Second Laboratory

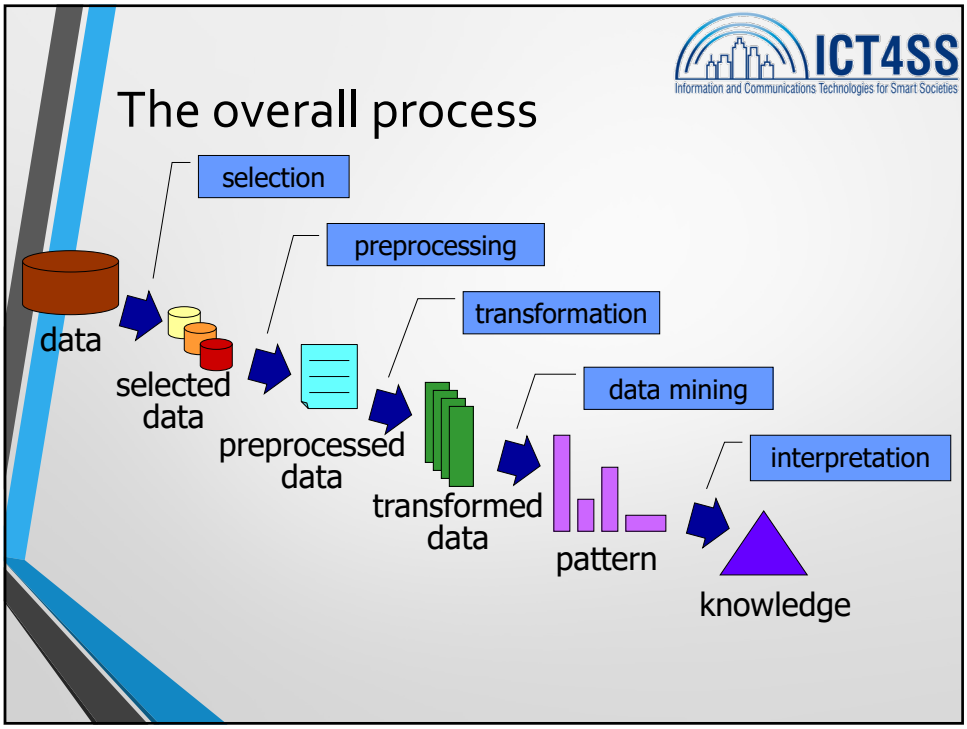
57

Lab 2 description

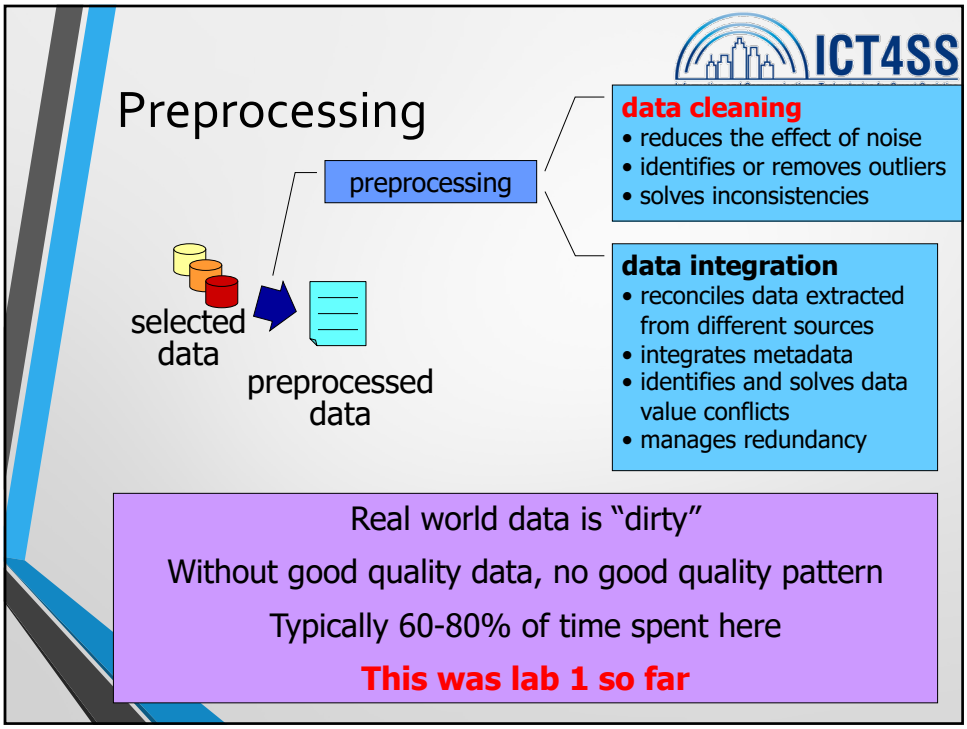
Goal: design an analytics that is able to predict the number of rentals in the future

- **Given** the time series of the number of rental in the past
- **Find** the best ARIMA model to predict the future number of rentals
- **Test** it properly

58

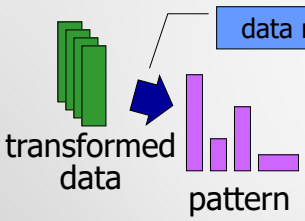

ICT4SS
Information and Communications Technologies for Smart Societies
The overall process
selection
preprocessing
transformation
data mining
interpretation
data
selected data
preprocessed data
transformed data
pattern
knowledge

59


ICT4SS
Preprocessing
preprocessing
selected data
preprocessed data
data cleaning
• reduces the effect of noise
• identifies or removes outliers
• solves inconsistencies
data integration
• reconciles data extracted from different sources
• integrates metadata
• identifies and solves data value conflicts
• manages redundancy
Real world data is "dirty"
Without good quality data, no good quality pattern
Typically 60-80% of time spent here
This was lab 1 so far

60

Data mining



Supervised learning
(Requires labeled data)
•Classification
•**Regression – with ARIMA**

Unsupervised learning
• Clustering
• Rule mining
• Anomaly detection

Which approach to use?
Which tools?
Typically 20-40% of time spent here

Lab 2 will be designing a sound regression model

61

Lab assignment

Given the time series of the number of rental in the past

- **Design** and **test** a prediction analytics
  - Each group has so far worked to get 3 time series for rentals in 3 different cities
  - Test with each of this time series
- *What are the best performance you can get?*
- *What are the impact of the parameters?*
  - *(p,d,q) and the training window size*
- *Why in some cities you get better/worse performance?*

62

Model design

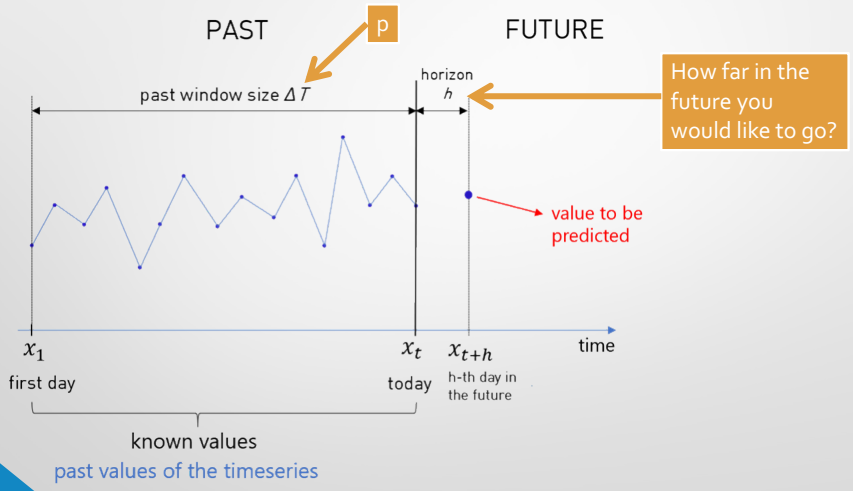
$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

Find the best ARIMA model to predict the future number of rentals

- How to choose (p,d,q) ?
  - p is the lag in the AR model => how much X_t depends on the past samples?
  - d is the number of differentiation steps to make the model stationary
  - q is the lag in the MA model => is there correlation in the noise?

63

Prediction in general



64

Choosing Model Specification

- ACF and PACF can be used for determining $ARMA$ model hyperparamters p and q

	$AR(p)$	$MA(q)$	$ARMA(p,q)$
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

  - Note that the selection for p and q is not unique

65

Model design

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j w_{t-j}$$

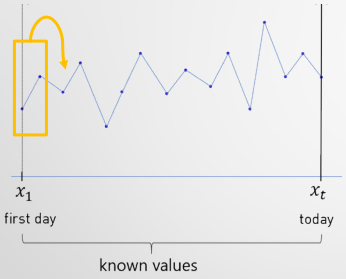
Find the best ARIMA model to predict the future number of rentals

- How to choose (p,d,q)?
  - p is the lag in the AR model => how much x_t depends on the past values?
  - d is the number of differentiation steps to make the model stationary
  - q is the lag in the MA model => is there correlation in the noise?
- **How to train the model?**
  - How much of past data you need to use to find ϕ_i, θ_j?

66

Training in practice

p=2



Derive the training dataset

X_{t-2}	X_{t-1}	Y_t
X_1	X_2	X_3
X_2	X_3	X_4
X_3	X_4	X_5
…	…	…
X_{t-2}	X_{t-1}	X_t

Given t past observations, we can build t-p training rows
All rows are our training set

67

Training in practice

$$X_t = w_t + \sum_{i=1}^{p} \phi_i X_{t-i}$$

We have 2 unknowns (ϕ_1, ϕ_2), so we have

$X_3 = \phi_1 X_2 + \phi_2 X_1 + w_3$

$X_4 = \phi_1 X_3 + \phi_2 X_2 + w_4$

$X_5 = \phi_1 X_4 + \phi_2 X_3 + w_5$

…

$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + w_t$

This are our t–p equations

X_{t-2}	X_{t-1}	Y_t
X_1	X_2	X_3
X_2	X_3	X_4
X_3	X_4	X_5
…	…	…
X_{t-2}	X_{t-1}	X_t

We use the Maximum Likelihood Estimation or
Least Square Error to find the $\{\phi_i\}$ which minimize the error

$r_i = y_i - (\phi_1 X_{i-1} + \phi_2 X_{i-2}) \Rightarrow MSE = \Sigma(r_i(\phi_1, \phi_2))^2$

which is solved by imposing the partial derivative equal to 0

68

Training and testing in practice

Recap: given past observations from 1 to t

- I can get $(t\text{-}p)$ rows to **train my model** and obtain (ϕ_i)
- This allows me to predict $X'_t = \Sigma\ \phi_i\ X_{t-i}\ + w_t$

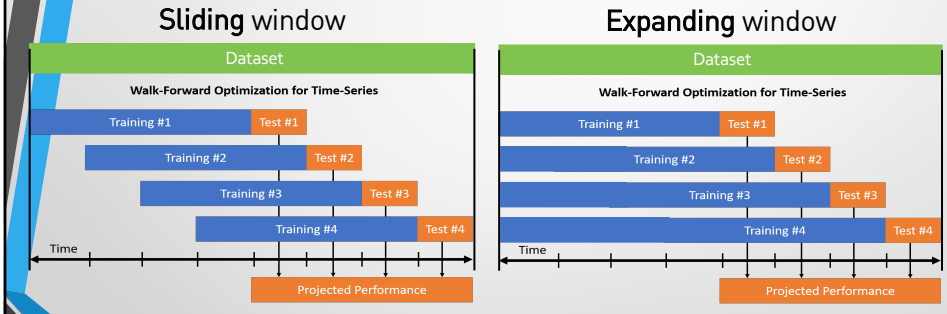
- Then time passes and I get the actual observation X_{t+1}
- I **compute the error** $r_i = X'_t - X_t$ => *this allows me to evaluate the prediction accuracy via MSE, MAPE, MAE, …*

- And then I can **compute a NEW model**, since now I have one more observation => one more line in my training set
  - Sliding window or expanding window
  - How big shall be the initial training window?

69

Train & test method

Walk – forward validation



70

Lab assignment

Given the time series of the number of rental in the past

- **Design** and **test** a prediction analytics
  - Each group has so far worked to get 3 time series for rentals in 3 different cities
  - Test with each of this time series
- *What are the best performance you can get?*
- *What are the impact of the parameters?*
  - *(p,d,q) and the training window size*
- *Why in some cities you get better/worse performance?*

71

Arima models in python

72

Python and Scikitlearn

- Python offers easy to use libraries to deal with data processing
  - pandas for data management
  - matplotlib for easy visualisation
  - scikitlearn for ML functions

73

Generic work flow

1. Read and prepare the data
   1. Import you time series
   2. Check for missing data (ARIMA cannot deal with missing data!)
   3. Explore data
2. Train and test the ARIMA mode
   1. Do a grid search to optimize p,d,q
   2. Check the impact of window size
   3. Compare expanding vs sliding window
3. Visualise results

74

Some useful functions

```
import pandas as pd
pd.read_csv(filepath, sep=',', usecols=None, dtype=None))
    df = pd.read_csv('TorinoMoved.csv', usecols=['date','Moved'],
        dtype={'Moved':np.float64})
pd.plotting.autocorrelation_plot(df)
```

75

Some useful functions

```
# Compute ACF and PACF
from statsmodels.tsa.stattools import acf, pacf
lag_acf = acf(df, nlags=48)
lag_pacf = pacf(df, nlags=48, method='ols')

# Train an ARIMA model
from statsmodels.tsa.arima_model import ARIMA
# instatiate the model
model = ARIMA(df, order=(p,d,q))
# Fit it
model_fit = model.fit(disp=0, method={'css-mle','mle','css'}, maxiter=500, )
```

This is the loglikelihood to maximize. If "css-mle", the conditional sum of squares likelihood is maximized and its values are used as starting values for the computation of the exact likelihood via the Kalman filter. If "mle", the exact likelihood is maximized via the Kalman Filter. If "css" the conditional sum of squares likelihood is maximized.

The resulting model

Print results or not

Maximum number of iterations for solving

76

Some useful functions

```
#Example
from statsmodels.tsa.arima_model import ARIMA
p=2; d=0; q=2
# fit model
model = ARIMA(df, order=(p,d,q))
model_fit = model.fit(disp=0)

#plot results
fig = plt.figure(figsize=(15,5))
plt.plot(df)
plt.plot(model_fit.fittedvalues, color='red')
```

This is fitting ONE model on all time series.
The fitted values then run the ONE model through the all time series, getting X'_t from $\{X_{t-i}\}$

77

Some useful functions

The resit method returns the residuals r_i

```
# plot residual errors
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
# and check their distribution
residuals.plot(kind='kde') =
```

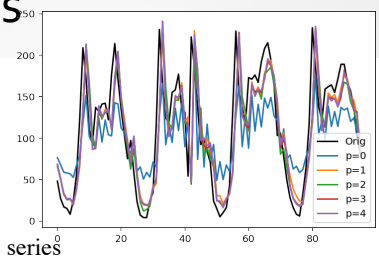
78

Some useful functions

```
%%time   # print how much time was needed to run all this
X = df.values.astype(float)   # extract the time series
size = 24*7*2 # amount of data in the initial window for training
print("training on size", size)

test_len = 96 # The number of tests to be done for which we repeat the train and test
lag_orders = (0,1,2,3,4) # We want to check the parameters - so we repeat this for different values of p
predictions = np.zeros( (len(lag_orders), test_len) )  # just initialize an array to store prediction for each lag_order

d=0; q=1  # fix d and q, then do a grid search on p
for p in lag_orders:
    print('Testing ARIMA order (%i,%i,%i)' % (p,d,q))
    train, test = X[0:size], X[size:(size+test_len)] # split the train and test - we use an expanding/sliding window
                                        # approach –  data in train is used to train
                                        # then predict now+1. We then expand/slide the time and repeat
    history = [x for x in train] # this is the past data used for training

    for t in range(0, test_len): # repeat for all tests we want to do
        model = ARIMA(history, order=(p,d,q))  # make a new model
        model_fit = model.fit(disp=0, maxiter=500, method='css')  # fit it using the desired parameters
        output = model_fit.forecast() # here we get the forecasted data at t+j

        yhat = output[0] # get t+1
        predictions[lag_orders.index(p)][t]=yhat #and store it
        #print('t=%i: prediction=%.2f, actual=%i, error=%1.2f (%.2f%%) -- ' % (t, yhat, obs, yhat-obs, 100*(yhat-obs)/obs))
        obs = test[t] # slide over time, by putting now+1 into past
        history.append(obs)
        history=history[1:] # and drop the first sample if you wish to use sliding window. For expanding – just comment this line
```

79

Some useful functions

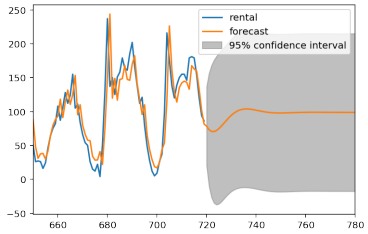



```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

plt.plot(test, color='black', label='Orig') # plot the real time series
for p in lag_orders:  # now for each model compute performance and plot predictions
    print('(%i,%i,%i) model => MAE: %.3f -- MAPE: %.3f -- MSE: %.3f -- R2: %.3f'
                        % (p,d,q,
                        mean_absolute_error(test, predictions[lag_orders.index(p)]),
                        mean_absolute_error(test, predictions[lag_orders.index(p)]) / test.mean()*100,
                        mean_squared_error(test, predictions[lag_orders.index(p)]),
                        r2_score(test, predictions[lag_orders.index(p)])))

    plt.plot(predictions[lag_orders.index(p)], label='p=%i' % p )
plt.legend()
```

80

Some useful functions

```
# now let the prediction go for now just horizon 1 – but for more time
fig, ax = plt.subplots()
ax = df.loc[650:].plot(ax=ax) # plot the data from step 650 up to now

# plot now the predicted data for the future days – we have 720 sample – so let's go up to 780
fig = model_fit.plot_predict(650, 780, dynamic=False, ax=ax, plot_insample=False)
```



81