



УНИВЕРСИТЕТ ИТМО

**Специализированные технологии машинного  
обучения  
Machine learning**

**Lecture 1 – Time Series Analysis and Forecasting**

## Outline:

---

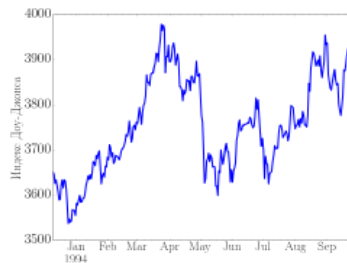
1. Time-series and their properties
2. “Classic” methods for time-series analysis
3. Time-series forecasting as machine learning problem
4. Multivariational forecasting
5. Metrics, uncertainties and quality analyses
6. Some practice...

"Prediction is very difficult,  
especially if it's about the future."

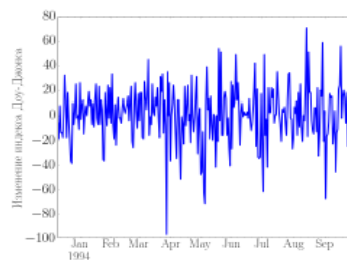
Niels Bohr

# Time series examples:

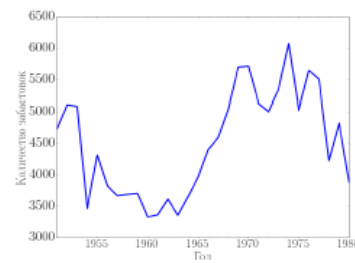
A time series is a sequence  $S$  of historical measurements  $y_t$  of an observable variable  $y$  at equal time intervals



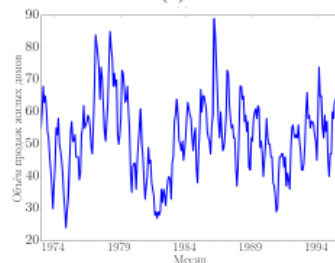
(a)



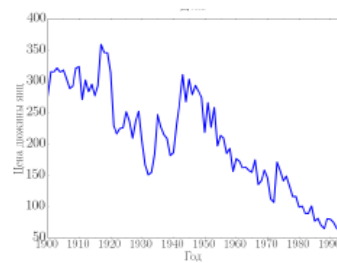
(b)



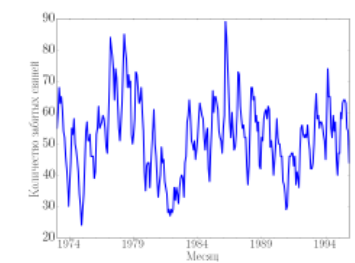
(c)



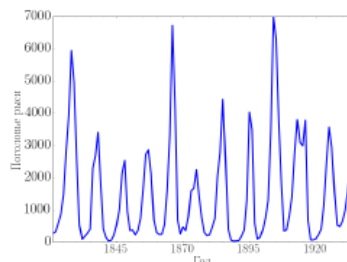
(d)



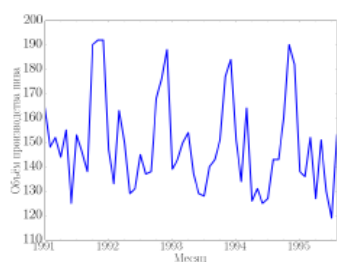
(e)



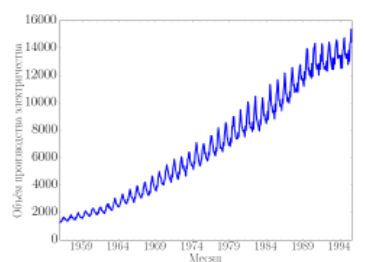
(f)



(g)



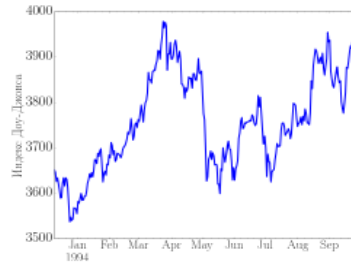
(h)



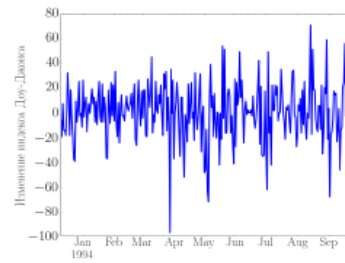
(i)

- **Trend**
  - smooth *long-term change of row level*
- **Seasonality**
  - cyclical level changes of a series with a constant period
- **Cycle(s)**
  - changing the level of a series with a variable period
- **Errors (noise)**
  - unpredictable random component of a series
- **Stationarity**
  - time independence
- **Autoregression**
  - dependence between the values of a series at adjacent points

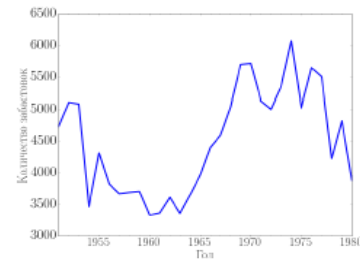
# Time series examples: trend and seasonality



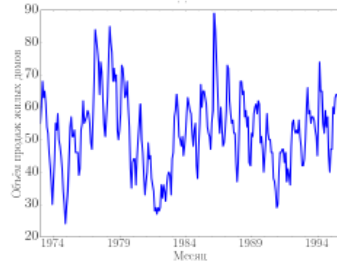
(a)



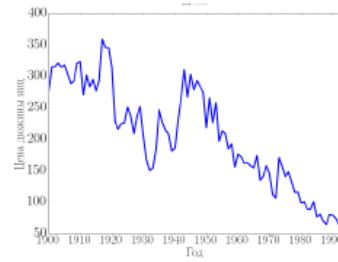
(b)



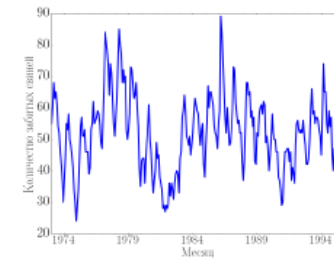
(c)



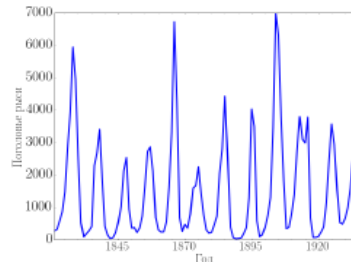
(d)



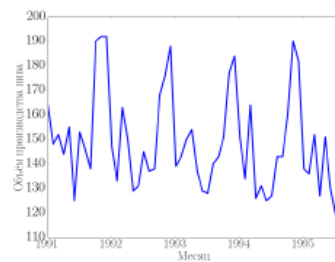
(e)



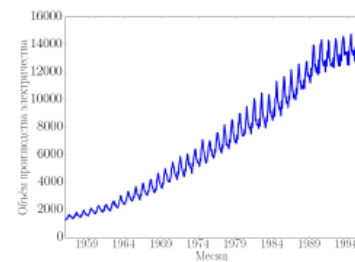
(f)



(g)

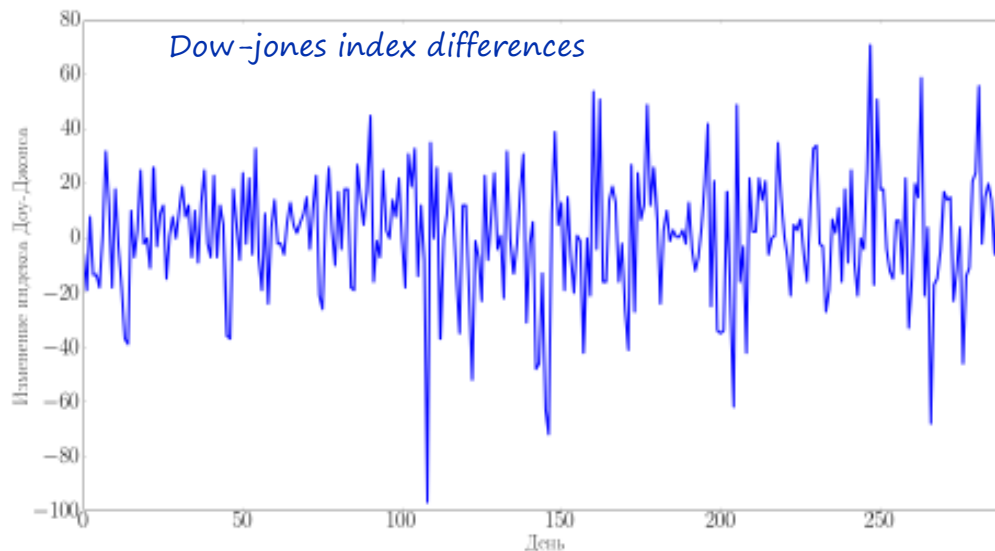


(h)



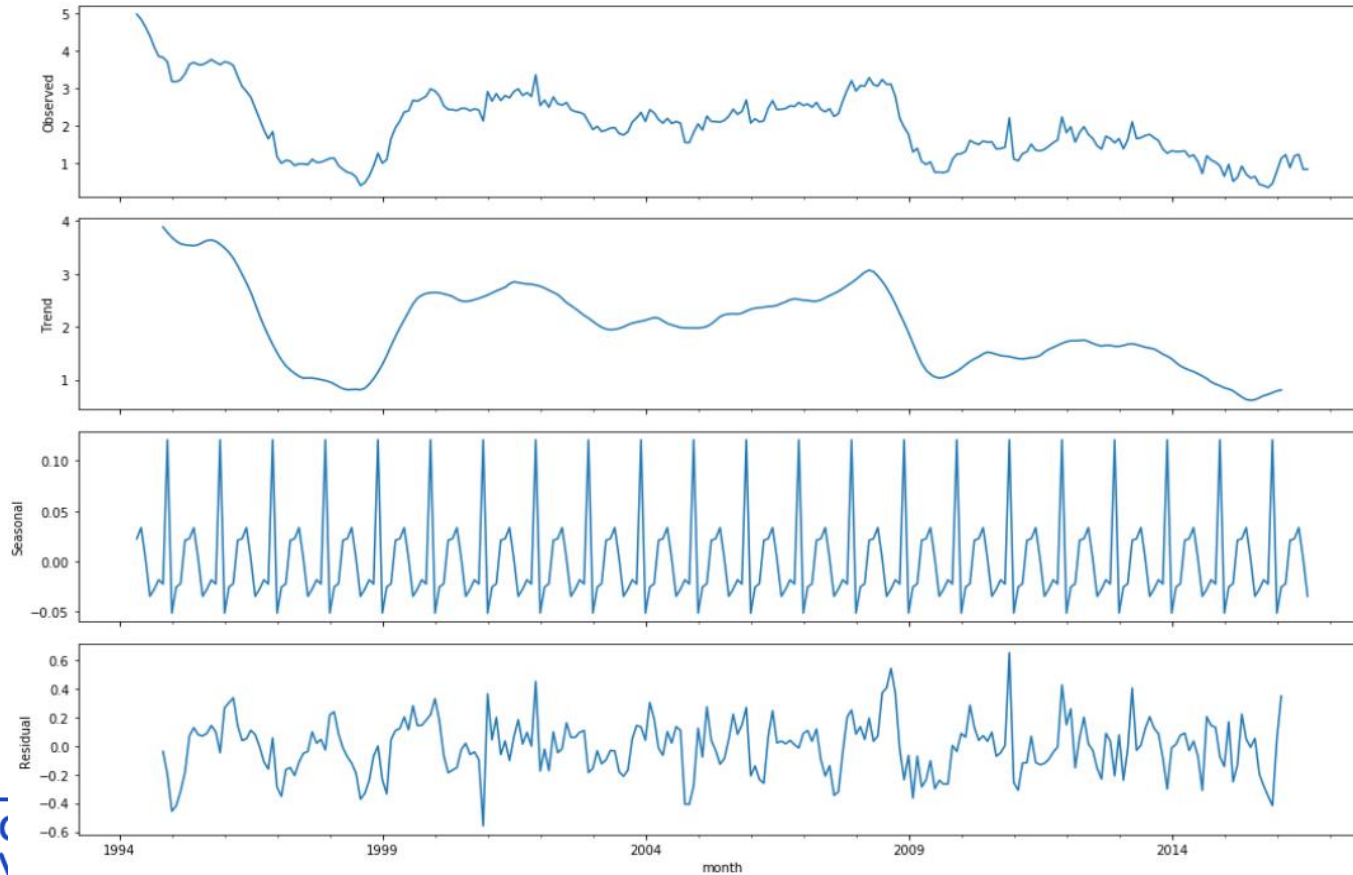
(i)

# Time series examples: noise



- non-systematic behavior: no trend, no seasonality, no cycles...
- random component over the signal;
- ~small deviations;

# Decomposition of the time series

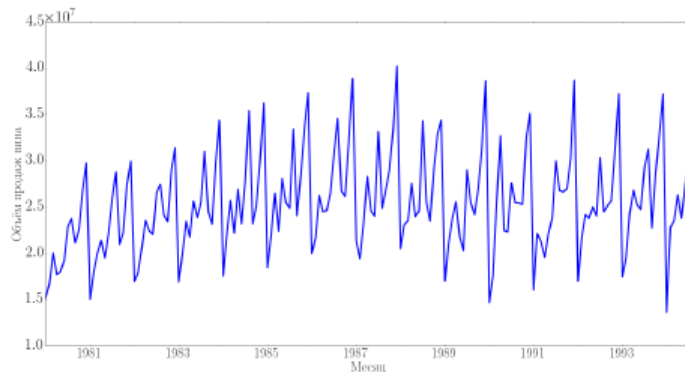




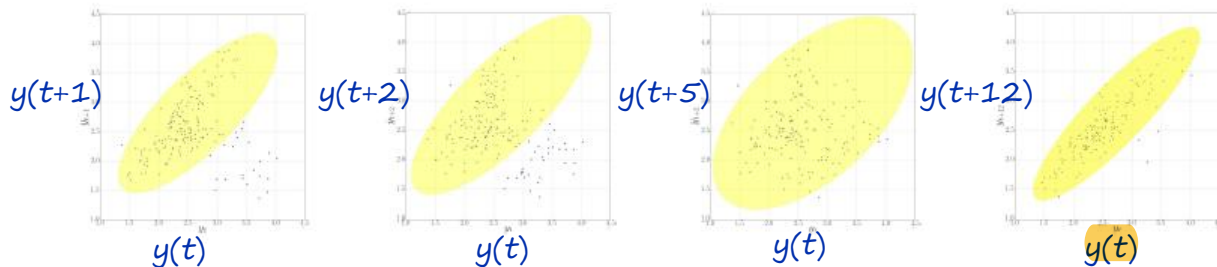
- *Changepoints in time series*
- *Correlations with external features (news, external variables, currency value etc.)*
- *Local trends*
- *(local) Maximums and Minimums*
- *Anomalies*
- *FORECASTING*

# Autocorrelation (I)

Monthly value of  
wine sales in  
Australia  
(# bottles)



Dependence of the  
values from the  
previous steps



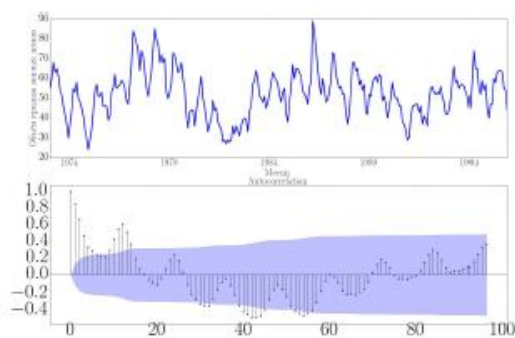
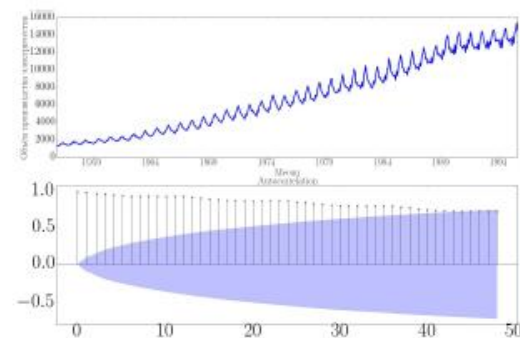
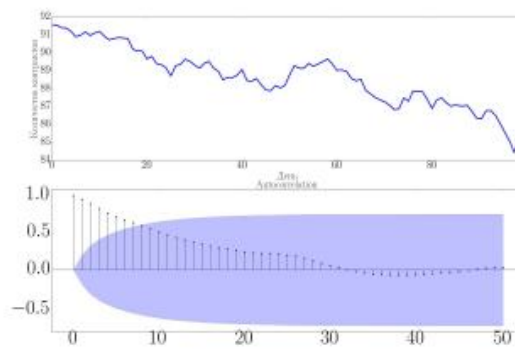
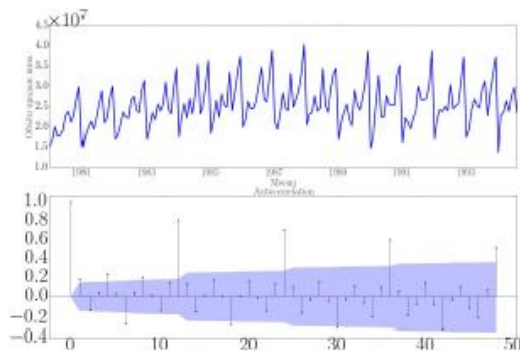
Autocorrelation  
function for lag  $\tau$ :

$$r_\tau = \frac{\sum_{t=1}^{T-\tau} (y_t - \bar{y})(y_{t+\tau} - \mathbb{E}y)}{\sum_{t=1}^{T-\tau} ((y_t - \bar{y}))^2}.$$

Pearson correlation function  
between time-series value at time  
 $(t)$  and  $(t+\tau)$ .

# Autocorrelation (II)

- correlograms:



# Operations with time series

- Difference (derivative):

$$y'_t = y_t - y_{t-1}.$$

- Seasonal derivative:

$$y'_t = y_t - y_{t-s}.$$

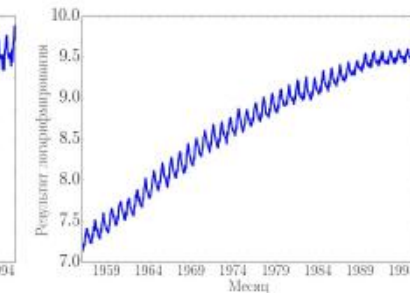
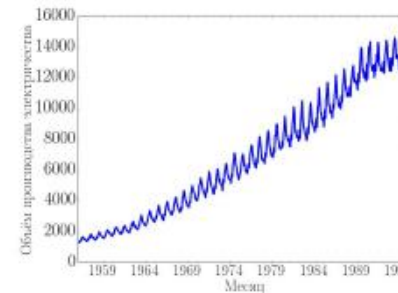
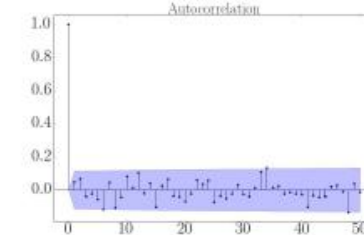
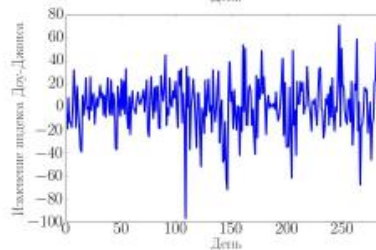
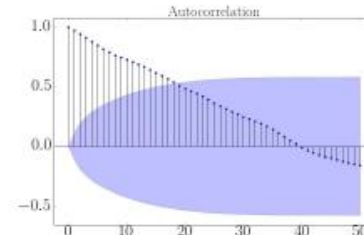
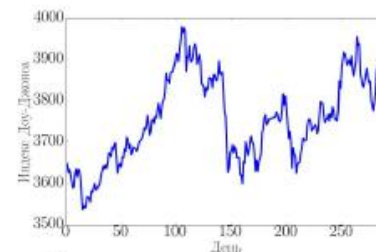
- Dispersion normalization (Box-Cox transformation):

$$y'_t = \begin{cases} \ln y_t, & \lambda = 0, \\ (y_t^\lambda - 1) / \lambda, & \lambda \neq 0. \end{cases}$$

- Stationarity test (Dickey-Fuller test) (some statistics):

$H_0$  – non-stationarity

$H_1$  – stationarity



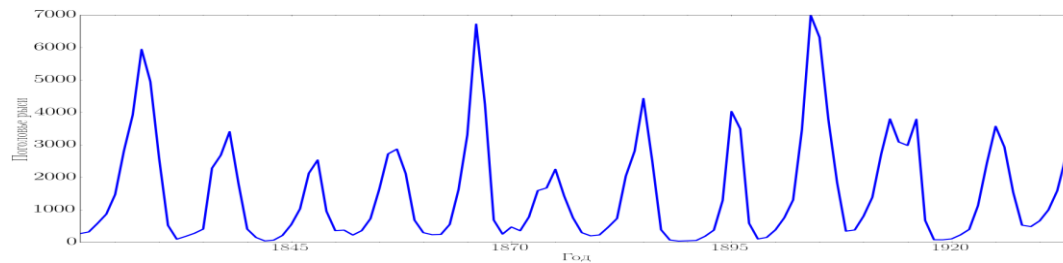
- still pretty good in forecasting autoregressive time series with strong seasonality;
- need custom fine-tuning for every new example;

- AR( $p$ ): 
$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t.$$

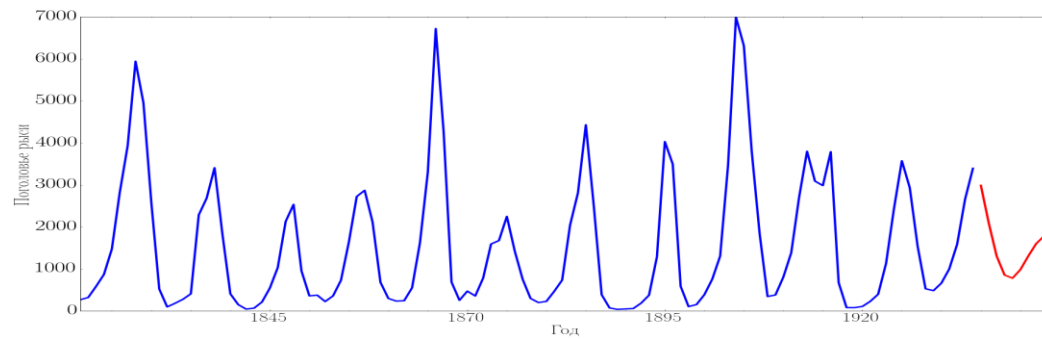
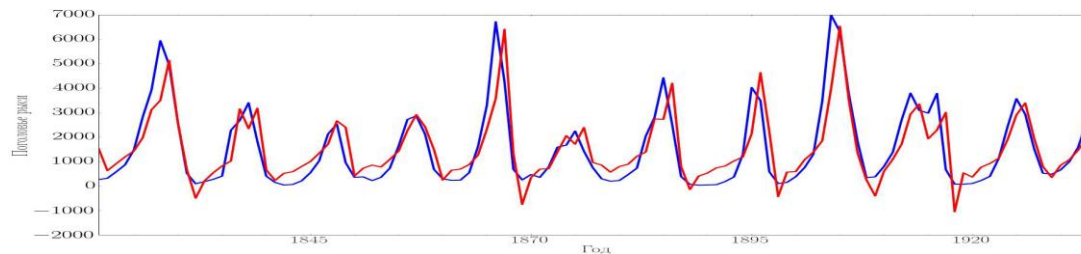
- MA( $q$ ): 
$$y_t = \alpha + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q},$$

- ARMA( $p, q$ ): 
$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q}.$$

# ARIMA models (II)



ARMA(2,2)



# ARIMA models (III)

Wold's theorem:

Every stationary time series can be approximated by ARMA(p,q) model with predetermined accuracy

⇒ We need **stationary** time series!

- Box-Cox transformation (log())
- Derivative (one-step or seasonal)

⇒ ARIMA(p,d,q) – ARMA model for **d-times** derivative time-series

$$y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q}, \quad \text{ARMA}(p,q)$$

Seasonality:

$$+ \phi_S y_{t-S} + \phi_{2S} y_{t-2S} + \dots + \phi_{PS} y_{t-PS} \quad + P \text{ components with period } S$$

$$+ \theta_S \epsilon_{t-S} + \theta_{2S} \epsilon_{t-2S} + \dots + \theta_{PS} \epsilon_{t-PS}. \quad + Q \text{ components with period } S$$



**SARMA(p,q)x(P,Q)**

$SARMA(p,q)x(P,Q)$  +  $d$  – times derivative  
+  $D$  – times seasonal derivative

=  $SARIMA(p,d,q)x(P,D,Q)$  model

- need to find  $(P,Q,p,q)$  ;
- minimize AIC:

$$AIC = - 2 \ln L + 2k,$$

Where:

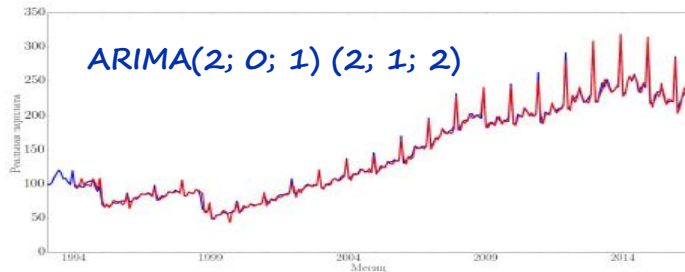
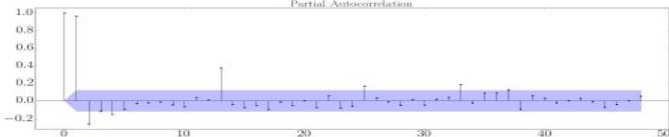
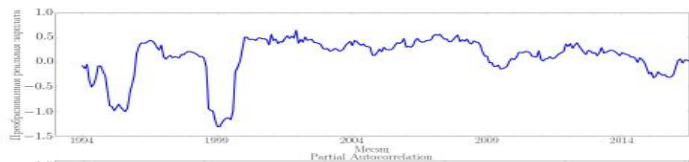
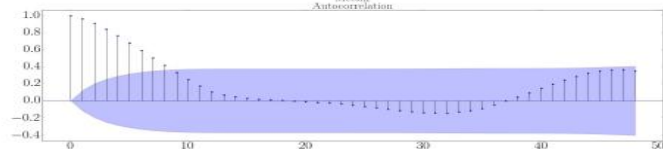
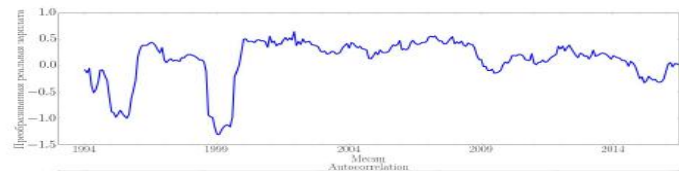
$L$  – Likelihood function

$k = P + Q + p + q + 1$  – number of model parameters

Best model – model  $ARIMA(p,q)x(P,Q)$  with min AIC.



# ARIMA models (IV)



$Q^*S=Q'$  – last seasonal lag with significant autocorrelation  
 $Q' = 0 \Rightarrow Q=0$

$q$  – last non-seasonal lag with significant autocorrelation  
 $q=8$

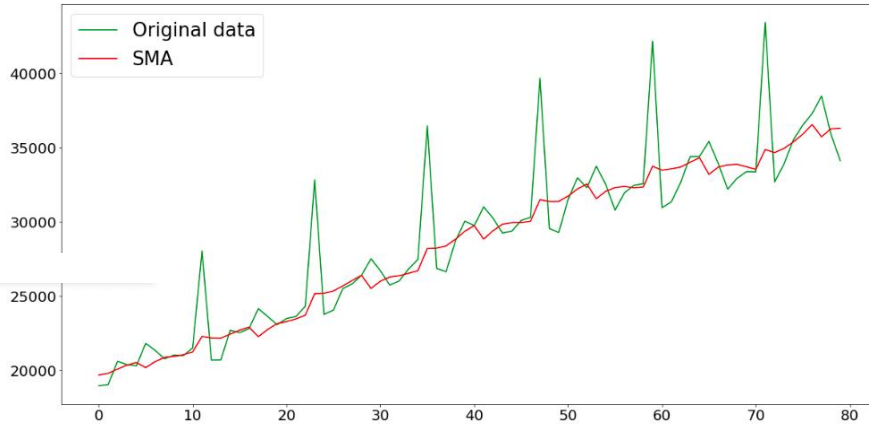
$P^*S = P'$  – last seasonal lag with significant partial autocorrelation  
 $P' = 24 \Rightarrow P=2$

$p$  – last non-seasonal lag with significant partial autocorrelation  
 $p=2$

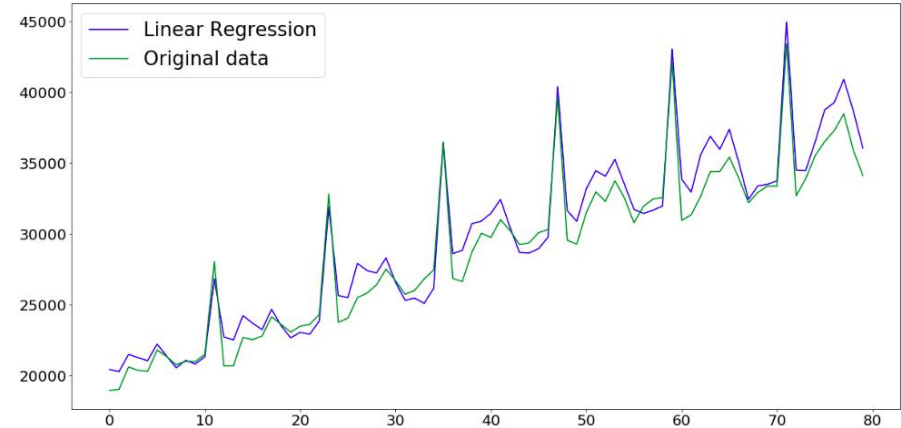
$AIC \rightarrow \min(p, d, q, P, D, Q)$

$\Rightarrow \text{ARIMA}(2; 0; 1) (2; 1; 2)$  – best model

*Plot 1. Simple moving average*



*Plot 2. Linear Regression*



There are several *metrics*:

- $R^2$
  - MSE (RMSE) – mean squared error
  - MAE – mean absolute error
  - MAPE – mean absolute percentage error
  - SMAPE – symmetric mean absolute percentage error
- 
- MLAPE, RSMLAPE etc.... – for more advanced error analysis

## $R^2$

$R^2$  is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

- "R squared" or *coefficient of determination*
- Commonly used for linear regression models
- $0 < R^2 < 1$
- The higher the better.  $R^2 = 1$  – ideal.

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$RSS = \sum_{t=1}^n e_t^2 = \sum_{t=1}^n (y_t - \hat{y}_t)^2$$

$$TSS = \sum_{t=1}^n (y_t - \bar{y}_t)^2$$

```
1 from sklearn.metrics import r2_score
2
3 print("Linear Regression R^2:", round(r2_score(y, y_pred_lr), 3))
4 print("SMA R^2:", round(r2_score(y, y_sma), 3))
```

Linear Regression R^2: 0.942

SMA R^2: 0.822

## MSE

Mean squared error (MSE) measures the average of the squares of the errors—that is, the average squared difference between the prediction and actual values.

- It is always non-negative
- Values closer to zero are better.

$$MSE = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2$$

```
1 from sklearn.metrics import mean_squared_error
2
3 print("Linear Regression MSE:", round(mean_squared_error(y, y_pred_lr), 3))
4 print("SMA MSE:", round(mean_squared_error(y, y_sma), 3))
```

Linear Regression MSE: 1882343.713

SMA MSE: 5774211.042

## RMSE

Root-mean-square error (RMSE) is the **root** of the average squared difference between the prediction and actual values.

- It is always non-negative
- Values closer to zero are better.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2}$$

```
1 from sklearn.metrics import mean_squared_error
2
3 print("Linear Regression RMSE:", round(np.sqrt(mean_squared_error(y, y_pred_lr)), 3))
4 print("SMA RMSE:", round(np.sqrt(mean_squared_error(y, y_sma)), 3))
```

Linear Regression RMSE: 1371.985  
SMA RMSE: 2402.959

## MAE

Mean absolute error (MAE) is the average vertical distance between each point given by estimator and actual line.

- Usually expresses accuracy as a **percentage**

$$MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t|$$

```
1 from sklearn.metrics import mean_absolute_error
2
3 print("Linear Regression MAE:", round(mean_absolute_error(y, y_pred_lr), 3))
4 print("SMA MAE:", round(mean_absolute_error(y, y_sma), 3))
```

Linear Regression MAE: 1148.816

SMA MAE: 1341.285

## MAPE

Mean absolute percentage error (MAPE) shows average share of the error in actual value. MAPE usually expresses accuracy as a percentage.

- It cannot be used if there are zero values because there would be a division by zero.
- For forecasts which are too low the percentage error cannot exceed 100%, but for forecasts which are too high there is no upper limit to the percentage error.

$$MAPE = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right|$$

```
1 def mean_absolute_percentage_error(y_true, y_pred):
2     return round(np.mean(np.abs((y_true - y_pred) / y_true)) * 100, 3)
3
4 print("Linear Regression MAPE:", mean_absolute_percentage_error(y, y_pred_lr))
5 print("SMA MAPE:", mean_absolute_percentage_error(y, y_sma))
6
```

Linear Regression MAPE: 4.0  
SMA MAPE: 22.493



## SMAPE

Symmetric mean absolute percentage error is an accuracy measure based on percentage.

The **absolute difference** between actual value and predicted value is **divided by half the sum** of absolute values of the actual value and the forecast value. The value of this calculation is summed for every fitted point  $t$  and divided again by the number of fitted points  $n$ .

$$SMAPE = \frac{100\%}{n} \sum_{t=1}^n \frac{|\hat{y}_t - y_t|}{\frac{1}{2} * (|y_t| + |\hat{y}_t|)}$$

```
1 def smape(y_true, y_pred):  
2     return round(np.mean(np.abs((y_pred - y_true)) / ((np.abs(y_true)) + np.abs((y_pred)) / 2)), 3 )  
3  
4 print("Linear Regression SMAPE:", smape(y, y_pred_lr))  
5 print("SMA SMAPE:", smape(y, y_sma))
```

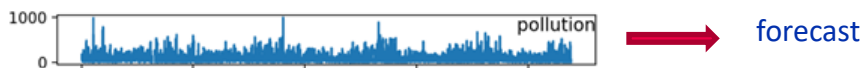
Linear Regression SMAPE: 0.026

SMA SMAPE: 0.147

# Univariate and Multivariate forecasting

## Univariate:

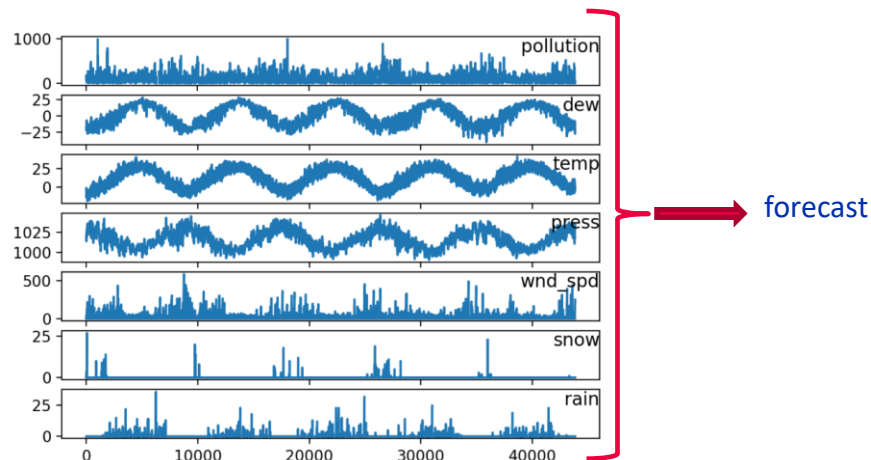
- One target time series
- Predicting based only on it



## Multivariate:

- One target time series
- Collect several features for the same time period that can influence the result  
(currency rate, temperature, unemployment rate, etc)

Forecast is based on full data



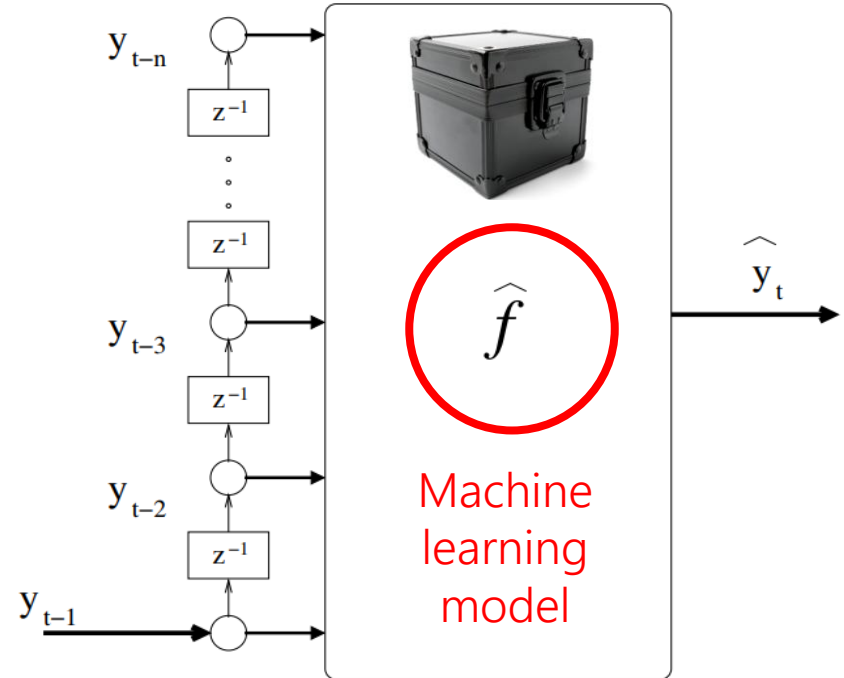
ML models !

# Forecasting as machine learning problem (I)

- Let's assume one-step forecasting
- Supervised learning problem: we need train set (inputs)  
with labels (outputs)  
..and test set
- Time series:  $S: [y_0, y_1, \dots, y_{t-2}, y_{t-1}]$
- Trying to predict  $\langle y_t \rangle$

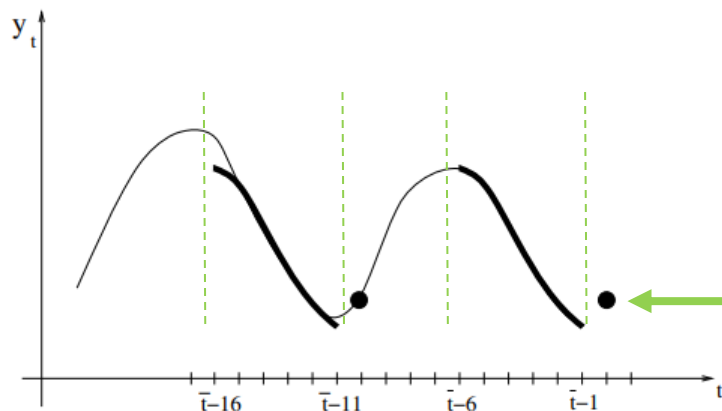
$$X = \begin{bmatrix} y_{N-1} & y_{N-2} & \dots & y_{N-n-1} \\ y_{N-2} & y_{N-3} & \dots & y_{N-n-2} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & y_{n-1} & \dots & y_1 \end{bmatrix} \quad Y = \begin{bmatrix} y_N \\ y_{N-1} \\ \vdots \\ y_{n+1} \end{bmatrix}$$

*inputs*                      *outputs*



## Algorithm:

- given a time series  $y_t$  up to time  $(t-1)$
- we want to predict the next value of the series  $y_t$
- let's initialize the number of neighbors:  $n=6$
- create template  $T$  of  $n$  previous values:  $T_n = \{y_{t-6}, y_{t-5}, \dots, y_{t-1}\}$
- search for the most similar (in some metric) pattern in the previous data

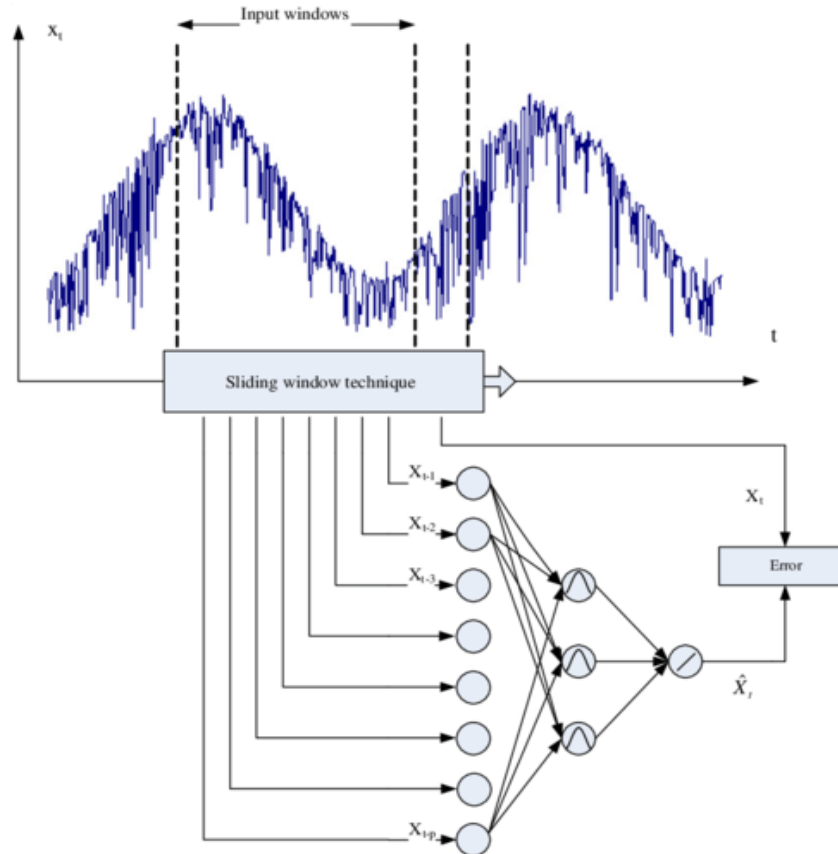


Prediction is  
 $y_t = y_{t-10}$

The pattern  $\{y_{t-16}, y_{t-15}, \dots, y_{t-11}\}$  is the most similar to the pattern  $\{y_{t-6}, y_{t-5}, \dots, y_{t-1}\}$

# Dense model (MLP)

- take every train example as independent input for the network;
- Predicting one or several values in the future – can be one-step or multi step;
- sliding window technique
- Train on a full dataset



# Recurrent neural networks. Motivation.

Example:



- I have been to **Paris!**
- And how was it?
- It was wonderful! In childhood I spent a lot of time there.  
I have a perfect opportunity to learn \_\_\_\_\_ language.

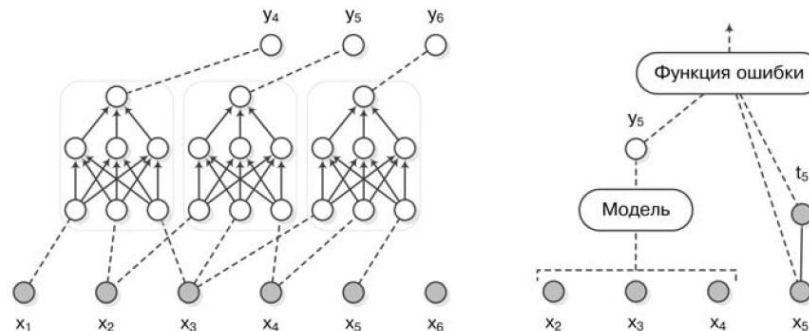
*23 words later – “Paris”*

*– it is the only reason to put  
“French” into the gap.*

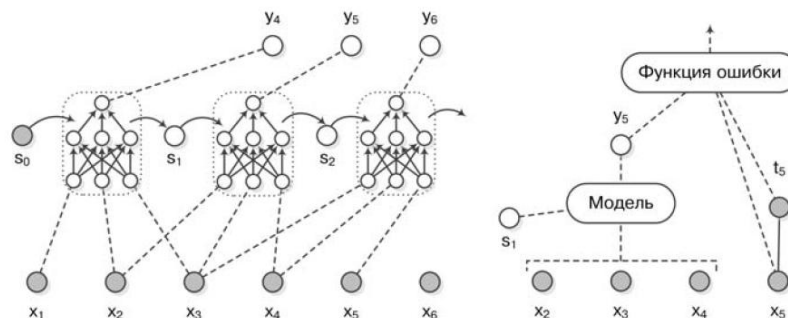
- *The model should ‘remember’ old (and very old) inputs to be able to use them when it is the reason for it*
- *In classic dense networks the weights of “early” words is exponentially small.*
- *How to remember..?*

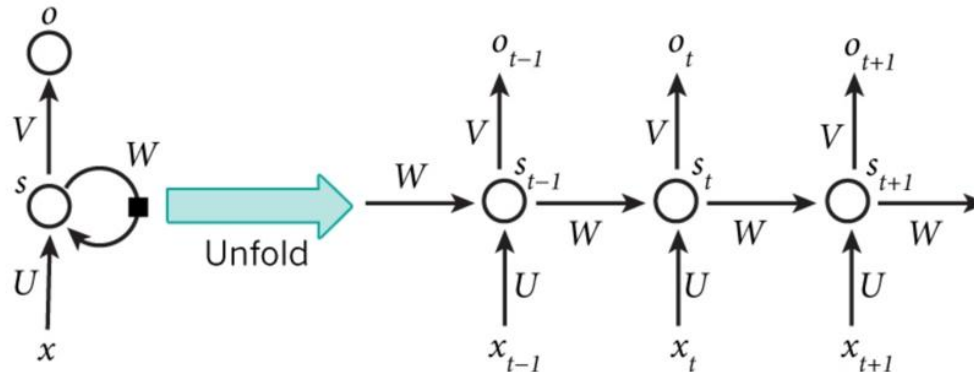
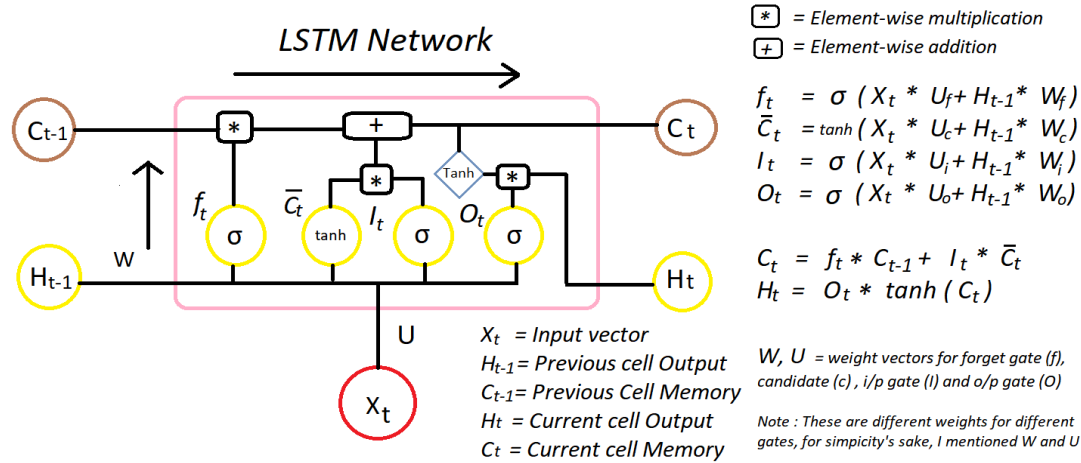
# RNN and MLP ("dense networks")

Dense  
model:



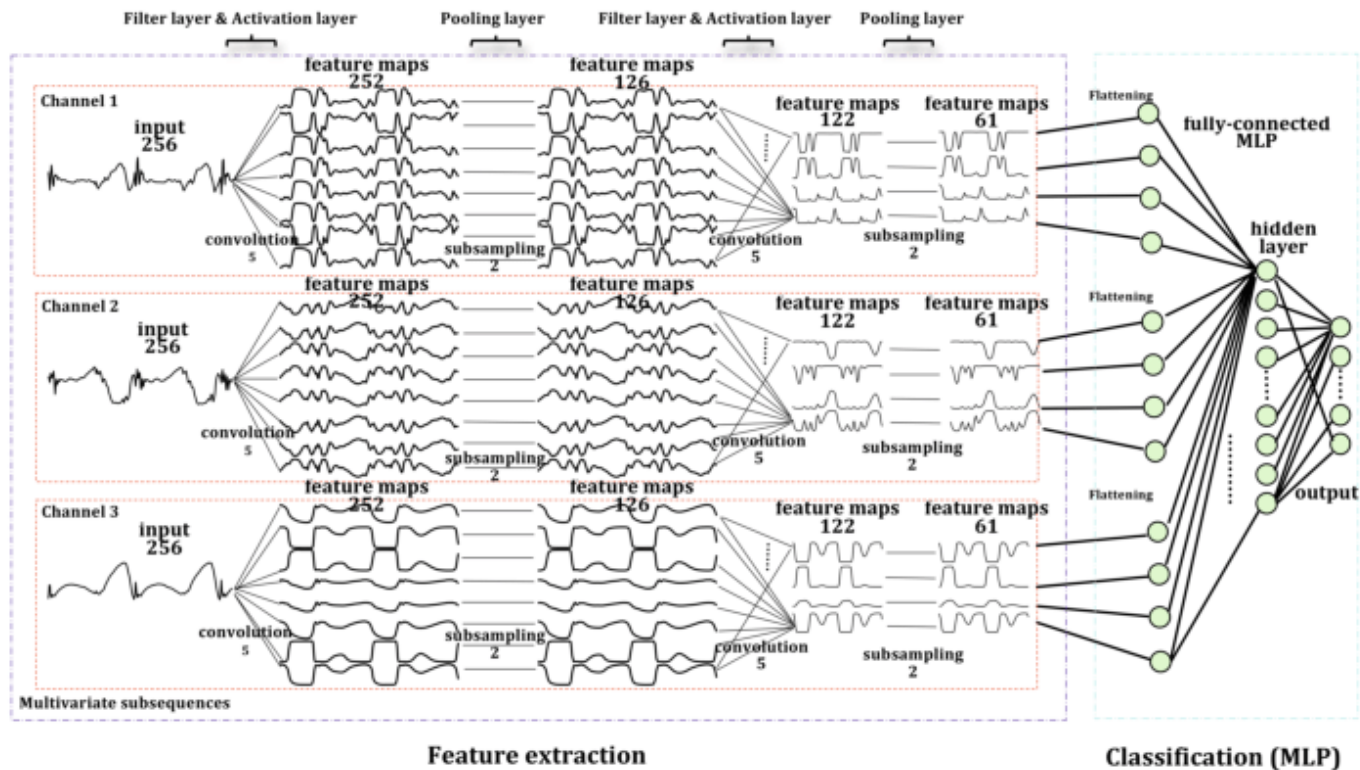
Recurrent  
model:







# CNN for time-series classification

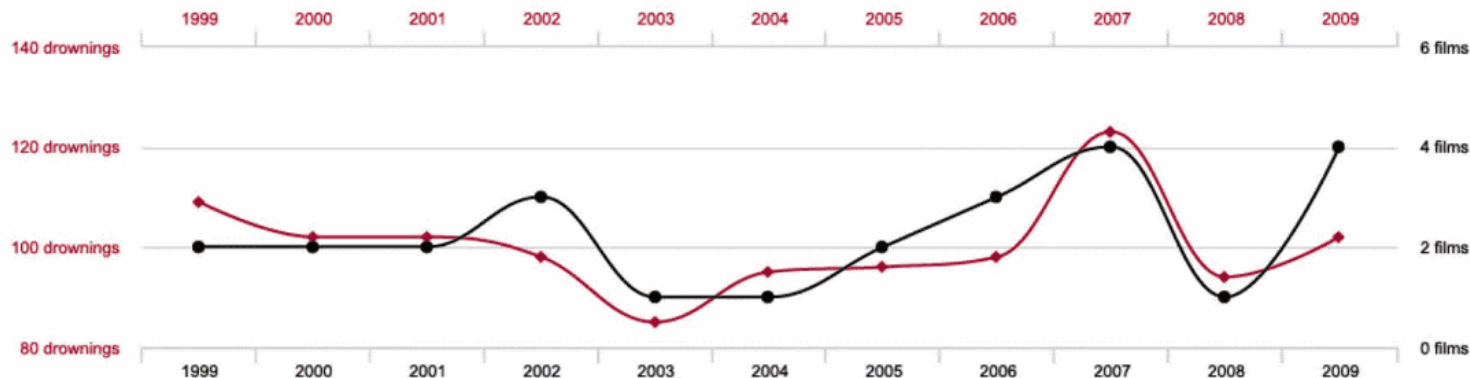


# Multivariate time series and causality

# Correlations and causality



66.6%  
CORRELATION



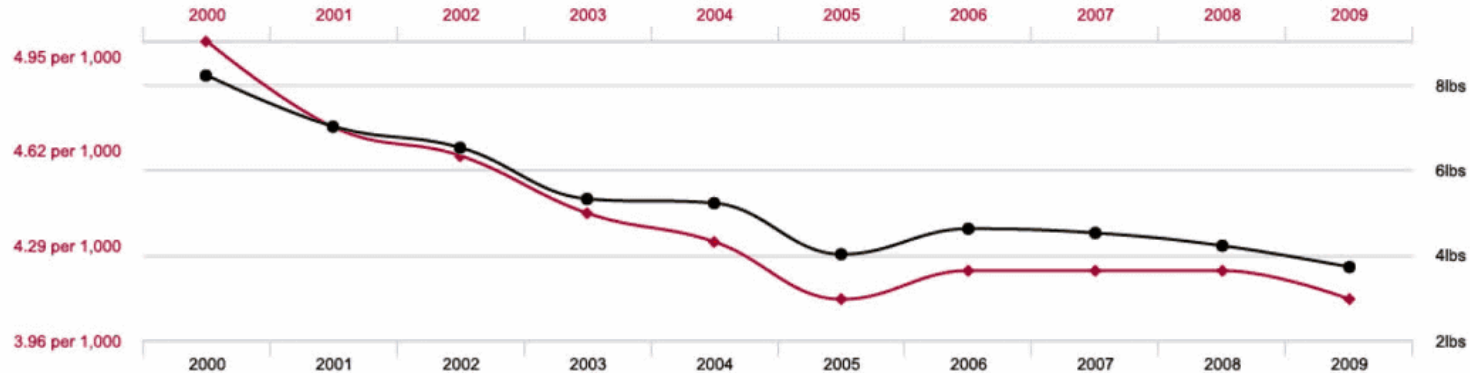
Source: [tylervigen.com/spurious-correlations](http://tylervigen.com/spurious-correlations)

# Correlations and causality



DIVORCE  
IN  
MAINE

99.26%  
CORRELATION



Source: [tylervigen.com/spurious-correlations](http://tylervigen.com/spurious-correlations)

## Granger test

- two regressions are constructed: in each regression of the dependent variable is one of the variables checked for causality, and the lags of both variables are the regressors
- test two null-hypotheses: the coefficients for the lags of the second variable are simultaneously zero for goal-predictor( $X \rightarrow Y$ ) and predictor-goal ( $Y \rightarrow X$ ) regressions.

$$X_1(t) = \sum_{j=1}^p A_{11,j} X_1(t-j) + \sum_{j=1}^p A_{12,j} X_2(t-j) + E_1(t)$$

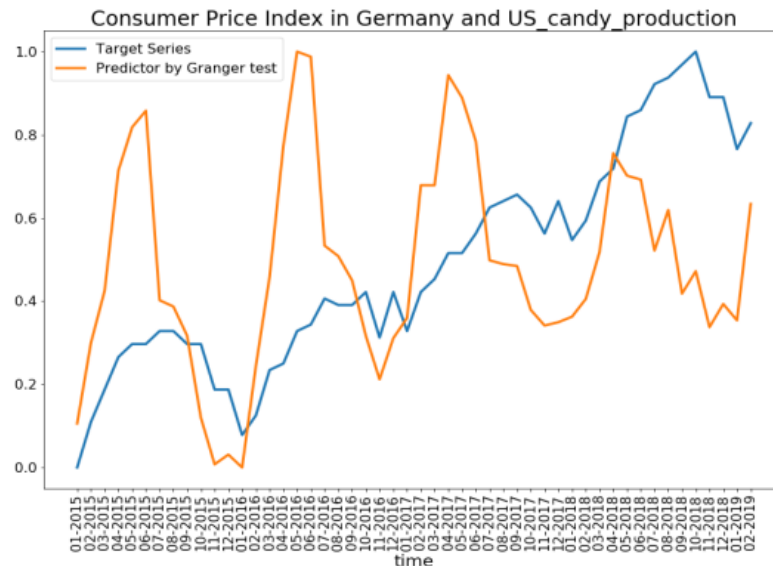
$$X_2(t) = \sum_{j=1}^p A_{21,j} X_1(t-j) + \sum_{j=1}^p A_{22,j} X_2(t-j) + E_2(t)$$

## Cross-correlation

- evaluate correlation between time series (Y) and values of p-predictors (X) at different lag positions

## Convergent Cross Mapping test

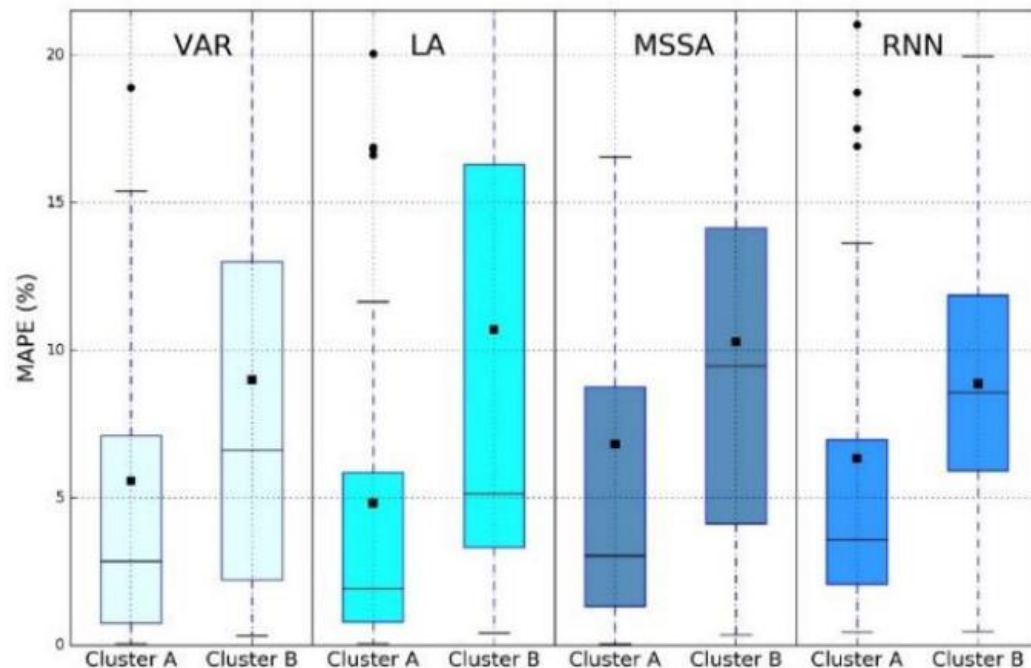
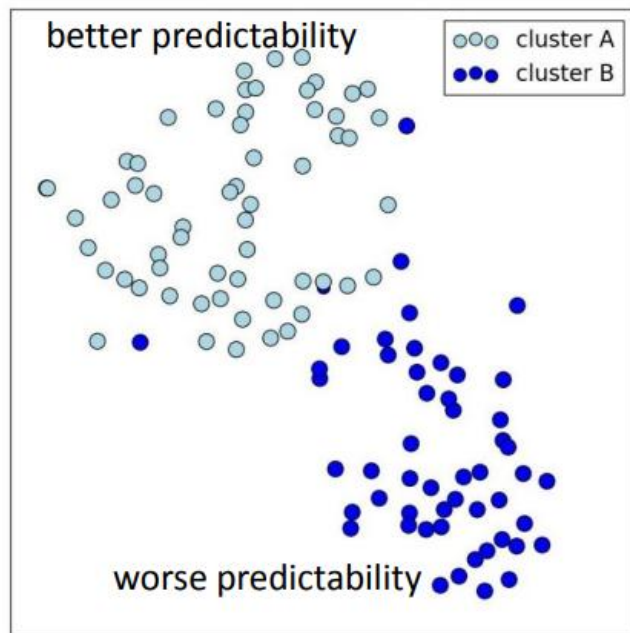
- Create the shadow manifold for X, called  $M_x$ ;
- Find the nearest neighbors to a point in the shadow manifold at time  $t$ ;
- Create weights using the nearest neighbors approach;
- Estimate Y using the weights (this estimate is called  $Y|M_x$ );
- Compute the correlation between Y and  $Y|M_x$ ;



Eichler M. Causal inference in time series analysis // Causality: Wiley Series in Probability and Statistics. 2012. P. 6–28.

<https://royalsocietypublishing.org/doi/full/10.1098/rsta.2011.0613>

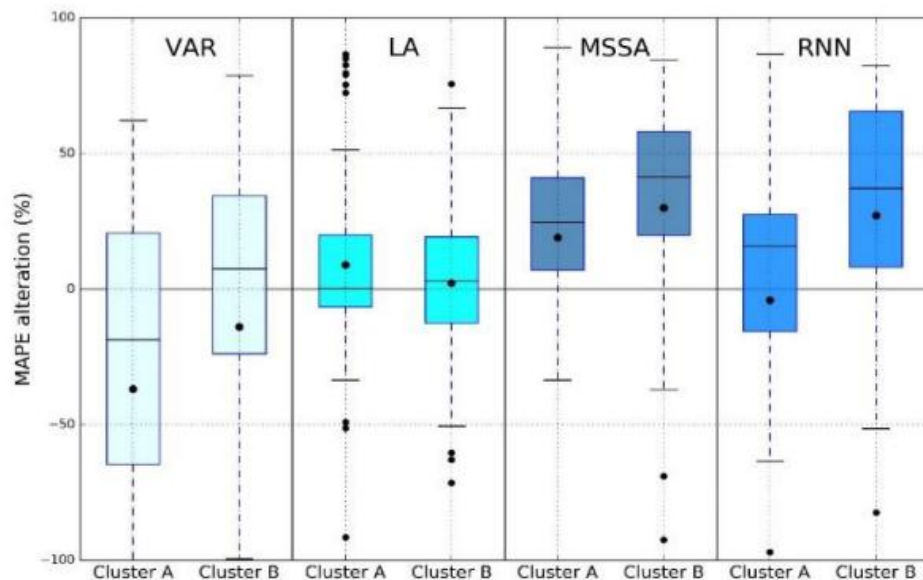
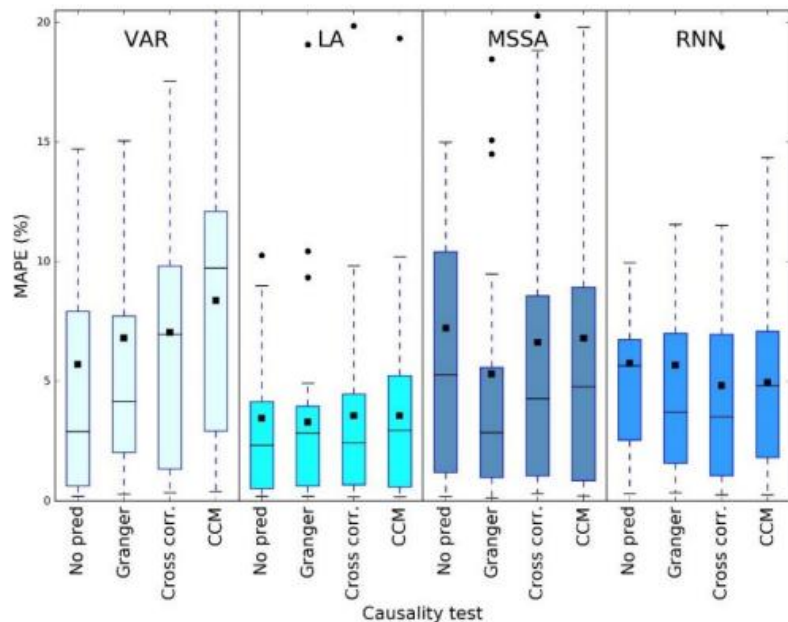
# Clustering the predictability



Experimental result of t-SNE visualisation for k-means clustering. Each point corresponds to a single time series.

MAPE distribution in the clusters

# Causality test dependence and MAPE alteration



MAPE distribution for the four models with one additional series-predictor. Three different causality approaches were tested.

Relative MAPE alteration caused by using of the predictor time-series in two clusters.



# 3 types of uncertainty of ML models for forecasting

- *Model uncertainty*
  - captures the scenario with unknown parameters and the properties of the data,
  - can be reduced as more samples being collected
- *Inherent noise*
  - the uncertainty in the data generation process and is irreducible.

BUT:

  - by developing a good noise model one can perform denoising operation in order to do forecast on the signal!
- *Model misspecification*
  - the scenario where the testing samples come from a different population than the training set, which is often the case, for example, in time series anomaly detection



# Thank you!