Real Time Electric Load Forecasting
Data Mining Techniques on Distributed Computing

Dr. Eng. Felice Tuosto
felice.tuosto@eng.it
OUTLINE

• Introduction

• Problem statement

• Proposed solution

• Experimental results
Introduction/ Aim & Benefits

Aim:
To estimate the electric load along a future time horizon on the basis of the available information (actual system state and history and so on)

Benefits:
Several players of the energy sector could benefit from prompt, accurate and reliable predictions of the electric load (demand).

Benefits are both technical (management/control) and financial.
Introduction/ Needs of network operators and utilities

- Real time dispatching
- Resources supply
- Power reserve supply
- Trading on the Electric Market
- Asset Planning

Needs of network operators and utilities
Introduction/ Load Forecast (LF): brief description

**Customer:**
- Generic transmission/distribution system operator

**Project:**
- SW to *predict/archive* the electric load demand

**Aims:**
- Real time - very short term forecasting ($T_s=15\text{min}$) of the *zonal* active power (6 forecasts within 5min)
- Real time - very short term forecasting ($T_s=15\text{min}$) of the *nodal* active and reactive power (500 forecasts within 5min)
- Real time - very short term forecasting ($T_s=15\text{min}$) of the significant *weather* variables (6 forecasts within 5min)
- Real time *errors* estimation
Introduction/Load Forecast (LF): brief description

**Forecasts**

**System Operator**

**Actions**

**Market Operator**

**Sensors**

- Weather Data
- Load Demand Data
- Economic Factors
- Special Events

**Stabilize**

**Optimize**
Beginning from methodology/ CRISP-DM

Load Forecasting (LF) is a complex problem:

The selection of the most appropriate LF model requires the deep knowledge of the electric power system and its own control and management strategies: Electric energy cannot be stored.
Learning with Python…
Problem Statement/ **Six things to know about prediction**

1. Load Forecasting (LF) is a stochastic problem
2. All the predictions are error affected
3. Some predictions are useful
4. All the predictions can be improved
5. The accuracy is never ensured
6. Having multiple models is always preferable
Problem Statement/ LF forecasting problem

Load Forecasting (LF) is a complex problem:

- **Non-linear** dynamics are involved
- Depends from several variables not directly measurable
- Depends from several further variables that must be forecasted

The output from a wind and solar farm are highly unpredictable
Problem Statement/ LF problem classification

Methodologies classification:

- Statistical (es. time series AR, MA, ARMA, SARIMAX)
- Half heuristics (es. similar days)
- Machine learning (ANN, SVM, FL, etc.)

Temporal classification:

- **Very-short term prediction**: from one minute to some hours
  (es. real time power system control)
- **Short term prediction**: from one hour to one day
  (es. GU daily costs optimization)
- **Medium term prediction**: from one week to one year
  (es. out of service programming, network maintenance etc..)
- **Long term prediction**: more than one year
  (es. Investments plans, Reinforcement plans etc..)
Problem Statement/ LF problem classification

Spatial classification:
- National
- Zonal
- Nodal
Problem Statement/ Variables involved

**Customers typology:**
- Residential
- commercial
- industrial

**Meteorological factors:**
- $dT=1^\circ C \Rightarrow dP=1\%$
- overcast sky than a clear sky $\Rightarrow dP=3-4\%$
- $dw=4\text{Km/h} \Rightarrow dP=1\%$

**Load historical values:**
- It’s always necessary to **learn** from history

**Time factors:**
- Time of day / day of week (working / eves / holidays)
- Cyclical features: daily, weekly, seasonal, annual
Problem Statement/ Variables involved

Special events:
• Strikes
• sporting events
• political events
• ...

(Some events are very difficult to consider in a standard LF model hence they should be placed in a dedicated model ...)

Economic factor:
• commodity prices
  • oil
  • gas
  • coal
• emissions
• growth
• population
• GDP
• income
Proposed solution/ Engineering solution

**LoadForecast:** SW *automatically* and in *real time* provides the electric load demand predictions (active and reactive electric power)

- **Spatial resolution**
  - National, zonal and nodal demands (single Autotransformer 380/220/150 kV)
  - Easily scalable and characterizable for *Smart Grid* applications

- **Time horizons**
  - Time intervals: from 5 minutes to some days
  - Sampling time: from few minutes to some hours

**Use:** provide data to the EMS:
- OPF (Optimal Power Flow)
- ORPF (Optimal Reactive Power Flow)
- AD (Advance Dispatching)
Case study. Data source.
Proposed solution/ Variables involved

- **Spring**
- **Spring -> Summer**
- **Summer**

- **$P_{cons}$ [MW]**
- **$T_{a}$ [$^\circ$C]**
- **$C_{op}$ [adim]**
Proposed solution/ Off line forecasting

- Acquire csv files containing all the measurands necessary for the prediction
- Set the data structure needed by the learner (Matrices: <features,target>, state, param. etc.)
- Initiate the calibration procedure of hyperparameters and training (offline-training)
- Initiate the forecasting procedure
- Archive the results in csv file
Proposed solution/ Off line forecasting

It is used off-line for **best model selection**. It provides useful **functionalities** for:

- Time series analysis
- Data-merge modules definition, reconstruction and downsampling ($T_c=\text{cost.}$)
- Filtering
- Data clustering
- Features construction and filtering (preselection of the most important features)
- LF model set-up (statistical, MachineLearning, Hybrid)
- Parameters calibration
- Forecasting **scenario simulation** (es. hottest and coldest days)
Proposed solution/ Real time forecasting

**Supervisor** controls the operating status and synchronizes all the SW modules:

**Initialization stage:**
1) To **acquire** all the current and historical data needed to the prediction from the LF-DB
2) To set the data structure needed by the learner (Matrices: <features,target>, state, param)
3) To initiate the calibration procedure of **hyperparameters** and training (offline-training)
4) To initiate the forecasting procedure
5) To archive results in LF-DB

**Steady-state stage:**
1) To acquire from the last updated data from LF-DB
2) To **update** the data structures needed by the learner (Circular matrices)
3) To initiate the **adaptive training** procedure (**online-training**)
4) Return to step 1)
Proposed solution/ Project requirements

Precise/Accurate:
Mean and standard error below a certain threshold value

Multi-platform:
Unix, Linux, Windows

Modular:
Simplifies the development, testing and maintenance operations

Scalable vertically/horizontally:
Fundamental requirements in BigData applications

Efficient:
Fast and compact data structures to manage them in main memory mass
A forecast made after 5 min is a wrong prediction!!!

Reliable/Resilient:
Applications used in control room (num.pref.ok/pred. even on few data)
Proposed solution/ Project requirements

Adaptive: Forecasts must adapt in real time to the dynamics of stochastic process

Automatic: Must acquire and store predictions when new measurements are provided without external intervention

Flexible: Must be fully-configurable (connection parameters, archiving, forecasting models, computational tree)

General: Must give the possibility to integrate various SW components (filters, clustering models, features of the reduction models, forecasting models etc.) allowing to build new LF models also by non-expert users

Integrable: Must provide standard interfaces to access the various data related to calculations
The approach followed in this project is based on the following methodologies:

- **Data Mining** (DM)
- **Software Engineering** (SE).

An *iterative* (and *interactive*) procedure based on the main steps here reported was used:

1) **Quick Prototyping:**
   Rapid development of a prototype that meets only the requirements on the forecast *errors* (precision and accuracy) → DM

2) **Engineering:**
   Engineering of the prototype in order to meet the remaining requirements → SE
Proposed solution/ Quick prototyping

Objective (Validation Set):

\[
MSE(VS) = \frac{1}{N} \sum_{k=1}^{N} (P_{cons_k} - P_{prev_k})^2 \approx \sigma^2 + ModelBias^2 + ModelVariance^2
\]
Proposed solution/ Quick prototyping

Objective (ValidationSet):

\[
MSE = \frac{1}{N} \sum_{k=1}^{N} (P_{\text{cons}_k} - P_{\text{prev}_k})^2 \approx \sigma^2 + \text{ModelBias}^2 + \text{ModelVariance}^2
\]

**Iterative procedure:**
- Study and quick test of several DM techniques
- In each iteration you will have a **Prototype**\((k)\)
- The procedure is repeated several times
- The realization of the generic P\((k)\) is useful for the realization of P\((k+1)\)
- MSE\((k+1)\) < MSE\((k)\)

**Benefits:**
- One obtains quickly several working versions
- It promotes *discussion* with the customer and industry experts
- One can immediately exploit the expert advices (often dictated by the *experience/ intuition*)
Proposed solution/ Quick prototyping: Iterative procedure

Problem identification / solution used

Data acquisition

Data exploration

Data filtering

Data transformation

Features generation

Features selection

Model & Trainer selection

DataSet separation: TraingSet/TestSet

Hyperparameter tuning

Training on TrainingSet

Predictions on TestSet

Validation on ValidationSet
Problem statement and solution identification:
- Classic methodologies
- Machine Learning methodologies

Data acquisition:
- From different database (Oracle, MySQL)
- From file (CSV, XML, HDFS)

Data exploration:
- To understand the basic properties of the available time-series
- No one optimal scheme:
  - knowledge of the stochastic process
  - based on experience
- Time-consuming
- Some methodologies can help:
  - multiple graphs (box plot, scatter plot, histogram)
- Frequency analysis (periodicity: daily, weekly, yearly, etc.)
- Some statistical indices summarize some hidden features in Big-data (es. mean, variance, correlation and so on)
- Outliers detection
- Bad data detection
Proposed solution/ Quick prototyping: Iterative procedure

Data Filtering:
- *Critical* for the success of any learning algorithm
- Outliers, spike, bad-data detection and removing
- Different sampling time
- Noise filtering
- Missing *data reconstruction* (spline, Kalman-Filter, Machine-learning...)
- Short and corrupted time-series are removed *automatically*
Data transformation:

- Irrelevant or redundant time series removal/reduction via:
  - Correlation analysis, cross-correlation
  - Algorithm based on DT, Gradient Boosting Trees (GBT), Principal component analysis (PCA)

- Normalization, *dummy* variables codification

- Clusterization:
  - day of the year
  - type of the day
  - hour of the day
Proposed solution/ Quick prototyping: Iterative procedure

Features generation:

➔ The most important step in a Machine Learning project
➔ The aim is to represent the complex relationship between available information and future information (encoded by DataSet)
➔ The predictor will learn these relationships (learn by example)

The dataset is composed by taking from different databases all the data pairs:

\[ <z(k), y(k)> \]

\[ y(k) = \text{target value} \]

\[ z(k) = \text{vector of features} \]

known values at the k-th step (that from the statistical analysis we assume are related to the target)

\[ \text{DataSet} = \{ <z(k), y(k)> : k = 1...N \} \]
Proposed solution: Quick prototyping: Iterative procedure

Features generation:

- ACF, PACF, XCF can help but there are no general rules (considerations based on specific knowledge in the field)

- **Composition**: to make up new features from the available ones (es. environmental temperature $T(k)$ and also $T(k)^2$, $T(k)^3$...)

- **Decomposition**: proper codification (es. codification of the weak of the day, hour of the day)

![Diagram of proposed solution](image-url)
Proposed solution/ Quick prototyping: Iterative procedure

**Features generation: historical load**

Historical load is used as input for model prediction

The learner uses various range of data:

- the most recently available load
- the load of yesterday (in a range of same h)
- the load of the same day of the last week
- the load of the same hours for one week
- the same values for the last few years
Proposed solution/ Quick prototyping: Iterative procedure
Proposed solution/ Quick prototyping: Iterative procedure

Features generation: composition example

- **Weekday index:** is an important load affecting factor in view that different days of a week generally have different load shapes.

- **Hour index:** different hours of a day generally have different load shapes.

- **Month index:** winter, summer
**Features generation: weather information  WINTER**

- Includes wind-chill temperature, humidex, wind speed, cloud cover, and precipitation.
- Temperature that is felt could be much lower than air temperature in winter cause of wind, so wind-chill temperature is used for winter.

\[ T_{wc} = 35.74 + 0.6215T_a - 35.75v^{0.16} + 0.4245T_a v^{0.16} \]

- \( T_{wc} \) = wind-chill temperature
- \( T_a \) = air temperature
- \( v \) = wind speed
Proposed solution/ Quick prototyping: Iterative procedure

**Features generation: weather information** SUMMER

- The combined effects of heat and humidity cause high level of discomfort in summer
- Therefore, **humidex** that measures the combined effect of heat and humidity is used for summer

\[ H = T_a + 0.5555 \times \left( 6.11 \times e^{5417.753 \times \left( \frac{1}{273.15 - \frac{T}{D}} \right) - 10} \right) \]

- \( H = \) humidex
- \( D = \) dev. point
Proposed solution/ Quick prototyping: Iterative procedure

Features selection and data clustering:

- To reduce the **cardinality** of the problem:
  - speed up the response of the learning algorithm
  - improve the accuracy of the algorithm
  - improves the comprehensibility of the results

- Ad hoc (by hand) methods based on industry specific knowledge or **intuition**
  - difficult if there is a large number of characteristics
  - every hypothesis must be validated

- Automatic methods:
  - Univariate feature selection
  - Recursive feature elimination
  - Tree-based feature selection (**GBT**)
Proposed solution/ Quick prototyping: Iterative procedure

Data clustering example of ad hoc methods:

**Similar Day-Based Load Input Selection:** days are selected so as to have:
- the same weekday index
- similar weather
- similar index day-of-a-year (stagionality)

\[ \min_{i} \sum_{t=1}^{24} \left| w^f(t) - w^i(t) \right|, \text{where } i \in \Theta \]

- Scatter plot and correlation coefficient of common and proposed methods

<table>
<thead>
<tr>
<th>Common (versus yesterday)</th>
<th>Proposed (versus similar day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient = 0.65</td>
<td>Correlation coefficient = 0.91</td>
</tr>
</tbody>
</table>
Proposed solution/ Quick prototyping: Iterative procedure

Model & Trainer selection:

- Time-consuming
- SARIMAX (Kalman Filter)
- Linear regression LassoLarsCV
- K nearest neighbors
- MARS
- Support Vector Machines
- Neural Networks
- Decision Trees
- Random Forest
- Gradient Boosting Trees
- Deep Learning

DataSet split

- TrainingSet:
- TestSet:
Proposed solution/ Quick prototyping: Iterative procedure

Hyperparameters tuning:

- Research of parameters that minimize RMSE (k-fold Cross Validation)
- Time-consuming
- Several research methods
  - GridSearch
  - RandomSearch
  - GeneticAlgorithm
- Good results but slow
- Not applicable online but they give us a first idea
Proposed solution/ Quick prototyping: Iterative procedure

Training on *TrainingSet*:

- Several algorithms more or less quick

Forecasting/Validation on *ValidationSet*:

- RMSE calculation (precision and accuracy)
- if RMSE>threshold => go back to the previous step
- else => END
Proposed solution/ Quick prototyping: choosing the right estimator

**classification**
- SVC
- Ensemble Classifiers
- KNeighbors Classifier
- SGD Classifier
- Naive Bayes
- Text Data
- Linear SVC

**regression**
- SGD Regressor
- ElasticNet
- Ridge Regression
- SVR(kernel='rbf')
- Ensemble Regressors

**clustering**
- KMeans
- MeanShift
- MiniBatch KMeans
- VBGMM

**dimensionality reduction**
- Randomized PCA
- Isomap
- Spectral Embedding
- LLE

**scikit-learn algorithm cheat-sheet**

START

get more data

>50 samples

predicting a category

few features should be important

<100K samples

predicting a quantity

<100K samples

looking

<10K samples

number of categories known

<10K samples

just luck

<10K samples

tough luck

<10K samples

Generalized Linear Models

Not Working

NOT WORKING

NOT WORKING

NOT WORKING

NOT WORKING
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Stability</th>
<th>Training Time</th>
<th>Memory Usage</th>
<th>Adaptivity</th>
<th>Automatic Features Selection</th>
<th>Forecast Error</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SARIMAX</td>
<td>med3</td>
<td>veryhigh4</td>
<td>low3</td>
<td>low3</td>
<td>low1</td>
<td>high1</td>
<td>discharged13</td>
</tr>
<tr>
<td>LassoLars</td>
<td>high4</td>
<td>high3</td>
<td>low3</td>
<td>high3</td>
<td>high3</td>
<td>med3</td>
<td>accepted19</td>
</tr>
<tr>
<td>K nearest neighbors</td>
<td>high4</td>
<td>high3</td>
<td>med2</td>
<td>med2</td>
<td>med2</td>
<td>high1</td>
<td>discharged14</td>
</tr>
<tr>
<td>MARS</td>
<td>med3</td>
<td>low1</td>
<td>med2</td>
<td>low1</td>
<td>low1</td>
<td>med3</td>
<td>discharged11</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>med3</td>
<td>med2</td>
<td>med2</td>
<td>low1</td>
<td>med2</td>
<td>med3</td>
<td>discharged13</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>low1</td>
<td>low1</td>
<td>high1</td>
<td>low1</td>
<td>med2</td>
<td>med3</td>
<td>discharged9</td>
</tr>
<tr>
<td>Extra Trees</td>
<td>high4</td>
<td>med2</td>
<td>med2</td>
<td>med2</td>
<td>high3</td>
<td>med3</td>
<td>accepted16</td>
</tr>
<tr>
<td>Random Forest</td>
<td>high4</td>
<td>low1</td>
<td>med2</td>
<td>med2</td>
<td>high3</td>
<td>med_high2</td>
<td>discharged15</td>
</tr>
<tr>
<td>Gradient Boosting Trees</td>
<td>high4</td>
<td>med2</td>
<td>med2</td>
<td>high5</td>
<td>high3</td>
<td>low4</td>
<td>accepted20</td>
</tr>
<tr>
<td>LassoLars + Gradient Boosting Trees</td>
<td>high4</td>
<td>high3</td>
<td>med2</td>
<td>veryhigh4</td>
<td>veryhigh4</td>
<td>verylow5</td>
<td>accepted22</td>
</tr>
</tbody>
</table>
Proposed solution/ Quick prototyping: the final hybrid model

**HybridModel** = **LassoLars** + **GradientBoostingTrees**

- **Integrate** statistics and artificial intelligence methods
- LL: is good when relationships are **linear** and there are many **redundant** features \((N_f >> N_e)\)
- LL: takes only the most relevant and linear features => training time is very low
- GBT: **uses** the results of LL model (features/linearity/errors) to reduce errors and training time
- GBT: capture **nonlinearities** and similar days (Trees)
- The model is **understandable** (no black-box models)
- Realtime optimization parameters using CV
- LL: for each iteration it updates the regularization coefficient (being fast)
- GBT: only in the initialization phase it estimates all the parameters
- GBT: for each iteration update only the most dynamic parameter: the number of trees (being fast)
Proposed solution/ Model flow

1. Start
2. Data Acquisition → Data Preprocessing → Data Clustering
3. Model Features Selection → Model Input Generation
4. EnsembleLearner-LN → Ensemble → Learner_{1} → Iperparameter Optimization (Offline-CrossValidation)
5. EnsembleLearner-NL → Ensemble → Learner_{N}
6. Supervisor (Online-Training)
7. RMSE<Th → Yes → Model Deploy
8. RMSE<Th → No → Iperparameter Optimization (Offline-CrossValidation)
9. Learner_{1} → Ensemble → Learner_{N}
10. Forecast
Experimental results/ Load demand prediction

- $t_0=2015-07-06\ 06:00:00$  horizon=$24h$  Tcamp=$15\text{min}$

### Exogenous variable series
Experimental results/ Load demand prediction

- \( t_0 = 2015-07-06 \ 06:00:00 \)  horizon=24h  \( T_{\text{camp}} = 15\text{min} \)

\[
\text{RMSE}_{\text{prev}} = 516 \text{ MW}
\]
Experimental results/ Load demand prediction

- $t_0=2015-07-06\ 08:00:00$  horizon=$24h$  $T_{camp}=15\text{min}$

 RMSE$\text{prev} = 336\text{ MW}$
Experimental results/ Load demand prediction: errors

![Graph showing Root Mean Square Error over time](time.png)
Proposed solution/ Software Engineering: distributed computing

Distribution of multiple processes over different Server and CPU

Performance: 500 forecasts within 5min

Predictor N.10 works on server N.1 and CPU N.3
predictor(svr=1,cpu=3,pred=10)
Proposed solution/ Software Engineering: memory management

- No Data
- New Data 1
- New Data 2

1. **MC**
   - Supervisor(k)
     - g(k,1)
     - g(k,2)

2. **CPU**
   - CPU1: 3%
   - CPU2: 3%

3. **MC**
   - Supervisor(k)
     - g(k,1)
     - g(k,2)
     - p(k,1,1)
     - p(k,2,1)

4. **MC**
   - Supervisor(k)
     - g(k,1)
     - g(k,2)
     - p(k,1,2)
     - p(k,2,2)

5. **HD**
   - p(k,1,1) p(k,2,1)
   - p(k,1,2) p(k,2,2)

   - New Data 1
   - New Data 2

   - HD
     - X X
     - X X

   - HD
     - p(k,1,1) p(k,2,1)
     - p(k,1,2) p(k,2,2)

   - HD
     - X X
Proposed solution/ Software Engineering: memory management

No Data → New Data 1 → New Data 2

1. Supervisor(k) → g(k,1) → CPU1: 3%
   → g(k,2) → CPU2: 3%

2. Supervisor(k) → g(k,1) → p(k,1,1) → CPU1: 100%
   → g(k,2) → p(k,2,1) → CPU2: 100%

3. Supervisor(k) → g(k,1) → p(k,1,2) → CPU1: 100%
   → g(k,2) → p(k,2,2) → CPU2: 100%

Memory usage:
- 62.9 GB RAM installed
- RAM used: 6894 MB last, 6837 MB avg, 7149 MB max
- SWAP used: 0 MB last, 0 MB avg, 0 MB max
- Page tables: 53 MB last, 53 MB avg, 55 MB max
- RAM+SWAP+PT used: 6947 MB last, 6890 MB avg, 7203 MB max
- Memory mapped: 59 MB last, 59 MB avg, 70 MB max
- Committed: 6639 MB last, 6642 MB avg, 6810 MB max
- Shared Memory: 23 MB last, 23 MB avg, 25 MB max
Proposed solution/ Software Engineering: CPU Utilization
Proposed solution/ **Software Engineering: archivers configuration**

- **Daemon**
  - invokes **Manager Archives**
- **Manager Archives**
  - invokes **Archiver**
- **Archiver**
  - Data to be acquired
  - Data extraction, processing and backup logic
  - Saved data

**Database**
- **DB ARCHIVER**
- **DB ARCHIVER_STATUS**
- **File di configurazione**

---

© 2014 Engineering Group

WWW.ENG.IT
If $N_{cpu}=32$ and $Speed=2.0GHz$ then $t_{exec/cpu}=18sec$

**Requirement:** $N_{pred/cpu}*t_{exec/cpu} < 5min \implies N_{pred/cpu} < 60*5/18 = 16$

If $N_{pred/cpu}=15$ then $N_{pred} = N_{pred/cpu}*N_{cpu} = 15*32 = 480$
Proposed solution/ Software Engineering: technology

Core implemented in **Python**

LFOR uses scientific libraries and DataMining (numpy/scipy, pylab scikit-learn, xgboost, ffnet) tools developed in **Python/C++**

Obviously the rough codes are modified and adapted on the basis of our needs!!!!!!!

<table>
<thead>
<tr>
<th></th>
<th>C++</th>
<th>Python</th>
<th>R/Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program rate</td>
<td>+++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Rapid prototipation</td>
<td>---</td>
<td>+++</td>
<td>++</td>
</tr>
<tr>
<td>Integrated scientific env.</td>
<td>+</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Support to advanced programmation</td>
<td>+++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>Parallelization code libraries</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>API interfaces to OS</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>General purposes</td>
<td>+++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>GUI</td>
<td>+++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>Generators</td>
<td>++</td>
<td>+++</td>
<td>-</td>
</tr>
<tr>
<td>Interacts</td>
<td>-</td>
<td>+++</td>
<td>+++</td>
</tr>
<tr>
<td>Wrapping C++</td>
<td>+++</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>
THANK YOU

Dr. Eng. Felice Tuosto
felice.tuosto@eng.it