at different frequencies of 2, 5, and 10 HZ
- Numpy fft and fftreq

Feature Engineering

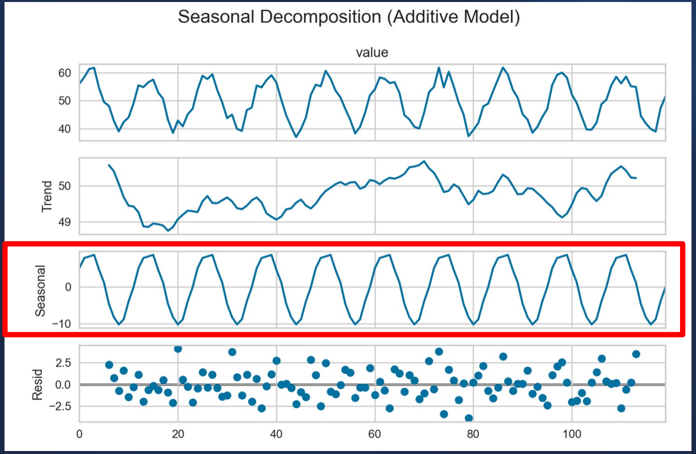
Seasonality Feature

- Decomposition Models (Additive & Multiplicative)
- Breakdown a time series into the Trend + Seasonality + Residual / Noise components
- Isolate Seasonality as a Feature / supports the model learn seasonality explicitly

```
[31]: # Decompose the series (monthly → period=12)
      decomp = sm.tsa.seasonal_decompose(df_season['value'], model='additive', period=12)

[36]: # Extract and display the seasonal component
      df_season['seasonal'] = decomp.seasonal
      df_season.head(20)
```

	date	value	seasonal
0	2010-01-31	55.993428	5.105323
1	2010-02-28	58.383725	7.910412
2	2010-03-31	61.295377	8.329753
3	2010-04-30	61.706314	8.762467
4	2010-05-31	54.531693	4.690788
5	2010-06-30	49.531726	1.210407
6	2010-07-31	48.158426	-4.648775
7	2010-08-31	42.874615	-8.249412
8	2010-09-30	39.061051	-10.249620
9	2010-10-31	42.424866	-8.855133
10	2010-11-30	44.073165	-3.936090



statsmodels.tsa.seasonal.seasonal_decompose

Example Structure

- Times Series Structure
- Uses Seasonality, Lag and Rolling Window Statistics Features (3-Mo)
- Y-Target may be any time in the future, e.g., T+3 Months.
- Model data must begin were the Lag Ends, no NaNs.

	date	value	seasonal	lag1	lag2	lag3	rolling_mean	rolling_std	y_target
0	2010-01-31	55.993428	5.105323	NaN	NaN	NaN	NaN	NaN	70.196688
1	2010-02-28	58.383725	7.910412	55.993428	NaN	NaN	NaN	NaN	79.579267
2	2010-03-31	61.295377	8.329753	58.383725	55.993428	NaN	NaN	NaN	89.000534
3	2010-04-30	61.706314	8.762467	61.295377	58.383725	55.993428	59.344711	2.679357	33.799516
4	2010-05-31	54.531693	4.690788	61.706314	61.295377	58.383725	58.979277	3.313444	37.558295
5	2010-06-30	49.531726	1.210407	54.531693	61.706314	61.295377	56.766278	5.838059	9.398194
6	2010-07-31	48.158426	-4.648775	49.531726	54.531693	61.706314	53.482040	6.128793	57.828014
7	2010-08-31	42.874615	-8.249412	48.158426	49.531726	54.531693	48.774115	4.792593	3.594227
8	2010-09-30	39.061051	-10.249620	42.874615	48.158426	49.531726	44.906455	4.839613	46.559802
9	2010-10-31	42.424866	-8.855133	39.061051	42.874615	48.158426	43.129740	3.759603	54.264463
10	2010-11-30	44.073165	-3.936090	42.424866	39.061051	42.874615	42.108424	2.147392	28.654125
11	2010-12-31	49.068540	-0.070119	44.073165	42.424866	39.061051	43.656906	4.167288	59.083326
12	2011-01-31	55.483925	5.105323	49.068540	44.073165	42.424866	47.762624	5.871617	3.050025
13	2011-02-28	54.833694	7.910412	55.483925	49.068540	44.073165	50.864831	5.367845	3.734819
14	2011-03-31	56.550164	8.329753	54.833694	55.483925	49.068540	53.984081	3.352546	82.260056
15	2011-04-30	57.535679	8.762467	56.550164	54.833694	55.483925	56.100865	1.189804	36.019064

References

1. **Rahul Holla -** Advanced Feature Engineering for Time Series Data *[Medium]*
2. **Francesca Lazzeri -** Introduction to feature engineering for time series forecasting *[Medium]*
3. **Roshmita Dey -** Understanding Time Series Data and Key Concepts *[Medium]*
4. **Donato_TH -** The Journey into Time Series: Dancing with Trends and Seasons *[Medium]*
5. **Khairul Omar -** Deconstructing Time Series using Fourier Transform *[Medium]*
6. **Nayeem Islam** - Comprehensive Guide to Time Series Data Analytics and Forecasting with Python *[Medium]*

Thank you