Machine Learning Project

Alessandro Alberti, Miguel Alcalá Belmonte, Giovanni Scognamiglio
a.alberti17@studenti.unipi.it, m.alcalabelmonte@studenti.unipi.it, g.scognamiglio2@studenti.unipi.it
Mathematics (Modeling), Data Science Bachelor (Erasmus), Data Science & Business Informatics.
ML course (654AA), Academic Year: 2021/22
Date: 24/01/2022
Type of project: **B**

Abstract

The following is an analytical comparison of the most widely adopted machine learning models with an emphasis of the effect of ensemble methods and regularisation. The research is built on extensive empirical testing based on cross-validation and random inner test set for model selection. The final model used for ML Cup prediction is AdaBoost on Decision Tree.

1 Introduction

The aim of this project was to gain a deep understanding of the various machine learning languages. More in depth, our intent was to explore and discover the representational potential and flexibility of various machine learning languages with respect to (a) complexity control requirements and (b) efficiency and time consumption.
The models on which our study is based are (1) Decision-Tree and its related Ensemble-based methods (2) Kernel based methods SVM and (3) Neural Networks.
The validation schema we followed for all models was based on (a) hold out for plotting learning curves in order to understand the absolute effect of single hyper parameters, (b) 5-k fold cross validation for graphically assessing the relative effect of different parameters combinations and (c) minimum (5-7)-k fold for automated grid searches . For the ML-CUP, we first extracted a random 10% split from the training data to use as inner test set. Furthermore, all inner test set results reported in the tables for all models are the average results of a 20 time initialisation repetition.
For each predictive problem (MONK's and ML-CUP), we fine tuned a learning algorithm with both graphical and automated approaches such as plotting graphs matrix and grid search computations to obtain the best possible hypothesis to approximate the target function of the problem.

2 Method

The project was built using Python and the main workspace was Jupyter Lab hosted on two Google VM instances with 8vCPUs each which allowed us to make heavy usage of task parallelisation for independent tasks. Python libraries such as Numpy and Matplotlib were used along with Sklearn library for modeling the tree-based algorithms and Keras for training the neural networks. Dask and IpyParallel libraries were used for parallelising tasks on multiple cores.
For the neural networks, multiple architectures (mostly three layers models), were developed as they tended to provide the best performance whilst having still acceptable training time. Rectified Linear Unit (ReLu) activation function was used throughout all the networks built for the ML-CUP. SGD with Nesterov Momentum and learning decay was the most used search algorithm as it provided high flexibility. Minibatch sizes of 32,64 and 128 were used as they minimised validation waiting time whilst allowing stable learning curve. No regularization, Tikhonov and Drop-out regularizations were tested. Widely adopted Glorot initialization schema was tested along with customs initialisation range. Early-stopping on loss validation and retraining on mean training loss value for the structural stabilised and Tikhonov regularised models, whilst a 10% validation hold-out was used to stop retraining the Drop-Out regularised model.

3 Experiments

3.1 Monk Results

Neural Networks results

Task	Hyperparameters					BCE		MSE		Accuracy	
	n	lr	mom	l-decay	lam	TR	TS	TR	TS	TR	TS
Monk1	4	0.5	0.5	2eˆ-4	0	0.017	0.03	0.003	0.005	0.998	0.995
Monk2	4	0.1	0.9	1eˆ-4	0	0.001	0.024	3eˆ-6	0.006	1	0.994
Monk3	4	0.15	0.7	0	0	0.136	0.175	0.035	0.049	0.952	0.937
Monk3+lam	4	0.05	0.5	0	0.0025	0.261	0.257	0.054	0.053	0.935	0.950

Table 1: Average (off 20 initializations) results obtained for the MONK's tasks with a single hidden layer NN.

SVM results

Task	Hyperparameters			Accuracy	
	Kernel	C	gamma	TR	TS
Monk1	'rbf'	94.222	0.007	1.0	1.0
Monk2	'rbf'	1eˆ7	0.001	1.0	1.0
Monk3	'rbf'	1.0	0.1	0.934	0.972

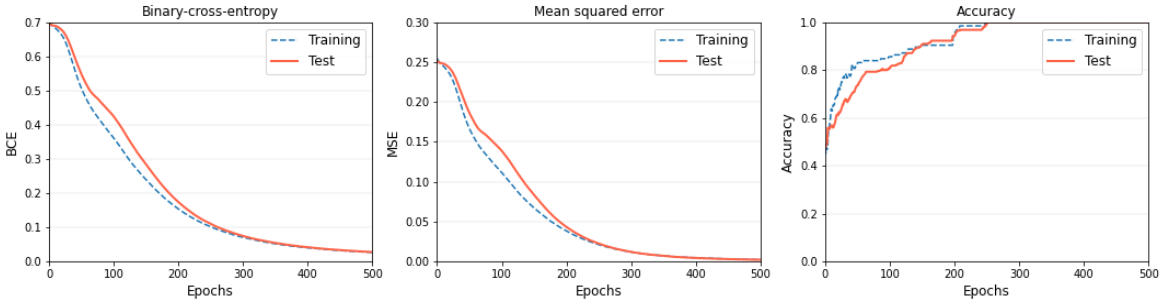
Table 2: SVM results obtained for the MONK's tasks.

Decision Tree based models results

	Task	Hyperparameters				Accuracy	
		Criterion	min_sample_split	min_sample_leaf	n_estimators	TR	TS
D.T	Monk1	'gini'	2	1	-	1.0	0.956
	Monk2	'gini'	2	1	-	1.0	0.697
	Monk3	'gini'	20	12	-	0.934	0.950
R.F.	Monk1	'gini'	2	1	52	1.0	0.991
	Monk2	'gini'	2	1	150	1.0	0.625
	Monk3	'gini'	18	8	10	0.934	0.950
Ada	Monk1	'gini'	13	13	24	1.0	1.0
	Monk2	'gini'	22	15	103	1.0	0.802
	Monk3	'gini'	24	36	10	0.975	0.891

Table 3: Average (off 20 initializations) results obtained for the MONK's tasks with Single Decision tree classifier, Random Forest and AdaBoost tree classifier.

MONK1



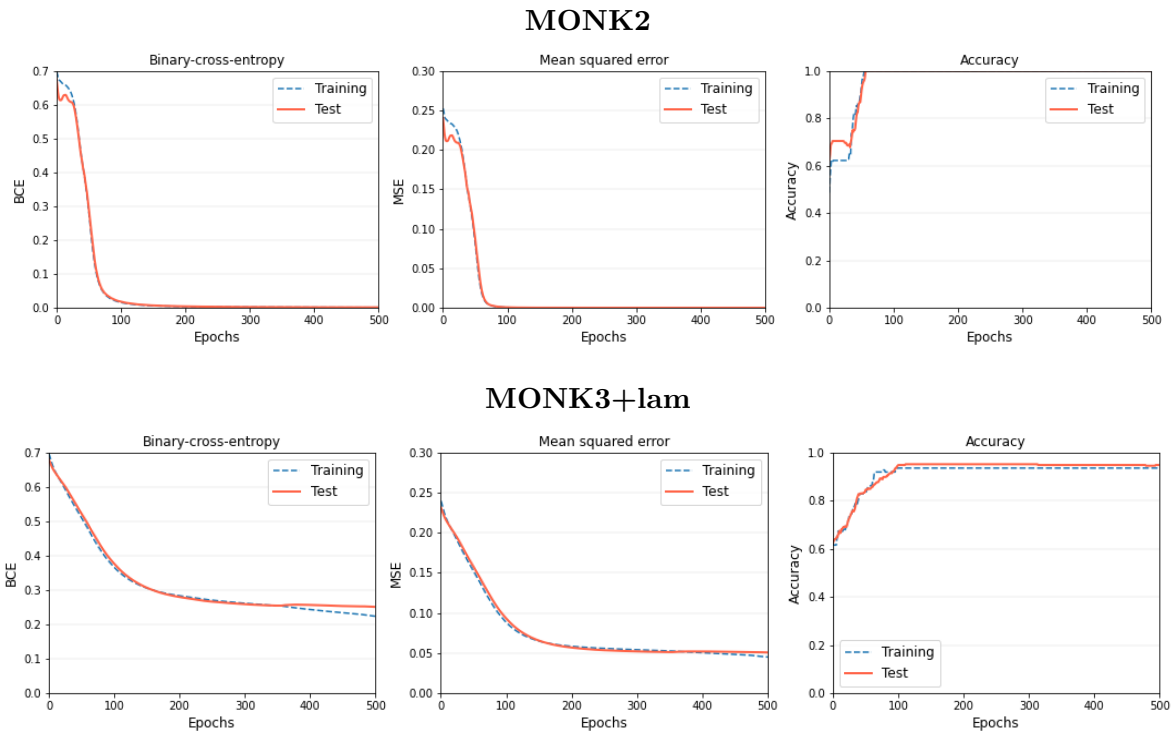


Figure 1: Learning curves obtained with best NN models for each of the MONK's problems. We represent the binary-cross-entropy (used loss), the MSE and the accuracy both for the training and test scores.

3.2 Cup Results

Neural Network Cup results

The screening phase was based on extensive empirical testing through visualisations and eventually automated targeted grid searches. We first tried to understand the model's behaviour with an increasing number of neurons and layers. We saw that a single layer was not sufficient to fully fit the training data, whilst a two layer model had much more representational capability even with fewer neurons (Figure 6 in Appendix).

We proceeded with two different approaches to find the best hypothesis. The first approach was to proceed with a bottom-up strategy and find a model that would generalise well simply by constraining the model's toponomy (structural stabilisation). We added an increasing number of neurons and layers before the model went into overfit. We further searched for a better hypothesis by applying learning decay. After seeing that increasing learning decay allowed us to have a much larger model without incurring in overfitting, we tested whether such a model would yield a better generalisation than the simple model obtained with the structural stabilisation. In Figure 7 in Appendix, we see reported the best models obtained from our graphical screening phase for both

approaches. After graphically assessing a good-enough model, we performed a targeted gridsearch.

Neural Network - Structural stabilization					
	n_neu	l_rate	mom	MEE TR	MEE VAL
Search range (Time: 12H)	[30,45]	[0.001,0.003]	[0.7,0.9]	-	-
Top 3 results	41-38-40	0.0015	0.8	1.042	1.137
	36-42-40	0.0010	0.8	1.041	1.138
	35-44-40	0.0010	0.8	1.040	1.138

Table 4: Best gridsearch results for structural stabilization NN models

In the second approach, we used a large toponomy that overfitted the data and applied regularisation to achieve the best possible generalisation. We first tried to use only Tichonov L2 regularisation and made a gridsearch for best hyperparameters (the best 3 results are in Table 10 in Appendix while the best result is tested on the inner test in Table 6). Here we used Early-stopping by validation loss and we retrained stopping at the mean training loss.

Finally both Tikhonov L2 weight penalisation and modern Dropout regularisation were extensively tested both graphically and trough gridsearch. As suggested by Srivastava et al. [2] we used large networks (3 hidden layers) with a relatively high learning rate and learning rate decay (fixed at the value of $l\_rate/10$ in keras SGD function). Because of the trembling effect on the training loss due to Dropout, we used a validation hold-out set to retrain the model before testing it: we used 10% of the training dataset to use Early-stopping on the validation loss.

Neural Network with Drop-out regularization							
	n_neu	l_rate	mom	lam2	dp_rate	MEE TR	MEE VAL
Search range 1 (Time: 12H)	(50,70,100,150)	[0.0003,0.003]	(0.7,0.9)	[0.0,0.1]	[0.08,0.4]	-	-
Search range 2 (Time: 12H)	(100,150)	[0.0005,0.003]	(0.7,0.9)	[0.0,0.005]	[0.1,0.3]	-	-
Top 3 results	150 (x3)	0.003	0.9	0.0025	0.225	0.822	1.033
	150 (x3)	0.003	0.9	0.002	0.275	0.825	1.034
	150 (x3)	0.003	0.9	0.0015	0.3	0.842	1.039

Table 5: Best gridsearch results for structural Dropout NN models

We identified the best NN model by looking at the inner test MEE results and plotted the corresponding learning curves (Figure 2). In the table below, NN1 refers to the best structural stabilisation model, NN2 refers to the best model with only Tikhonov weight decay and NN3 refers to the best Dropout+Tikhonov regularization model.

Models	Hyperparameters					Validation MEE			Inner test MEE	
	n_neu	l_rate	mom	lam2	dp_rate	MEE TR	MEE VAL	VAL STD	TR	TS
NN1	41-38-40	0.0015	0.8	0	0	1.042	1.137	0.026	0.971	1.151
NN2	75 (x3)	0.0017	0.9	0.0040	0	0.747	1.112	0.052	0.784	1.138
NN3	150 (x3)	0.003	0.9	0	0.225	0.822	1.033	0.065	0.811	1.073

Table 6: Inner test results for the three approaches best models.

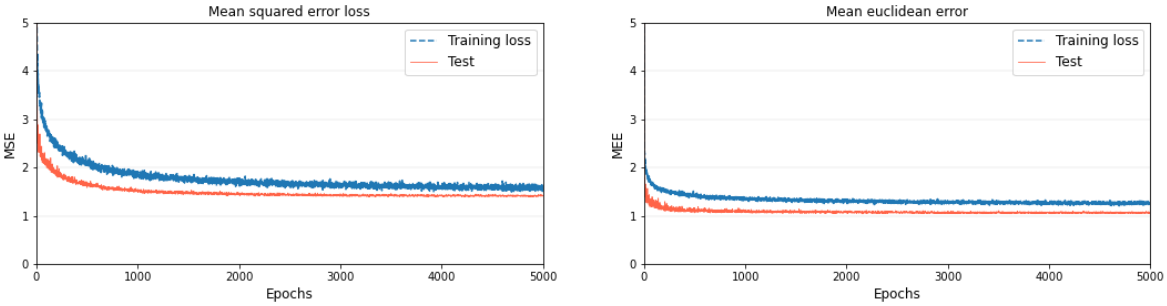


Figure 2: Learning curves of the NN3 model.

SVM results:
In the implementation of the SVM we compared different models using a 5-fold Cross Validation on the TR+VL set. We used 'MultiOutputRegressor' function (sci-kit learn library) to extend the SVM to a multi-output problem, as the CUP one. We searched through the hyperparameters space with the following validation schema:

- We made a large-range grid of values of C and γ with fixed ϵ (to default value 0.1) and kernel ('rbf'). We plotted the values through a heat maps (Figure 9).
- We made a large-range grid of values of ϵ and C with fixed γ (to default value 'auto') and kernel ('rbf'). We plotted the results through heat maps (Figure 10).
- We made a large-range grid search on ϵ, γ, C and 'kernel' and then a thinner grid search on the best range for the hyperparameters. We finally tried to improve our results running a random search (1000 trials) on smaller ranges:

SVM				
	Kernel	C	γ	ϵ
Search Range 1 (Time: 20min)	'rbf','lin','sigmoid'	[1e^-4,1e^8]	[e^-4,e^5]	[e^-5,e^7]
Search Range 2 (Time: 1h 40min)	'rbf'	[1,50]	[0.01,1]	[0.001,1]
Search Range 3 (Time: 25min)	'rbf'	[1,20]	[0.05,0.15]	[0.01,1]

Table 7: Hyperparameters ranges for the two grids (1-2) and the ranges used for the final random search (3).

SVM						
	Kernel	C	γ	ϵ	MEE TR	MEE VAL
Top 3 Results	'rbf'	12.462	0.089	0.596	0.985	1.135
	'rbf'	16.517	0.072	0.610	1.001	1.136
	'rbf'	12.960	0.097	0.308	0.916	1.137

Table 8: Best three models obtained with the validation schema

- We selected the best result by the validation euclidean error (MEE val) and then tested on the internal test set. Final results:
  TRAIN MSE: 0.792 – TEST MSE: 0.883
  TRAIN MEE: 0.979 – TEST MEE: 1.032

DecisionTree and DecisionTree Ensemble Cup results

We started by applying a fully unregularized single Decision Tree Regressor and observed a noticeable variance in the results. On one thousand hold-out tests, we obtained 1.45 validation MEE with 0.12 standard deviation. We imagined that the single unregularized tree probably overfit the data and thus created an adaptable complexity scale on which to plot the results of multiple runs of the single tree (Figure 3).

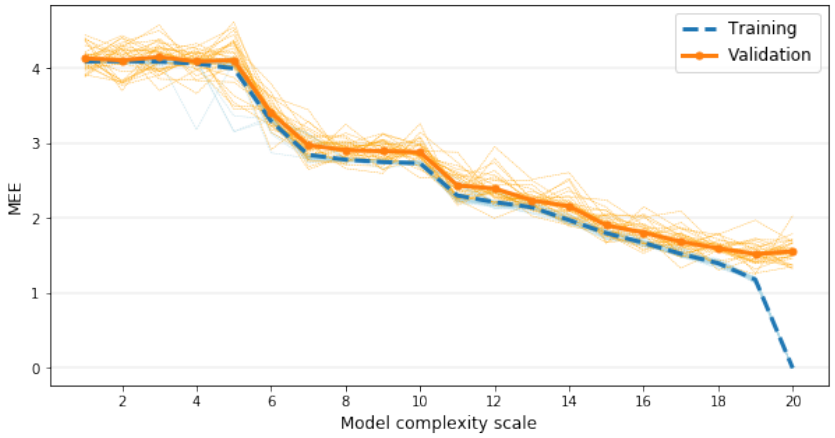


Figure 3: Single decision tree regressor MEE results by increasing complexity

This plot allows us to (1) quickly get a sense of the generalization capability of the model (2) to get very thin parameter range on which to perform a gridsearch.
From the Figure 3, we can see that our model slightly overfits when being fully unregularised and that our model's best generalization is achieved when slighlty regularized.
We thus performed a targeted gridSearch and the results are reported hereafter:

SINGLE DECISION TREE					
	min_leaf	min_split	max_features	MEE TR	MEE VAL
Search Range (Time: 2 min)	[1,42]	[2,12]	[0.7,1]	-	-
Top 3 Results	9	5	0.8	0.982	1.398
	9	9	0.7	0.962	1.399
	11	10	0.8	1.043	1.399

Table 9: Best three models obtained with Single Decision Tree

The next model we analysed was RandomForest model, a bootstrap aggregation ensemble tree-based model. We first graphically compared the difference between the above fine-tuned tree and a RandomForest with the same tuning with increasing number of estimators.

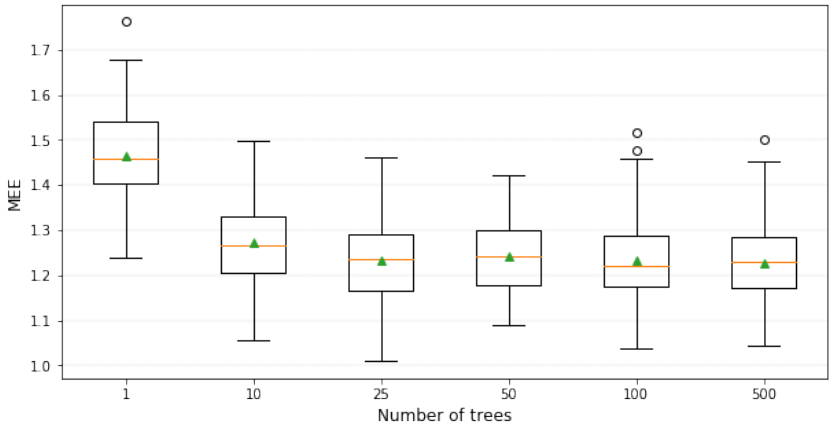


Figure 4: Fine-tuned single decision tree (N trees = 1) vs RandomForest with increasing number of estimators

From Figure 4 we clearly understand the advantage the ensemble method can provide without almost any regularisation. We see that by simply using an aggregation of 10 models trained on a re sampled data set, we reduce the mean MEE from around 1.4 to 1.2 along with a strong reduction in variance. In order to get a further idea of the behaviour of RandomForest for various level of complexity and number of estimator we plotted a matrix of plots based on Figure 3 for different estimators (see Figure 11 in Appendix). From Figure-RandomForest-matrix, we noticed that the model achieved its best generalisation level when being fully unregularised. Contrary to the single tree that overfit the data when fully unregularised, we evidentiate that the bootstrap aggregation prevents overfitting by nature. We thus performed a targeted gridsearch with unregularized parameters and obtained the following results:

RANDOM FOREST						
	min_leaf	min_split	max_features	n_estimators	MEE TR	MEE VAL
Search range (Time: 2 h)	[1,4]	[2,5]	[0.7,1]	[50,1050]	-	-
Top 3 Results	2	4	0.7	550	0.495	1.080
	1	4	0.1	750	0.453	1.083
	1	2	0.8	650	0.409	1.085

Table 10: Best three models obtained with Random Forest

When running a thousand random trials with the best tuned model, we obtained 1.11 validation MEE with 0.074 standard deviation. Thus evidentiating the variance reduction effectiveness of the ensemble method and the superior generalisation gained from it. Finally, we applied the AdaBoost with DecisionTree ensemble. Following our framework, we plotted the matrix of plots of validation MEE with respect to complexity and various number of estimators. (see Figure 12 in Appendix)
From Figure 12 in Appendix, we saw that, similarly to RandomForest, the best performing model is a fully unregularised model and we thus proceeded with our usual targeted gridsearch.

ADABOOST						
	min_leaf	min_split	max_features	n_estimators	MEE TR	MEE VAL
Search Range (Time: 3 h)	[1,5]	[2,5]	[0.8,1]	[500, 800]	-	-
Top 3 Results	1	4	0.8	625	0.074	1.037
	1	3	0.9	775	0.062	1.038
	2	4	0.8	625	0.135	1.043

Table 11: Best three models obtained with AdaBoost

From one hundred trials on different random splits, we obtained a 1.037 validation MEE with 0.087 standard deviation.

Overall, as summarised in Table 12, we saw that the a clear advantage of tree-ensemble methods over the single decision tree both in terms of variance reduction and most notably in terms of improved bias.

Models	Hyperparameters				Validation MEE			Inner test MEE	
	m.s.s	m.s.l	m.f	n.est	TR	VAL	VAL STD	TR	TS
Decision Tree	5	9	0.8	-	0.943	1.445	0.118	0.938	1.363
RandomForest	4	2	0.7	550	0.488	1.086	0.072	0.483	1.084
AdaBoost	4	1	0.8	625	0.072	1.048	0.087	0.071	1.021

Table 12: Best models for each Tree Baased models

Finally, the best performing parameters extracted from the gridsearch for the three models mentioned above are reported in table tree-based-best along the performance they achieved on the initial hold-out inner test set.

Final model selection

The inner test set was used as a selection criteria. We first saw that the deep neural network with Drop-out, reported in the gridsearch table, appeared to have a very strong performance; however, when trying to retrain the model in order to use it for predictive, we noticed high variance in the results obtained. After extensive trials, we evidentiated that the high variance was due to the nature of the Drop-out regularization paired to the high learning rate used. We managed to lower the variance by lowering the learning rate; however it decreased the advantage gained from the Drop-out as the validation performance decreased. Finally, we thsus prefered to use the AdaBoost tree based model as it provided a low bias and low variance hypothesis that we deemed more reliable than the deep neural network for the problem at hand. Hereafter are the hyperparameters values of the model applied along with the results obtained from the cross validation and the inner test sets results.

Models	Hyperparameters				Cross-validation MEE			Inner test MEE		
	m.s.s	m.s.l	m.f	n.est	TR	VAL	VAL STD	TR	TS	TS STD
AdaBoost	4	1	0.8	625	0.072	1.058	0.087	0.071	1.021	0.006

4 Conclusion

We started our study with the intent of gaining practical knowledge on the application of different machine learning models. What followed was an intense and highly instructive journey through the complex world of predictive modelling where we deepened our knowledge about many practical and theoretical concepts seen in class. Most notably, we stumbled across the stark contrast between a model's flexibility and time requirement to tune the model. Overall, around 80 percent of the time spent on this project was spent on the neural network model selection. We discovered that, although selecting a very flexible language and shifting the bias to the search algorithm improves the chances of having the true hypothesis in the hypothesis space, in practice this has significant drawbacks in terms of time and resources consumption when compared to model with higher language bias. In short, larger hypothesis space translates into having to search through a larger number of hypothesis and thus requires more time. We evidentiated that, in this case, other models like SVM can provide a competitive performance with reasonable tuning times. This can further be seen by the performance we obtained with Decision Tree that can leverage its simplicity and efficiency to be combined with an ensemble technique so as to obtain a strong performance with much less resources required than the neural network.

Blind test results: our group's nickname is *LosTurrones* and the file containing our predictions is named *losturrones-ML-CUP21-TS.csv*.

Acknowledgments

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

References

[1] Prechelt L.: *Early Stopping - but when?*. Neural Networks: Tricks of the Trade (2012).

[2] Srivastava N., Hinton G.,Krizhevsky A., Sutskever I.,Salakhutdinov R.: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research 15 (2014) 1929-1958.

Appendix A

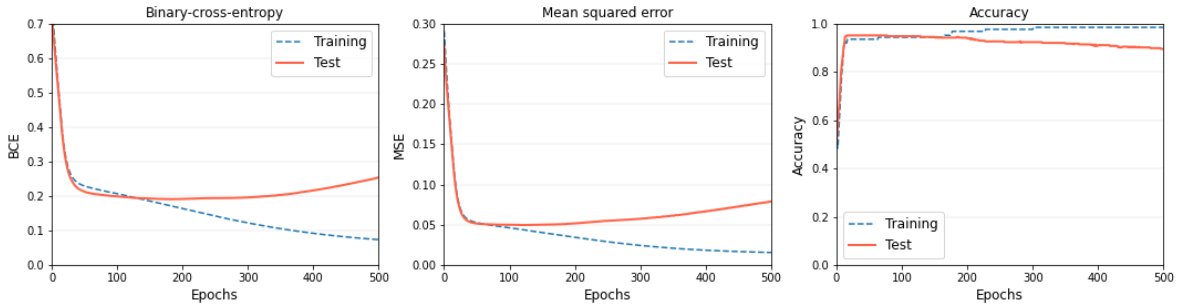


Figure 5: MONK3 learning curves for the best model without Tikhonov regularisation

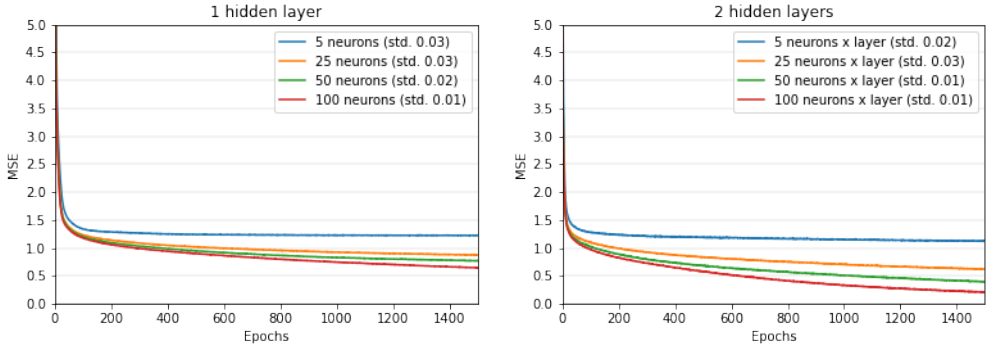


Figure 6: Screening phase of structural stabilisation approach

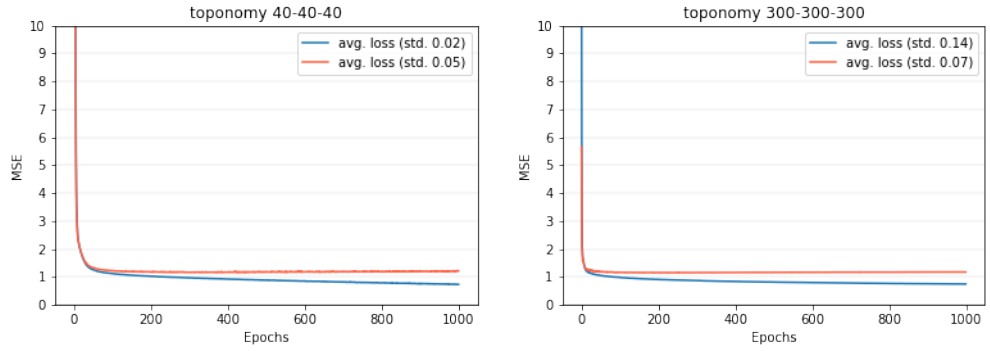


Figure 7: Screening phase of structural stabilisation approach

Neural Network - Structural stabilization with learning decay						
	n_neu	l_rate	mom	l_decay	MEE TR	MEE VAL
Search range (Time: 20H)	300	[0.001,0.009]	0.9	[1eˆ-2, 9eˆ-4]	-	-
Top 3 results	300	0.014	0.9	6eˆ-3	1.043	1.211
	300	0.012	0.9	6eˆ-3	1.057	1.215
	300	0.013	0.9	6eˆ-3	1.052	1.217

Table 13: Best gridsearch results for structural stabilization with learning decay approach

Neural Network with Tikhonov regularization						
	n_neu	l_rate	mom	lam2	MEE TR	MEE VAL
Search range (Time: 12H)	(40,50,75)	[0.0001,0.002]	(0.7,0.9)	[0,0.005]	-	-
Top 3 results	75 (x3)	0.0017	0.9	0.0040	0.747	1.112
	75 (x3)	0.0013	0.9	0.0040	0.751	1.120
	75 (x3)	0.0017	0.9	0.0025	0.720	1.125

Table 14: Best gridsearch results for Tikhonov regularized NN

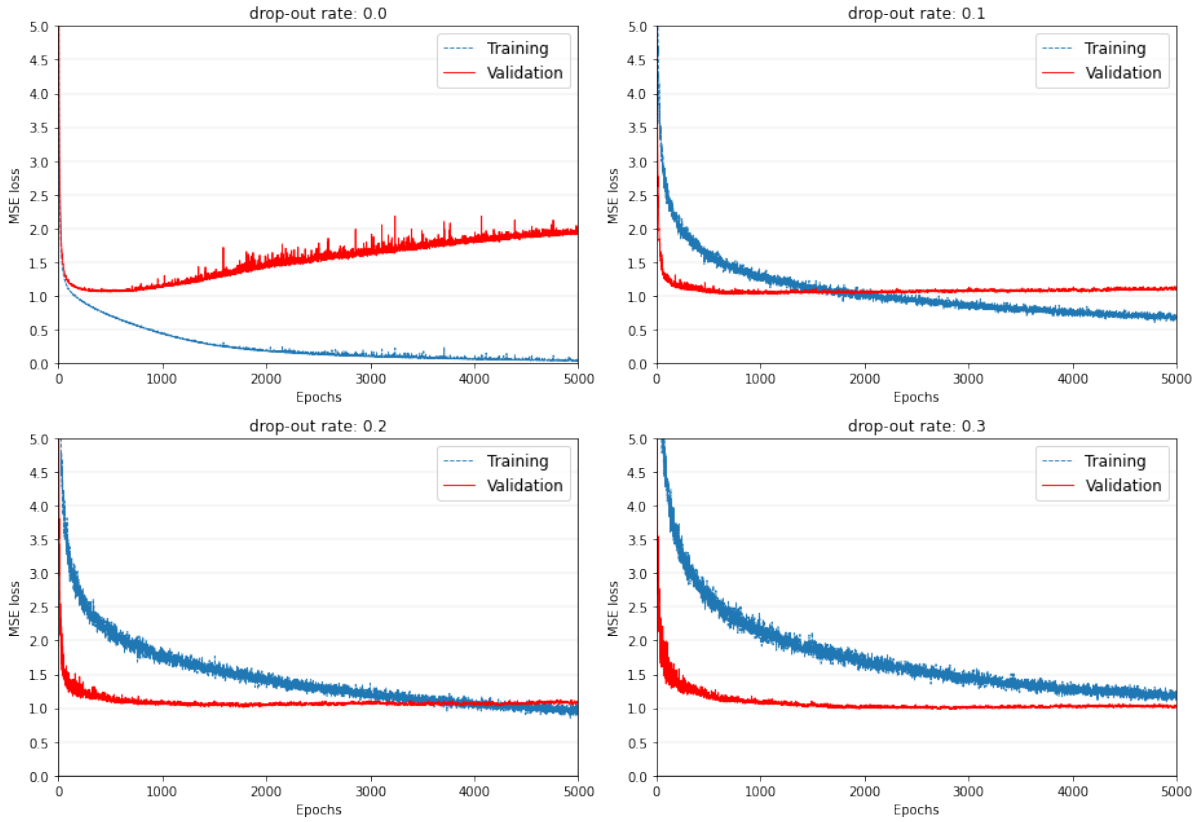


Figure 8: Learning curves for different values of Dropout parameter on a 100-100-100 neural network with 0.0005 learning rate and no Tikhonov regularisation.

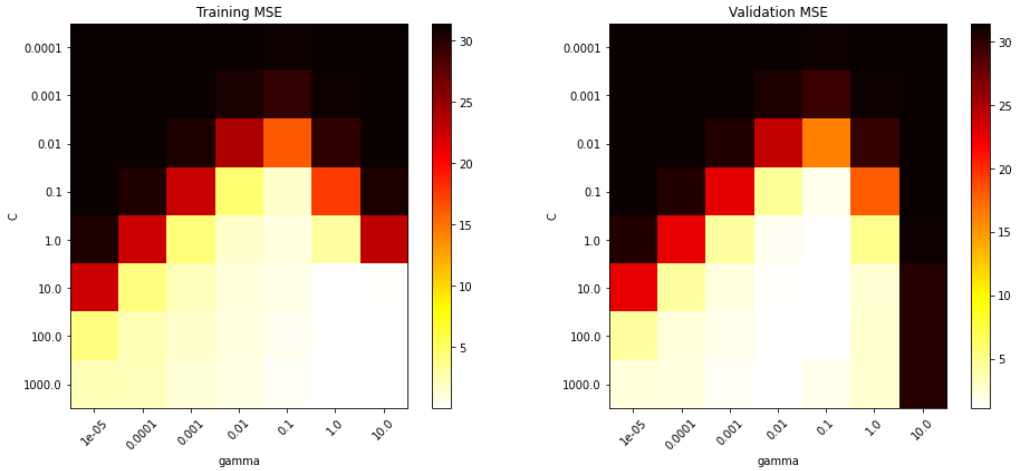


Figure 9: Training and 5-fold cross validated MSE with different values of C and γ

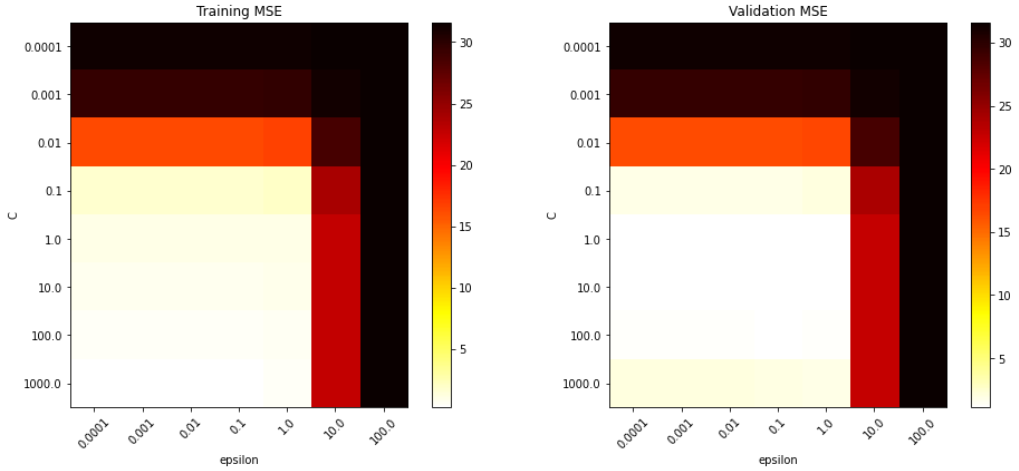


Figure 10: Training and 5-fold cross validated MSE with different values of and C.

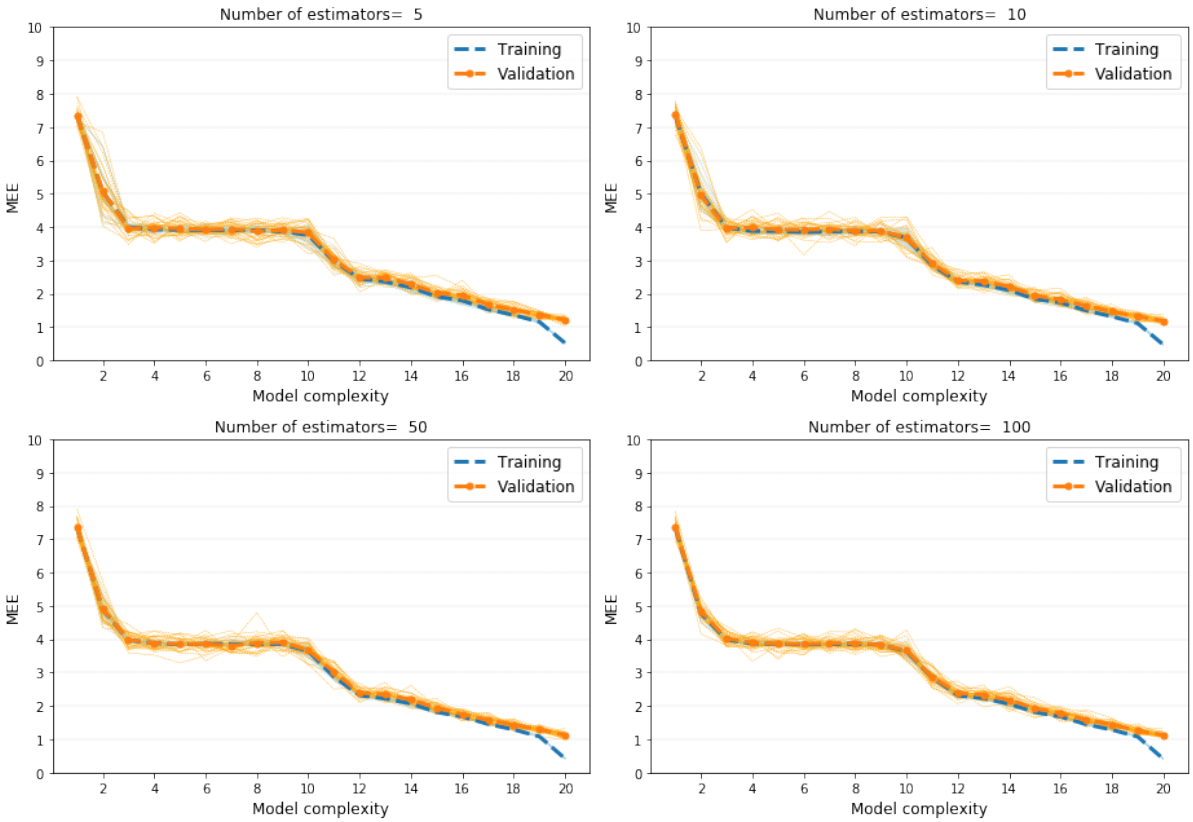


Figure 11: Matrix of RandomForest model behaviour by complexity scale and by increasing number of estimators

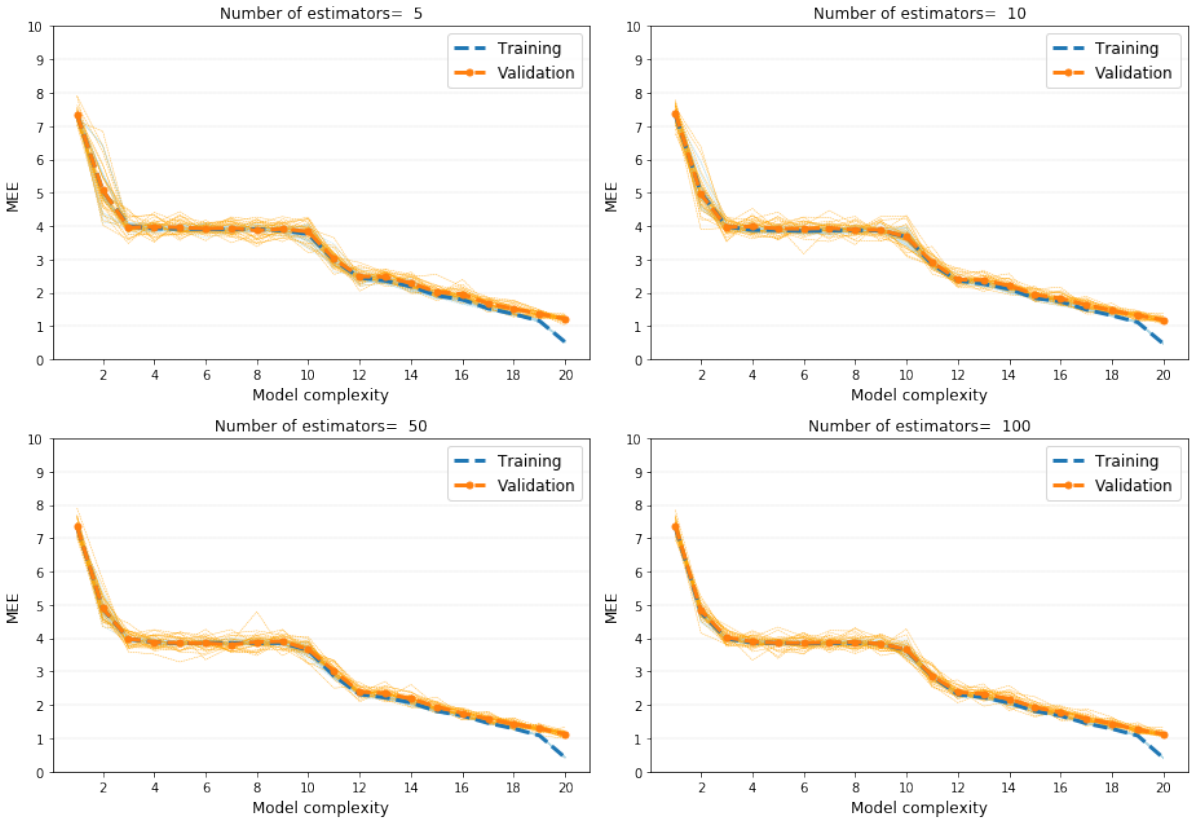


Figure 12: Matrix of AdaBoost with Decision Tree model behaviour by complexity scale and by increasing number of estimators