University of Pisa
Artificial Intelligence Course

Machine Learning Project Report
Course 654AA A.A 2020/2021 Project Type B

Giacomo Antonioli ID:619292 Email: g.antonioli3@studenti.unipi.it
Yohannis Kifle ID:621821 Email: y.telila@studenti.unipi.it

January 25, 2021

Abstract

This report is an analysis of the performance of different types of learning models applied to two different tasks. The first task is a binary classification problem on a MONK data-set[MON]. The second task is a binary output regression problem on a CUP data-set. The construction and development of the machines used to solve these problems were made by using two main Machine Learning frameworks: Keras and Scikit-Learn.

I. Introduction

The aim of the project was to gain a practical knowledge and experience of what the course explained and in particular to evince the main differences between the explored models and how the training process should be. The MONK data-set was used to experiment the some models and some training techniques, make some considerations on how to handle the data and how to train the models.
For both tasks the a methodical analysis was made:

1. Data preprocessing
2. Models selection
3. Model training
4. Model evaluation
5. Model testing

To train the Hyper parameters of all models two different solutions were chosen: a Grid Search approach was used, defining the ranges of the chose parameters and progressively narrowing the range of values around the best results obtained in the various search; a Randomized Search approach was also used to introduce some stochastic randomness to evade from saddle points and local minimums in the model function that was being tested. Once the models were all evaluated, according to the results obtained, the best ones were chosen.

II. Method

I. MONK

To analyze the MONK task 3 different algorithms were implemented. A custom Multi Layer Perceptron Model algorithm was tested as a Neural Network. The number of layers and the number of perceptrons in each layer were obtained through training and validation. Learning rate, momentum, activation functions, dropout rate, weight activation mode and number of epochs, early stopping and early stopping tolerance

hyper parameters were tuned in order to obtain the best model according to the accuracy percentage obtained on the Validation set.
A Support Vector Machine (from Scikit Learn) approach was used to implement another type of learning algorithm. To search through it's hypothesis space the following hyper parameters were used to improve the model: C - the regularization parameter, kernel - the degree of the polynomial kernel (used only to test the poly kernel), gamma - (Kernel coefficient for rbf, poly and sigmoid),coef0 (used for the poly and sigmoid kernels), tol - parameter for early stopping. The last implemented algorithm was KNeighbours Classifier (from Scikit Learn) where the following hyper parameters were tuned to optimized the models: n_neighbors, the number of neighbours, the weight function used for the prediction phase and the algorithm used to compute the nearest neighbour.

II. CUP

Since the main objective of this task is to try different models and get hands on experience on ML problems, we compiled every possible models that could solve this task. We selected ?13? models and compared their results. We started by re-scaling our data using using min-max scalar of scikit-learn library. We then started testing with off-the-shelf regressors(since our task is regression task) from scikit-learn library; RR(Ridge Regressor), SVR(Support vector regressor), KNR(K neighbors regressor), RFR(Random forest regressor), DTR(Decision tree regressor), ETR(Extra tree regressor), LR(Lasso regressor), RR(Ridge regressor), ENR(Elastic net regressor), BRR(Bayesian ridge regressor), SGDR(Stochastic gradient descent regressor), GBR(Gradient boosting regressor) and MPLR(Multi-layer Perceptron regressor). We optimized each models through gridsearch and We were able to get good results from some of the the regressors and few of them didn't perform well on our data.

Before we start to train our model, we did data pre-processing to check the distribution, correlation and outliers in the data. For the preprocessing task we used seaborn, pyplot and sklearn-preprocessing.

We then selected the top models that perform well on our exeperiment and created another model stacking the them together. We implemented this using scikit learn feature-union transformer object. The output of each previous model will be the feature for our new model.

II.1 Data preprocessing and analysis

Since we know little about our data, we decided to do some data analysis to get more intuition of our data. We know that our data has 10 features and 2 target variables and we started with analysing data distribution. We also did a correlation analysis to seek for relevant correlations between the given data-set, but no important correlation was found an no columns where removed or merged.

II.2 Data distribution

III. EXPERIMENTS

I. MONK Resutls

I.1 Preprocessing

The three MONK data-sets (already divided in training and testing data) are composed by six attributes of categorical features. A one-hot encoding technique was applied to prepare the data for the task. As a result the number of feature in each input vector grew to 17 binary features.

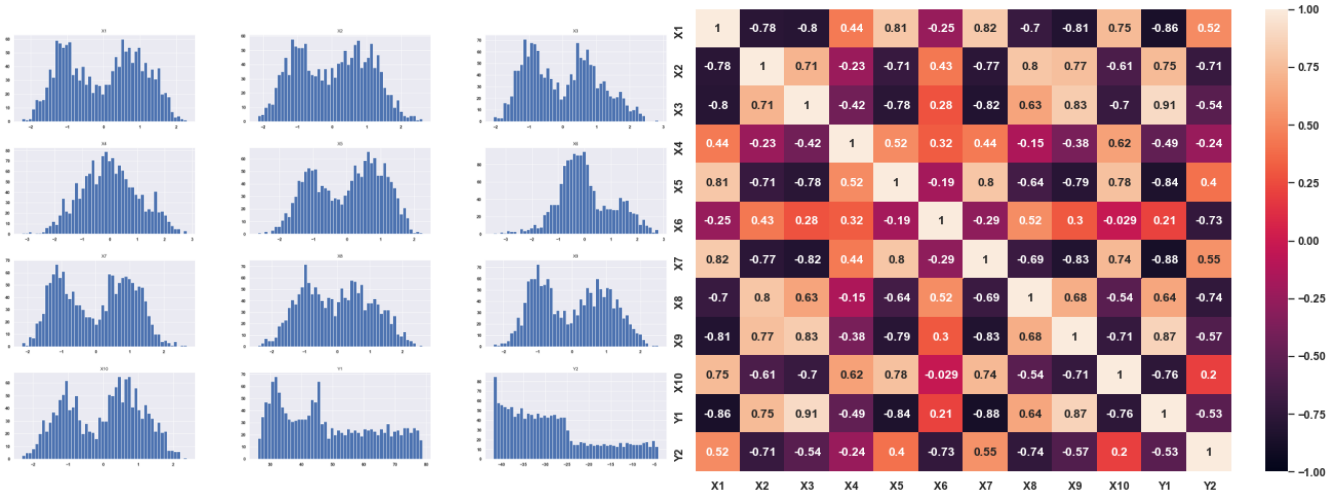

Figure 1: Data Analysis

I.2 Neural Network

For the Neural Network algorithm many parameters were tested using a grid search approach, starting from a coarse grid to shrink the range to more precise surroundings of the best parameters values. For each data-set a variable number units in the hidden layer were tested. More layers of hidden units were tested, but due to the simplicity of the problem a simpler but still effective model was preferred. A large range of values for the Learning rate and the Momentum were tested and the same policy used for the these parameters. The activation functions use in the model where decided before the testing phase according to the typology of the problem. Before each test the data was scaled with a min-max scaler in a range between (0,1).

Neural Network MONK results with parameters							
Task	Hidden Units	Lr	γ	Mse Tr	Mse Ts	Acc Tr	Acc Ts
Monk 1	5	0.9	0.4	0.0106	0.045	100%	93.98%
Monk 2	4	0.3	0.4	0.0052	0.0180	100%	98.15%
Monk 3	5	0.1	0.1	0.055	0.0425	93.44%	96.99%

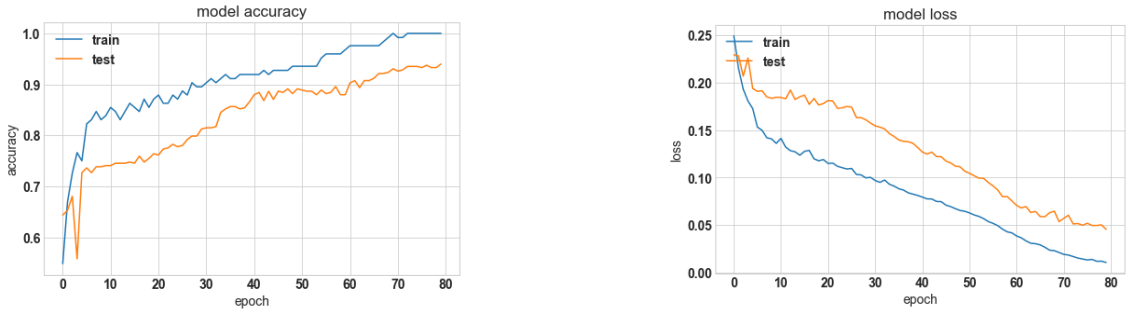

Figure 2: MONK1 Accuracy and MSE Loss with best Parameters

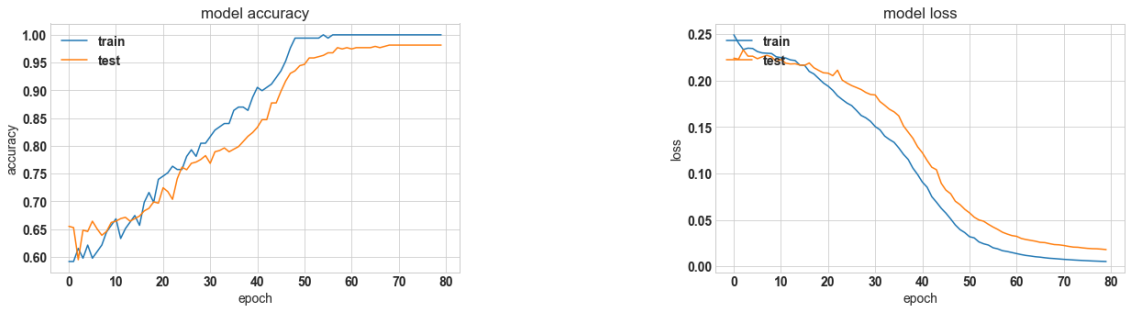

Figure 3: MONK2 Accuracy and MSE Loss with best Parameters

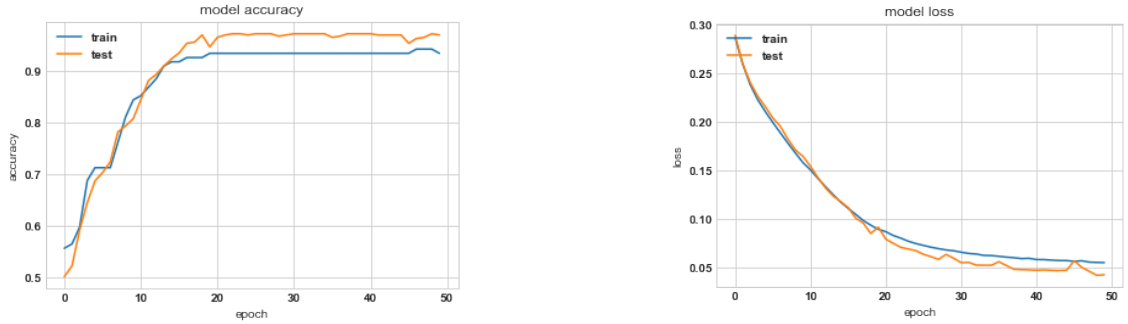

Figure 4: MONK3 Accuracy and MSE Loss with best Parameters

I.3 K Nearest Neighbour

For the KNN classifier a wide range of K values were tested, as well as different weight functions used in prediction phase,the algorithms used to compute the nearest neighbors.

K-Nearest Neighbour MONK results with parameters				
Task	Number of Neighbors	Weights	Algorithm	Accuracy
Monk 1	6	uniform	auto	81.71%
Monk 2	1	uniform	auto	75.23%
Monk 3	11	distance	auto	90.74%

I.4 Support Vector Machine Classifier

For the Support Vector classifier a wide range of values of Regularization parameter C were tested. The range went form a strong regularization (1) to a lose one (1000). Many kernel degrees where tested as well as some specific parameters for some specific kernels (gamma,degree,coef0). The tolerance for the stopping critereon was also used as an hyper parameter in the testing phase.

Support Vector Machine Classifier MONK results with parameters							
Task	C	kernel	degree	gamma	coef0	tol	Accuracy
Monk 1	10	rbf	—	0.1	—-	0.001	100%
Monk 2	1000000	rbf	—-	0.001	—	0.001	100%
Monk 3	1	linear	—	scale	—	0.001	93.5%

II. CUP Results

The CUP data-set was divided in a training set (60%) and a test set(40%). For the training of each model a cross validation technique the data was split into 10 folds, using 90% of the data as training set and 10% for validation, repeated 10 times over 10 different configurations and averaged.

II.1 Screening phase

Many of the selected algorithms where tested and some, even after an intensive hyper parameter search, were discarded. The models were tested using the extremes of the selected parameters and then the more useful ranges for each parameters were selected to be used in further grid searches.

II.2 Models

From our list of model experiments we selected SVR, Ridge, KNeighborsRegressor, RandomForestRegressor and MLPRegressor for further training. We stacked the output of these models and fed to the next model which is KNeighborsRegressor. We ran a grid search over the final model and we were able to optimize its parameter.The tables below show the grid search ranges and parameters used to train each model (the reported ranges are the ones of the final grid searches).

SVR Grid Search	
Kernel	'rbf','sigmoid'
C	1,12,23,34,45,56,67,78,89,100
γ	0.01, 0.1, 1.0, 10.0, 100.0, 'scale'
ϵ	0.1,0.2

Ridge Regressor Grid Search	
α	0.02, 0.024, 0.025, 0.026, 0.03,1,5,10,50,100,200, 230, 250,265, 270, 275, 290, 300, 500
Solver	'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga'
Max Iter	50,100,250,500,1000,1500,2000
Normalize	True,False
Intercept	True,False

KNearestNeighbor Grid Search	
Neighbors	range(3,50)
Leaf Size	range(3,30)
Weights	'uniform', 'distance'
P	1,2

Random Forest Regressor Grid Search	
Estimators	.linspace(start = 80, stop = 200, num = 5)]
Max Features	'auto', 'sqrt'
Max Depth	linspace(10, 100, num = 5)
Min Samples Split	2,5,10
Min Samples Leaf	1,2,4
Bootstrap	True, False

Multi Layer Perceptron Regressor Grid Search	
Hidden Layers	8,10,12,14,16,18
Activation	'tanh','relu'
Learning Rate Init	0.1,0.01,0.001,0.2
Max Iter	1000,2000,3000
Momentum	0.1,0.01,0.5
Tol	1e-2,1e-3,1e-4,1e-5

The fig below shows the training and cross validation score of the final output model.

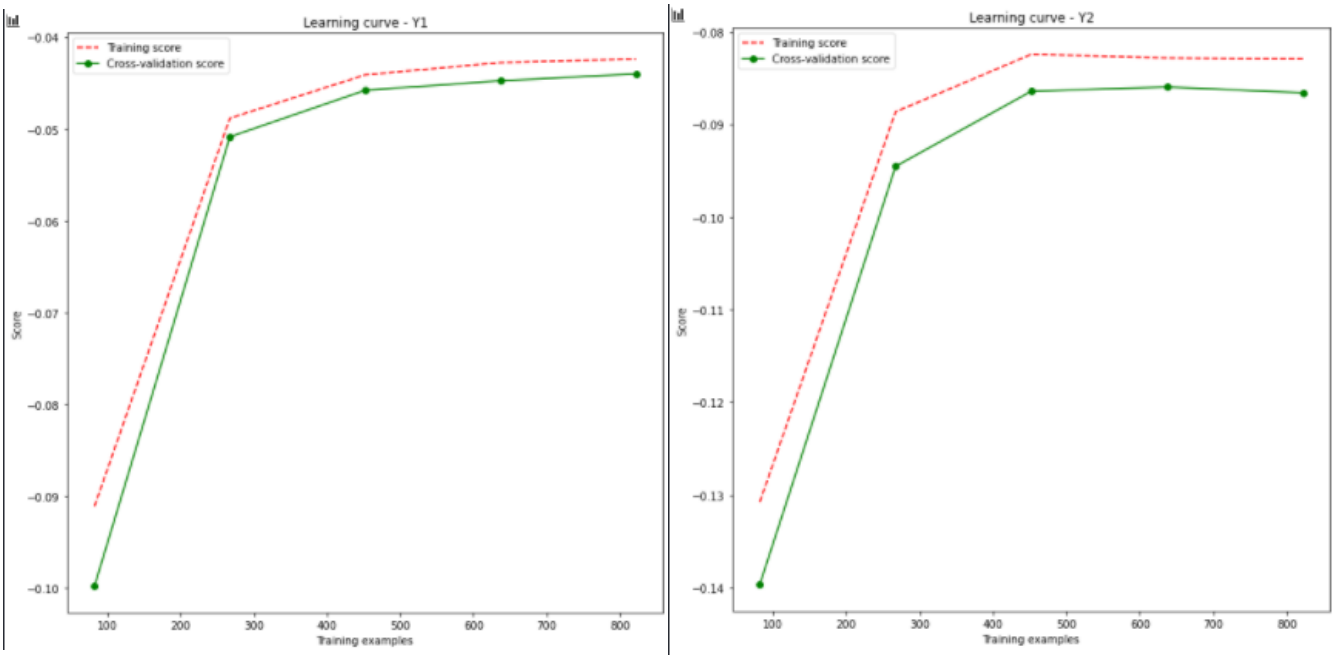

Figure 5: Accuracy for Y1 and Y2

II.3 KNN Regressor

We observed that KNN Regressor performed very well from all of our regression models. Then we decided to do another experiment with it. We wrapped KNN Regressor in a RegressorChain model. This will create two regressor models which will help us to predict two target variables. Since we already got the best parameter for this model optimizing it wasn't so difficult. The fig below represents the learning curve for the model.

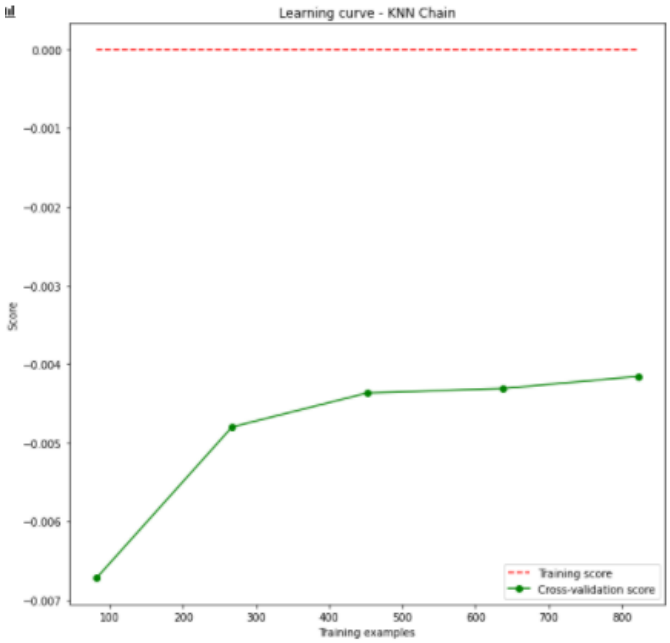

Figure 6: Final Model MSE Accuracy

The Final model was obtained stacking the best selected models in a first layer and training another model on a second layer feeding him the previous predictions of the pre-trained models. This decision was taken because having many accurate problems and teaching a new model to recognize the most accurate output of the models is accurate at least accurate as the worst selected model, but it also could outperform each single model taken on its own. The tables below are the best parameters configuration of all the models.

SVR Model Best Parameters				
Output	C	Kernel	γ	ϵ
Y1	1	rbf	0.1	0.1
Y2	23	rbf	0.01	0.1

RidgeRegressor Model Best Parameters					
Output	α	Solver	Max Iter	Normalize	Fit Intercept
Y1	50	sparse_cg	50	False	True
Y2	50	safa	50	False	True

KNearestNeighbor Model Best Parameters				
Output	Neighbor	Leaf Size	Weights	P
Y1	18	3	Distance	2
Y2	10	3	Distance	2

Random Forest Regressor Model Best Parameters						
Output	Bootstrap	Max Depth	Max Features	Min Samples Split	Min Samples Leaf	Estimators
Y1	False	32	sqrt	2	1	170
Y2	True	55	sqrt	2	1	200

Multi Layer Perceptron Regressor Model Best Parameters						
Output	Hidden Layer size	Activation	Learning Rate Init	Max Iter	Momentum	Tol
Y1	20	tanh	0.1	3000	0.001	1e-05
Y2	10	tanh	0.1	2000	0.001	0.0001

Models Accuracy			
Model	MSE Train(Y1/Y2)	MSE Test(Y1/Y2)	Time(Minues)
SVR	-0.003/-0.005	0.003/0.007	1.6/1.7
RR	-0.004/-0.008	0.003/0.008	4.2/5.0
KNR	-0.002/-0.007	0.001/0.003	6.0/5.4
RFR	-0.002/-0.007	0.001 /0.004	112.4/95
MLPR	-0.002/-0.007	0.003/0.008	35.5/29.9

Random Forest Regressor Model (second layer) Best Parameters			
Neighbors	Algorithm	Leaf Size	Weights
10	brute	1	Distance

Final Model Accuracy				
Model	MEE Train	MEE VAL	MEE Test	Time(minutes)
SVR	0.0719	0.34	0.4610	1.24

III. Discussion

During our experiment we implemented dimension reduction techniques to see if our model could do better with this approach. From all of our models Ridge regressor and Lasso regressor models are the one who showed a great difference. We this technique, the two models were able to converge to the optimal hyperparameter. But on the other hand, the learning time has increased significantly.

IV. CONCLUSIONS

From this project we learned how to interact with the available simulators on the market, and how various algorithm can be more or less effective with regard to the nature of the problem, the importance of the hyper parameters and how an efficient search strategy mixed with heuristics from a theoretical background can be time saving instead of using a brute force approach.
The results obtained on the blind test are in the file named "Boca-chica_ML-CUP20-TS.csv". The Group Nickname is Boca-chica.

V. Acknowledgment

We agree to the disclosure and publication of my name, and of the results with preliminary and final ranking.

References

[MON] MONK. *MONKS data-sets*. URL: https://archive.ics.uci.edu/ml/datasets/MONK's+Problems. (accessed: 24.01.2021).