Project EE4C12 EPE ES

Benoît Jeanson
Employee 930283
b.jeanson@tudelft.nl

October 27, 2023

1 Summary

This project consists in designing machine learning models to assess the risk factor and corresponding risk state of a power grid (see Figure 1). The way the risk factor is determined is not known. Therefore, we use a statistical approach to evaluate it in contexts that are within the distribution of the training data. To this end, we follow the pipeline as depicted (see Figure 2).

This project report is structured according to the project description. The first part will describe the *task 1* and consists in preparing the training: the data set is analyzed in order to identify what would be the relevant features to be used for the training process. The second part will describe the approach for the assessment of the *Risk Factor* – as a *regression* problem. The third part will describe the approach for the assessment of the *Risk State* – as a *classification* problem. A conclusion of the project implementation will then be proposed.

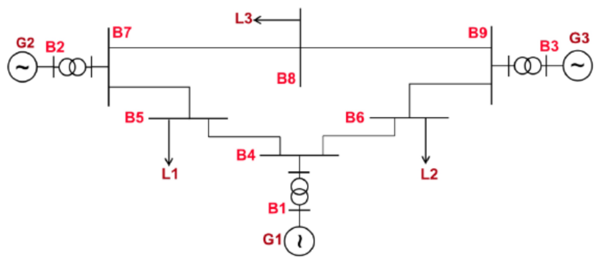


Figure 1: 9-buses power grid

All the simulations are performed using the python *scikit-learn* framework, and run on a *M1-chip MacBook Air*.

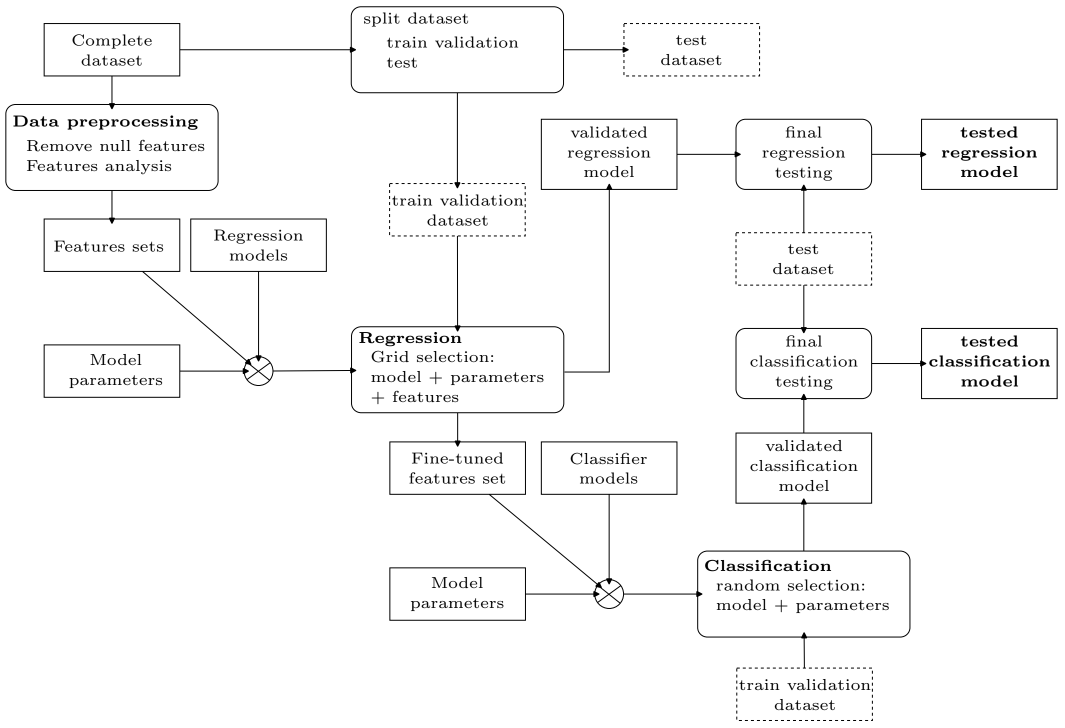


Figure 2: Workflow of the approach

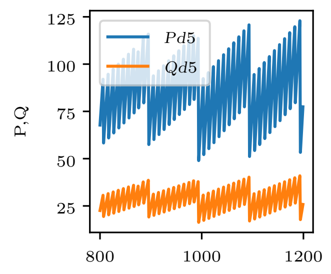
2 Task 1: preprocessing activities

Before entering into any machine learning process, the input data has to be cleaned up and prepared. First as pointed out in the project statement, all combinations of 2 buses are provided in the initial data set – including those combinations for which no line exists. In that case $P_{ij} = Q_{ij} = 0$. We first remove all the rows that contains only zeros. This reduces the number of rows from 218 in the initial data set to 66. Then the output data *Risk Factor* is extracted to properly isolate the input data (65 remaining rows) and the output one.

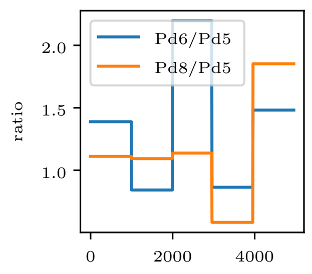
The next step consists in analyzing each remaining feature to identify if they are relevant for training. Some data can be misleading, some others are proportional to one another, which could impede the training because of the collinearity of features.

First, we proceed to an analysis of the *demand*. By plotting Pd_i and Qd_i for i in $\{5, 6, 8\}$ (e.g. Figure 3a for demand-5) we can identify:

- For each i, Qd_i looks proportional to Pd_i, which suggests a fixed $tan\varphi = Qd_i/Pd_i$. The amplitudes of $tan\varphi$ values throughout all the samples being lower than 10^{-7} confirms that observation. Therefore, it is not necessary to keep both P and Q as they provide the same information.
- The demands are correlated per time slots of 1000 samples – which becomes obvious with figure 3b. This observation is actually not useful for the training since the generation time series patterns are not correlated to the demand ones.



(a) P and Q for demand-5 – which suggests proportionality.



(b) Ratio of P for demand 6 and 8 vs 5 – demands are proportional by piece of 1000 samples.

Figure 3: Demand related curves

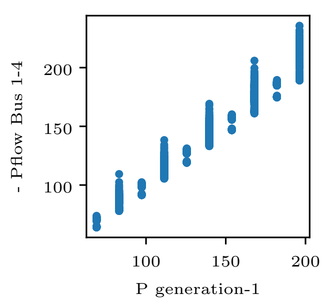
Regarding the *generation* plan: the data set provides the active power P for the 3 units: $\{1, 2, 3\}$, and Q values only for units 2 and 3.

- Generation-1 is the *slack* node. Indeed, the voltage is perfectly sustained at $1.3p.u.$. Moreover, when comparing the Pg_1 with $Pflow_{1-4}$, one can see (figure 4a) that the output flow (on the line) differs from Pg_1 – which shows that Pg_1 is the generation command value and not the generation output.
- The voltage can also be considered steady for generation 2 and 3 with amplitudes lower than $10^{-5}p.u.$
- As for the demand, the $tan\varphi$ look steady for generation 2 and 3, which would contradict the previous point as it is impossible in an evolving context to sustain both voltage and

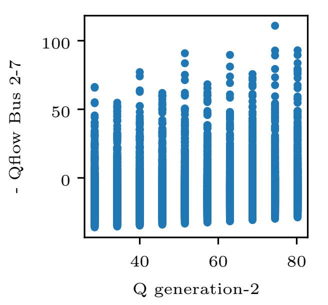
reactive power at the same time. Figure 4b shows that actually, the Q value given is actually not followed: this is certainly the time series of the command that would be followed if the generator had been in P, Q mode – which is not the case. The Q values for the generators shall not be used. They would actually not mislead the models – as the model does not rely the physics – this would only be redundant information with P time series.

- The P flows of Generation 2 and 3 follow their respective commands (e.g. Figure 4c).

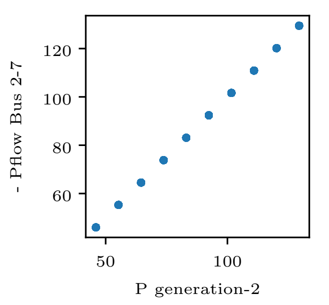
Thus, we can conclude the P, Q data provided for the generators, as well as the voltages on their connected buses, are useless as redundant with the data provided for their output flows are physically more relevant and better help in understanding the *'Risk Factor'*.



(a) P Generation-1: output vs command value: the generator does not follow the command, it is actually the slack node.



(b) Q Generation-2: output vs command value: the generator is in P, V mode and does not follow the command.

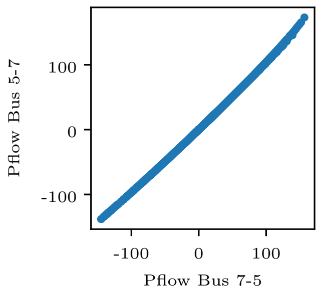


(c) P Generation-2: output vs command value: the flow follows the command.

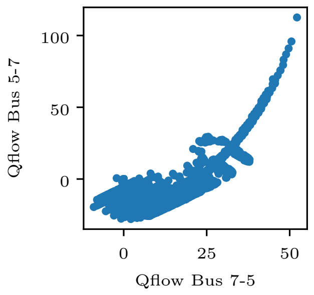
Figure 4: Generation related curves

Regarding the *voltage*, one can point out that all the information on the state of a power system is embedded in the *voltage angles* and *magnitudes* at each bus. Therefore, in theory, the *Risk Factor* could be deduced using only the voltage data. However, in practice, as the equations connecting the various physical values to the voltage are nonlinear, leveraging other intermediate physical values (i.e. between the voltage and the Risk Factor) in the design of the models improves the results. Plotting the various curves of the voltage brings these observations: the voltage magnitudes on the generation buses that are steady – and will not be used – and the voltage profiles are evolving presenting regularities similar to the risk factor profile.

Regarding the *branches*, P and Q values are given on both sides of each. The active power (Figure 5a) on both sides are similar (few losses), while the reactive power behavior (Figure 5b) between both sides is much more volatile – except for the branches that are directly connecting generation buses, which are probably short and are of low impedance.



(a) P Branch 5-7 vs 7-5



(b) Q Branch 5-7 vs 7-5

Figure 5: Branches related curves

Based on that investigation, the features that are candidates for the training are: the voltage magnitudes of all buses except those of the generators, the voltage angles of all nodes except for *bus-1* which is the slack node, the active flows of the branches connected to buses hosting a generator, all the reactive flows on branches, the active power of the demand, and – probably useless, to be tested – one active power per branch. The efficiency of the learning could be further investigated trough various combinations of these features.

3 Task 2: Estimation of the risk factor R

To identify the risk factor, simulations were made with various set of features – according the recommendations of Task 1. 6 combinations (Table 1) of features were tested with 3 kinds of model: a regression tree, a linear regression, and a neural network – MLP – (Table 2). The first two were only tested with the default parameters whereas latest was tested with various parameters.

Set	Features
A	Voltage angles and magnitudes
B	(P, Q) flows on generators
C	Voltage angles and magnitudes, (P, Q) flows on generators
D	Voltage angles and magnitudes, P flows on generators, all Q flows
E	Voltage angles and magnitudes, P flows on generators, all Q flows, P demands
F	Voltage angles and magnitudes, P flows on generators, all Q flows, P demands, P flows direct

Table 1: Features combinations set

Note that, *voltage magnitudes* only concern buses that do not have generation, and that *voltage angles* are for all the buses except *bus-1*, (P, Q) *flows on generators* are taken from the branch that is connecting the corresponding generator, and P *flows direct* are taken on one side of each branch.

Model	Paramètres	
TreeRegression	default	
LinearRegression	default	
MLP	hidden_layer_sizes	(100, 100), (100, 100, 100), (100, 100, 100, 100)
	alpha	0.01, 0.1
	max_iter	300, 500, 1000

Table 2: Tested regression models and their parameters

A systematic grid search is performed among the combinations of features and all the models and parameters. This was done 3 times which highlighted the influence of the random seed for the *MLP* model. The table 3 shows the results for the 5 best combinations of features and model parameters. In this top-5, the results are very close. Moreover, regarding the fact that they are ranked in different orders from one training to another, we will consider them equivalent. The only model remaining is *MLP* with $Alpha = 0.1$, the number of max iterations has no visible influence, and the architecture with 3 or 4 layers have similar results. Regarding the features combinations, D, E and F are equally represented, therefore, D contains all the information needed to asses the *Risk Factor* .

With the features set D, the results for the *TreeRegression* are $MSE = 7.79 \times 10^{-3}$ and $R2 = 0.920$, and for the *LinearRegression*, $MSE = 15.8 \times 10^{-3}$ and $R2 = 0.839$ – both with training times around 100 ms. These results show that, even if the MLP is outperforming them by almost one order of magnitudes on the MSE, these models are already very efficient, and have the merit of the simplicity. Regarding the MLP parameters, once again *max_iter* has no influence on the performances of the training, whereas *alpha* has a significant impact: all the training with a value of 0.01 have the worst performances ($MSE \simeq 4 \times 10^{-3}$ and $R2 \simeq 0.95$). The 2 layers MLP have good performances, but remain below the MLP with 3 and 4 layers.

Training	Group	Layers	Alpha	max_iter	MSE ($\times 10^{-3}$)	R2	Training Time
1	D	(100, 100, 100, 100)	0.1	300	2.32	0.976	10s
	F	(100, 100, 100)	0.1	300	2.37	0.975	6s
	D	(100, 100, 100, 100)	0.1	500	2.42	0.975	9s
	E	(100, 100, 100, 100)	0.1	300	2.44	0.975	7s
	D	(100, 100, 100)	0.1	500	2.47	0.974	8s
2	E	(100, 100, 100)	0.1	1000	2.35	0.975	5s
	D	(100, 100, 100, 100)	0.1	1000	2.37	0.975	5s
	F	(100, 100, 100, 100)	0.1	500	2.41	0.975	8s
	F	(100, 100, 100, 100)	0.1	300	2.42	0.975	7s
	E	(100, 100, 100)	0.1	500	2.42	0.975	4s
3	E	(100, 100, 100, 100)	0.1	1000	2.34	0.976	4s
	F	(100, 100, 100, 100)	0.1	300	2.35	0.975	13s
	D	(100, 100, 100, 100)	0.1	500	2.36	0.975	8s
	E	(100, 100, 100)	0.1	300	2.37	0.975	4s
	E	(100, 100, 100, 100)	0.1	500	2.38	0.975	6s

Table 3: The 5 best combinations of model parameters and features for 3 trainings

The final test is then performed with the features set D, a *4-layer MLP*, $max\_iter = 1000$, and $Alpha = 0.1$. On the test set, the results are $MSE = 3.13 \times 10^{-3}$ and $R2 = 0.967$.

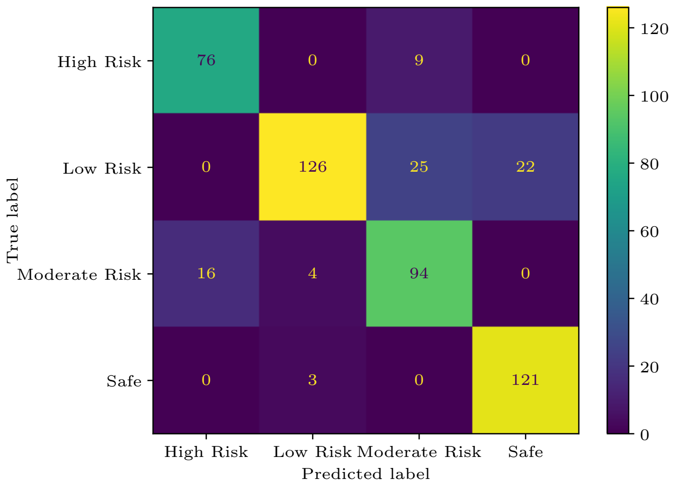


Figure 6: Confusion matrix with the regressor trained in task 2

4 Task 3: Estimation of the risk state: *Class*

This part consist in estimating the *Risk State*, which is function of the *Risk Factor* according to the following thresholds:

$$safe < 0.1 \leq LowRisk < 0.35 \leq ModerateRisk < 0.7 \leq HighRisk$$

First, as a regressor was trained in the previous task, it can provide an easy first classifier, by assessing the risk factor of the situation and classify with the appropriate thresholds. The results are: accuracy: 0.84, precision: 0.83, recall: 0.85, F1: 0.84 and the confusion matrix: Figure 6

Then, for this part, the intention was to test the function `RandomizedSearchCV` with *SVC classifier*. Two sets of parameters were tested to assess the best set of parameters between *SVC classifier* non-polynomial kernels (i.e. *linear* and *rbf*) that do not have the parameter *degree* and the polynomial one.

`RandomizedSearchCV` was so stunning and easy to implement, it was worth testing it also with the *MPLClassifier* which turned out to outperform the performances of the *SVC* classifiers.

The results are in the Table 4

Classifier	Param	Best	Accuracy	Precision	Recall	F1
Non-Poly SVC	C: `loguniform(1e-5, 1e2)` kernel: 'linear', 'rbf'	0.764 'rbf	0.854	0.852	0.855	0.853
Poly SVC	C: `loguniform(1e-5, 1e2)` degree: `np.arange(1, 5)`	18.636 2	0.818	0.821	0.819	0.819
MLP Classifier	hls: (100, 100, 100) (100, 100, 100, 100) alpha: `loguniform(1e-4, 1)` activation: 'logistic', 'tanh', 'relu'	(100, 100, 100) 0.003 'logistic'	0.943	0.934	0.941	0.937

Table 4: parameters explored with `RandomsizedSearchCV` and results

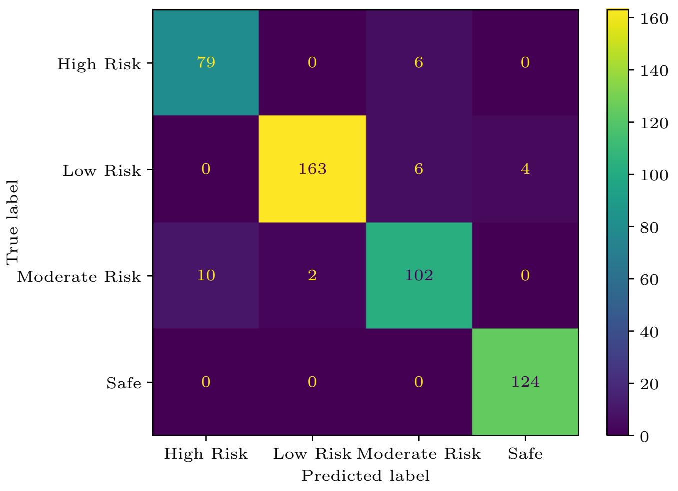


Figure 7: Confusion matrix with the *MLPClassifier* trained in task 3

Out of curiosity, considering the excellent performances of the parameters identified for the *MPLClassifier* by `RandomsizedSearchCV`, the parameters were tested to see if these parameters were also relevant for the regressor of task 2. The results are: $MSE = 4.70 \times 10^{-3}$ and $R2 = 0.951$, which does not outperform the previous models. Moreover, if the corresponding regressor is itself used for the classification, its performances (accuracy = 0.82) are poor compare to what was reached.

The final model is therefore the *MPLClassifier* with the parameters identified by the random search. The confusion matrix is presented in Figure 7

5 Conclusion

This study on a small case turned out to be pretty complex and allows me exploring numerous aspects of the machine learning. Many options were not yet implemented that could have been very interesting. Regarding the preprocessing (task 1), the exploration could also include for example *PCA* analysis to analyze the features through a more statistical approach. I preferred relying on my own experience of the domain – which can be misleading: sometimes naive approaches turn out to be very fruitful. The work on the regressor (task 2) completed the analysis of the features by confirming a narrowed set of relevant features. The development of the grid search was done manually – which gave a lot of control on the process and on the information retrieved. In task 3, I kept the features fine-tuned in task 2 and I could appreciate the power of `RandomsizedSearchCV` – which is very efficient, especially allowing parallelization which significantly sped up the training. Mixing up the models and parameters between task 2 and 3 – reuse of *MLPRegressor* of task 2 for a classifier, use of the parameters identified in task 3 for a new regressor – confirmed that principle that a model shall be trained as much as possible in strict accordance with its final usage.