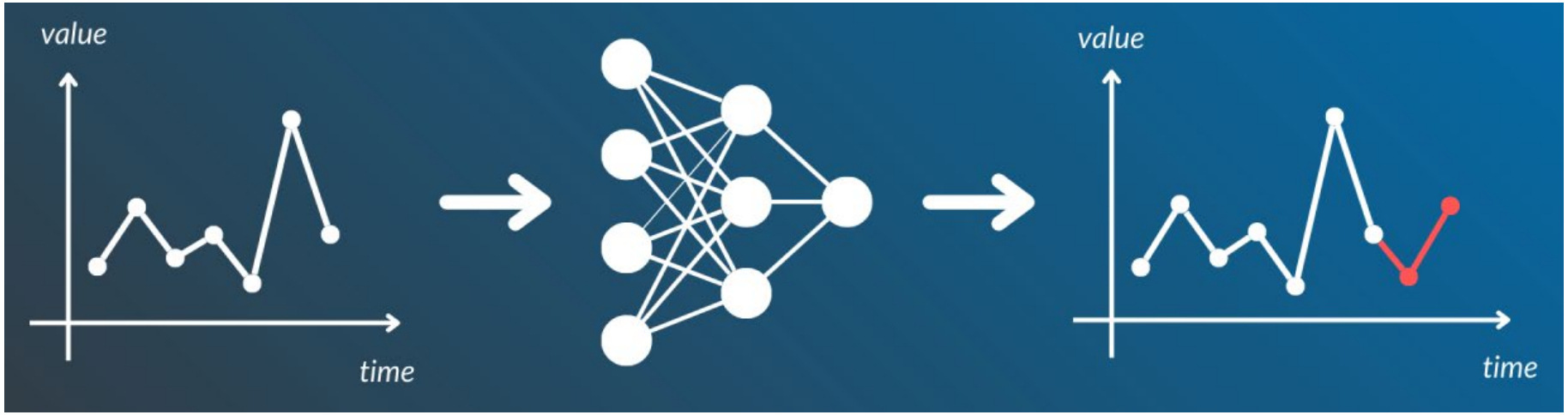
Time Series — Learning from the Past to Predicting the Future

V2. Updated Oct 27



https://www.linkedin.com/pulse/data-windowing-technique-used-time-series-forecasting-alejandro/

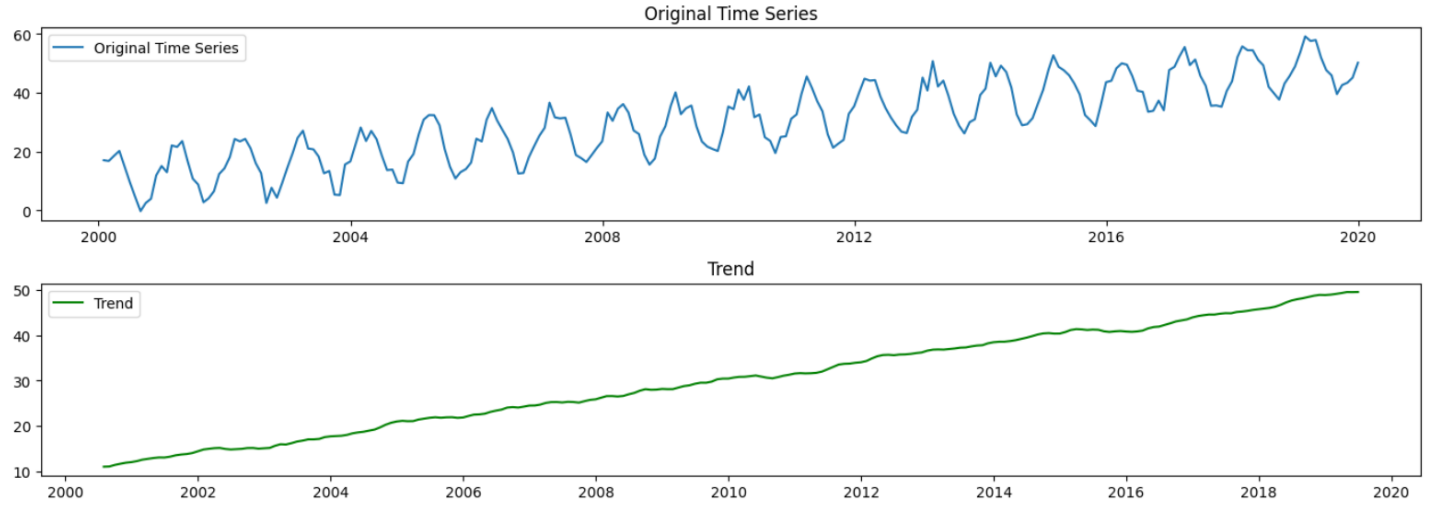
Time Series Basics – TimeSeries_basics.ipynb

- Trend
- Seasonality
- Cyclicality
- Noise

https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

Time Series Basics - Trend

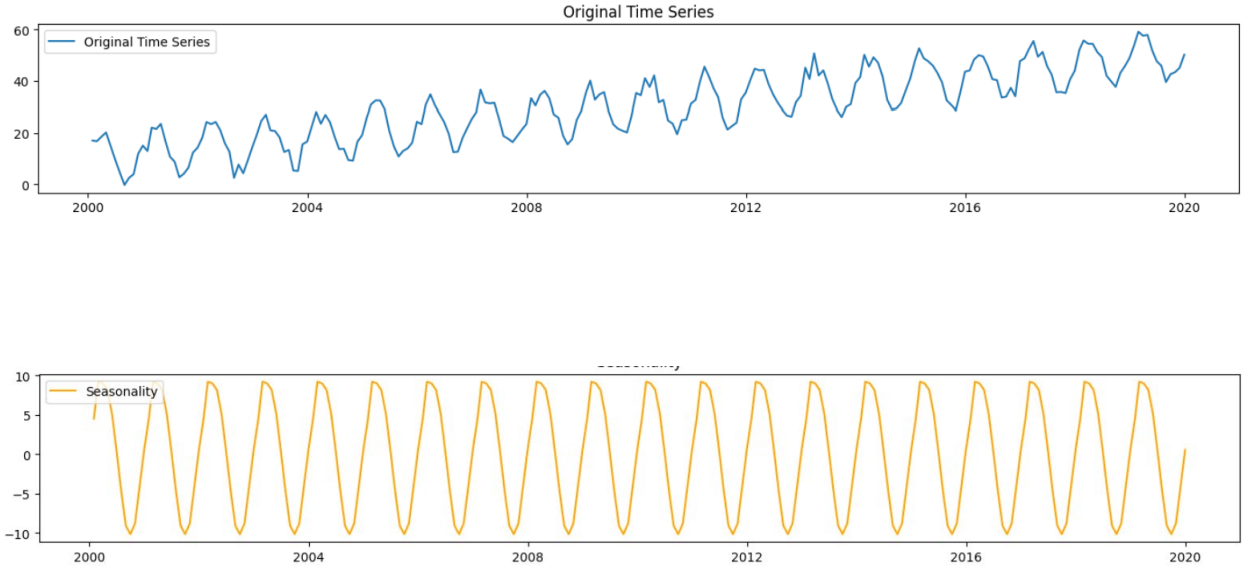
- Similar to how we use “trend” in day-to-day conversations, trend in the context of time series also refers to the directional increase or decrease of the data over time.
- For example:
  - if we record the height of an individual from age of 3 to 18 annually, we will see an overall increasing trend in the data.
  - If we continue recording the height through the age of, let’s say 50, we will see that the increase or inclining trend continues up to a certain age and then once growth slows down and eventually stops, the height almost flattens.



https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

Time Series Basics - Seasonality

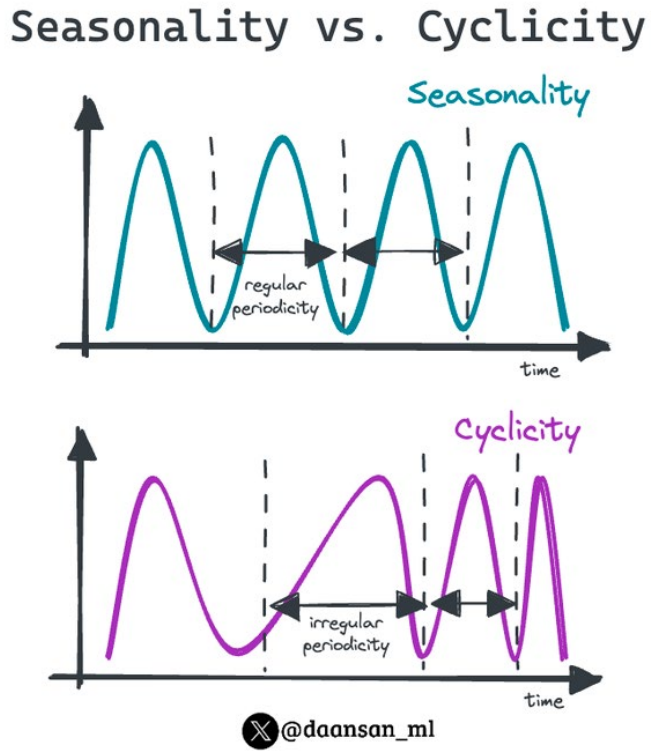
- As the name suggests, refers to regular patterns in the underlying data during a specific period of time.
- For example, if we look at monthly weather data for Seattle over the past 10 years, we will see how the temperatures tend to increase during spring and summer time and then decrease during fall and winter months, which is repeated annually — this is a seasonality.



https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

Time Series Basics - Cyclicality

- This one is similar to seasonality but it does **not** happen in a fixed schedule.
- For example, economic business cycles do not follow and repeat a regular time line, which makes them harder to forecast.

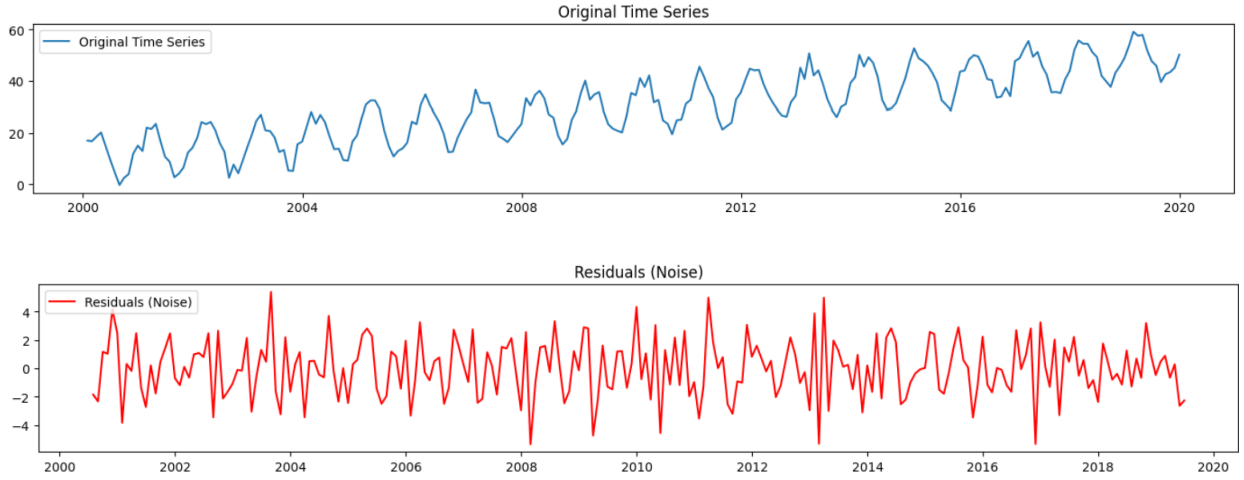


https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

https://x.com/daansan_ml/status/1777984758729134213

Time Series Basics - Noise

- Noise or residuals represent irregular fluctuations that do not follow a pattern.
- For the more statistically-inclined readers, noise is the random variation in the data, which makes predictions challenging.
- For example, a sudden surge in the sales data due to a one-day promotion is considered noise.
- We usually remove noise from the data to better understand the underlying patterns.



https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

HW5: Find optimal LSTM or GRU models (HPO problem) for a multivariate time series dataset - to predict Total Nitrogen Oxides, NOx

Dataset to use for our example code to predict Total Nitrogen Oxides, **NOx** (1 /2)

- Open source air quality data set from UC Irvine machine learning depository, available under a CC BY 4.0 license
- https://archive.ics.uci.edu/dataset/360/air+quality
- Contains 9,358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device.
- The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded from March 2004 to February 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses.
- Multivariate observation timeseries (cf. Univariate)

https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

Dataset to use for our example code to predict Total Nitrogen Oxides, **NOx** (2 /2)

- Ground Truth hourly averaged concentrations for CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx) and Nitrogen Dioxide (NO2) and were provided by a co-located reference certified analyzer.
- Evidences of cross-sensitivities as well as both concept and sensor drifts are present as described in the following paper (citation required) eventually affecting sensors concentration estimation capabilities.

  Vito, Saverio De, Ettore Massera, Marco Piga, Luca Martinotto and Girolamo Di Francia. "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario." Sensors and Actuators B-chemical 129 (2008): 750-757.

  https://www.semanticscholar.org/paper/a90a54a39ff934772df57771a0012981f355949d
- Missing values are tagged with **-200** value.
- This dataset can be used exclusively for research purposes. Commercial purposes are fully excluded.

https://towardsdatascience.com/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

(7485, 13)

DateTime	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-03-10 18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
2004-03-10 19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2004-03-10 20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
2004-03-10 21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
2004-03-10 22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888
2004-03-10 23:00:00	1.2	1197.0	38.0	4.7	750.0	89.0	1337.0	96.0	1393.0	949.0	11.2	59.2	0.7848
2004-03-11 00:00:00	1.2	1185.0	31.0	3.6	690.0	62.0	1462.0	77.0	1333.0	733.0	11.3	56.8	0.7603
2004-03-11 01:00:00	1.0	1136.0	31.0	3.3	672.0	62.0	1453.0	76.0	1333.0	730.0	10.7	60.0	0.7702
2004-03-11 02:00:00	0.9	1094.0	24.0	2.3	609.0	45.0	1579.0	60.0	1276.0	620.0	10.7	59.7	0.7648
2004-03-11 03:00:00	0.6	1010.0	19.0	1.7	561.0	-200.0	1705.0	-200.0	1235.0	501.0	10.3	60.2	0.7517

(Index)

AirQuality_LSTM_GRU.ipynb
on Canvas

"Lags (= past time step)" in Time Series Analysis

- **lags** refer to the past values in a dataset used to help forecast future values. Specifically, a "lag" represents how many time steps back a particular observation is from the current time step.
- Lags help capture patterns, trends, and dependencies within the data that span multiple time steps, which is especially useful for forecasting.
- **Lag Notation**:
  - "Lag 1" refers to the value at the previous time step.
  - "Lag 2" refers to the value from two time steps back, and so on.
  - If today is *t*, then Lag 1 would be the value at *t-1*, Lag 2 would be at *t-2*, etc.

Example

Let’s say you have a daily temperature series, and you want to predict today's temperature. You could use:

- Temperature, 1 day ago (Lag 1)
- Temperature, 2 days ago (Lag 2)
- Temperature, 7 days ago (Lag 7, weekly lag)

Lagged Feature Example

A temperature time series

Day	Temp	Lag 1	Lag2
Jan 1	29°F	-	-
Jan 2	30°F	29°F	-
Jan 3	32°F	30°F	29°F
Jan 4	33°F	32°F	30°F

Why Lagged Features Are Useful

- **Autocorrelation**: Time series data often exhibit autocorrelation, meaning values at one point are correlated with values at previous points. Lag features capture this structure.
- **Seasonality** and **Trends**: For data with recurring patterns (like daily or yearly cycles), lag features can help the model learn and account for these cycles.
- **Stationarity**: In some cases, lag features help make the data more stationary (stable mean and variance over time), which is beneficial for modeling.

(7485, 13)

DateTime	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-03-10 18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
2004-03-10 19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2004-03-10 20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
2004-03-10 21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
2004-03-10 22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888
2004-03-10 23:00:00	1.2	1197.0	38.0	4.7	750.0	89.0	1337.0	96.0	1393.0	949.0	11.2	59.2	0.7848
2004-03-11 00:00:00	1.2	1185.0	31.0	3.6	690.0	62.0	1462.0	77.0	1333.0	733.0	11.3	56.8	0.7603
2004-03-11 01:00:00	1.0	1136.0	31.0	3.3	672.0	62.0	1453.0	76.0	1333.0	730.0	10.7	60.0	0.7702
2004-03-11 02:00:00	0.9	1094.0	24.0	2.3	609.0	45.0	1579.0	60.0	1276.0	620.0	10.7	59.7	0.7648
2004-03-11 03:00:00	0.6	1010.0	19.0	1.7	561.0	-200.0	1705.0	-200.0	1235.0	501.0	10.3	60.2	0.7517

(Index)

Number of lags = 3

CO	PT.S1	NMHC	C6H6	PT.S2	PT.S3	NO2	PT.S4	PT.S5	T	RH	AH	Lag1	Lag2	Lag3	y
2.2	1376	80	9.2	948	1092	122	1584	1203	11	60	0.786	131	103	166	172

(7482, 15)

Why 7482? The first 3 rows are NOT used for training

(7485, 13)

DateTime	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-03-10 18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
2004-03-10 19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2004-03-10 20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
2004-03-10 21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
2004-03-10 22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888
2004-03-10 23:00:00	1.2	1197.0	38.0	4.7	750.0	89.0	1337.0	96.0	1393.0	949.0	11.2	59.2	0.7848
2004-03-11 00:00:00	1.2	1185.0	31.0	3.6	690.0	62.0	1462.0	77.0	1333.0	733.0	11.3	56.8	0.7603
2004-03-11 01:00:00	1.0	1136.0	31.0	3.3	672.0	62.0	1453.0	76.0	1333.0	730.0	10.7	60.0	0.7702
2004-03-11 02:00:00	0.9	1094.0	24.0	2.3	609.0	45.0	1579.0	60.0	1276.0	620.0	10.7	59.7	0.7648
2004-03-11 03:00:00	0.6	1010.0	19.0	1.7	561.0	-200.0	1705.0	-200.0	1235.0	501.0	10.3	60.2	0.7517

(Index)

Number of lags = 3

CO	PT.S1	NMHC	C6H6	PT.S2	PT.S3	NO2	PT.S4	PT.S5	T	RH	AH	Lag1	Lag2	Lag3	y
2.2	1376	80	9.2	948	1092	122	1584	1203	11	60	0.786	131	103	166	172
1.6	1272	51	6.5	836	1205	116	1490	1110	11.2	59.6	0.788	172	131	103	131

(7482, 15)

(7485, 13)

DateTime	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-03-10 18:00:00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
2004-03-10 19:00:00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2004-03-10 20:00:00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
2004-03-10 21:00:00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
2004-03-10 22:00:00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888
2004-03-10 23:00:00	1.2	1197.0	38.0	4.7	750.0	89.0	1337.0	96.0	1393.0	949.0	11.2	59.2	0.7848
2004-03-11 00:00:00	1.2	1185.0	31.0	3.6	690.0	62.0	1462.0	77.0	1333.0	733.0	11.3	56.8	0.7603
2004-03-11 01:00:00	1.0	1136.0	31.0	3.3	672.0	62.0	1453.0	76.0	1333.0	730.0	10.7	60.0	0.7702
2004-03-11 02:00:00	0.9	1094.0	24.0	2.3	609.0	45.0	1579.0	60.0	1276.0	620.0	10.7	59.7	0.7648
2004-03-11 03:00:00	0.6	1010.0	19.0	1.7	561.0	-200.0	1705.0	-200.0	1235.0	501.0	10.3	60.2	0.7517

(Index)

Number of lags = 3

CO	PT.S1	NMHC	C6H6	PT.S2	PT.S3	NO2	PT.S4	PT.S5	T	RH	AH	Lag1	Lag2	Lag3	*y*
2.2	1376	80	9.2	948	1092	122	1584	1203	11	60	0.786	131	103	166	172
1.6	1272	51	6.5	836	1205	116	1490	1110	11.2	59.6	0.788	172	131	103	131
…															…
1.0	1136	31	3.3	672	1453	76	1333	730	10.7	60	0.770	62.0	89.0	131	62

(7482, 15)

12+3=15

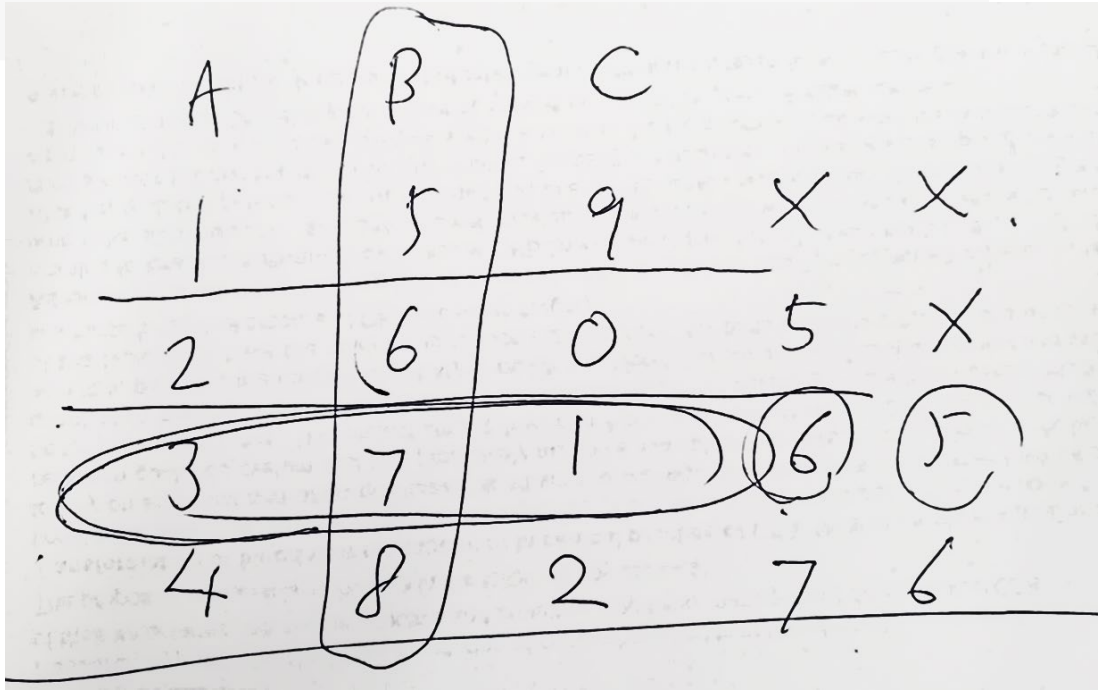
```python
# dataframe shift() test
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8], 'C': [9, 0, 1, 2]})
print(df)
# Shift the data down by 1 row
df_shifted_down = df.shift(1)
print(df_shifted_down)
```

```
   A  B  C
0  1  5  9
1  2  6  0
2  3  7  1
3  4  8  2
     A    B    C
0  NaN  NaN  NaN
1  1.0  5.0  9.0
2  2.0  6.0  0.0
3  3.0  7.0  1.0
```

```
n = 2
df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8], 'C': [9, 0, 1, 2]})
for i in range(1, n + 1):
    df[f'lag_{i}'] = df['B'].shift(i)
df_shifted_down = df.shift(1)
print(df_shifted_down)
```

```
     A    B    C  lag_1  lag_2
0  NaN  NaN  NaN    NaN    NaN
1  1.0  5.0  9.0    NaN    NaN
2  2.0  6.0  0.0    5.0    NaN
3  3.0  7.0  1.0    6.0    5.0
```



```python
n_lags = 3

def create_lagged_features_lstm(data, n_lags):
    lagged_data = data.copy()
    for i in range(1, n_lags + 1):
        lagged_data[f'NOx_lag_{i}'] = lagged_data['NOx(GT)'].shift(i)
    lagged_data = lagged_data.dropna()
    return lagged_data
```

12+3 = 15 cols

CO	PT.S1	NMHC	C6H6	PT.S2	PT.S3	NO2	PT.S4	PT.S5	T	RH	AH	Lag1	Lag2	Lag3
2.2	1376	80	9.2	948	1092	122	1584	1203	11	60	0.786	131	103	166
1.6	1272	51	6.5	836	1205	116	1490	1110	11.2	59.6	0.788	172	131	103
…														
1.0	1136	31	3.3	672	1453	76	1333	730	10.7	60	0.770	62.0	89.0	131

Note: **in some cases, you *explicitly insert* lag features**

- That happens when you're using **models that do not natively handle sequences**, such as:
  - Linear regression
  - Random forest
  - XGBoost
  - Tabular deep networks (MLP)
- In these cases, you manually create **lag columns** and append them to the training and test sets as extra features.

AirQuality_LSTM_GRU.ipynb
AirQuality_LSTM_GRU_f25a.ipynb
on Canvas

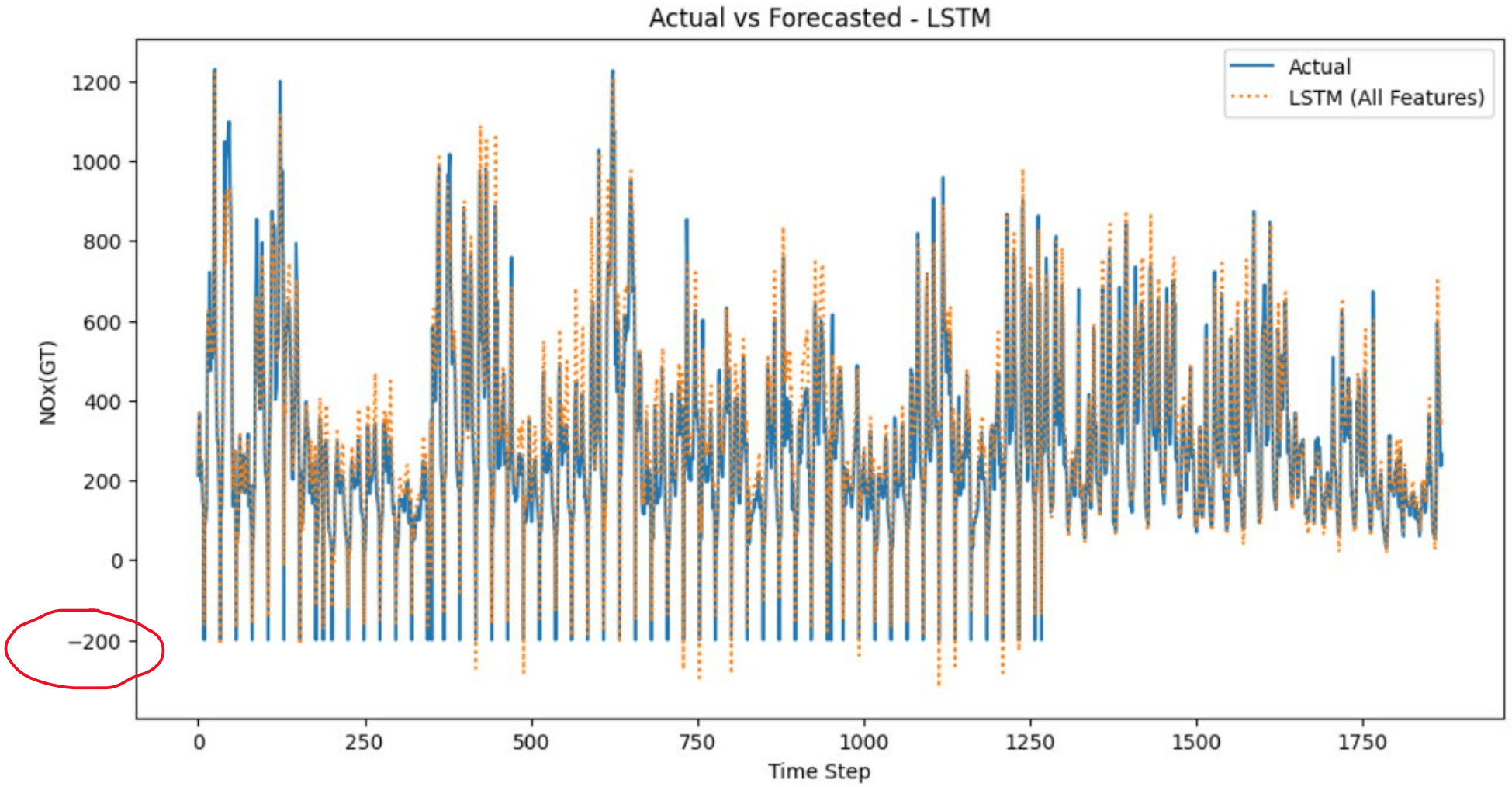
One of the class projects to find optimized models

Farzad Nobar, Time Series — From Analyzing the Past to Predicting the Future, https://medium.com/data-science/time-series-from-analyzing-the-past-to-predicting-the-future-249ab99ec52d

```
model_all_features = Sequential()
model_all_features.add(LSTM(50, activation='relu', input_shape=(n_lags, X_train_lstm_all_features.shape[2]))) # 3, 5
model_all_features.add(Dense(1))
model_all_features.compile(optimizer='adam', loss='mse')
```

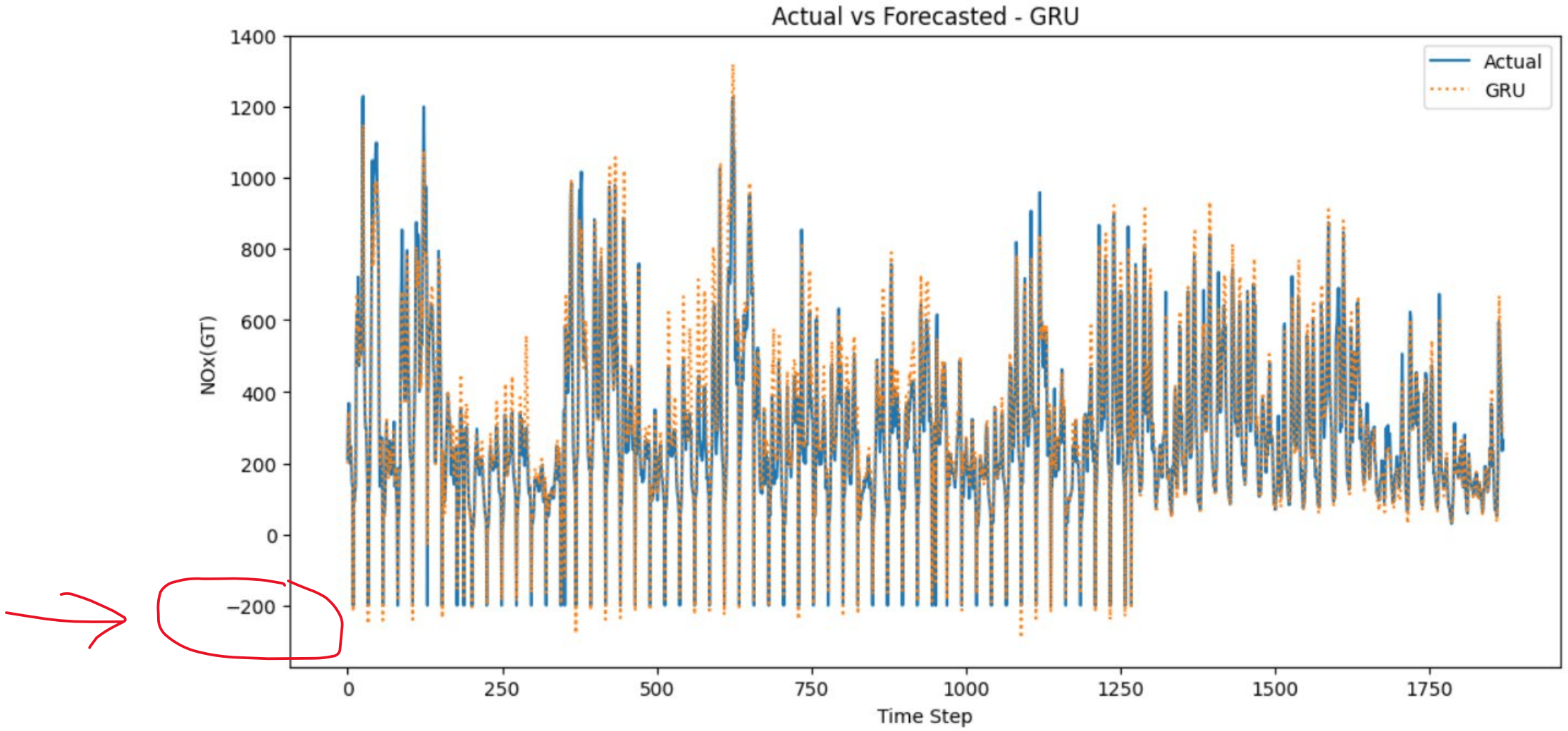
3×5=15

```
RMSE calculated from m.evaluate: 78.19492183499673
```



```
# build GRU
model_gru_all_features = Sequential()
model_gru_all_features.add(GRU(60, activation='relu', input_shape=(n_lags, X_train_gru_all_features.shape[2])))
model_gru_all_features.add(Dense(1))  # Output layer predicting one value (NOx(GT))
model_gru_all_features.compile(optimizer='adam', loss='mse')
```

```
Root Mean Squared Error (RMSE) for GRU (All Features): 67.7883385066943
```



How to handle missing values (-200)?

1. Dropping - removing the row!
2. Forward or backward fill - Use the previous (forward fill) or next (backward fill) valid value to fill the missing data. This works well for time series with slow changes.
3. Interpolation: Use linear, spline, or polynomial interpolation to estimate missing values based on neighboring data points.
4. Imputation - Fill missing values with mean, median, mode, or using predictive models.

For 2-4 above, an Indicator Feature can be added. It is a binary flag that denotes whether a particular value in the dataset is missing or made up. It is an additional **column** added to the dataset, where:

- 1 (True) indicates a missing value (or value is made up)
- 0 (False) indicates the presence of a value.

Dropping using Panda

```
data_cleaned = data_cleaned.replace(-200, np.nan)
data_cleaned = data_cleaned.dropna()
```

Forward fill, next valid value

```
# forward fill, next valid value
data_cleaned = data_cleaned.replace(-200, np.nan).ffill()
```

Interpolation: Use linear, spline, or polynomial interpolation to estimate missing values based on neighboring data points.

```
# values with the average (the linearly calculated value)
# between the previous and next valid points.
data_cleaned = data_cleaned.replace(-200, np.nan).interpolate()
```

How to Tune LSTM Hyperparameters with Keras for Time Series Forecasting

- Please read https://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting (Shampoo Sales dataset)
- Univariate (Usually DL does not perform better than ML)
- Possible HPs
  1. # epochs
  2. Batch size
  3. # neurons
  4. Layers
  5. # of lags
  6. Optimizer
  7. Loss function
  8. Dropout rate
  9. Validation patience for Early stopping
  10. Regularization

HW5 Requirements

1. The given code was not using a separate validation dataset. Using the test dataset for validation is not recommended. It can lead to overfitting the model to the test set and provide a biased evaluation of the model's performance on unseen data. **Use a separate validation set** during training for hyperparameter tuning and early stopping, and keep the test set aside for a final, unbiased evaluation of the trained model.
2. Hyperparameter optimization using either LSTM or GRU
3. Must use at least ~~7~~ 6 parameters (See slide 26)
4. Must handle missing values (-200)
5. Use ES(1+1) with 1/5 rule

Recommended way for validation instead of creating a separate validation dataset.

```
# split the data (80% train, 20% test)
train_size = int(len(data_cleaned) * 0.8)
train_data = data_cleaned[:train_size]
test_data = data_cleaned[train_size:]

…

history_gru_all_features =
  model_gru_all_features.fit(X_train_all_features_4m,
                             y_train,
                             epochs=150,
                             batch_size=32,
                             #validation_data=(X_val_all_features-4m, y_val),
                             validation_split=0.2,
                             callbacks=[early_stopping],
                             verbose=0)
```