BIG HUMAN

Machine Learning Project

2023-2024

Big Human
Date: 15/01/2024
Type of project: B

Authors:
Francesco Caprari 580154 — Master Degree CS, Big Data, f.caprari@studenti.unipi.it
Francesco Botrugno 545713 — Master Degree CS, Big Data, f.botrugno1@studenti.unipi.it
Agnese Camici 559788 — Master Degree Digital Humanities a.camici1@studenti.unipi.it

Introduction & objectives

GOAL

Compare the performance of different machine learning **models**, for classification and regression tasks, focusing the attention on **Neural Networks**

DATASETS

Regression: **`ML-CUP23`**

Binary classification: **`MONK`**

Methods: models

Binary classification: **MONK**

- Neural Networks (Keras)
- Support Vector Machine
- K-Nearest Neighbors

Regression: **ML-CUP23**

- Neural Network (Keras, Pytorch)
- Support Vector Regression
- K-Nearest Neighbors
- Random Forest
- Bagging Regressor
- ElasticNet Regression

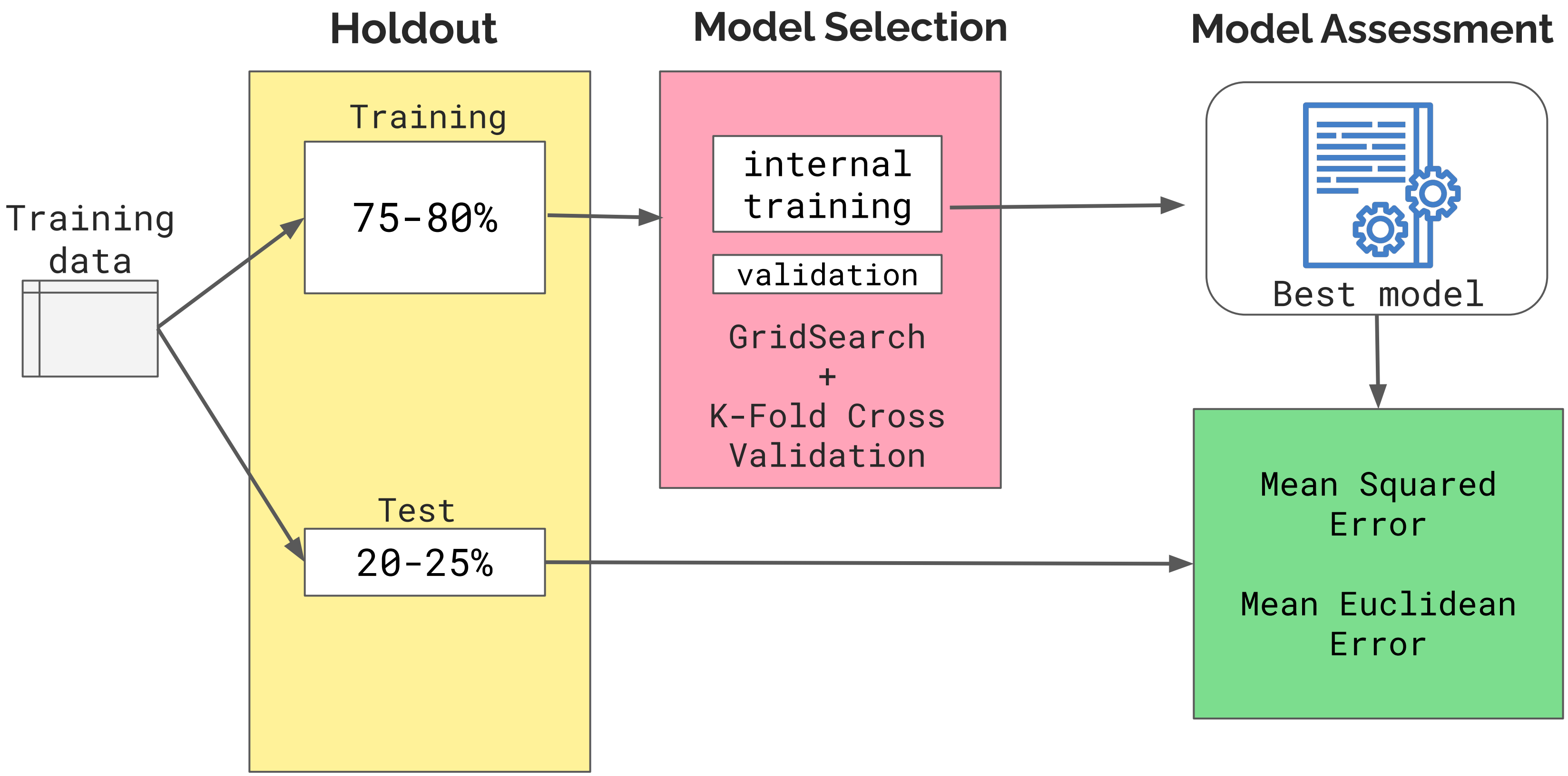
Methods: libraries

The project has been developed on the *Google Colaboratory* platform using `Python 3.12`

Different libraries have been used:

- *Pytorch* & *Keras* for MLP
- *Scikit-learn* for the KNN, SVM, Ensemble Methods and ElasticNet, and to compute the Grid Search
- *Matplotlib* for plot the result of the MSE and MEE in correlation with the epochs

Methods: workflow



Methods: CUP Neural Network choices

- Both in Keras and Pytorch we chose to implement **Stochastic Gradient Descent** with **mini-batch** and with weight decay applying **Ridge Regularization** as optimization algorithm

- For initialization in Keras, we used **Glorot initialization** as the default initialization, while in Pytorch the **weight are randomly** sampled from a normal distribution with **mean zero**.

- As a stopping condition, we set a **fixed number of epochs**.

MONKS Experiments: NN Keras

Number of hidden layers: 1
Activation function: sigmoid
Learning algorithm: stochastic gradient descent + mini batch
Batch size: 16 for MONK1, 25 for MONK2 and MONK3
Fixed number of epochs

Multilayer Perceptrons (MLP) implemented With Keras

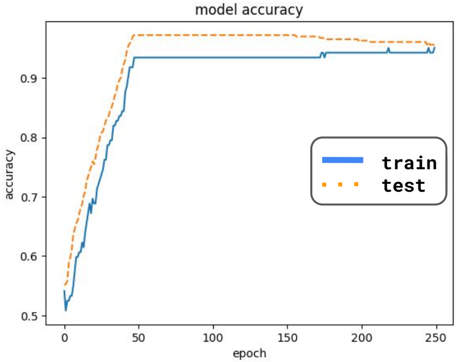
Task	Eta	Momentum	Epochs	MSE (TR/TS)	Accuracy (TR/TS)%
MONK1	0.15	0.75	150	0.0019/0.0019	100%/100%
MONK2	0.25	0.77	100	0.0020/0.0026	100%/100%
MONK3	0.1	0	250	0.0478/0.0436	94%/97%

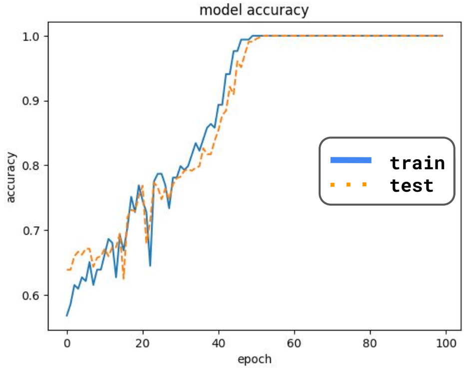
MONK 1

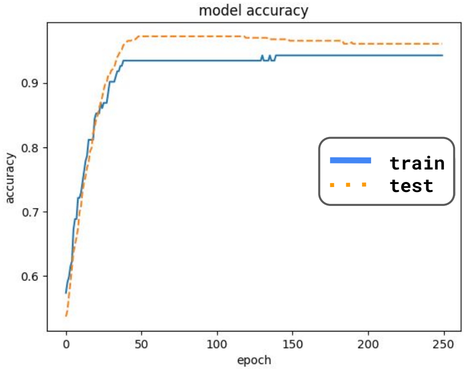
MONK 2

MONK 3

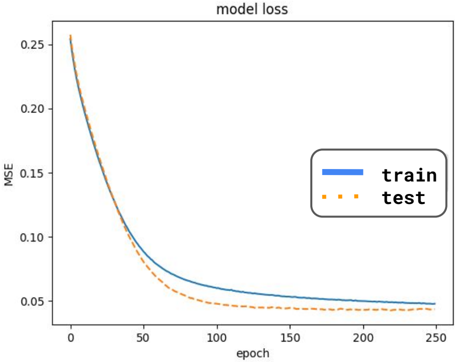
ACCURACY

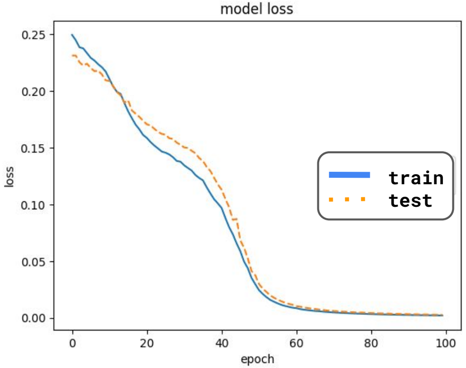


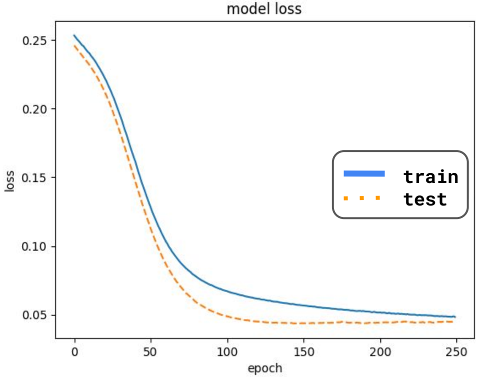




LOSS







MONKS Experiments: Neural Networks

MONKS Experiments: SVM & KNN

Monk	C	Kernel	gamma	Accuracy (TR/TS)
1	50	rbf	auto	100%/100%
2	10	poly	scale	100%/77%
3	1	rbf	scale	94%/97%

Table 1. SVM results

Monk	algorithm	leaf size	metric	n° neighbors	weights	Acc. (TR/TS)
1	auto	10	minkowski	8	uniform	81%/74%
2	kd_tree	30	minkowski	1	uniform	100%/77%
3	auto	10	manhattan	24	distance	100%/92%

Table 2. KNN results

MONKS Experiments: Final Results

	accuracy values on TEST SET		
Model	**MONK1**	**MONK2**	**MONK3**
Neural Network Keras	**100%**	**100%**	96%
Support Vector Machine	**100%**	77%	**97%**
K-Nearest Neighbors	74%	77%	92%

CUP Experiments: Validation & Grid Search

- The dataset was divided into two parts: **25%** was left for **internal testing** and the remaining portion for training and validation (development set).

- To perform model selection and choose the parameters, we used **grid search** and **K-Fold cross-validation** on the development set, while for PyTorch and Random Forest we use **Randomized cross-validation**

- Once the hyperparameters were obtained, the development set was split into training and validation subsets to perform the refit with the **best-found hyperparameters** and then run the model on the internal test dataset

CUP Experiments: NN Keras - Grid Search

For NN with Keras library we used **Grid Search with 3-fold Cross Validation**, the whole process took **3:30 hours**. After fixing the first parameters, through a **first model selection**, we performed a separated **second model selection** to choose the remaining parameters (weight decay and dropout) to reduce the global time of computation

1st model selection

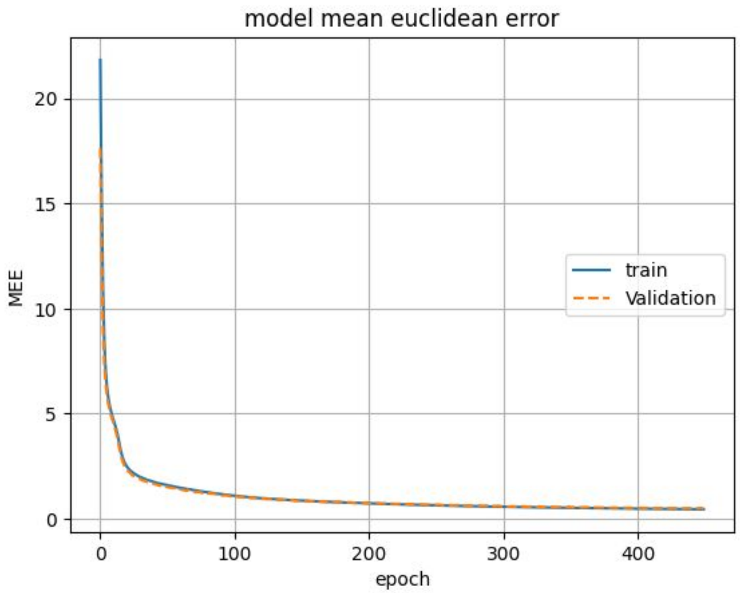
Parameters	Values
batch size	[8,16,32]
eta	[0.002, 0.01, 0.0008]
momentum	[0.0, 0.05, 0.08]
hidden layers	[1,2,3]
units	[30,50,100]
epochs	[200]

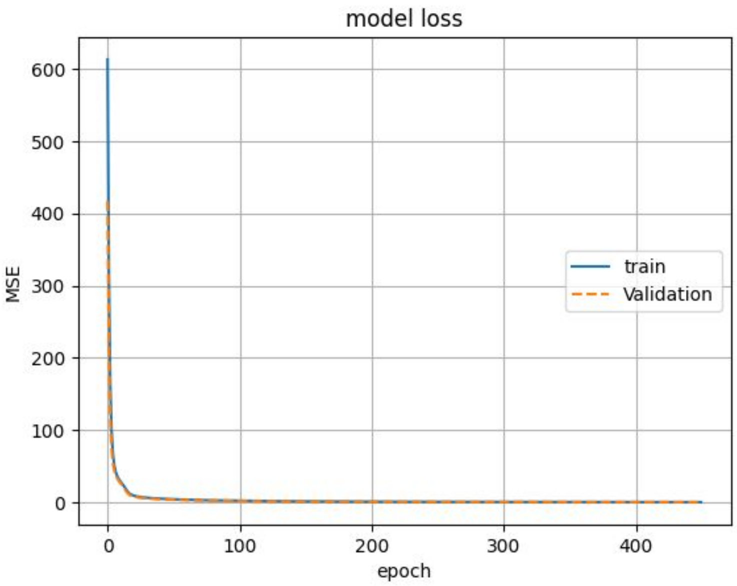
2st model selection

Parameters	Values
dropout rate	[0, 0.1, 0.03]
weight decay	[0.00001, 0.001, 0.01]

CUP Experiments: NN Keras - Results

epochs	activation	eta	momentum	units	hidden layers	weight decay	MSE (TR/VL/TS)	MEE (TR/VL/TS)
450	tanh	0.002	0.5	100	3	0.001	0.298/0.562 /0.379	0.469/0.551 /0.615





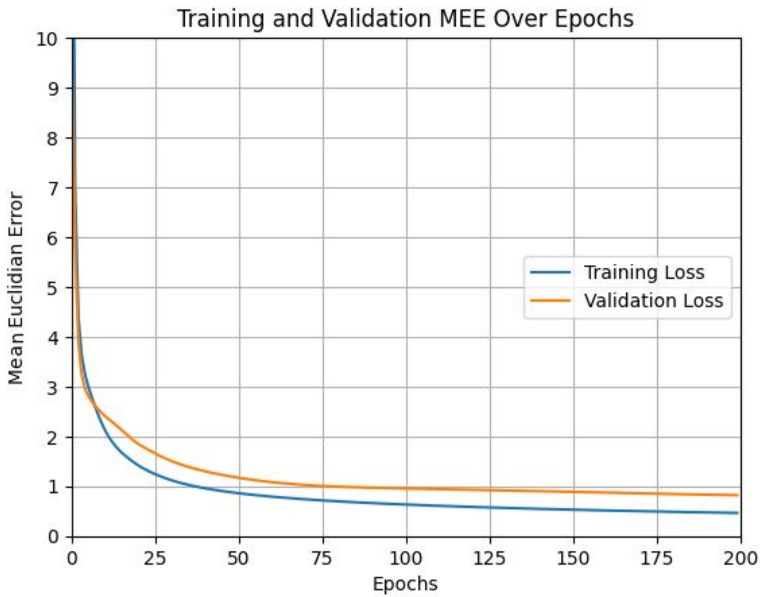
CUP Experiments: NN PyTorch - Grid Search

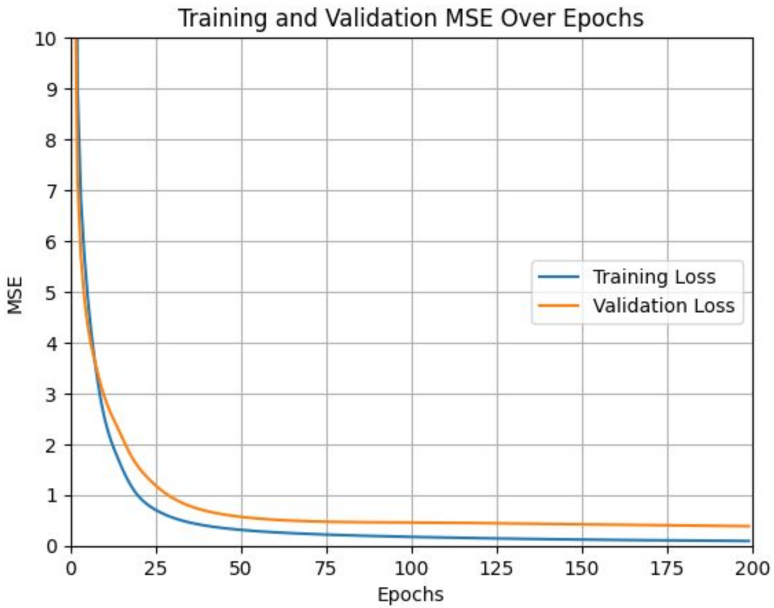
For NN with PyTorch library we computed **Randomized Grid Search CV** which took **1:30 hours**, some parameters were fixed like the number of hidden layers and the activation function

Parameters	Values
`eta`	`[0.002, 0.01, 0.0008,0.005,0.03]`
`momentum`	`[0.0, 0.05, 0.8]`
`weight decay`	`[0.0001, 0.001, 0.01, 0.1]`
`units`	`[50,100,150]`
`epochs`	`[200]`

CUP Experiments: NN PyTorch - Results

epochs	activation	eta	momentum	units	hidden layers	weight decay	MSE(TR/VL/TS)	MEE(TR/VL/TS)
200	tanh	0.005	0.5	150	3	0.001	1.58/1.03/0.52	0.95/1.21/1.24

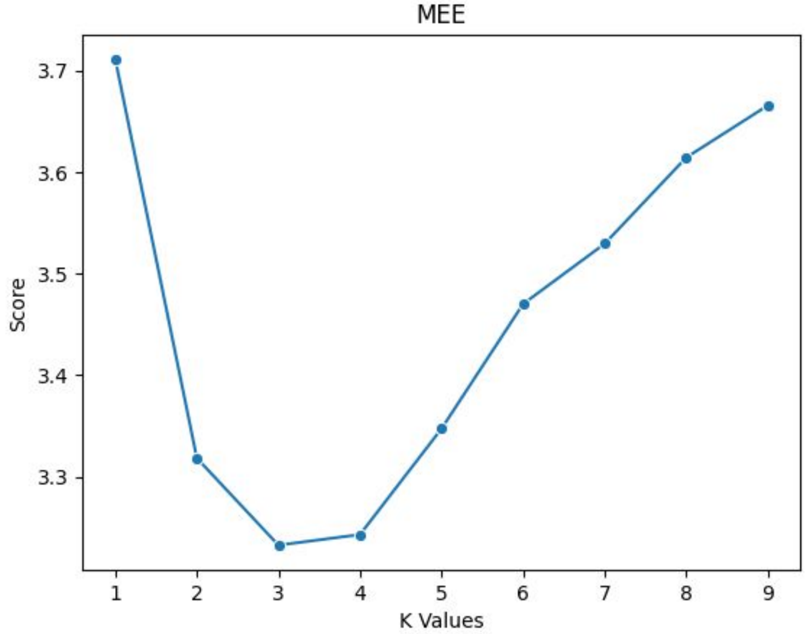




CUP Experiments: KNN - Grid Search

For KNN we first conducted a preliminary test by taking the average values of MEE, followed by a more in-depth Grid Search, time to search 10 minutes.

Parameters	Values
n° of neighbors	range (1, 30)
algorithm	[auto, ball_tree, kd_tree, brute]
leaf_size	[10, 20, 30]
weights	[uniform, distance]
p	range (2,10)



CUP Experiments: KNN - Results

algorithm	n ° neighbors (k)	leaf size	p	weights	MSE (TR/VL/TS)	MEE (TR/VL/TS)
auto	4	10	2	distance	0.0/3.62/ 4.96	0.0/2.56/ 2.83

CUP Experiments: SVR - Grid Search

For Support Vector we performed two different grid searches with K-Fold Cross Validation with k=5, Time of computation: 1 minutes for the first one and 5 minutes for the second one

1° Grid Search - 2 Parameters

Parameters	Values
C	[1,10,100]
Kernel	[linear,rbf]

2° Grid Search - 4 Parameters

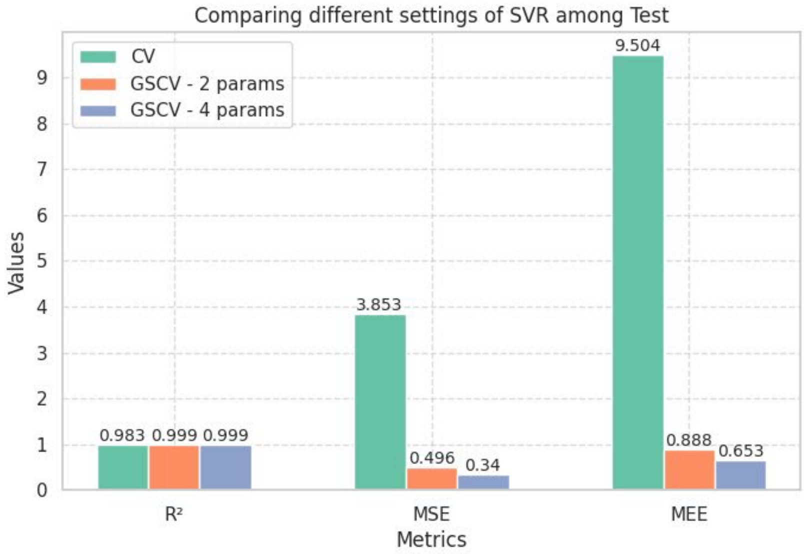
Parameters	Values
C	[100,250,500]
Kernel	[linear,rbf]
gamma	[0.1, 1, scale]
epsilon	[0.1,0.2]

CUP Experiments: SVR - Results

C	Kernel	gamma	epsilon	R (TR/VL)	MSE (TR/VL/TS)	MEE(TR/VL/TS)
100	rbf	scale	0.1	0.99/0.99	0.17/0.36/0.50	0.56/0.78/0.89
500	rbf	scale	0.1	0.99/0.99	0.06/0.22/0.34	0.32/0.58/0.65

We also compared the results of SVR with only Cross Validation without tuning parameters, discovering a very **high drop of the error rate after Grid Search**

Between the two GS we chose the second model since in the internal test, we obtained a value of `MEE=0.6` respect `MEE=0.88` of the first model



CUP Experiments: Ensemble Methods - Grid Search

We compared the results of two ensemble methods: one, the Bagging Regressor, and the other, Random Forest

Parameters	Values
n° estimators	[50,100,150]
max samples	[0.2, 0.5, 0.8]
max features	[0.1, 0.5, 0.8]
bootstrap	[True, False]
bootstrap features	[True, False]

Table1. Grid Search Bagging Regressor

Parameters	Values
n° estimators	[100,200,250]
criterion	[squared error,absolute error]
max samples split	[2, 5, 10]
max samples leaf	[1, 2, 4]
max depth	[10, 40, 80]
max features	[auto, sqrt]
bootstrap	[True, False]

Table2.Grid Search Random Forest

CUP Experiments: Ensemble Methods - Results

n° estimator	max samples	max features	bootstrap	bootstrap features	MSE (TR/VL/TS)	MEE (TR/VL/TS)
50	0.8	0.8	False	False	0.20/0.31/0.42	0.60/0.79/0.84

Table 1. Estimator used is SVM with the parameters find in the Grid Search, Result for Bagging Regressor

n° estimator	criterion	min samples split	min samples leaf	max depth	max features	bootstrap	MSE (TR/VL/TS)	MEE (TR/VL/TS)
250	absolute error	2	1	40	sqrt	False	0.0/2.96/3.38	0.0/2.31/2.42

Table 2. Result for Random Forest

CUP Experiments: ElasticNet Regression

For ElasticNet we performed grid search with 10-Fold CV using MultiTaskElasticNetCV which uses **R^2 as metric.** The free parameters are: alpa and l1_ratio. Time of execution was 2 minutes.

Parameters	Values
alpha	[1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.0, 1.0, 10.0, 100.0]
l1 ratio	range(0, 1)

Table 1. Grid Search

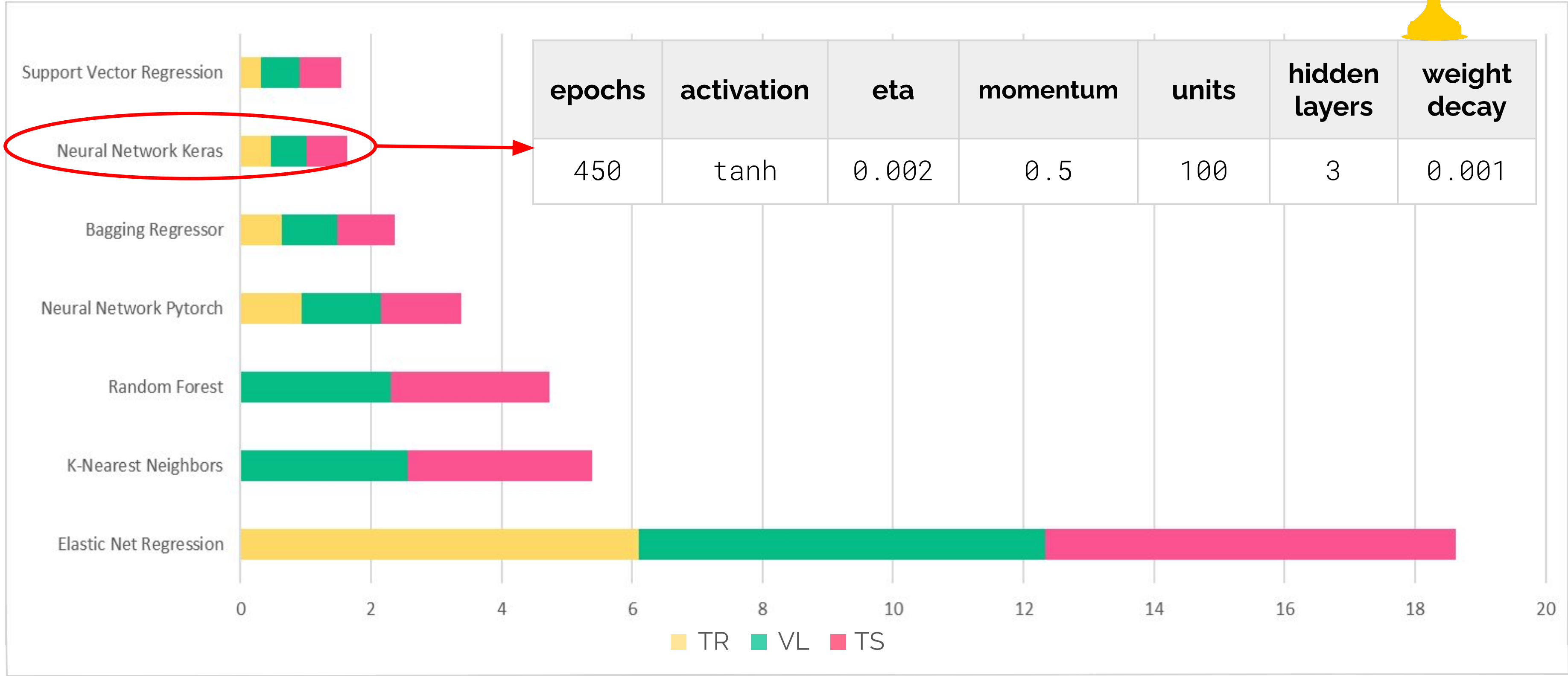
alpha	l1 ratio	MSE (TR/VL/TS)	MEE (TR/VL/TS)
0.01	0.91	18.0/17.3/19.33	6.1/6.24/6.29

Table 2. Results

CUP Experiments: Final Model Results

	MEE Score		
Model	**TR**	**VL**	**TS**
Neural Network Pytorch	0.94	1.21	1.24
Neural Network Keras	0.46	**0.55**	**0.62**
Support Vector Regression	**0.32**	0.58	0.65
K-Nearest Neighbors	0.0	2.56	2.83
Bagging Regressor	0.60	0.79	0.84
Random Forest	0.0	2.31	2.42
Elastic Net Regression	6.11	6.24	6.30

CUP Experiments: Final Model Results



epochs	activation	eta	momentum	units	hidden layers	weight decay
450	tanh	0.002	0.5	100	3	0.001

Conclusions

Tuning hyperparameters through Grid Search with k-Fold cross validation provides always **better results**

Neural Networks is the model which performed better both for classification on MONK and regression on ML-CUP datasets

For Neural Networks the Keras library was **easier** to use respect to PyTorch

To select the final model we chose the one with the **smaller Mean Euclidean Error (MEE)** on the internal **test set**, which, in our case, is the **Neural Network created with Keras**.

However also the **SVR model** provides good results and requires less time for tuning parameters through Grid Search (5 minutes vs 3 hours)

```
Blind Test File: Big_Human_ML-CUP23

Francesco Caprari
Francesco Botrugno
Agnese Camici
```

Bibliography

- Haykin, Simon S. *Neural networks and learning machines*. 3rd ed, Prentice Hall, 2009.
- Mitchell, Tom M. *Machine Learning*. McGraw-Hill, 1997.
- Goodfellow, Ian, et al. *Deep learning*. The MIT Press, 2016.
- Zou, Hui, e Trevor Hastie. *Regularization and Variable Selection Via the Elastic Net*. Journal of the Royal Statistical Society Series B: Statistical Methodology, vol. 67, fasc. 2, April 2005, pp. 301–20. https://www.jstor.org/stable/3647580. Accessed January 2024.
- PyTorch documentation, https://pytorch.org/docs/stable/index.html. Accessed January 2024.
- Keras documentation, https://keras.io/guides/. Accessed January 2024.
- Matplotlib documentation https://matplotlib.org/stable/index.html. Accessed January 2024.
- scikit-learn documentation https://scikit-learn.org/stable/user_guide.html. Accessed January 2024.

Thank you
for your attention