https://github.com/dlevonian/madrid-weather/blob/master/data_leakage_note.pdf

Note on Data Leakage

In time series forecasting, one of the dangers is data leakage, when the predictive model happens to have access to the future timestep it is trying to predict.

Data leakage contaminates the predictions by artificially inflating their accuracy, which in turn may lead to a whole host of problems, from wrong research decisions to wrong expectations in real-time production. This happens all the time – the algorithms are complex, new data sources are constantly emerging, array indexing is different across languages, etc.

The target variables can leak into the model at the inference stage, or both during training and inference. Preventing data leakage during training starts with separating and completely isolating the test time series at the very start – ideally, even before the preprocessing steps such as data imputation or scaling.

The clean train/validation/test split is necessary but far from sufficient. It does not protect us from the algorithm itself being faulty (due to any of the reasons above) and data leaking during the inference.

Inference-time leakage

The predictive models require legitimate access to the timesteps up until but not including the timestep it is predicting:

Â(t) = Model (A(t-1), A(t-2),... , E(t-1), E(t-2),...)
where Â(t) is the forecasted target, A(t) and E(t) are the actual target and other (exogenous) variables

The data leaks during inference when the model has access to the future timestep – either the target itself, or the exogenous variables, or both:
Â(t) = LeakedModel (A(t), A(t-1), A(t-2),... , E(t), E(t-1), E(t-2),...)

Detecting leakage

The worst types of data leakage are not immediately obvious. The model produces some predictions Â(t) that seem 'normal' and nothing signals that they are leaked. It may become obvious only later at backtesting stage, when you end up correctly predicting next-day stock movements 90% of the time (something is wrong!). In the worst case, data leakage may remain unnoticed for a while.

A quick way to detect leakage at the early development stage is the following:

- Pick a timestep T to be tested for leakage. Make sure the actual observations at T are typical for your data, e.g. within 2 standard deviations of the mean
    - if the data is relatively homogenous and the model is timestep-invariant, it's convenient to pick the last timestep to be predicted
- Distort the data for A(T) and E(T) significantly, e.g. multiply by 10^6 or 10^9
    - converting to zeros may not work because the leaked predictions would still have the model's bias which may be indistinguishable from the typical series A(t)
- Predict Â(T)
- Compare Â(T) to the typical range for A(t)
    - if Â(T) is not within the typical range, e.g. significantly larger or smaller than your typical observations, your model is leaking: Â(T) is a function of distorted A(T) and/or E(T)
    - try multiplying A(T) and E(T) by different multiples or testing leakage at a different timestep T
- If Â(T) stays within the typical range despite these distortions to A(T) and E(T), we conclude that the future timesteps do not affect our predictions and the model is leak-free.