Personal report: Power models based on Machine Learning for optimizing HPC systems

Author: Thang NGUYEN DUC,
Group member: Thang NGUYEN, Luis Palluel, Landry Amiard, Pratvi KALAVADIYA
{duc-thang.nguyen, luis.palluel, landry.amiard, pratvi.kalavadiya}@univ-tlse3.fr

Guided by: Prof. Georges Da COSTA, Prof. Jean-Marc Pierson

Toulouse, June 25, 2021

Abstract

With the increasing demand for HPC systems, the optimization problem for HPC systems will greatly reduce costs and improve platform performance effectively. The dynamic voltage and frequency scaling (DVFS) technique is a popular method used to control the performance of the HPC by frequency change. The project aims to build a model that sets the right frequency to optimize power consumption. Our experiments' best performance is Multi target learning in Deep neural network with Gaussian noise achieved 0.80 accuracy, 0.78 F1 score, and mean square error: 1.00 on test set.

1 Introduction and motivation

Nowadays, computational demands like Artificial intelligence, Big data, etc are increasing in High-performance computation systems. Therefore, the HPC systems face growing energy costs for computation, even for large cooling systems. The HPC system runs continuously 24/7, all the nodes are not always running at full capacity at all times. Therefore if there is an effective way to manage energy in these cases, there will be significantly reduced energy costs. For example: We can do somethings to save energy on non-task nodes. We can also determine which frequency is suitable for solving the task because it would be a waste when providing high performance for a simple math operation. The dynamic voltage and frequency scaling (DVFS) technique is commonly used in HPC systems to reduce power consumption by reducing processors' voltage and frequency when not used and speeding up when needed. However, it is not easy to come up with an effective strategy, and the system needs to have an agent that aggregates a lot of environmental parameters to provide the right decision.

Give some examples of excess energy caused by bottlenecks [1]. Typically as:

- If the bottleneck is CPU, which means the CPU cannot process the incoming tasks in time, it needs to increase computation speed by increasing the CPU frequency.
- Conversely, if the bottleneck is RAM, we need to reduce the frequency down to avoid wasting energy.
- The speed of sending and receiving packets over the network or disk is also an important factor in adjusting the frequency.
- etc

The project aims to develop a learning-based model to provide an effective strategy with DVFS for HPC systems' energy management using machine learning methods. The project will be experimental on the Grid'5000 system, which is a large-scale and flexible testbed for experiment-driven research in all areas of computer science with a large amount of resources: 15000 cores, 800 compute-nodes [1].

In this project, we have experimented on two main approaches: Tree-based machine learning techniques and deep neural networks.

1.1 Personal planned activity

we followed a workflow project. It is based on a standard process for machine learning projects, as shown in the figure 2. Specifically, the project includes six main phases:

1. Define the problem
2. Data collection
3. Exploratory data analysis
4. Data processing & data engineering & Data selection
5. Training model
6. Evaluation and testing

Table 1 shows the personal timeline for each task following the procedure shown in figure 1 and 2.

[1] https://www.grid5000.fr/w/Hardware

Table 1: Project timeline.

No.	Task name	Round 1		Round 2		Round 3		Round 4		Round 5	
		From	To	From	To	From	To	From	To	From	To
1	Data collection	10/10/20	20/10/20	20/10/20	20/11/20	20/11/20	30/12/20	-	-	-	-
2	EDA	21/10/20	25/10/20	7/11/20	7/11/20	14/11/20	14/11/20	-	-	-	-
3	Feature selection and engineering	26/10/20	28/10/20	7/11/20	8/11/20	14/11/20	14/11/20	-	-	-	-
4	Build & Train model	29/10/20	5/11/20	8/11/20	12/11/20	14/11/20	30/12/20	15/3/21	15/4/21	25/5/21	20/6/21
5	Evaluation	5/11/20	6/11/20	13/11/20	13/11/20	30/12/20	30/12/20	15/4/21	15/4/21	10/6/21	20/6/21

Round 1: Tree-based algorithm: XGBoost on single metric and single target
Round 2: Tree-based algorithm: LightGBM on single metric and single target
Round 3: Tree-based algorithm: LightGBM & XGBoost on all metrics and single target
Round 4: Deep neural networks on all metrics and single target
Round 5: Deep neural networks on all metrics and multi-targets

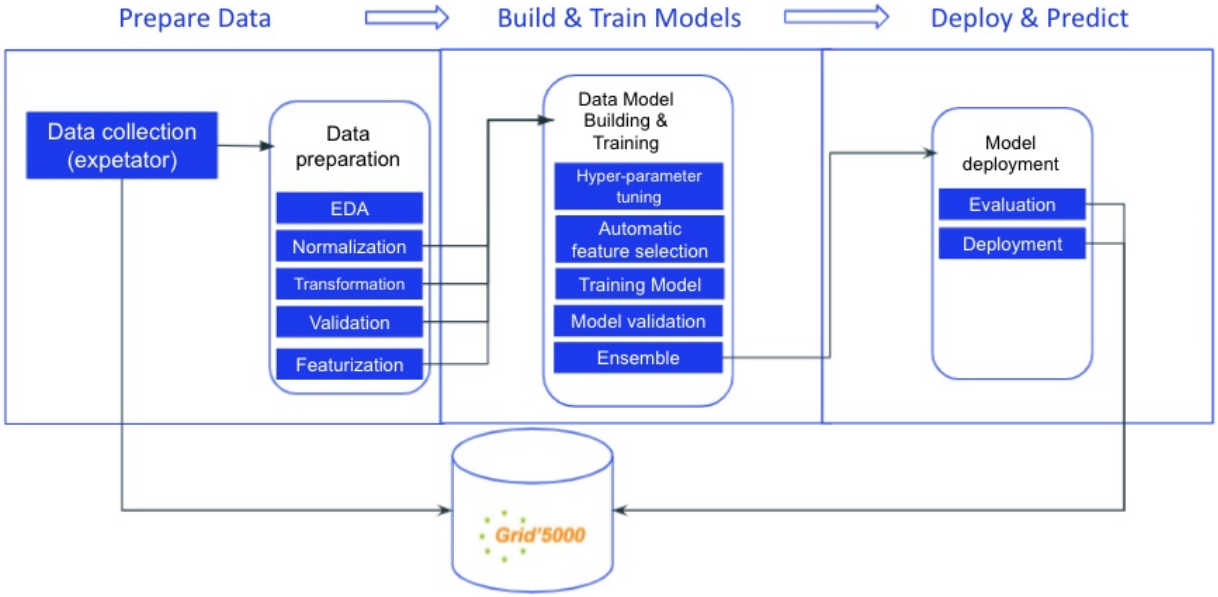


Figure 1: The diagram of our project pipeline.

Table 2: Dataset summary.

No.	Site	cluster name	Number of experiment	number of row
1	Nancy	grimoire	323	4199
2	Nancy	grisou	1060	13779
3	Rennes	paravance	150	44131
4	Lyon	taurus	99	1164
	Total		1632	63273

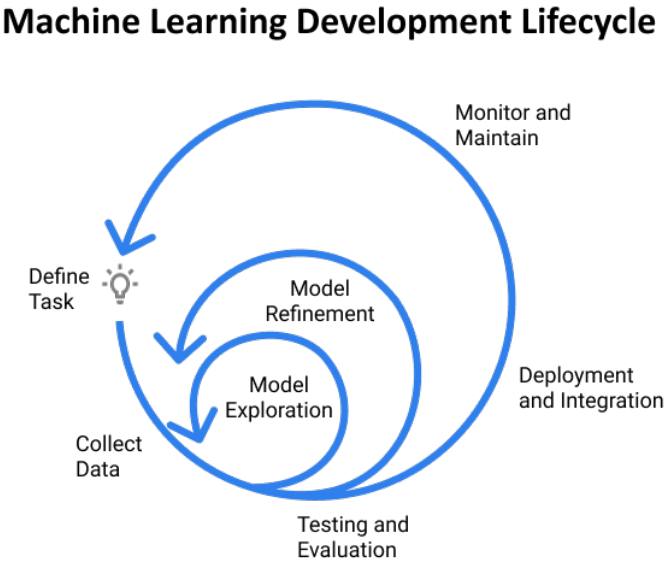


Figure 2: A machine learning development lifecycle

2 Dataset

2.1 Data collection

To build a good model, we need a quality dataset. Therefore, this is an important and time-consuming job of the project. I performed data collection using expetator tool [2] on Grid500 system with 1,632 experiments equivalent to 21,091 rows.

Table 2 presents basic statistics for the number of collected data per each cluster on Grid500. Actually, I have gathered many experiments, but they can not be used without power information.

2.2 Exploratory data analysis

Once we have our data in hand, the next step we need to do is Exploratory data analysis (EDA). This task is an important that allow us to understand the problem's data properties using data statistics, data visualization, ... and then build an effective strategy.

The collected data, I received is the result from experiments with different frequencies on a HPC's cluster and then record the environmental parameters included:

- Knowledges: contains information of each experiment such as: cluster information, duration, number of processing units, power/energy/etp consumption.

[2] https://pypi.org/project/expetator/

- Vectors:
  - **network_names** = rxb, rxp, txb, txp
  - **infiniband_names** = irxb, irxp, itxb, itxp
  - **rapl_names** = dram0, dram1, package-00, package-11
  - **load_names** = user, nice, system, idle, iowait, irq, softirq, steal, guest, guest_nice
  - **perf_names** = cpu_cycles, instructions, cache_references, cache_misses, branc_instructions, branch_misses, bus_cycles, ref_cpu_cycles, cache_l1d, cache_ll, cache_dtlb, cache_itlb, cache_bpu, cache_node, cache_op_read, cache_op_prefetch, cache_result_access, cpu_clock, task_clock, page_faults, context_switches, cpu_migrations, page_faults_min, page_faults_maj, alignment_faults, emulation_faults, dummy, bpf_output

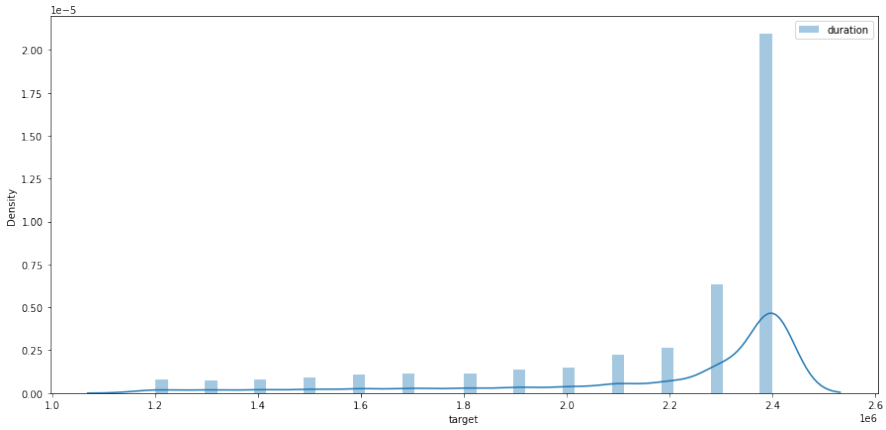


Figure 3: The distribution of target with duration metric

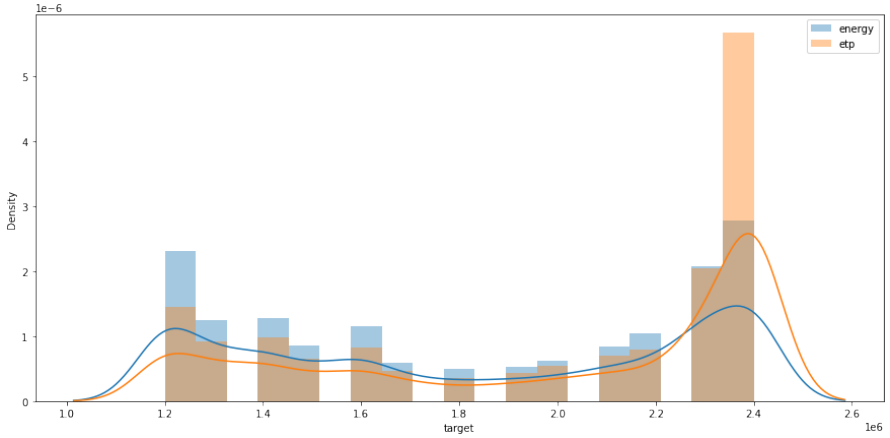


Figure 4: The distribution of target with energy and etp metrics

The project's aim is to build a model that can inferences an effective frequency based on the objective metric function are duration, energy, etp. You can see Figure 3, mainly experiments are focus more at high frequencies and understandable when the objective function is run time. However, with the objective function is energy and etp, figure 4 experiments tend to focus on both sides of the frequency set. We will build two strategies as follows:

- A multi-objective metrics model

- Each model for each metric

From the collected data based on experiments, we will get target and targetZ as the result of choosing the best frequency by different strategies through expetator tool:

- target: select the best frequency which has the best value by a selected metric in each collected experiment.
- targetZ: select the closed frequency which has the value less than $ratio * bestvalue$ by a selected metric in each collected experiment.

Therefore, the single-target or multi-targets model will also be our plan.

2.3 Feature preprocessing

Observation through a boxplot of numerical variables chart shows that there are outliers, which should be eliminated or replaced with the lower_lim(quantile = 5%) or upper_lim(quantile = 95%) of all the data.

Data will be standardized during training phase, making the learning process faster instead of one data with skewed data domains. Also, computationally large numbers will be more expensive than small numbers.

2.4 Feature selection

Figure 5, too highly correlated variables ($threshold \geq 0.9$) with one another should remove such as: cache_node, task_clock, cache_ll, page_faults_min, txp, txb,package-11, dram1, softirq. Because these can decrease the model's availability to learn, decrease model interpretability, and decrease generalization performance on the test set. We also use PCA, a statistical method to help dimensionality reduction.

2.5 Feature engineering

Feature engineering is one of the important tasks to increasing interpretability during the model's learning process.

- **Data binning** is a method of converting continuous data into discrete data. This technique is significant for the model, helping the model understand the category of that data.
- **Log transforms** is a technique commonly used in machine learning. When applied to skewed distributions, it makes sense since the log function tends to extend values in the low region and decrease values in the uplands.
- **Aggregating categorical** is a technique for creating new aggregate statistical features by a group. In this problem, I will group by experiment with statistical tools as min, max, sum, mean,...

3 Machine learning model

3.1 Tree-based methods

In this project, we chose tree-based methods to create a lightweight and efficient model. Specifically, we have experimented on 2 methods of gradient boosting algorithm: XGBoost and LightGBM. Gradient boosting is an ensemble technique by combining weak prediction models to build a stronger model, illustrated in Figure 7.

And I used the hyperparameter finder using GPEI-based Bayesian optimization to help find the best one for our model (e.g Optuna, hyperopt,etc).

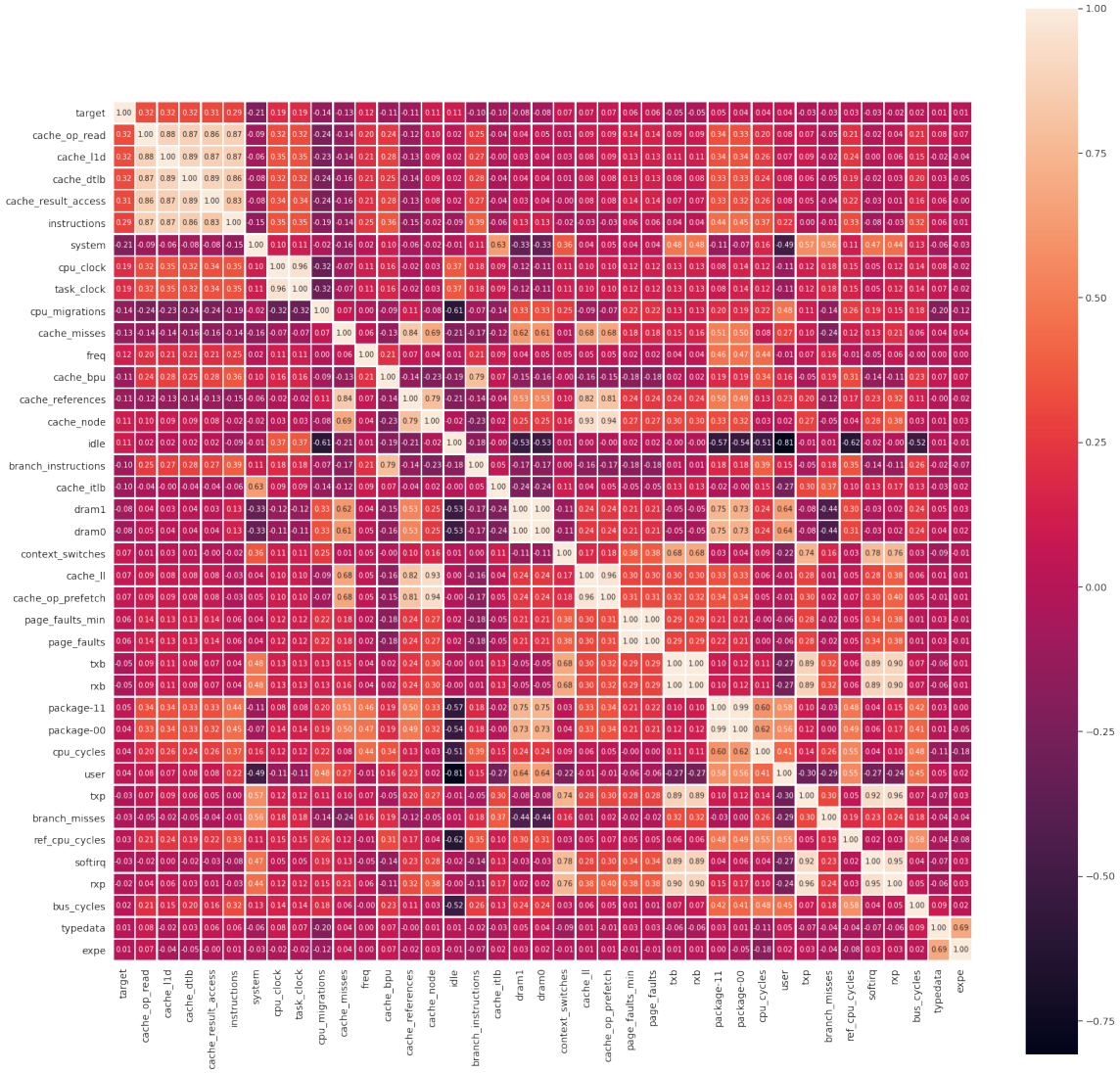


Figure 5: The correlated variables

3.2 Deep neural networks

When we mention the state-of-the-arts machine learning methods, the neural networks will not be absent. From starting project, We also have a plan about it on our experiment. From the analysis from the section data understanding, I establish the following two experiments:

- Single-target learning in Deep neural network is a common Deep neural network for only one task.
- Multi-targets learning in Deep neural network is a sub-group of machine learning in which multiple learning tasks are solved at the same time. In this problem, we have two tasks: target and targetZ.

My model architecture (Listing 1 and Figure 6) includes 6 layers and on each layer, I use BatchNorm and dropout techniques to improve performance of learning process and also avoid over/under-fitting problem.

```
_________________________________________________________________
      Layer (type)                 Output Shape              Param #
```

```
================================================================
       BatchNorm1d-1                  [-1, 190]             380
            Linear-2                 [-1, 1024]         195,584
       BatchNorm1d-3                 [-1, 1024]           2,048
           Dropout-4                 [-1, 1024]               0
            Linear-5                  [-1, 512]         524,800
       BatchNorm1d-6                  [-1, 512]           1,024
           Dropout-7                  [-1, 512]               0
            Linear-8                  [-1, 256]         131,328
       BatchNorm1d-9                  [-1, 256]             512
          Dropout-10                  [-1, 256]               0
           Linear-11                  [-1, 128]          32,896
      BatchNorm1d-12                  [-1, 128]             256
          Dropout-13                  [-1, 128]               0
           Linear-14                   [-1, 64]           8,256
      BatchNorm1d-15                   [-1, 64]             128
          Dropout-16                   [-1, 64]               0
           Linear-17                   [-1, 13]             845
           Linear-18                   [-1, 14]             910
================================================================
Total params: 898,967
Trainable params: 898,967
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 3.43
Estimated Total Size (MB): 3.48
----------------------------------------------------------------
```

Listing 1: The model architecture for Multi-targets learning in Deep neural network

3.3 Training methods

1. **Train test validation split**: To ensure the best performance and accurate comparison of the model after training. I have divided the data set according to each experiment into a training set and test set with an 8: 2 ratio. This division ensure that the two sets' distribution is similar and the data must be independent in both sets, meaning that no experimental data-row exist on both sets. The test set is used to evaluate the model independently after training. The training data will be divided into folds and guarantee the above condition. Each fold represents a validation set to help choose the model's best weight after learning the rest of the set, illustrated in Figure 8.

2. **k-Fold Cross-Validation**: Model selection strategy is also an interesting technique. We can once more assemble the trained model on each fold to create the best model to predict the test set (Figure 9, as known an ensemble voting method. This process is like a consultation from doctors with different knowledge and experience.

3. **Gaussian Nosie**: With a non-huge dataset, the model can memorize all training examples, in turn leading to overfitting and poor performance on a testset. Gaussian Nosie will be used in our experiments to help improve the robustness of the model, resulting in better generalization by adding noise randomly to data during training phase.

4. **Multi-target/loss function combination**: From the problem's target, we determine that the prediction's result is acceptable if the predicted value is close to the actual value. Therefore, I use both the metrics functions in figure 10b multi_logloss and mean squared error (mse) to help train the best model. With neural network loss function, we base on F1 score ($2 * \frac{precision * recall}{precision + recall}$) (figure 10c) to make a target loss of model during training to help balance between the precision and the recall.

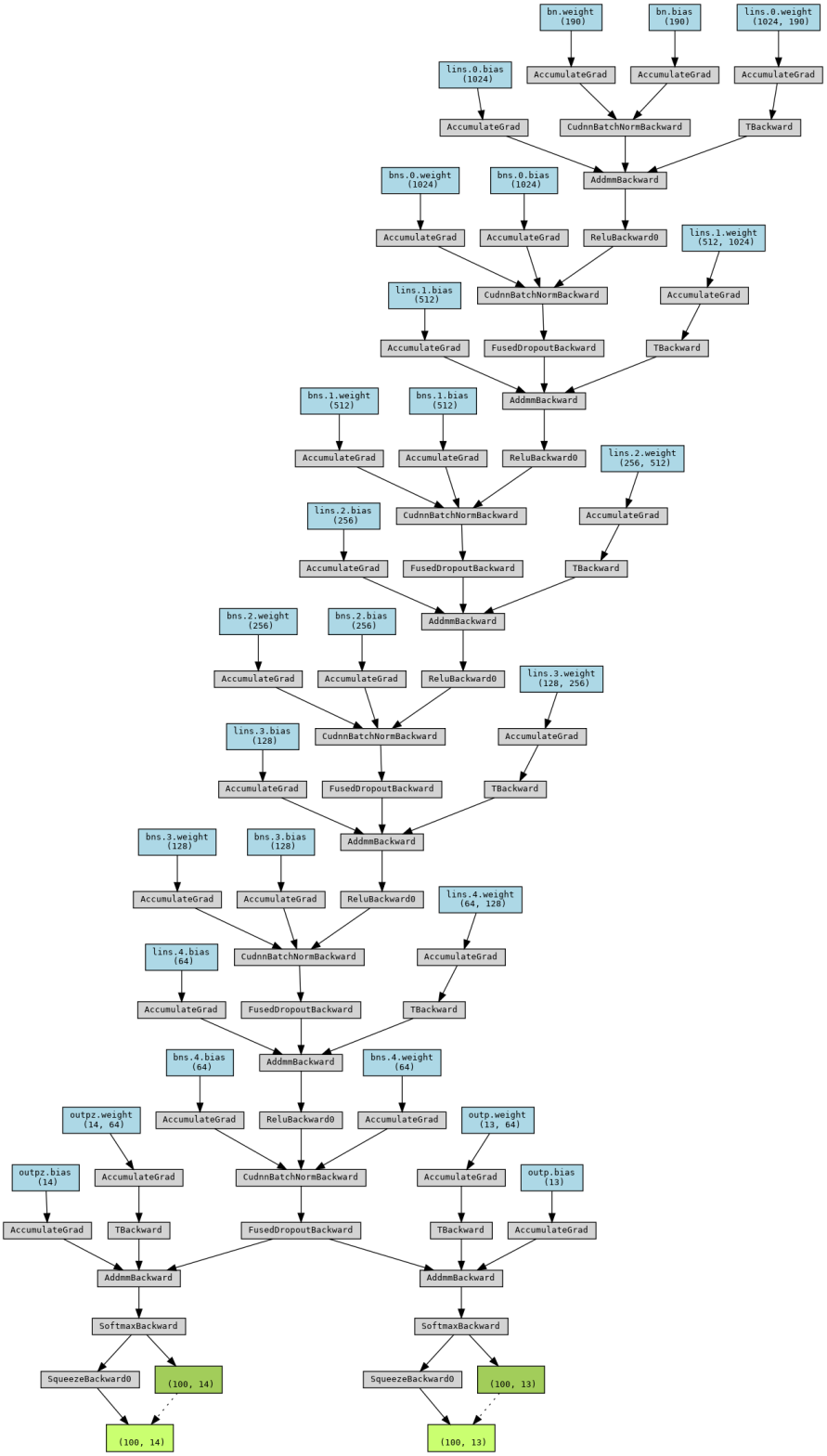


Figure 6: The Deep neural network architecture for Multi target learning.

5. **Metric validation**: During training phase, we need to track and keep the best weight measuring performance of current model on metrics. We use Receiver operating characteristic(AUC), F1 score, Accuracy, mse, ... during our experiments.

6. **Learning rate scheduler**: Gradual warmup scheduler is used in our neural network experiment, a way to reduce the primacy effect of the early training that help significantly performance, shown by [2].

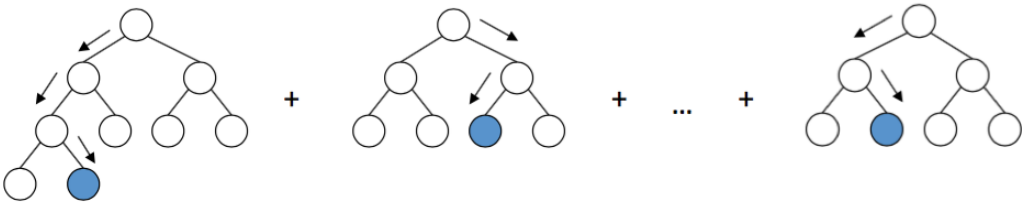


Figure 7: Gradient boosting algorithm

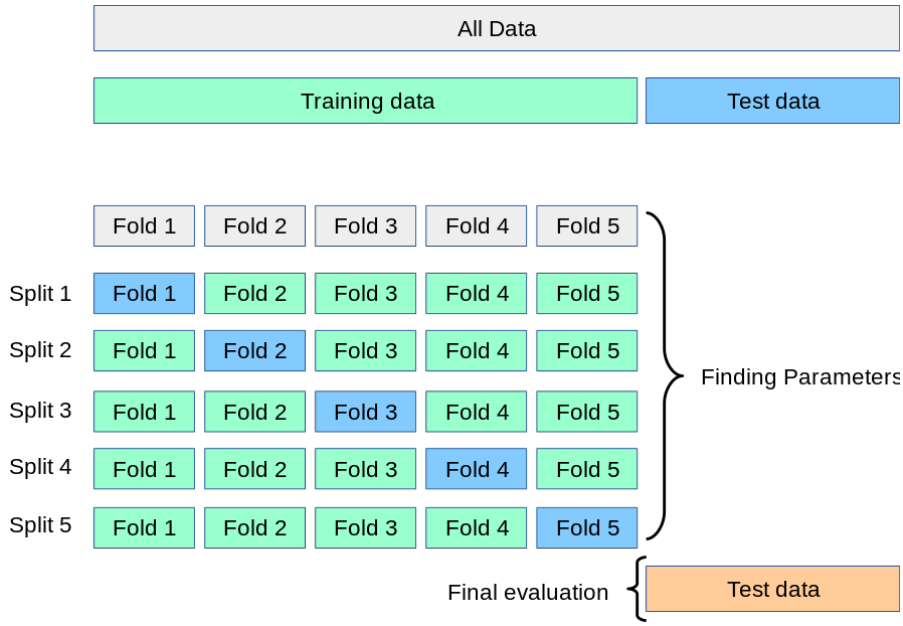


Figure 8: Cross-validation strategy

4 Results

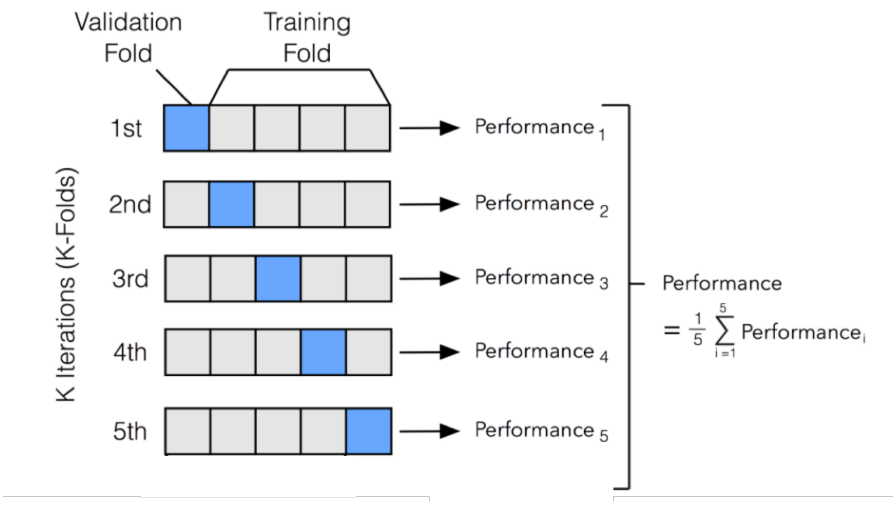


Figure 9: Model selection strategy

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(test set; predicted vaue; actual value)

(a) Mean squared error

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij} * \ log(p_{ij})$$

(b) Multi_logloss

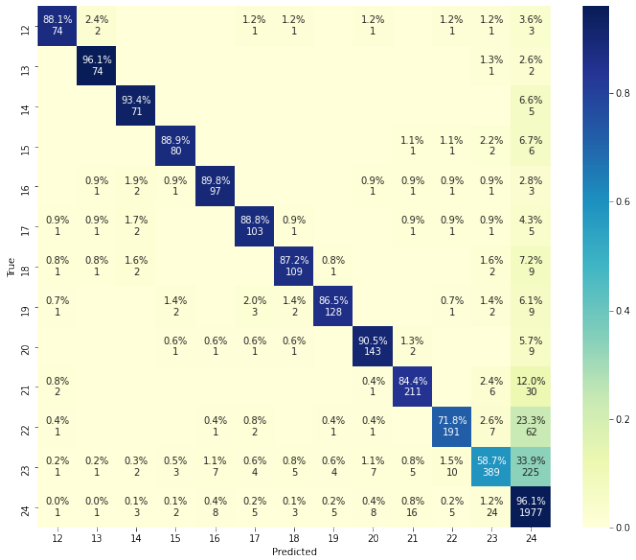
$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}}\right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

(c) F1 score loss

Figure 10: The metric functions

Table 3: All experiment results

No.	Methods	Objective	Valid	Test				
			Acc	F1	Acc	Ensemble F1	Ensemble Acc	Ensemble MSE
1	LightGBM	duration	0.86	0.85	0.86	0.86	0.87	2.19
2	LightGBM	energy/etp	0.72	0.70	0.71	0.72	0.73	3.54
3	Xgboost	all metrics	0.79	0.72	0.76	0.72	0.75	4.03
4	LightGBM	all metrics	0.81	0.73	0.77	0.74	0.78	3.40
5	DNN single-target	all metrics	0.81	0.75	0.77	0.76	0.77	2.15
6	DNN multi-targets	all metrics	0.82	0.77	0.78	0.77	0.78	1.24
7	**DNN multi-targets with Gaussian noise**	**all metrics**	**0.83**	**0.78**	**0.80**	**0.78**	**0.79**	**1.00**



(a) confusion matrix

==========> TTA on test acc: 86.44228490163546, mse: 2.033894287745911

	precision	recall	f1-score	support
0	0.90	0.88	0.89	84
1	0.91	0.96	0.94	77
2	0.87	0.93	0.90	76
3	0.90	0.89	0.89	90
4	0.85	0.90	0.87	108
5	0.87	0.89	0.88	116
6	0.89	0.87	0.88	125
7	0.92	0.86	0.89	148
8	0.88	0.91	0.89	158
9	0.89	0.84	0.87	250
10	0.91	0.72	0.80	266
11	0.89	0.59	0.71	663
12	0.84	0.96	0.90	2058
accuracy			0.86	4219
macro avg	0.89	0.86	0.87	4219
weighted avg	0.87	0.86	0.86	4219

(b) classification report

Figure 11: The model's performance for duration metric

Table 3 shows the results of each of my experiments. Specifically, Firstly experiment, I implement single objective model for each metric. The duration metric results showed the best performance with 0.86 accuracy (Figure 11, while the two metrics, energy, and etp, had lower results (Figure 12. This is easy to understand when looking at the distribution of the metrics in Figure 3 and 4.

The second training scenario is a multi-objective model for all metrics; we've shown that the performance is significantly increased, reaching 0.78 (Figure 13 compared to the strategy of creating three independent models. Figure 13 also show that the model produces a prediction accuracy at important frequency cases. Cases of the wrong classification are often concentrated close to the ground truth label.

Third experiment, our approach is deep neural networks. Where I implement two type of learning: single target and multi-target learning (target and targetZ). Results show that multi-target learning deep neural network give the state-of-the-art performance in our experiments, achieved (0.78 Accuracy and 0.77 F1 score) and 0.80 Accuracy, 0.78 F1 score, and 1.00 mse when using Gaussian noise technique, shown more in figure 14 and 15. With targetZ definition, We can see it's hard to predict a exactly frequency when |best frequency set|> 1, figure 14cd shown F1 score only 0.4, but the low mse (1.04) allow us to see model can have good prediction.

5 Conclusion

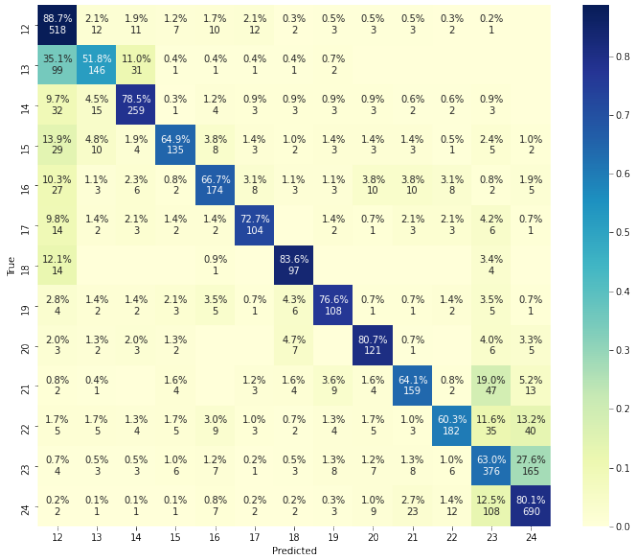
Our approaches are completely appropriate and got a good results with $0.8ACC$,0.78 F1 score and especially the wrong prediction cases are quite close to the ground-truth using Multi target learning in Deep neural network with Gaussian noise.

However, we have not yet completed the roadmap as we cannot deploy the results on real HPC for many reasons in the context of covid-19.

My Contribution:

- Collect data [3].
- EDA and data processing [4].

[3] https://github.com/georges-da-costa/defi/tree/master/src/csvs
[4] https://github.com/georges-da-costa/defi/tree/master/src/EDA%20and%20visualization



(a) confusion matrix

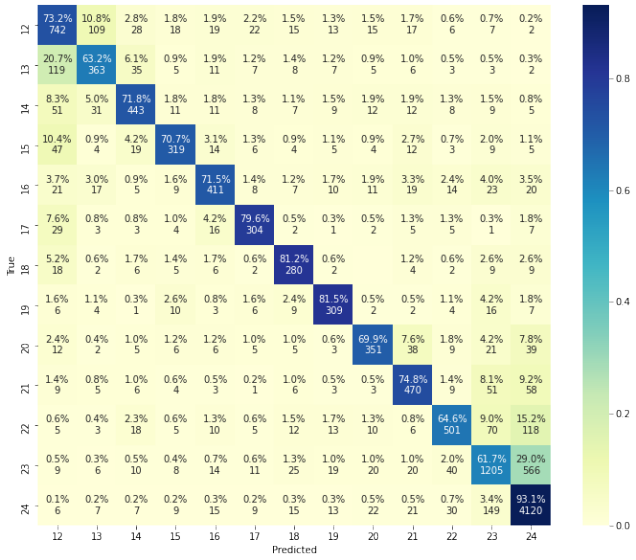
```
=====================> TTA on test acc: 72.6734548898887, mse: 3.5479
              precision    recall  f1-score   support

           0       0.69      0.89      0.77       584
           1       0.72      0.52      0.60       282
           2       0.79      0.78      0.79       330
           3       0.80      0.65      0.72       208
           4       0.76      0.67      0.71       261
           5       0.74      0.73      0.73       143
           6       0.73      0.84      0.78       116
           7       0.73      0.77      0.75       141
           8       0.72      0.81      0.76       150
           9       0.74      0.64      0.69       248
          10       0.83      0.60      0.70       302
          11       0.63      0.63      0.63       597
          12       0.75      0.80      0.77       861

    accuracy                           0.73      4223
   macro avg       0.74      0.72      0.72      4223
weighted avg       0.73      0.73      0.72      4223
```

(b) classification report

Figure 12: The model's performance for energy metric



(a) confusion matrix

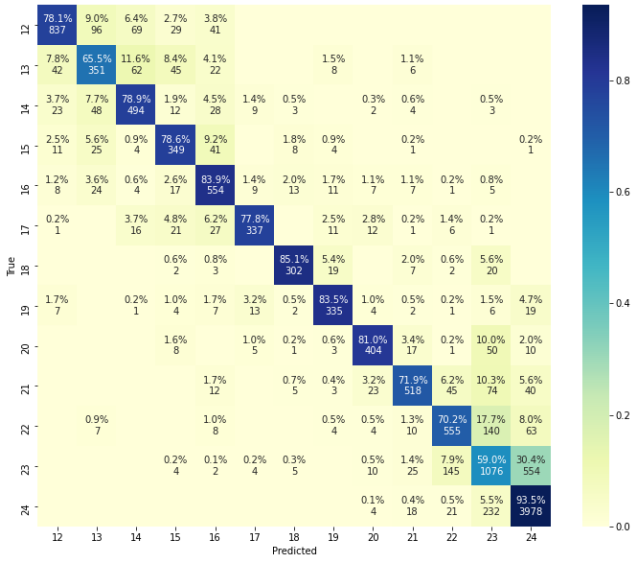
```
=====================> TTA on test acc: 77.80947852274528, mse: 3.4085
              precision    recall  f1-score   support

           0       0.69      0.73      0.71      1013
           1       0.65      0.63      0.64       574
           2       0.76      0.72      0.74       617
           3       0.77      0.71      0.74       451
           4       0.76      0.71      0.74       575
           5       0.77      0.80      0.78       382
           6       0.71      0.81      0.76       345
           7       0.76      0.82      0.79       379
           8       0.77      0.70      0.73       502
           9       0.74      0.75      0.75       628
          10       0.79      0.65      0.71       776
          11       0.77      0.62      0.68      1953
          12       0.83      0.93      0.88      4423

    accuracy                           0.78     12618
   macro avg       0.75      0.74      0.74     12618
weighted avg       0.78      0.78      0.77     12618
```

(b) classification report

Figure 13: The multi-objective model's performance for all metrics(duration, energy, etp)



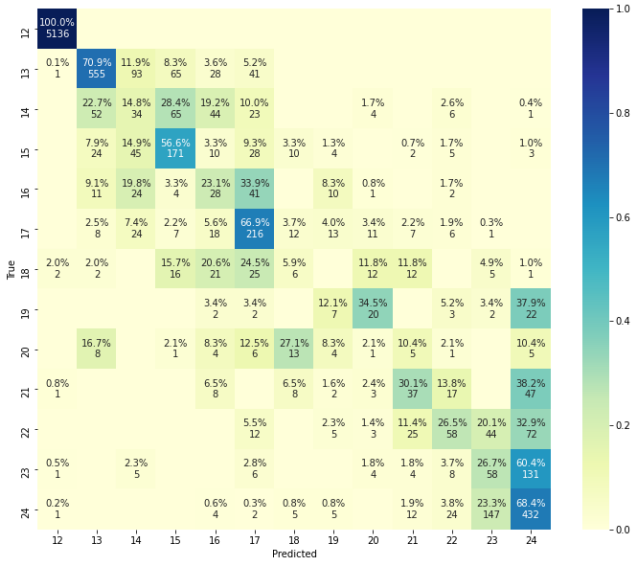
(a) confusion matrix on target

```
loss_valid: 0.37526, roc_auc_target: 0.908744, roc_auc_targetZ:
0.834736, total_roc_auc: 0.871740,    F1_target: 0.782243,F1_targetZ:
0.395831,accuracy_target: 0.799841, accuracy_targetZ: 0.799128,
MSE_target: 1.006341656757828,MSE_targetZ: 1.0442330558858501 .
              precision    recall  f1-score   support

           0       0.90      0.78      0.84      1072
           1       0.64      0.65      0.65       536
           2       0.76      0.79      0.77       626
           3       0.71      0.79      0.75       444
           4       0.74      0.84      0.79       660
           5       0.89      0.78      0.83       433
           6       0.89      0.85      0.87       355
           7       0.84      0.84      0.84       401
           8       0.86      0.81      0.83       499
           9       0.84      0.72      0.78       720
          10       0.71      0.70      0.71       791
          11       0.67      0.59      0.63      1825
          12       0.85      0.94      0.89      4253

    accuracy                           0.80     12615
   macro avg       0.79      0.77      0.78     12615
weighted avg       0.80      0.80      0.80     12615
```

(b) classification report on target



(c) confusion matrix on targetZ

	precision	recall	f1-score	support
0	1.00	1.00	1.00	5136
1	0.84	0.71	0.77	786
2	0.15	0.15	0.15	229
3	0.52	0.57	0.54	302
4	0.17	0.23	0.19	122
5	0.54	0.67	0.60	323
6	0.11	0.06	0.08	102
7	0.14	0.12	0.13	58
8	0.02	0.02	0.02	66
9	0.34	0.28	0.31	133
10	0.39	0.22	0.28	261
11	0.21	0.20	0.20	289
12	0.46	0.36	0.40	1191
13	0.83	0.92	0.87	3617
accuracy			0.80	12615
macro avg	0.41	0.39	0.40	12615
weighted avg	0.79	0.80	0.79	12615

(d) classification report on targetZ

Figure 14: The Multi-targets learning in Deep neural network's performance for all metrics

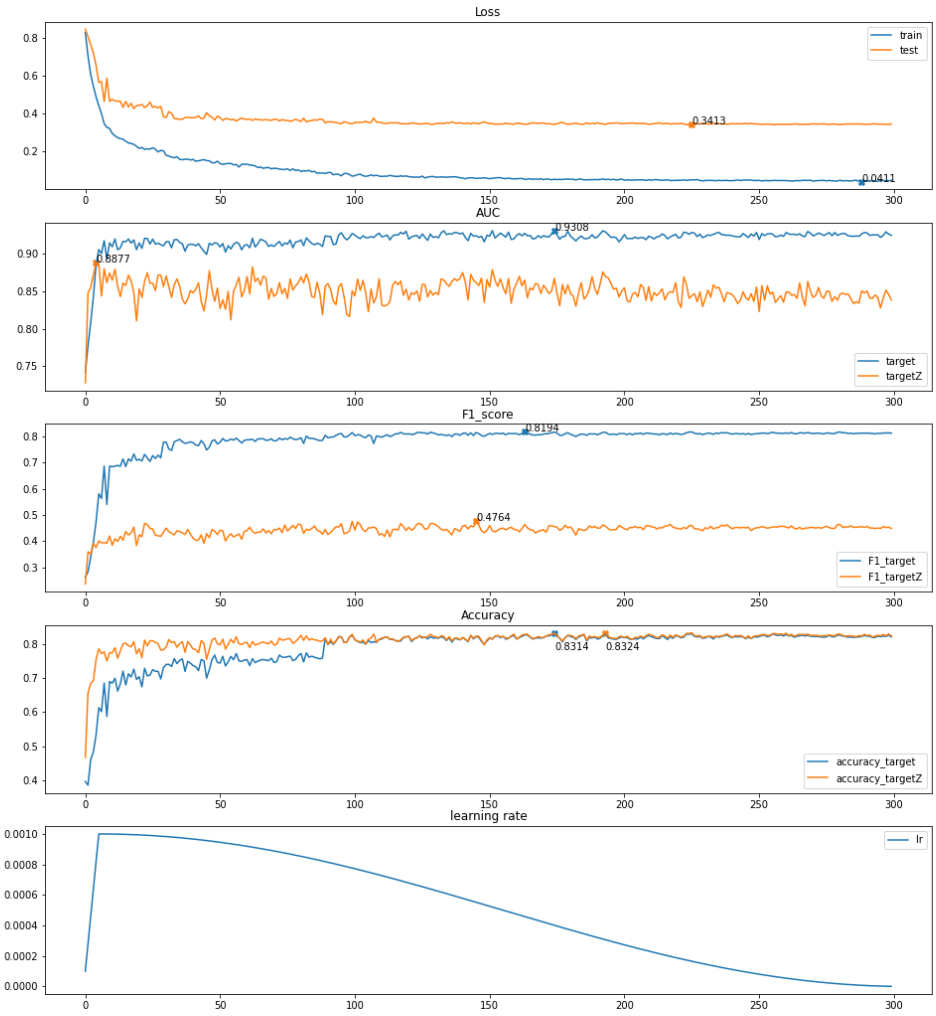


Figure 15: The Multi-targets learning in Deep neural network's log of performance during training

- Feature engineering [5].
- Build and train Tree-based methods: Xgboost & LightGBM with good performance[6].
- Build and train Deep neural networks to achieve height performance[7].

[5] https://github.com/georges-da-costa/defi/blob/master/src/Modeling%20and%20evaluation/NN-multi-targets/datapreparation.py
[6] https://github.com/georges-da-costa/defi/tree/master/src/Modeling%20and%20evaluation/Tree-based%20methods
[7] https://github.com/georges-da-costa/defi/tree/master/src/Modeling%20and%20evaluation/NN-multi-targets

References

[1] Georges Da Costa and Jean-Marc Pierson: “DVFS Governor for HPC: Higher, Faster, Greener”. In: *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. ISSN: 2377-5750. Mar. 2015, pp. 533–540. DOI: 10.1109/PDP.2015.73.

[2] Liyuan Liu et al.: “On the variance of the adaptive learning rate and beyond”. In: *arXiv preprint arXiv:1908.03265* (2019).

List of Figures

List of Tables