# Window feature overview

1. Window feature

* Lag feature -> can we summarize more than one past value into a single feature?
* Window feature: compute statistics (e.g., mean, min, max) using a window over the past data
* We can create window features from both the target and other features

1. Section overview

* How to choose a window size and statistics
* Window types: rolling vs. expanding windows
* Adding weights to the windows
* How to implement in Python

# Rolling window features

1. Rolling window features

* Apply a window to the time series
* Compute statistics from data inside the window
* To avoid data leakage, assign the statistics to timestamp AFTER the window

A table with numbers and numbers

Description automatically generated

* Move the window and iterate (i.e. roll) across the time series

1. Edge cases

* Option 1: treat as missing data
* Drop the rows with missing data
* Impute the missing data
* Pros:
  + All rolling statistics have the same window size
  + Simple
* Cons:
  + Reduces the amount of data if dropping rows
  + Edge cases could be quite different to rest of data
* Option 2: use smaller window sizes at the edges
* Drop the rows with missing data
* Impute the missing data
* Pros:
  + Less missing data
  + Simple
* Cons: Stats at edges are based on smaller window sizes

1. How to pick the window size?

* If the window size is too large:
* Won’t capture the local behavior of the time series
* Potentially a lot of missing data at the edges
* If the window size is too small:
* The summary statistic is computed over a small set of numbers and may not add much additional information
* Different window sizes can capture different behaviors in the data (e.g., short term vs long term trends) -> try different window sizes
* Heuristic: set the window sizes to be the same as the seasonal period(s) t smooth over the multiple seasonalities
* Try multiple window sizes, use feature selection methods, and assess the performance of the model
* Try nested window features where we use multiple window sizes on different time scales

A table with numbers and a number on it

Description automatically generated

1. Nested rolling window features

* Nested rolling windows can help capture info at different time scales
* Trend in short term (weeks) vs long term (months) in the target and features
* E.g. from finance:
* Momentum: tendency of an increasing/decreasing stock price to continue to increase/decrease
* One indicator of momentum is the difference between a shorter term moving average and a longer term moving average
* When to use nested window features?
* No theoretically correct answer
* Experiment, try feature selection, and check model performance
* Example: hourly electricity demand – Moving averages over different time scales capture different behaviors (e.g., the yearly average captures long term trends)

A graph showing a window

Description automatically generated

1. How to pick which statistics

* Mean – very common to smooth the data
* Standard deviation measures volatility which can sometimes be predictive
* Without prior knowledge, it is difficult to know if other summary stats is useful
* Keep it simple and use mean and standard deviation
* Is the accuracy good enough? Then don’t add more features than needed
* Try a variety of features and use feature selection and modeling techniques (e.g., LASSO) to reduce number of features

1. Rolling window features - Implementation in Python

* Pandas

A screenshot of a computer

Description automatically generated

* The ‘rolling’ method by default assigns the rolling statistics to the edge of the window
* Need to shift the output of the row down by 1 to avoid data leakage
* The ‘min\_periods’ argument allows us to use smaller windows at the edges
* Rolling window features with Feature-engine

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

* Note: std for window of 1 is undefined
* Rolling window features with sktime

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Output does not contain the original time series

# Expanding window features

1. Rolling vs. expanding window features

* Rolling window
* Uses only some of the previous values at any step
* Useful to capture recent behaviors of the time series
* E.g. average sales over the past one month
* Expanding window
* Uses all previous values at any step
* Useful when you need access to the entire time series
* E.g. aggregating a variable which has a cumulative effect, target encoding

1. Expanding window features

* Apply a window to the time series
* Compute statistics from data inside the window
* To avoid data leakage, assign the statistics to timestamp after the window
* Expand window and iterate across the time series

1. When is expanding window helpful?

* Uses all previous values at any step
* Useful when you need access to the entire time series
* E.g.
* Aggregating a variable which has a **cumulative** **effect**
* Target encoding

1. Implementation in python

* Pandas

A screenshot of a computer

Description automatically generated

* The ‘expanding’ method by default assigns the rolling statistics to the edge of the window
* Need to shift the output to the row down by one to avoid data leakage
* Feature-engine

A screen shot of a computer

Description automatically generated

1. Summary

* Expanding window include all the data prior to the end of the window
* Can be helpful to capture cumulative effects and are used in target encoding
* Not used as often as rolling windows as more recent data tends to be more predictive

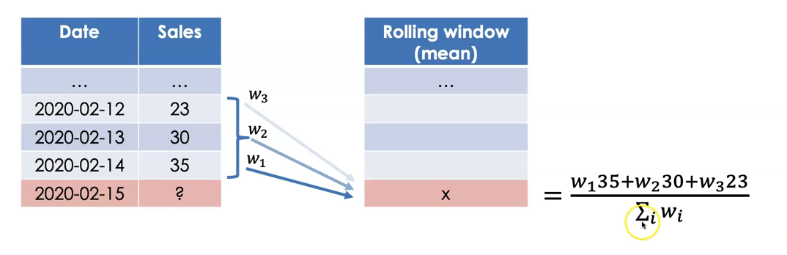
# Weighted window functions

1. Motivation

* What if we want to be more sensitive to recent observations? (e.g. to pick up changes in trend)
* We could use shorter window >< increase the variance of the new window feature
* A common solution: assign weights to the window, more weight to recent observations, to compute, e.g., a weighted mean

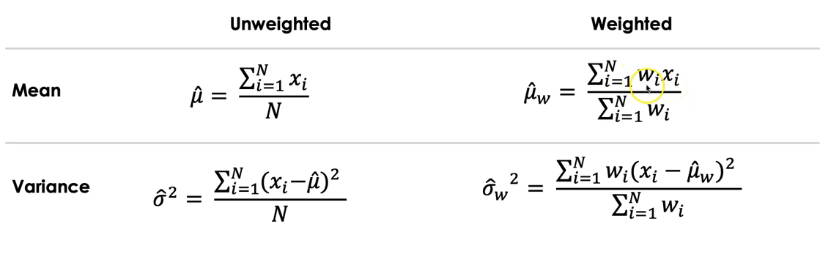
1. Weighted rolling mean

* Normalize by dividing by the sum of weights



1. Other weighted statistics

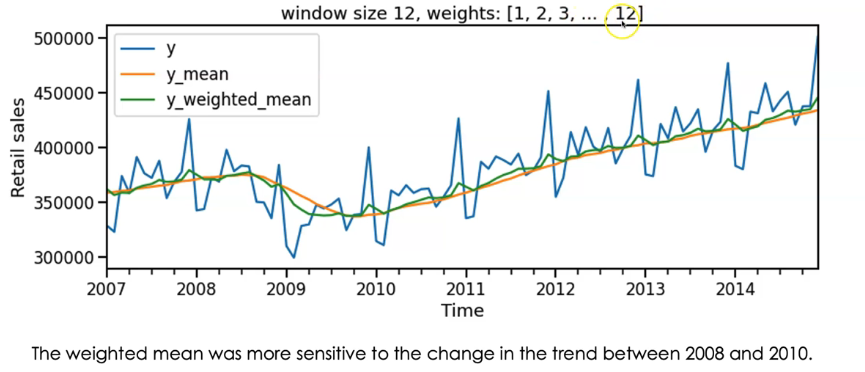
* Each statistic (e.g., the mean) has a formula for a weighted version



1. How to pick the weights?

* We can think of starting with 100% weight. We’re spreading this over the window
* How do we spread the weight?
* Domain knowledge
* Linear & exponential weights (weights decays linearly or exponentially at rate of decay )
* Try multiple weighting schemes and test performance (computationally prohibitive)
* The lack of a principled way to select weights is a downside of this approach

1. E.g. retail sales with linear weights



1. Aside: relation to distributed lags

* Distributed lags can result in many lag features. We also know we want more weight to be given to a recent lag relative to larger lags
* Creating a weighted window feature allows us to condense the same logic into a single feature, with the downside that we must manually specify weights

1. Summary

* Weighting our window features can make the feature more sensitive to recent changes
* There are multiple ways that the weight can be defined
* There is no easy way to pick the ‘best’ weights. Instead we can use heuristics, domain knowledge, and trial & error

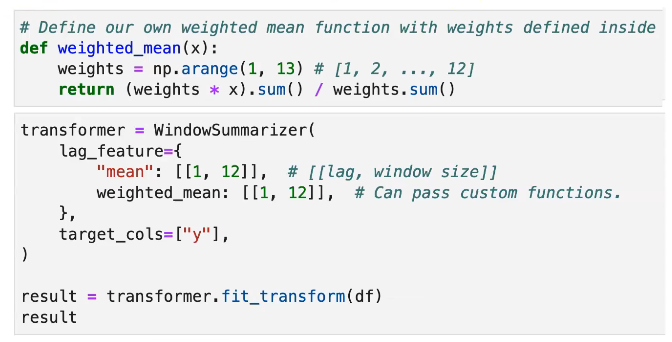
1. Implementation in Python

* Pandas

A computer code with numbers and letters

Description automatically generated

* Sktime



# Exponential weights

1. How to pick the weights?

A graph of a bar graph

Description automatically generated with medium confidence

1. Exponential weights

* User specifies the parameter
* Exponential weighted moving average (EWMA) at time t:
* If the window size is large, the denominator sum converges to
* EWMA at time t will be:

A diagram of a window size

Description automatically generated

1. Exponential weights and expanding windows

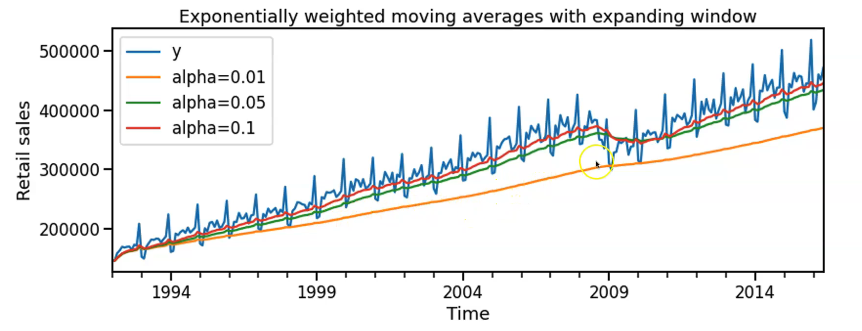
* Exponential weights are commonly applied to expanding windows
* Allows use of the whole history but weights recent values more than the distant past
* The weights change as the window expands
* Exponentially weighted moving statistics
* When computing the mean of the target -> Simple Exponential Smoothing (SES):
* SES is used commonly as a baseline forecasting model
* is learned by minimizing the forecasting error, e.g., MSE

1. How to pick ?

* If using target variable and the mean:
* The from a SES model is a good start
* Other could still be helpful in presence of other features though



* If using other features and / or other metrics:
* Then trial and error is required (e.g. try many values of and use a feature selection method like LASSO)
* E.g. retail sales



1. Implementation in python

* Pandas

A screen shot of a computer code

Description automatically generated

1. Summary

* Exponential weights requires specifying another parameter
* determines how quickly the weights decay going back in time
* Exponential weights can be used with expanding window functions to give less weight to the distant past