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

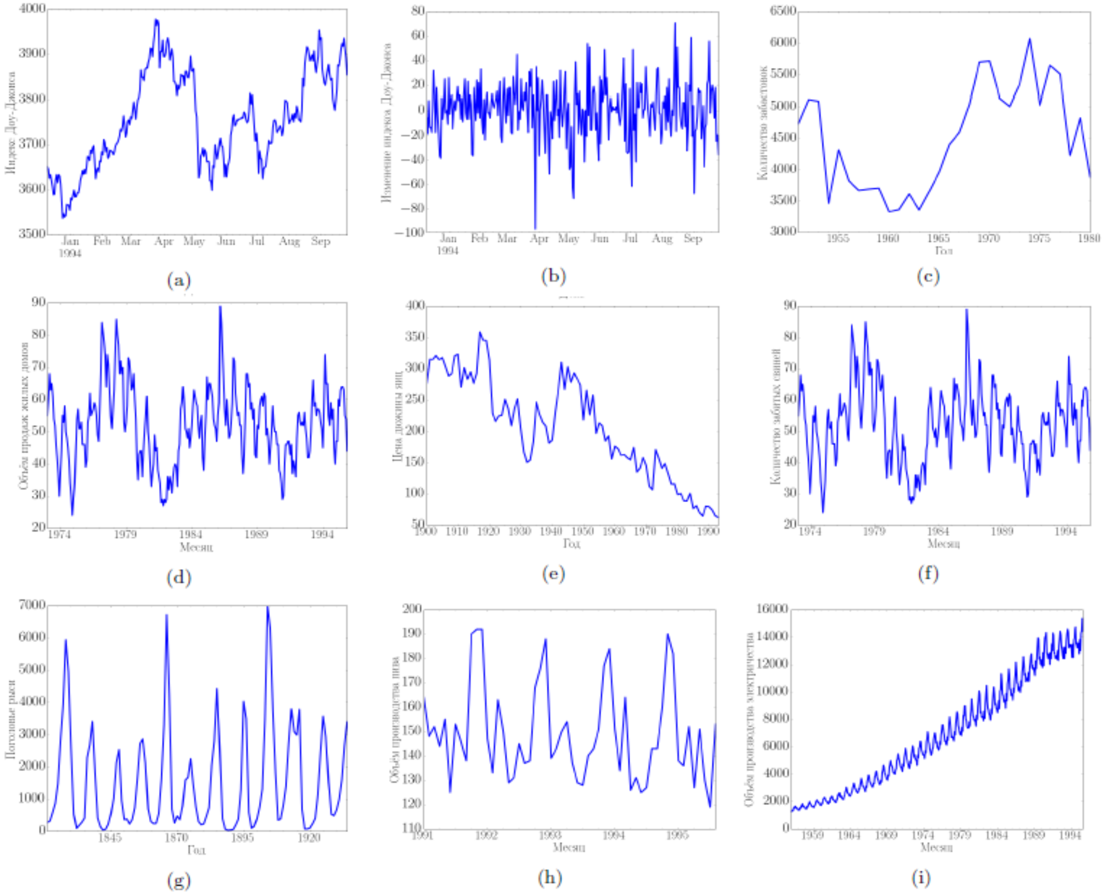
# Time series examples:

A ***time series*** is a sequence $S$

of historical measurements $y_t$

of an observable variable $y$

at <u>*equal time intervals*</u>



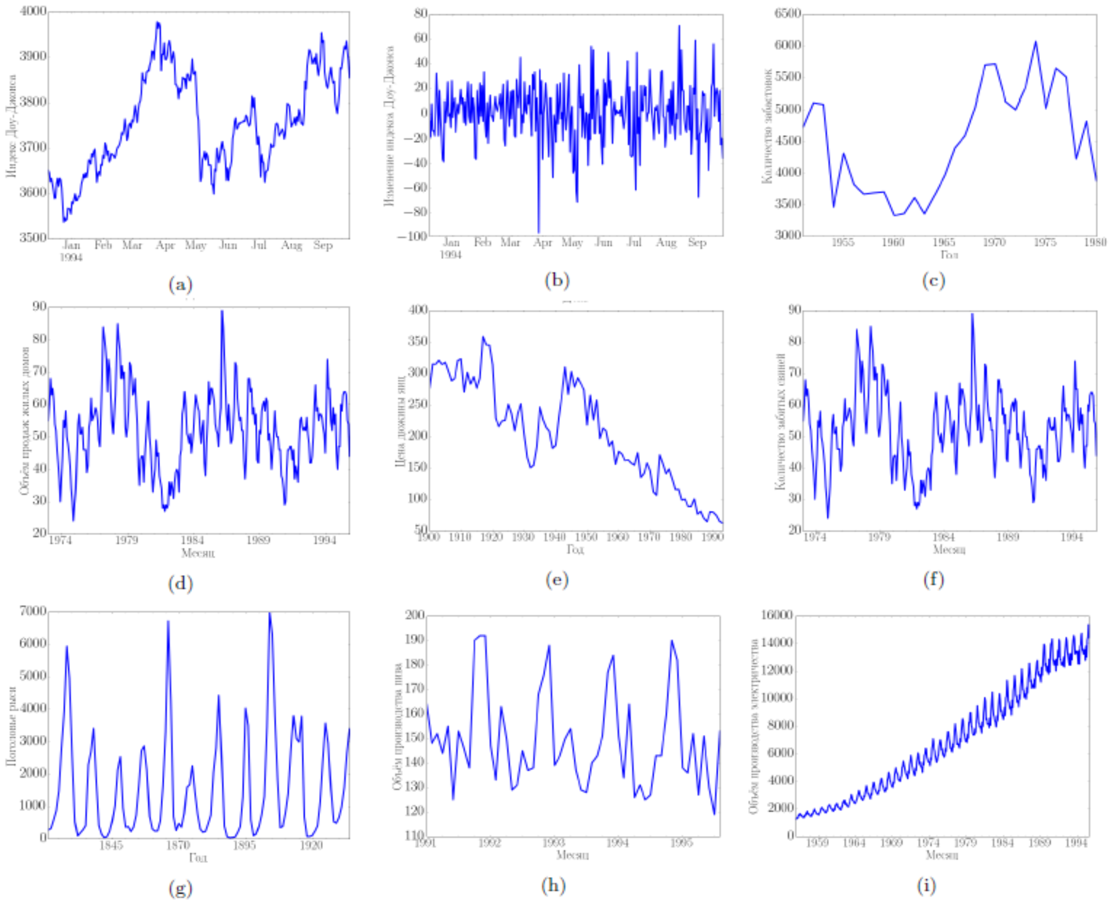
# Time series properties:



- **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

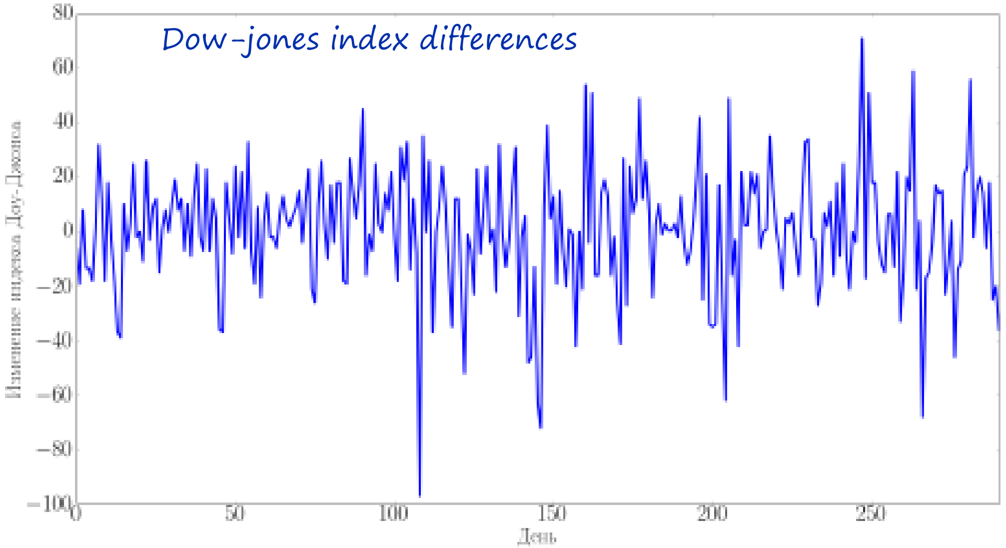
# Time series examples: trend and seasonality





# Time series examples: noise

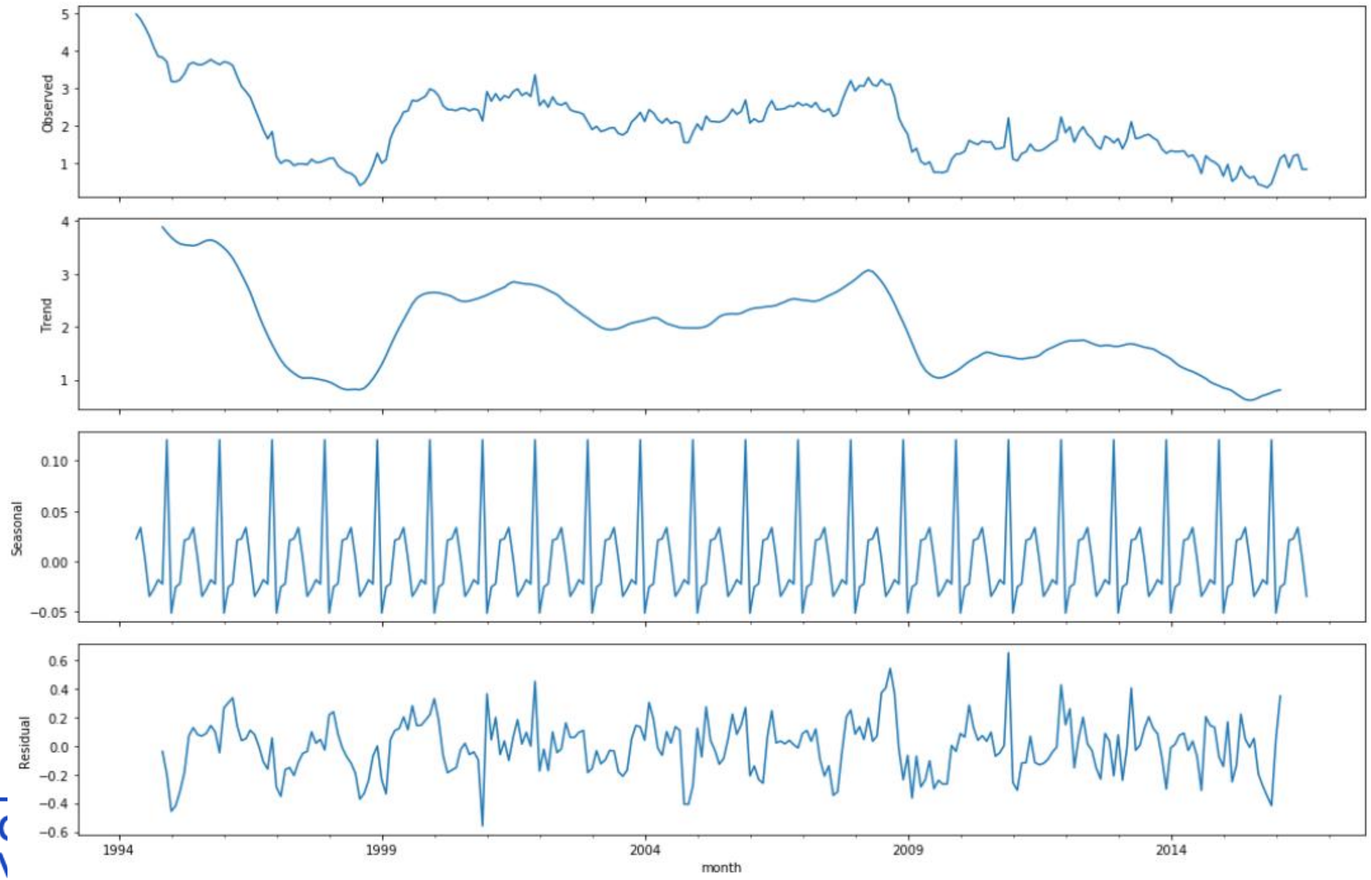





- non-systematic behavior: no trend, no seasonality, no cycles...
- random component over the signal;
- ~small deviations;

# Decomposition of the time series





# Points of interest

УНИВЕРСИТЕТ ИТМО

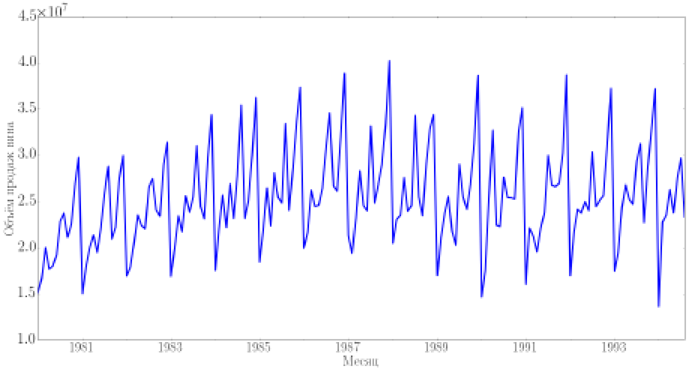
- *Changepoints in time series*
- *Correlations with external features (news, external variables, currency value etc.)*
- *Local trends*
- *(local) Maximums and Minimums*
- *Anomalies*
- *FORECASTING*

ITsMOre than a UNIVERSITY

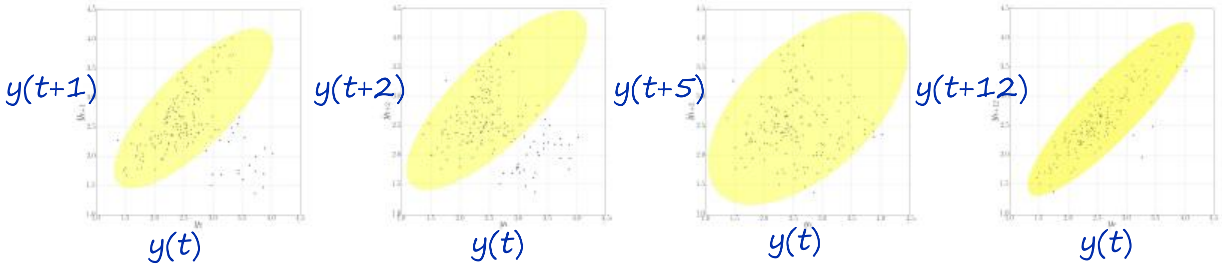
# Autocorrelation (I)



Monthly value of wine sales in Australia (# bottles)




Dependence of the values from the previous steps




**Autocorrelation** function for lag **τ**:

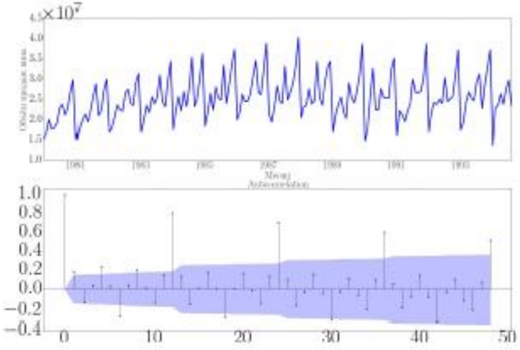
$$r_\tau = \frac{\sum_{t=1}^{T-\tau}(y_t - \bar{y})(y_{t+\tau} - \mathbb{E}y))}{\sum_{t=1}^{T-\tau}((y_t - \bar{y}))^2}.$$

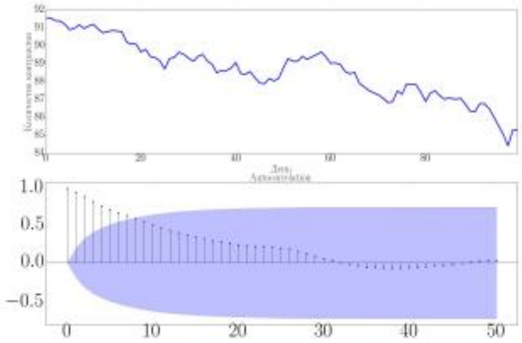
Pearson correlation function between time-series value at time (t) and (t+τ).

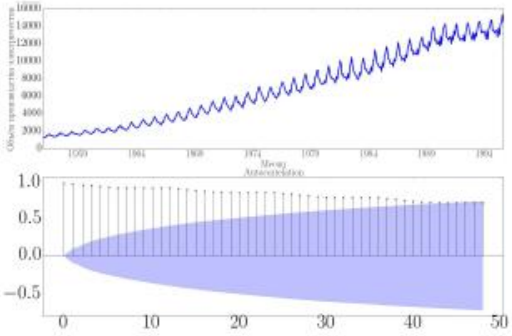
# Autocorrelation (II)

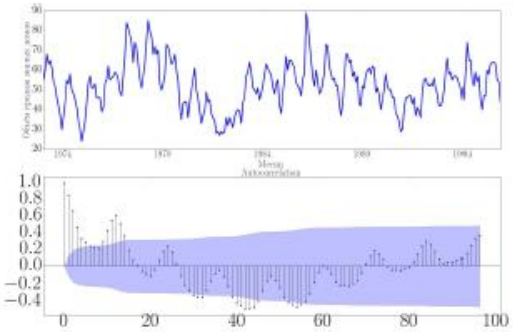
УНИВЕРСИТЕТ ИТМО

- *correlograms:*









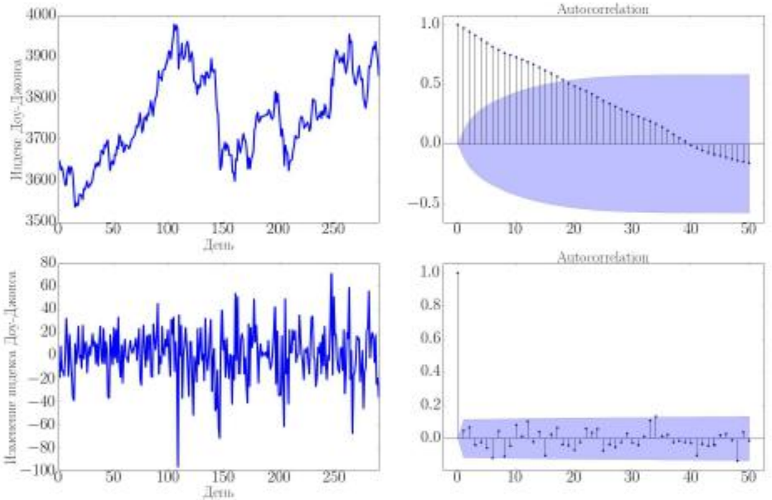
# Operations with time series

- *Difference (derivative):*

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

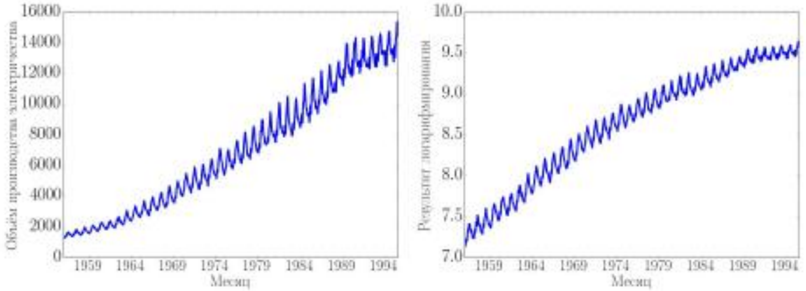
- *Seasonal derivative:*

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




- *Dispersion normalization (Box-Cox transformation):*

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




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

$H_0$ – *non-stationarity*
$H_1$ – *stationarity*

# ARIMA models (I)

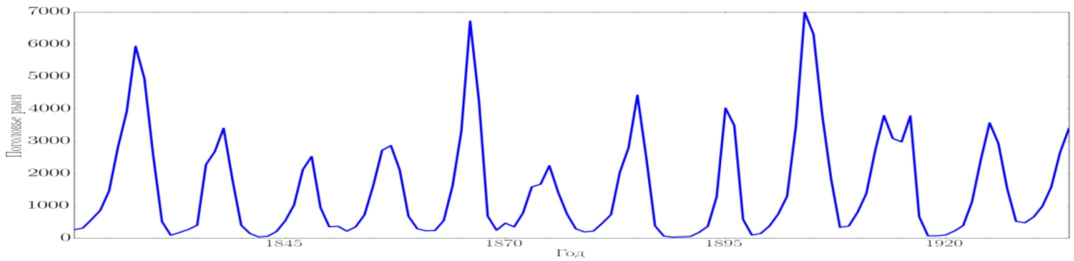
УНИВЕРСИТЕТ ИТМО

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

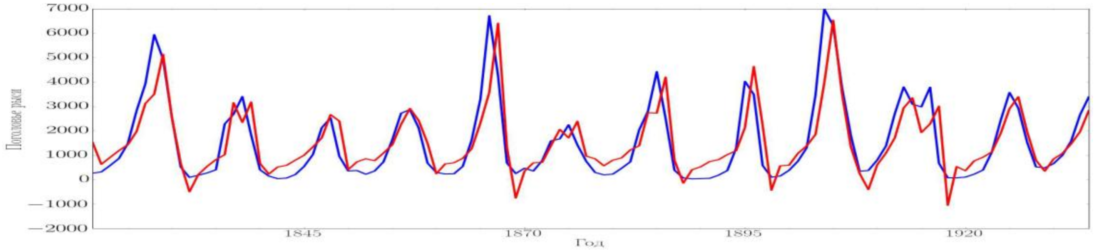
- AR(p): $y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t.$
- MA(q): $y_t = \alpha + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q},$
- ARMA(p,q): $y_t = \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} +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

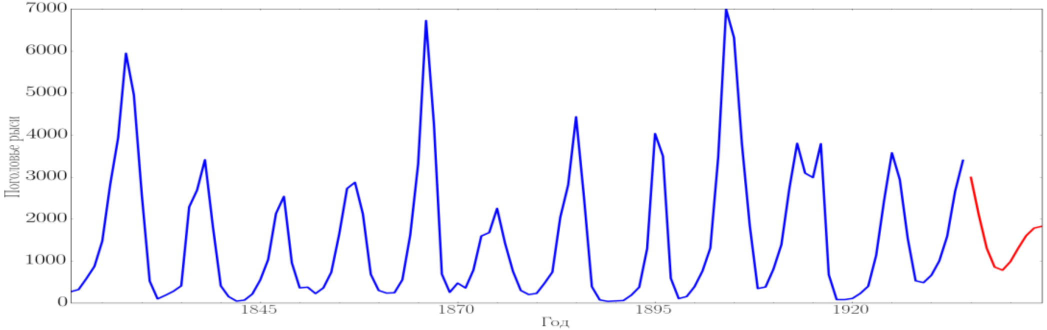
# ARIMA models (II)

УНИВЕРСИТЕТ ИТМО




ARMA(2,2)




ITsMOre than a UNIVERSITY

# ARIMA models (III)

УНИВЕРСИТЕТ ИТМО

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

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

Seasonality:

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

$$+\theta_S \varepsilon_{t-S} + \theta_{2S} \varepsilon_{t-2S} + \cdots + \theta_{PS} \varepsilon_{t-QS}.$$ + Q components with period S

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

ITsMOre than a UNIVERSITY

# ARIMA models (IV)



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:

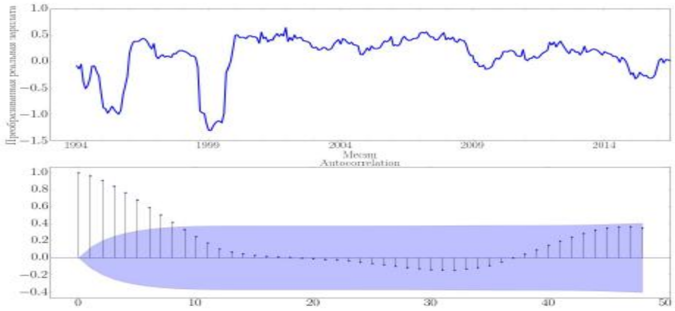
L – Likelihood function

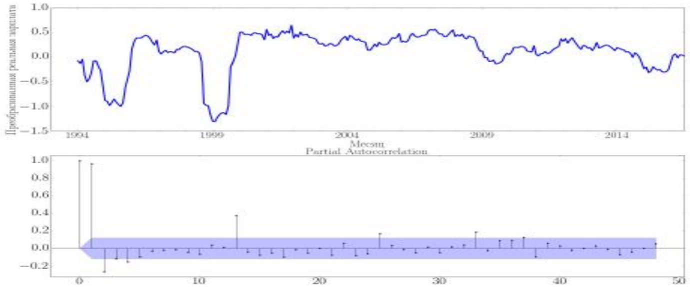
k = P + Q + p + q + 1 – number of model parameters

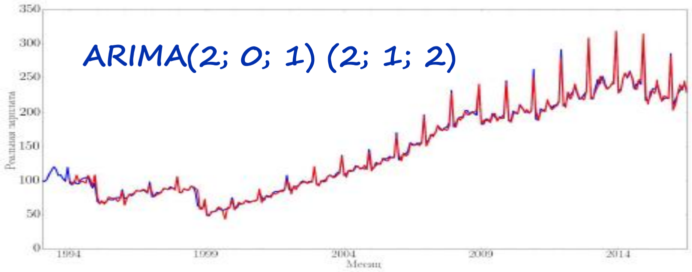
**Best model – model ARIMA(p,q)x(P,Q) with min AIC.**

# ARIMA models (IV)






**Q*S=Q’** – last seasonal lag with significant autocorrelation
**Q’ = 0 => Q=0**

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

**P*S = P‘** – last seasonal lag with significant partial autocorrelation
P‘ = 24 => P=2

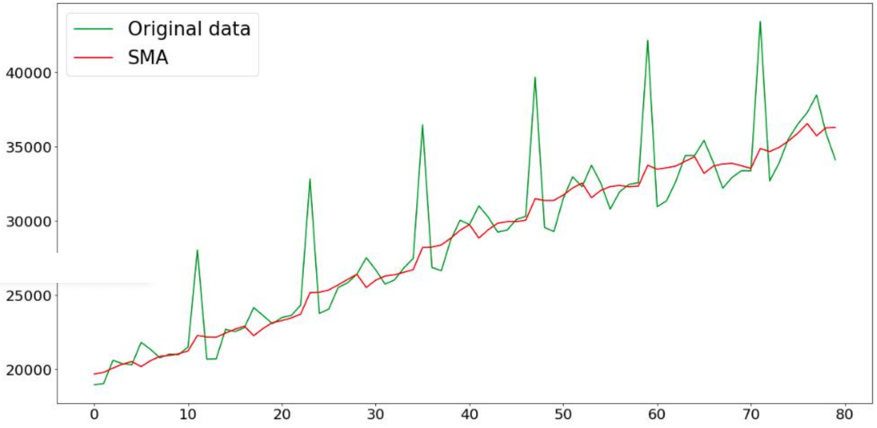
**p** – last non-seasonal lag with significant partial autocorrelation
p=2

AIC -> min(p,d,q,P,D,Q)

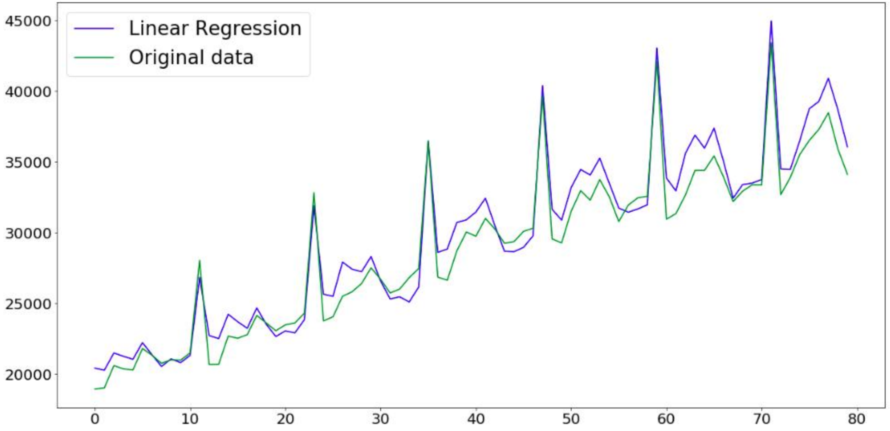
**=>ARIMA(2; 0; 1) (2; 1; 2) – best model**

# Simple forecasting models

Plot 1. Simple moving average




Plot 2. Linear Regression



# Metrics of forecasting quality



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

# Metrics of forecasting quality

УНИВЕРСИТЕТ ИТМО

## $R^2$

$R^2$ 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.

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

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

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

```
from sklearn.metrics import r2_score

print("Linear Regression R^2:",  round(r2_score(y, y_pred_lr), 3))
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.

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

```python
from sklearn.metrics import mean_squared_error

print("Linear Regression MSE:", round(mean_squared_error(y, y_pred_lr), 3))
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.*

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

```
from sklearn.metrics import mean_squared_error

print("Linear Regression RMSE:", round(np.sqrt(mean_squared_error(y, y_pred_lr)), 3))
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*

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

```python
from sklearn.metrics import mean_absolute_error

print("Linear Regression MAE:",  round(mean_absolute_error(y, y_pred_lr), 3))
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.

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

```
def mean_absolute_percentage_error(y_true, y_pred):
    return round(np.mean(np.abs((y_true - y_pred) / y_true)) * 100, 3)

print("Linear Regression MAPE:",  mean_absolute_percentage_error(y, y_pred_lr))
print("SMA MAPE:", mean_absolute_percentage_error(y , y_sma))

```

```
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.

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

```python
def smape(y_true, y_pred):
    return round(np.mean(np.abs((y_pred - y_true))/((np.abs(y_true)) + np.abs((y_pred)) / 2)), 3 )

print("Linear Regression SMAPE:",  smape(y, y_pred_lr))
print("SMA SMAPE:", smape(y , y_sma))
```

```
Linear Regression SMAPE: 0.026
SMA SMAPE: 0.147
```

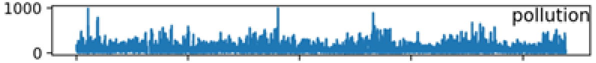
# Univariate and Multivariate forecasting



**Univariate:**

•One target time series

•Predicting based only on it



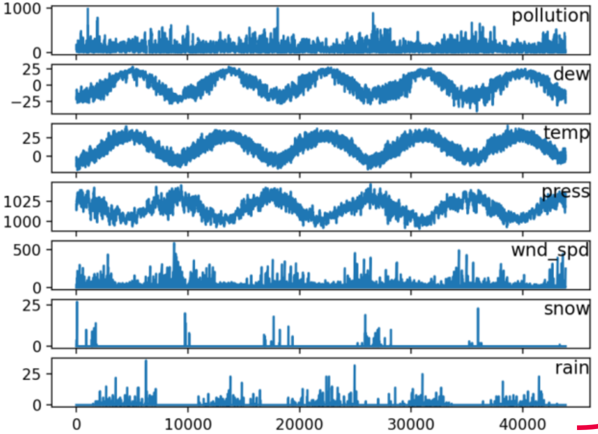

forecast

**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




forecast

**ML models !**

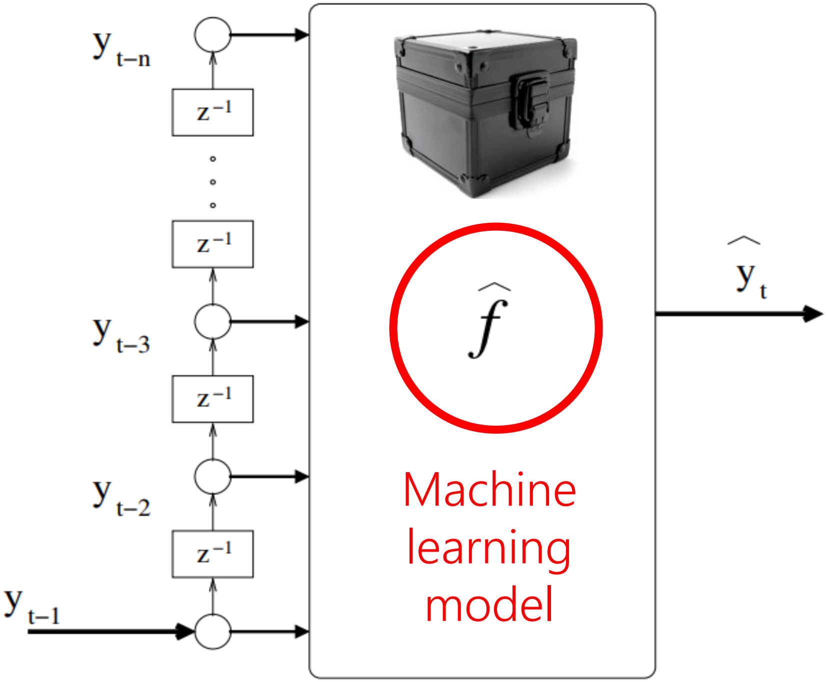
# Forecasting as machine learning problem (I)



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

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

*inputs* *outputs*

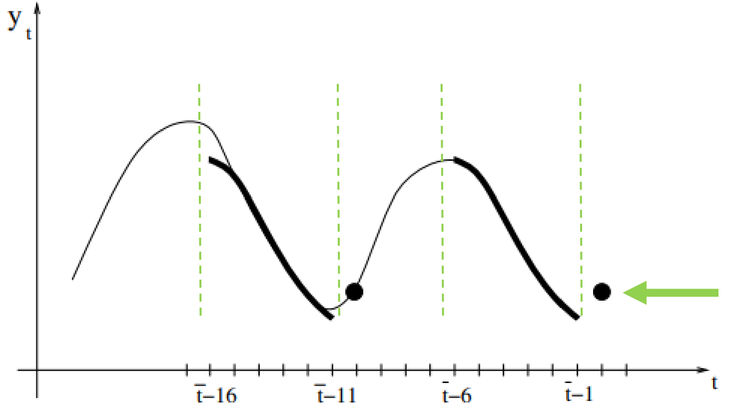


# Nearest neighbors model



## 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: $T_n = \{y_{t-6}, y_{t-5}, \ldots, y_{t-1}\}$
- search for the most similar (in some metric) pattern in the previous data



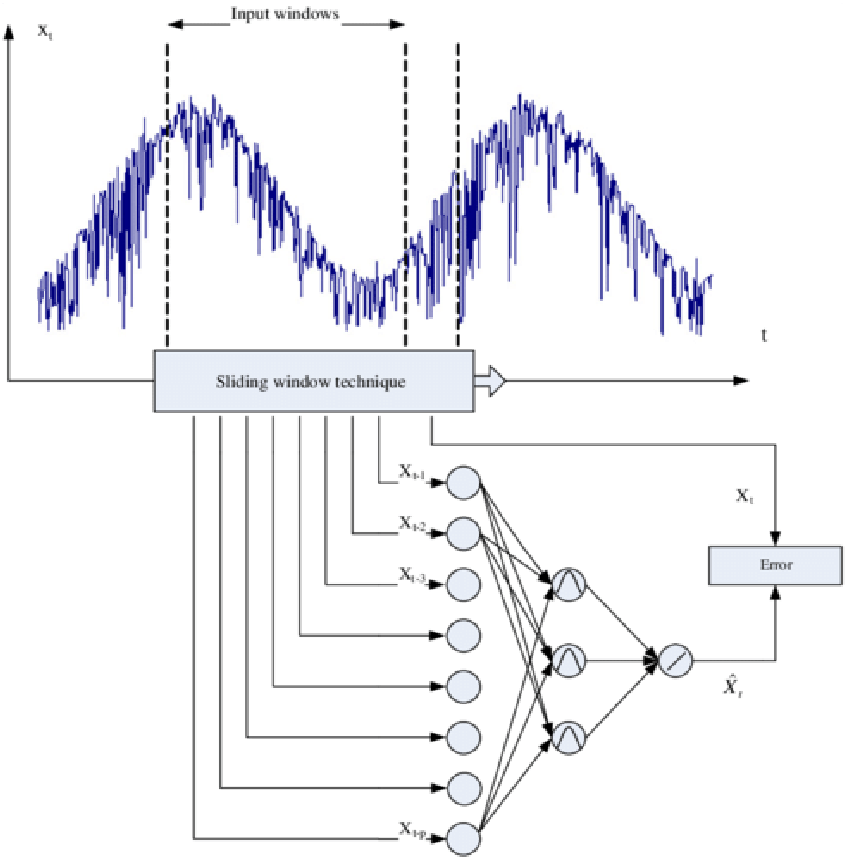

**Prediction is $y_t = y_{t-10}$**

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

# Dense model (MLP)

УНИВЕРСИТЕТ ИТМО

- *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*




ITsMOre than a UNIVERSITY

# 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"

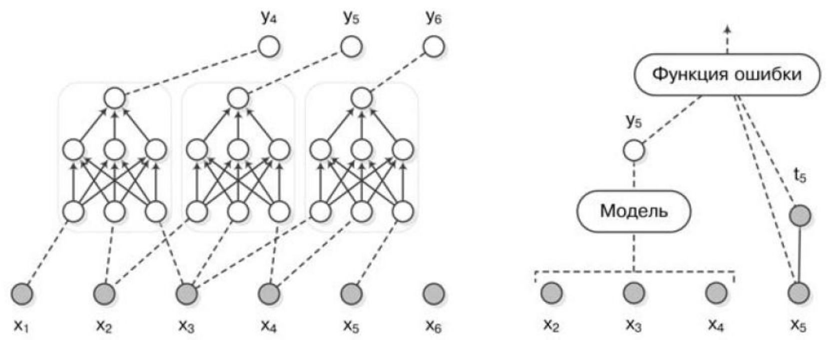
– it is the only reason to put "French" into the gap.

- 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..?

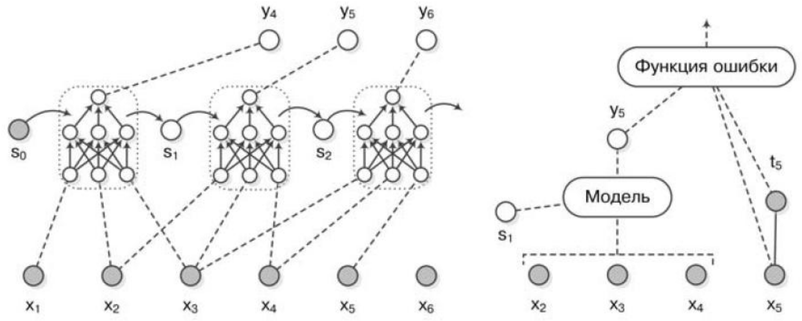
# RNN and MLP ("dense networks")



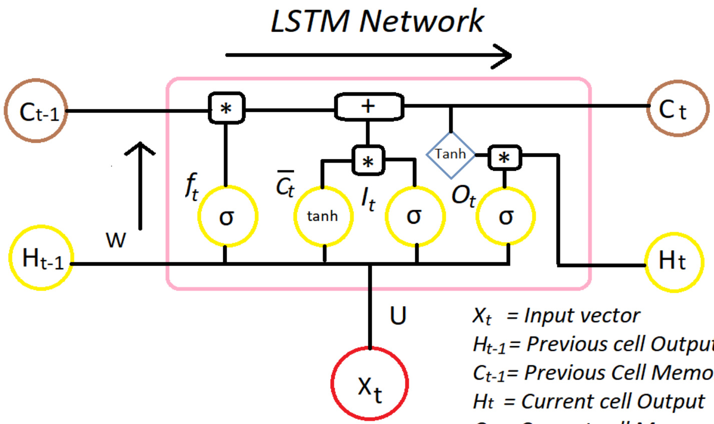
*Dense model:*




*Recurrent model:*



# RNN and LSTM




$\boxed{*}$ = Element-wise multiplication

$\boxed{+}$ = Element-wise addition

$$f_t = \sigma\,( X_t * U_f + H_{t-1} * W_f )$$
$$\bar{C}_t = \tanh\,( X_t * U_c + H_{t-1} * W_c )$$
$$I_t = \sigma\,( X_t * U_i + H_{t-1} * W_i )$$
$$O_t = \sigma\,( X_t * U_o + H_{t-1} * W_o )$$

$$C_t = f_t * C_{t-1} + I_t * \bar{C}_t$$
$$H_t = O_t * \tanh\,( C_t )$$

$X_t$ = Input vector

$H_{t-1}$ = Previous cell Output

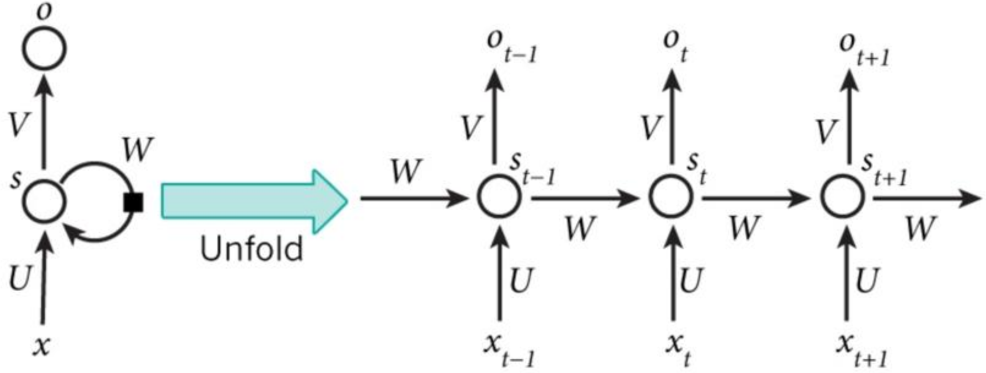
$C_{t-1}$ = Previous Cell Memory

$H_t$ = Current cell Output

$C_t$ = Current cell Memory

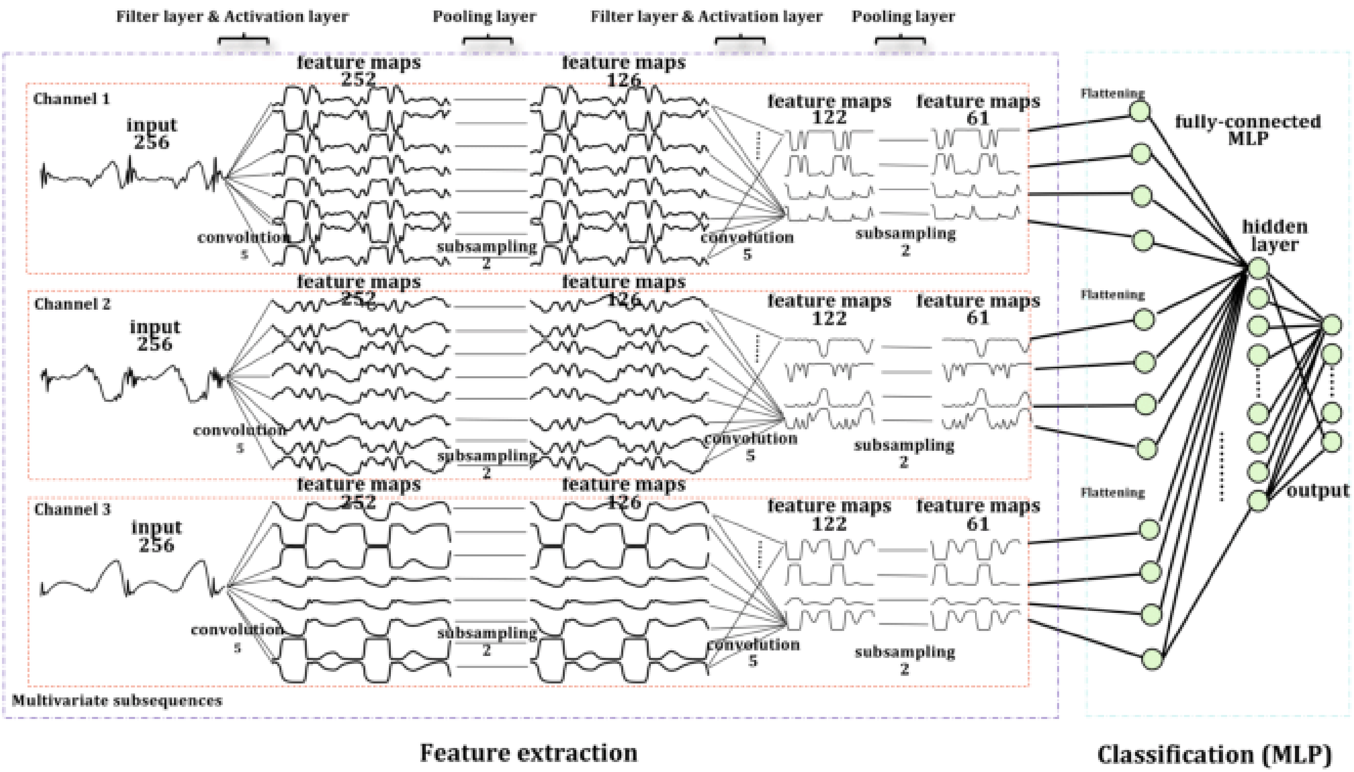
*W, U* = weight vectors for forget gate (f), candidate (c) , i/p gate (I) and o/p gate (O)

*Note : These are different weights for different gates, for simpicity's sake, I mentioned W and U*



# CNN for time-series classification

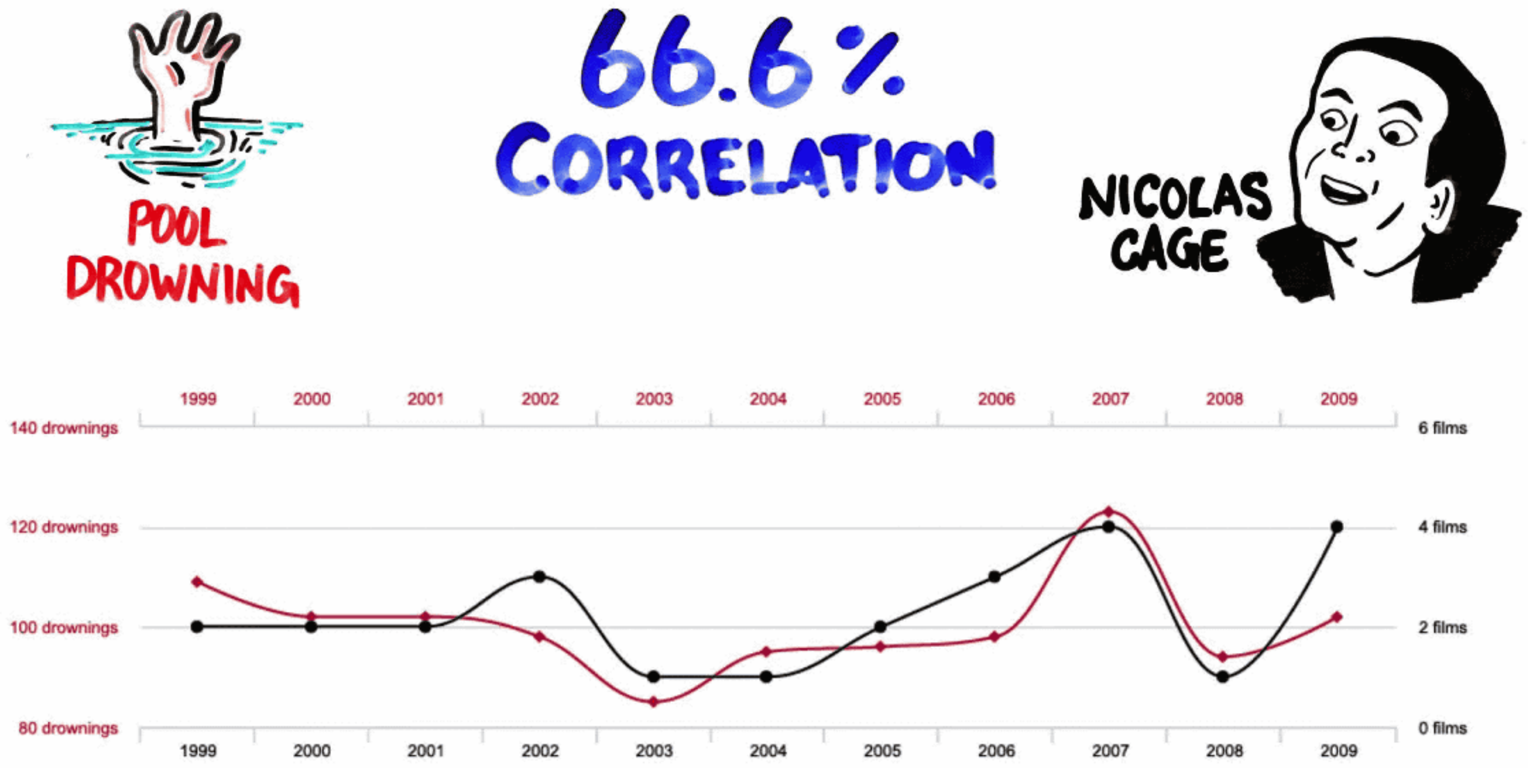





Convolution layers extract features of every time series for fully-connected MLP

# Multivariate time series and causality

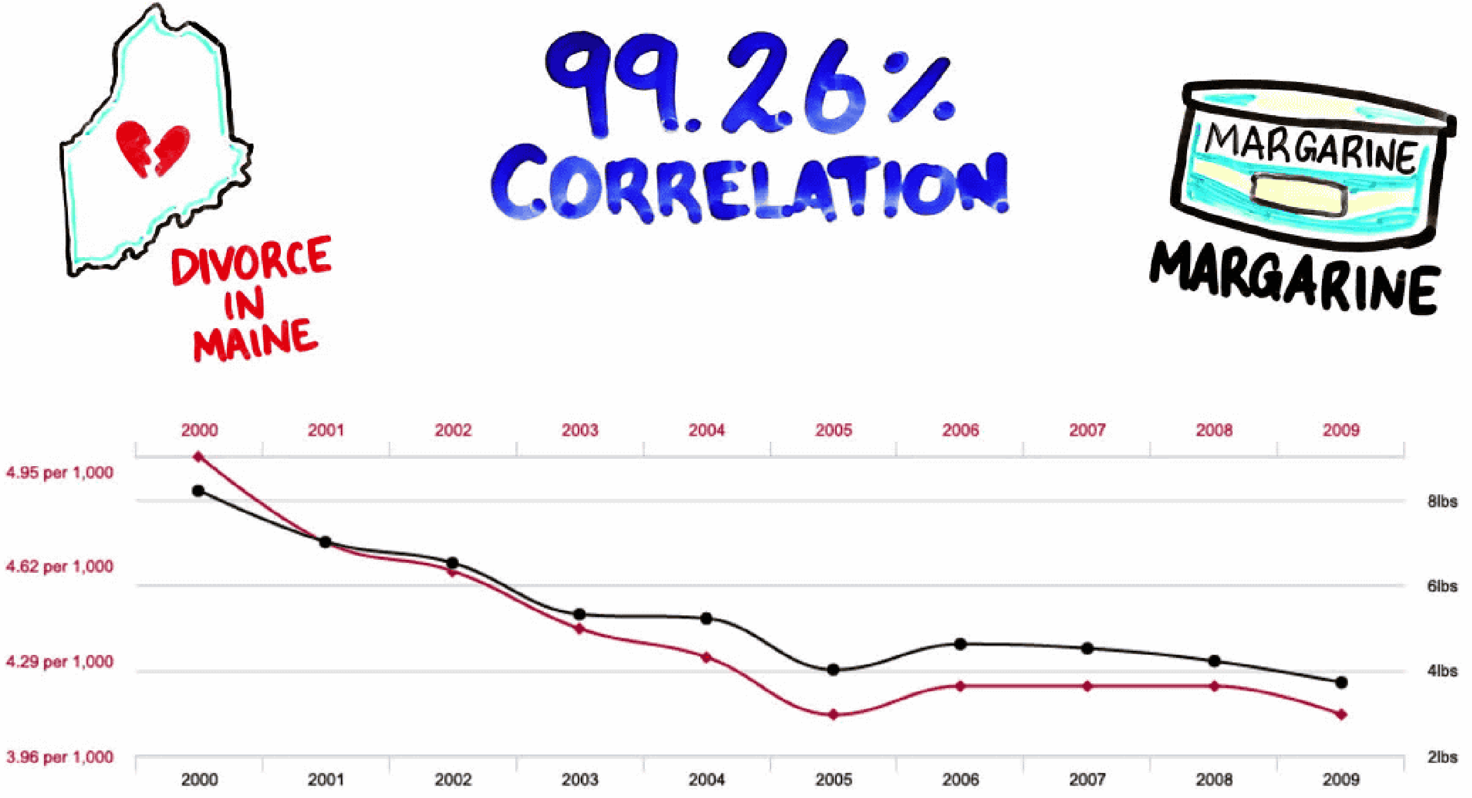
# Correlations and causality




Source: tylervigen.com/spurious-correlations

# Correlations and causality






Source: tylervigen.com/spurious-correlations

# Causality

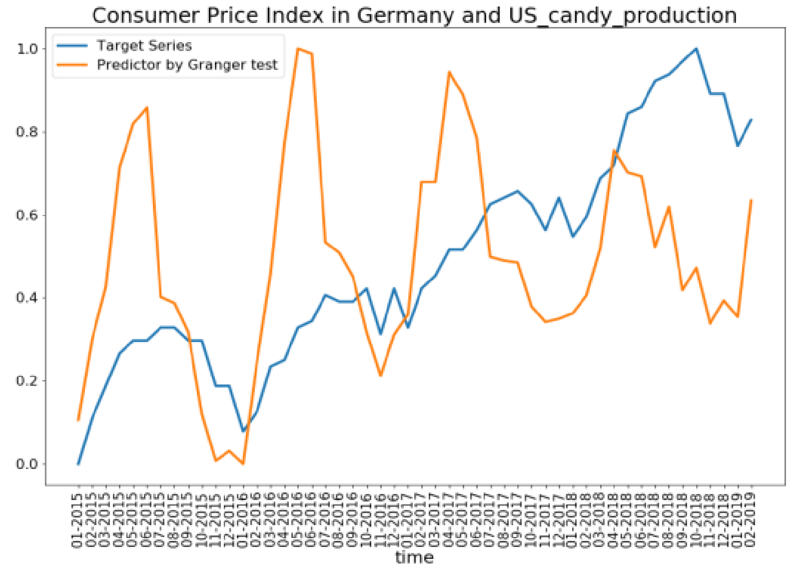


**Granger test**

- 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->Y) and predictor-goal (Y->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

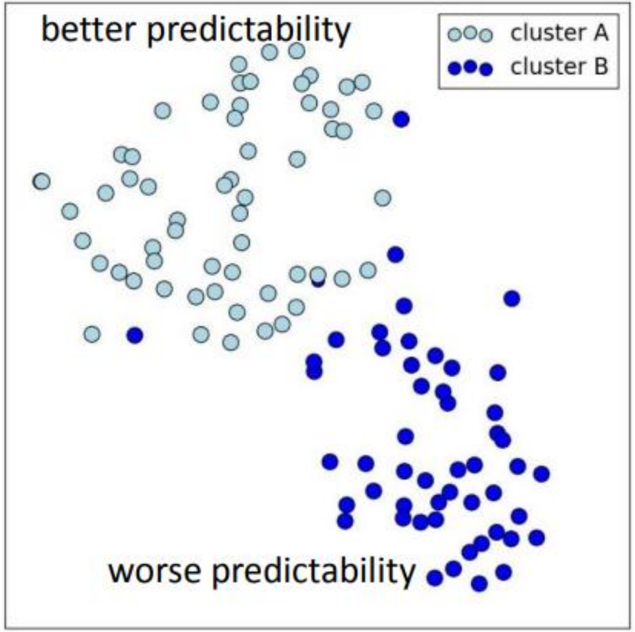
**Convergent Cross Mapping test**

- Create the shadow manifold for X, called Mx;
- 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|Mx );
- Compute the correlation between Y and Y|Mx;

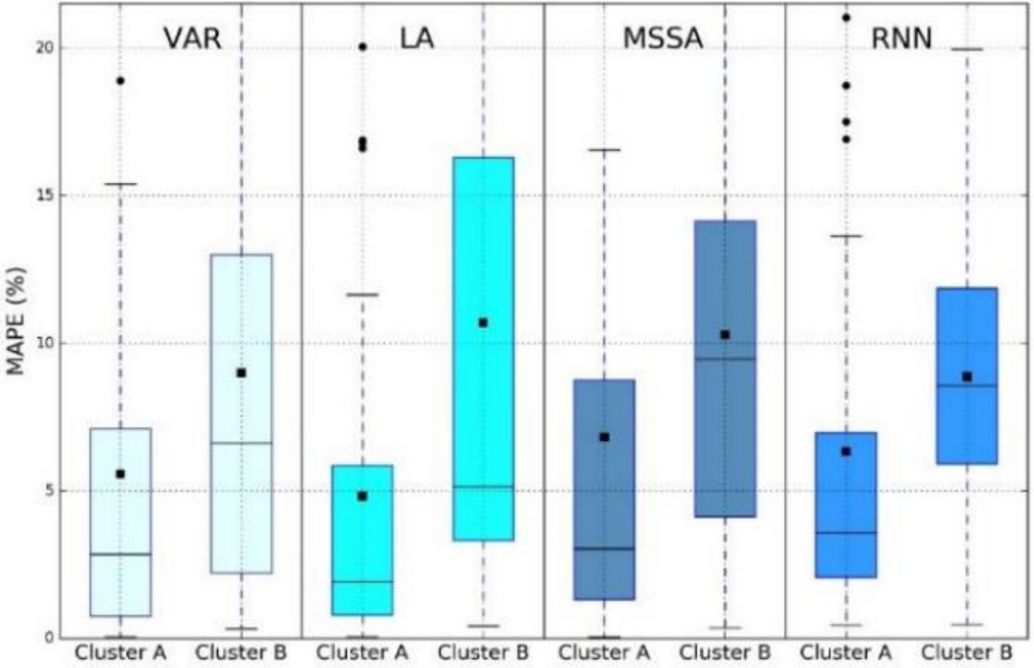
Eichler M. Causal inference in time series analysis // Causality: Wiley Series in Probability and Statistics. 2012. P. 6–28.
https://royalsocietypublishing.org/doi/full/10.1098/rsta.2011.0613

# Clustering the predictaility

УНИВЕРСИТЕТ ИТМО




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

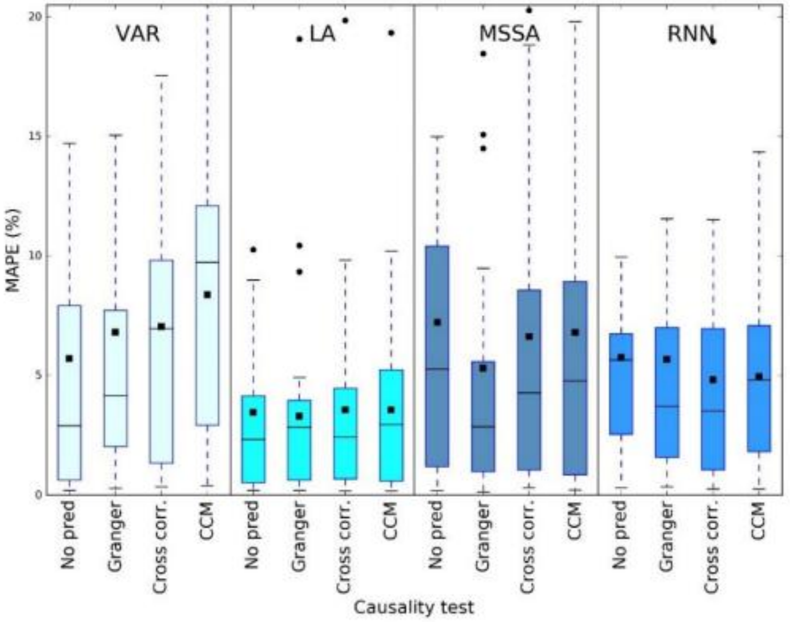



MAPE distribution in the clusters

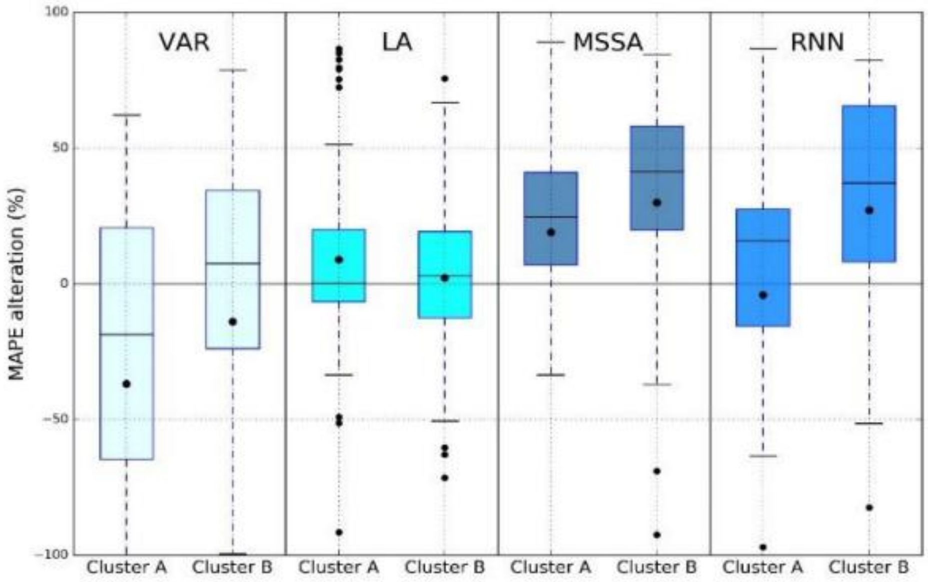
*P. Gladilin, A. Kovantcev, Analysis of multivariate time series predictability based on their features (2020)*

# 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.

*P. Gladilin, A. Kovantcev, Analysis of multivariate time series predictability based on their features (2020)*

# 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*

УНИВЕРСИТЕТ ИТМО

# Thank you!

ITsMOre than a UNIVERSITY