Machine Learning Project

Authors (Fabiana Chericoni, Ion Jornea, Alessandro Macchia)
Master Degree (Data Science & BI, Data Science & BI, Data Science & BI)
Emails: f.chericoni1@studenti.unipi.it - i.jornea@studenti.unipi.it - a.macchia3@studenti.unipi.it
ML course 654AA, Academic Year: 2022/2023
Date: 23/01/2023
Type of project: **B**

Abstract

The aim of this report is to present the performance of several models and compare them with each other. We performed the model selection through a grid search on various hyper-parameters which explored thousands of configurations by means of 5-fold cross validation. The best model for the CUP ended up being a 3-layer neural network with mini batch strategy.

1 Introduction

The ultimate goal of this project is to effectively tackle a multivariate regression task on the CUP dataset. This aim is pursued through the employment, tuning and comparison of various machine learning models, such as Random Forest, KNN, SVM and neural networks, with a particular focus on the latter. This is preceded by a classification task carried out on the benchmark MONK dataset, where a Naive Bayes model is also proposed. The emphasis is put on the comparison across different types of models.

2 Method

2.1 Code

The code was written in Python and organised through Jupyter Notebooks. Specifically, we opted for performing for each benchmark dataset a different notebook, where we applied a complete analysis through the usage of the models which have been cited above. It must be pointed out the fact that, due to the inherent nature of the project which required a lot of time to be correctly run, we took advantage of specific tools, namely Kaggle and Google Colab. Moreover, we used different Python libraries to support our project. In particular, for what concerns the management of the data and the assistance with the graphical representation, employed Pandas, NumPy, TensorFlow and Matplotlib. Furthermore, we exploited also other Python libraries, more focused on the development of the machine learning models, i.e., Keras, SciKeras and Scikit-Learn.

2.2 Preprocessing

We applied a *1-of-k* encoding on the MONK data set, so that all features ended up being in binary format. The CUP data set instead only had continuous attributes with mean close to 0 and standard deviation around 1. Hence, no preprocessing was needed.

2.3 Validation schema and preliminary trials

The MONK dataset came already split up into a design test and test set. The hyperparameters of the various models built on top of this data set were tuned through a grid search cross validation with 5 folds.

For the CUP data set instead we used 20% of the training set we were given as an internal test set. The remaining 80% represented our design set, on which we run again a grid search cross validation with 5 folds. For the neural network and SVM models some preliminary grid searches were run, and then we proceeded with a finer grid. This allowed us to test more parameters while containing computation times.

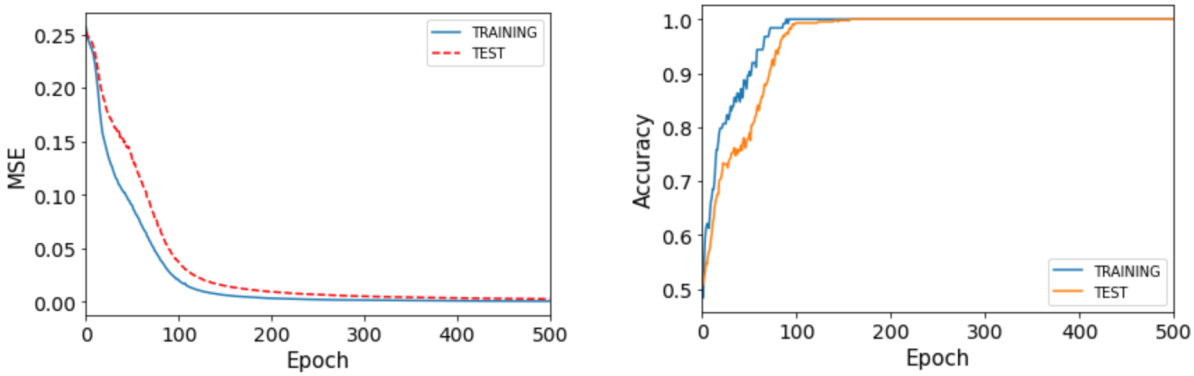
Preliminary hyperparameter tuning trials included the use of parameters of different orders of magnitude (e.g., 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, etc.).

3 Experiments

3.1 Monk Results

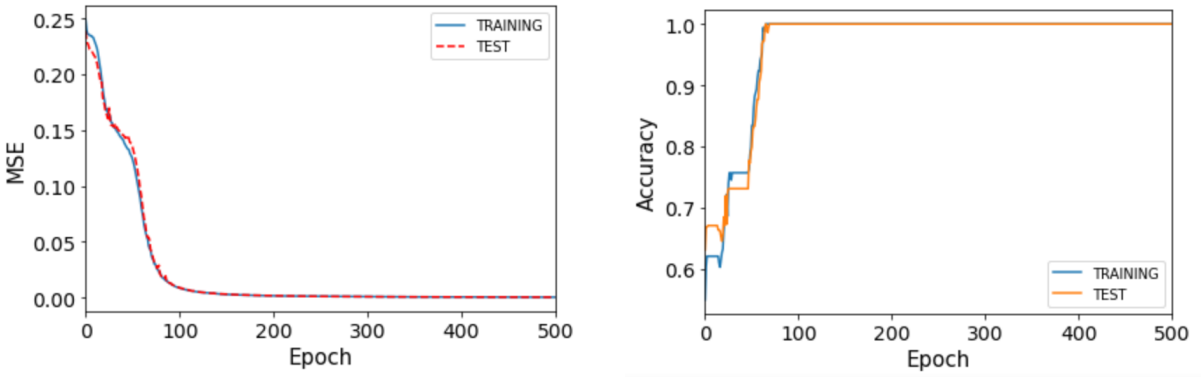
The performance of the various models of the MONK data set, coupled with the respective parameter settings, are shown in Table 1. The learning curves of the neural networks are depicted in Figures 1, 2, 3, and 4.

The neural network uses the ReLu activation function for hidden layers and the logistic activation function for the output layer. Moreover, we must point out the fact that the MLP (multilayer perceptron) has been developed through the usage of the sci-kit learn library, while the NN was devoloped thanks to the Keras library. Furthermore, we must state also that, for the MONK 3, in both the two models previously cited we were able to achieve better performances in test set via the utilization of the regularization parameter.



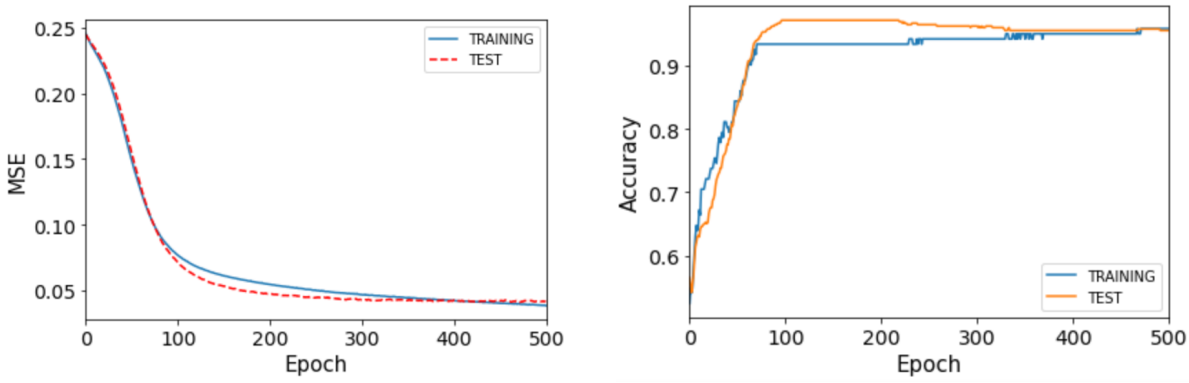
(a) MSE (b) Accuracy

Figure 1: Performance of the neural network on the MONK 1 data set



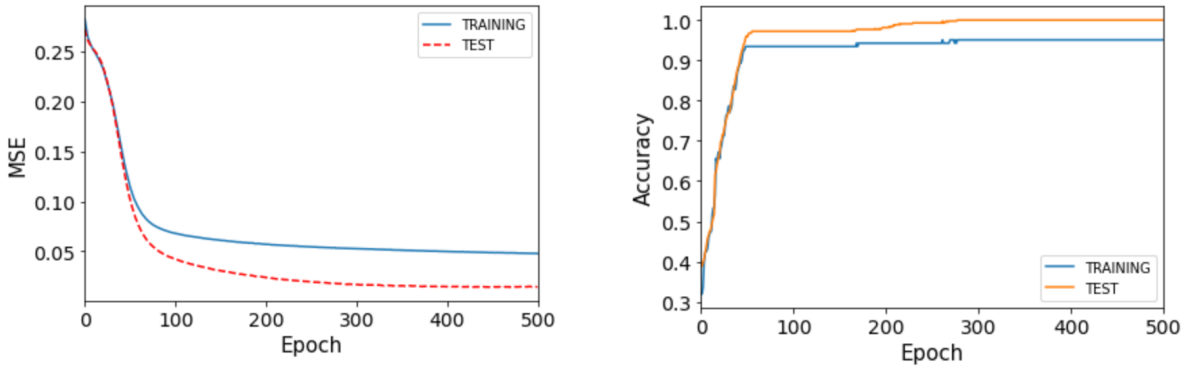
(a) MSE (b) Accuracy

Figure 2: Performance of the neural network on the MONK 2 data set



(a) MSE (b) Accuracy

Figure 3: Performance of the neural network on the MONK 3 data set



(a) MSE (b) Accuracy

Figure 4: Performance of the neural network on the MONK 3 data set with regularisation

Task	**Hyper-parameters**				**MSE**		**Accuracy**	
NN	*hidden_layer*	*eta*	*alpha*	*lambda*	TR	TS	TR	TS
MONK1	(4,)	0.2	0.7	0	0	0	100%	100%
MONK2	(2,)	0.7	0.01	0	0	0	100%	100%
MONK3	(4,)	0.1	0.01	0	0.04	0.04	96%	96%
MONK3+reg.	(2,)	0.1	0.5	0.001	0.05	0	95%	100%
MLP	*hidden_layer*	*eta*	*alpha*	*lambda*	TR	TS	TR	TS
MONK1	(3,)	1	0.4	0	0	0	100%	100%
MONK2	(2,)	0.4	0.5	0	0	0	100%	100%
MONK3	(2,)	0.4	0	0	0.01	0.05	100%	93%
MONK3+reg.	(3,)	0.9	0.01	0.001	0.01	0.06	99%	94%
NB	*alpha*				TR	TS	TR	TS
MONK1	253.2						76%	68%
MONK2	74.3						62%	67%
MONK3	0.1						93%	97%
KNN	*n.neighbors*	*p*	*weights*		TR	TS	TR	TS
MONK1	16	2	distance				100%	81%
MONK2	29	1	uniform				66%	67%
MONK3	41	1	distance				100%	95%
SVM	*kernel*	*degree*	*C*	*gamma*	TR	TS	TR	TS
MONK1	poly	2	3	scale			100%	100%
MONK2	poly	2	32	scale			100%	100%
MONK3	linear	1	1	auto			93%	97%
FOREST	*max_depth*	*criterion*	*min_leaf*	*n_estimators*	TR	TS	TR	TS
MONK1	14	gini	1	200			100%	96%
MONK2	15	gini	1	200			100%	75%
MONK3	8	gini	2	200			95%	97%

Table 1: Results obtained for the MONK tasks and parameter configuration

3.2 Cup Results

For what concerns the analysis that we have done in order to obtain the CUP results, we can state few key points. Firstly, we report that we split the cup file through a hold-out method, applying the following percentages: 80% for the training set and 20% for the test set. It should be remarked that we did not carry out any sort of data transformation. As it can be seen in Figure 3, all the models which we have been tested are listed below with their related parameters, which have been used for the grid search. In this regard, we emphasize the fact that for every grid search that has been performed we also computed a 5 fold cross validation so to get the best parameters possible. After retraining the model with the result gained from the hyper-parameter tuning, we executed each model assessment on the test set.

For what concerns the neural network we must mention that, due to the high complexity involved in searching for the best parameters, we opted for splitting it in different notebooks. Moreover, while trying to reduce as much as possible the time involved for the grid search, we further divided the cited notebook based on the batch strategy. Additionally we report that for the NN such as mb_50 and and mb_100 we applied the momentum, The momentum parameter was applied only for models using the batch strategy (not mini batch or online). For the training we opted for fitting the model for 500 epochs, using a 50 epoch patience parameter. This choice can be justified since we needed to speed up the computation of the grid search, which even with this speed-up needed a total of over 30 hours to complete. The latter has been mostly performed in remote, using platforms such as Kaggle, since it was impractical and inefficient running it in local on our personal devices. The virtual environment provided by Kaggle typically offered us around 12 GB of RAM memory, while the specifics of the CPU are undisclosed. Moreover, in NN models we used the Hyperbolic Tangent (Tanh) activation function for the hidden layers and linear activation function for the two output units. The loss was computed using the mean euclidean error (MEE).

3.3 Model Comparison

In Table 2 is displayed the performance of the instances (the ones with the best hyper-parameters according to our grid search cross validation) of the model classes proposed. The VL mean ($\pm$std) value is the result of the cross-validation and these values are fundamental for the model selection phase. The TR (training) and TS (test) values are computed after the retraining and the TS values are the ones used for the model assessment of the single best models. The model with the highest MEE is Random Forest, it cannot reach the best model's performance and it is very computationally slow. The slowest model to train for us was the neural network with online training strategy.

The best performance was achieved by the neural network with *minibatch* mode (with batch size = 100). The closest result was the one obtained by the KNN model, whose grid search however took much less time with respect to the neural network.

Model	MEE		
	VL mean (±std)	TR	TS
NN	1.4285 ± 0.0349	1.3071	1.4510
SVM	1.4639 ± 0.0185	1.2657	1.4776
RANDOM FOREST	1.4982 ± 0.0373	0.6826	1.5400
KNN	1.4390 ± 0.0276	0.0	1.4742

Table 2: Performance of the best model for each class on the CUP task

Grid-search						
NN						
Param	weights	architecture	eta	lambda	alpha	batch strategy
Range	[Random_Uniform, Glorot_Uniform]	$[(10,10),\ldots,(100,100,100)]$	$[0.00001,\ldots,0.9]$	$[0.001,\ldots,0.3]$	$[0,0.1,\ldots,0.9]$	[Batch, online mb_50, mb_100]
Best	RanUni	$(30,30,30)$	0.01	0.001		mb_100
KNN						
Param	n_neighbors	weights	p			
Range	$[1,\ldots,200]$	[unif,dist]	$[1,2]$			
Best	22	distance	2			
RFOREST						
Param	n_estimators	max_depth	min_samples_leaf	max_features		
Range	$[100,200,500]$	[None, $[1,\ldots,20]$]	$[1,\ldots,20]$	$[1,\ldots,18]$		
Best	500	20	2	3		
SVM						
Param	C	gamma	kernel	epsilon		
Range	$[1,\ldots,100]$	[scale,auto]	[poly,linear,rbf]	$[1,\ldots,100]$		
Best	4	auto	rbf	0.2		

Table 3: Tested and best parameters on the CUP task

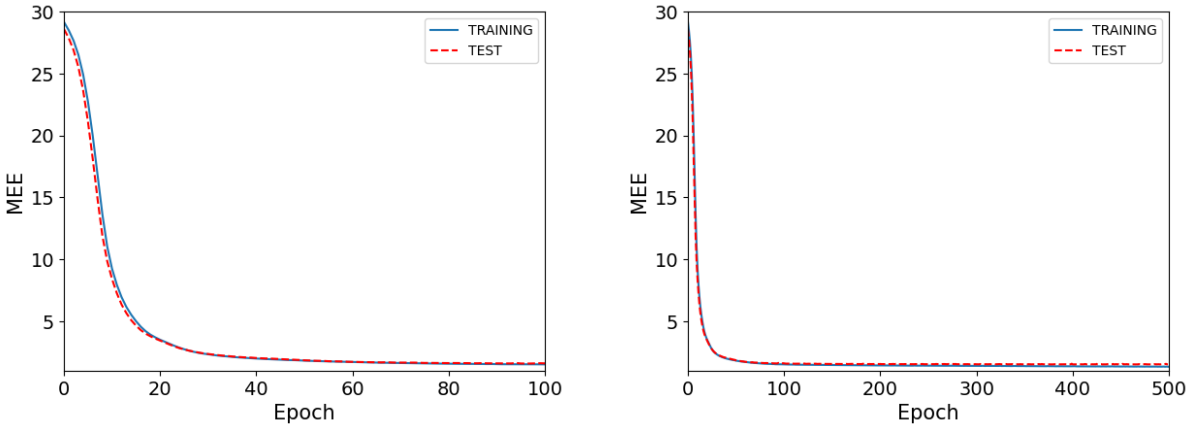
3.4 Best Model

For what concerns the choice of the best model, we opted to select the neural network with mini batch strategy, where tha batch size=100 (mb_100), whose parameters are visible in Table 4 (these include a three-layer architecture with 30 units each, learning rate of 0.01 and regularisation factor of 0.001). This decision has been taken due to the better performance on our internal test set, whose score can be seen in Table 2. Furthermore it can also be seen that, the neural network, is the model with the closest performance gap between the training and test score, possibly implying a reduced overfitting effect compared to the other models. The learning curve of the best model is shown in Figure 5: a close-up focus on the first 100 epoch is offered, to highlight the initial decline in the value of the loss function.

Given our results, we expect the MEE on the blind test predictions we have made with our model to be in the neighbourhood of 1.45.

Best Model:	**NN**				
Param	weights	architecture	eta	lambda	batch strategy
	Random_Uniform	$(30, 30, 30)$	0.01	0.001	mb_100

Table 4: Best Model parameters



(a) Detail of first 100 epochs

(b) Whole fitting

Figure 5: Learning curve of the neural network on the CUP data set

4 Conclusion

In conclusion we can state that even if we performed an extensive grid search for all the models which we computed, we realized that the random forest could present a situation of overfitting. The latter can be stated through the analysis of the MEE on the training set (the lowest of all the models) and the one on the test set (the highest of all the models). Moreover we agreed that even if we expected the neural network to be the best model to use for the CUP, we have to admit that also the SVM model has proven us to be valid and also extremely efficient, being much faster to train. This show us how there is no a perfect model that suits in every possible case, but rather a good possible model for every different scenario.

The nickname for our team is *MadCluster*.

BLIND TEST RESULTS: MadCluster_ML-CUP22-TS.csv

Acknowledgments

We agree to the disclosure and publication of our names, and of the results with preliminary and final ranking.