Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 9–18

Procedia Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Cross-domain Meta-learning for Time-series Forecasting

Abbas Raza Ali[a,*], Bogdan Gabrys[b], Marcin Budka[a]

[a]*Deparment of Computing and Informatics, Faculty of Science and Technology, Bournemouth University, United Kingdom*
[b]*Advanced Analytics Institute, University of Technology Sydney, Australia*

Abstract

There are many algorithms that can be used for the time-series forecasting problem, ranging from simple (e.g. Moving Average) to sophisticated Machine Learning approaches (e.g. Neural Networks). Most of these algorithms require a number of user-defined parameters to be specified, leading to exponential explosion of the space of potential solutions. Since the trial-and-error approach to finding a good algorithm for solving a given problem is typically intractable, researchers and practitioners need to resort to a more intelligent search strategy, with one option being to constraint the search space using past experience – an approach known as Meta-learning. Although potentially attractive, Meta-learning comes with its own challenges. Gathering a sufficient number of Meta-examples, which in turn requires collecting and processing multiple datasets from each problem domain under consideration is perhaps the most prominent issue. In this paper, we are investigating the situations in which the use of additional data can improve performance of a Meta-learning system, with focus on cross-domain transfer of Meta-knowledge. A similarity-based cluster analysis of Meta-features has also been performed in an attempt to discover homogeneous groups of time-series with respect to Meta-learning performance. Although the experiments revealed limited room for improvement over the overall best base-learner, the Meta-learning approach turned out to be a safe choice, minimizing the risk of selecting the least appropriate base-learner.

Keywords: Forecasting; Transfer Learning; Cross-domain Meta-learning, Meta-knowledge; Time-series; Cluster Analysis

1. Introduction

There exists a plethora of Machine Learning algorithms that can be used for the time-series forecasting problem. In order to use most of them, a number of user-defined parameters need to be specified, leading to exponential explosion of the space of potential solutions. One approach to finding a good algorithm for a given problem is trial-and-error, which is typically intractable. Hence researchers and practitioners need to resort to a more intelligent search strategy, with one option being to constraint the search space using past experience – an approach known as Meta-learning [16].

Learning at the base-level gathers experience within a specific problem, while Meta-learning is concerned with accumulating knowledge over several learning episodes [14]. The knowledge which is used by Meta-learner is ac-

* Corresponding author.
E-mail address: abbas.raza.ali@gmail.com

quired from previously solved problems, and each problem is characterized by several Meta-features. Meta-features combined with performance of learning algorithms (which act as targets or labels), form a Meta-knowledge database.

The purpose of this study is to evaluate the situations in which the use of additional data can improve performance of a Meta-learning system for time-series forecasting, with focus on cross-domain Meta-knowledge transfer. The culmination of previous work on Meta-learning within the group [17, 18] was the use of proposed approaches and data from NN3 and NN5 competitions[1] in [17], which supplemented by the available NN-GC1 data has led to our winning of the NN-GC1 international forecasting competition. In [18] it was stipulated (though not verified by any further analysis) that a particularly good predictive performance resulting from deploying the Meta-learning approach and a Meta-ranking algorithm on the NNGC-C dataset (monthly interval) and NNGC-E (daily interval) might have been due to additional use of the NN3 and NN5 (111 daily series each) datasets for generating Meta-knowledge and training Meta-learners. In this paper, we have concentrated on attempting to understand if the use of additional time-series from NN3/NN5 competitions have been the main reason behind the performance of our Meta-learner on series NNGC-C and NNGC-E of the NN-GC1 competition. Through an extended analysis of the results describing for which NN-GC1 time-series the Meta-learning performs best or worst, we have attempted to answer a more general question of when and under what circumstances the use of datasets from other domains (NN3/NN5 competitions in the current context) could be beneficial for recommending well performing forecasting methods for a problem at hand.

The remainder of this paper is organized as follows. Section 2 outlines the problem statement. High-level methodology applicable to cross-domain Meta-learning systems is presented in Section 3. The experimentation environment containing key components of a Meta-learning system are covered in Section 4. The results of various experiments are presented in Section 5. Section 6 focuses on cluster analysis of the Meta-examples in an attempt to identify homogeneous groups of time-series with respect to Meta-learning performance. The paper is concluded in Section 7.

2. Problem Statement

It has been widely accepted that gathering additional training data typically leads to an improvement in performance of predictive models. This stems from both theoretical arguments related to the bias-variance dilemma [6], where collecting more data is a way of avoiding the trade-off in the first place, and empirical evidence [12] suggesting that "it is not who has the best algorithm that wins – it is who has the most data", at least in some application areas.

In this study we evaluate the above hypothesis in a context of a Meta-learning system for time series forecasting, with a particular focus on using additional training data from other domains. The rationale behind such a scenario is that data from other domains is often much easier to find.

In particular, we perform an investigation to find whether the reason for good performance of a Meta-model built using the NN3 and NN5 datasets when applied to the NNGC-C and NNGC-E datasets was the similar frequency of observation recording or time-series sample-rate Meta-feature dominance over rest of the predictors. Investigation is also required to understand if increasing the size of training dataset could enhance the overall Meta-learning prediction accuracy. Additionally, it leads to another problem of not finding useful patterns from the cross-domain data, for example, NN3 and NN5 contain 222 instances which is a relatively small number with a lot of variations in the data points. It raises the question whether adding data from only the same domain can enhance Meta-level accuracy.

3. Methodology

In order to investigate the hypothesis stated in Section 2, an experimentation environment has been established. Figure 1 provides a high-level overview of the Meta-learning system setup for this work, which also includes cluster analysis on the Meta-database as mentioned earlier. The results of Meta-learning and cluster analysis have been analyzed to extract some evidence that could provide answers to the questions raised in the above section.

The Meta-learning workflow within the system is divided into two phases: i) Meta-modelling, and ii) Meta-ranking. For Meta-modelling two datasets from different domains are used: i) NN3 drawn from homogeneous population of monthly empirical business time-series, and ii) NN5 with history of cash machine transactions. Several Meta-features

[1] http://www.neural-forecasting-competition.com/

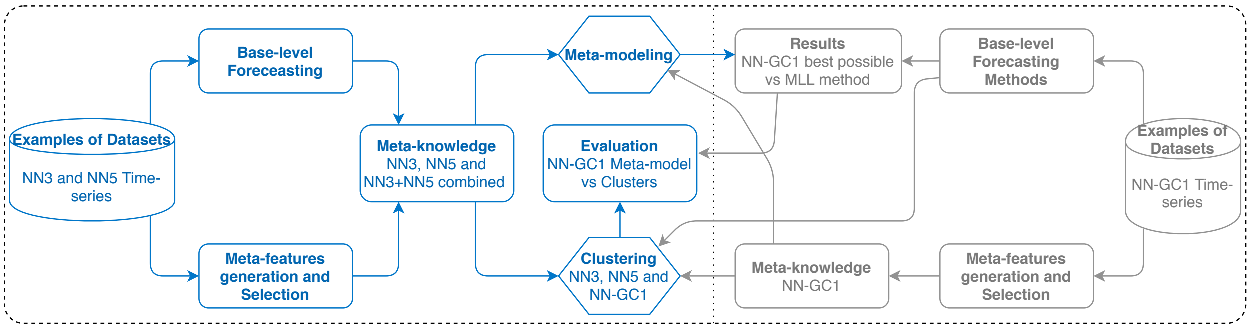
Base-level Foreceasting
Examples of Datasets
NN3 and NN5 Time-series
Meta-features generation and Selection
Meta-knowledge NN3, NN5 and NN3+NN5 combined
Meta-modeling
Evaluation NN-GC1 Meta-model vs Clusters
Clustering NN3, NN5 and NN-GC1
Results NN-GC1 best possible vs MLL method
Base-level Forecasting Methods
Examples of Datasets
NN-GC1 Time-series
Meta-knowledge NN-GC1
Meta-features generation and Selection

Fig. 1. Methodology of Cross-domain Meta-learning

and base-learner performance measures are computed from these datasets. The performances obtained from the base-learners are associated with the Meta-features extracted from each time-series to build a Meta-database for each dataset in separation as well as for a combined NN3+NN5 dataset. There are three different Meta-models trained against the resultant Meta-database: (1) Neural Network, (2) Decision Tree and (3) Support Vector Machine.

Within the Meta-ranking phase, the three Meta-models have been evaluated against six subsets of NN-GC1 which are from a different domain (i.e. transportation) than both NN3 and NN5. Furthermore, NN-GC1 has different observation sampling rates. The same Meta-features which are used in Meta-modelling phase, have been extracted from NN-GC1 for the purpose of Meta-ranking. The Meta-models, that are trained on NN3 and NN5, are used to estimate the most appropriate forecasting method for the Meta-examples of NN-GC1. These estimates are evaluated against the best possible forecasting method which is computed by independently and exhaustively evaluating all base-learners on NN-GC1 as depicted in the right-side of Figure 1.

Apart from Meta-modelling, cluster analysis has also been performed on three different combinations of Meta-datasets including NN3 versus NN-GC1, NN5 versus NN-GC1 and NN3+NN5 versus NN-GC1. A hierarchical clustering approach was applied with different linkage methods and distance similarity measures to extract the most interpretable cluster structures.

4. Experimentation Environment

The experimental environment comprising all the key components of a Meta-learning system has been setup to perform extensive set of experiments. The base-level forecasting algorithms and Meta-features used in this work are taken from [17] to enable comparative analysis.

4.1. Examples of Datasets

The Examples of Datasets (EoD) is a repository of usually large number of datasets from various domains. In this paper, the EoD consists of 222 univariate time-series from two different sources, NN3 and NN5 [10] competitions. Each data-source contains 111 series, with NN3 dataset consisting of monthly business observations and NN5 consisting of daily financial observations. These two data-sources have been used for training of the Meta-model used in further experiments.

The Meta-model has been tested on 6 NN-GC1 [10] competition subsets consisting of 66 univariate time-series in total. Each subset consists of 11 series with different frequency of observations and prediction horizon. Table 1 summarizes the basic properties of all datasets used in this study.

4.2. Base-level Forecasting Methods (Base-learners)

Performance of five base-level forecasting methods has been estimated for each of the time-series. Those algorithms vary from simple (Moving Average and Single Exponential Smoothing) to more complex (Automatic Box-Jenkins, Structural and Neural Networks). These algorithms were evaluated using SMAPE meaus which was averaged over the time-series and its standard deviation. The evaluation protocol consisted of training the models on first 75% of the

Table 1. NN3, NN5 and NN-GC1 datasets

Datasets	Series	Observations	Frequency	Horizon
NN3 and NN5 datasets used for Meta-Modelling				
NN3	111	52-126	Monthly	18
NN5	111	735	Daily	56
NN-GC1 datasets used for model evaluation				
NNGC-A	11	23-37	Yearly	0
NNGC-B	11	31-148	Quarterly	4
NNGC-C	11	48-228	Monthly	12
NNGC-D	11	527-1181	Weekly	52
NNGC-E	11	377-747	Daily	7
NNGC-F	11	902-1742	Hourly	24

series and then applying it to the remaining 25%. Several R [21] libraries have been used to compute the performances, where the configuration of the algorithms is given in the following subsections.

4.2.1. *Simple time-series Algorithms*

Moving Average (MA) is a simple time-series method where the arithmetic mean of the last k observations has been computed iteratively. The optimal value of k is selected using grid-search from 3 to 24 where the step size is 3. At each value of k, mean squared error (MSE) has been calculated on validation-set and the k is selected where the error value is lowest.

4.2.2. *Complex time-series Algorithms*

Three complex time-series algorithms are also included:

1. Autoregressive Integrated Moving Average Models (ARIMA) [8], where the configuration used in this work was obtained by performing a grid-search over possible models within the first and second differences with starting stepwise value of 1 [15]. The lag value that produced the lowest MSE on the validation set has been automatically selected. The reason for selecting maximum second-order difference as described in [17] is that the data usually only involves non-stationarity at maximum second-level.

2. Structural technique, which is a linear state-space model for univariate time-series based on various components of the series such as trends etc. [20]. The maximum likelihood estimates of the local level model is used to get the time-varying slope dynamics. The structural technique produces fitted Kalman filter and smoother [23].

3. Feed-forward Neural Network with a single hidden layer containing 12 neurons and up to 12-observations lag, to reflect weekly or yearly seasonality for NN5 and NN3 respectively. The predictions were averaged over ten randomly initialised networks to obtain the forecasts.

The summary of various parameters that are used by the base-learning forecasting methods and their performances have been depicted in Table 2. As it can be seen, the Moving Average and ARIMA performed best on average for NN3 and NN5 respectively, with MA being the top-performer for 41% time-series in NN3 and ARIMA outperforming other methods for 72% NN5 time-series. There is not much difference in overall SMAPEs and standard deviations of the remaining four algorithms. While the standard deviation of NN3 tends to be almost double that of NN5, the SMAPE for the former is roughly two times smaller. The reason of high standard deviation of NN3 dataset is most likely the length of the time-series, which are comparatively much shorter than NN5, making the models less stable.

4.3. *Meta-feature Generation*

There are three different groups of Meta-features extracted from univariate time-series, including descriptive statistics, frequency domain and auto-correlation features [17]. The features were computed using methods available in R [21]. Table 3 contains the list of features and their descriptions that have been extracted from the time-series.

4.3.1. *Descriptive Statistics*

The descriptive statistics have been computed on time-series detrended using polynomial regression as explained in [17]. Statistics that are computed using detrended series include standard deviation, skewness and kurtosis. Another

Table 2. Methods and their configurations that are used to compute performance measures

Configuration				NN3		NN5	
Methods	Parameter	Description	Value	SMAPE	Std Dev	SMAPE	Std Dev
MA	*k*	Number of observations	2-24	**15.68**	**14.74**	35.17	7.73
ARIMA	maxQ	Maximum number of order difference	2	18.83	15.88	**28.02**	**8.17**
Structural	type	Maximum likelihood estimates	level	17.57	15.35	36.09	9.05
NN	neurons and seasonality	No. of neurons in hidden layer and lag	12 and 12	17.05	13.56	34.89	7.37

feature, *trend*, has been calculated to measure the amount of variability of the time-series. The turning points and step changes of time-series have been computed as described in [22]. The turning points provide information of local minima or maxima within a series while a step change is detected when mean of the series is greater than twice the standard deviation at each observation of the series. The number of turning points and step changes have been cumulated within a series and normalized by the number of observations. Furthermore, the Durbin-Watson test and non-linearity measure have been calculated on polynomial regression of order three [17].

4.3.2. *Frequency Domain and Auto-correlations*

In frequency domain, two features of the Fast Fourier Transform have been extracted from the detrended time-series. These are the maximum value of power spectrum and the number of peaks greater than 60% of the maximum value. The maximum value of the power spectrum provides strength of the strongest seasonal or cyclic component, while the top 40% peaks in the power spectrum identify the number of times strong recurring components in a time-series [17] are found. There are five auto-correlation and partial auto-correlation features to capture information of stationarity and seasonality of the time-series. These correlations are computed for lags 1 and 2. Additionally, a lag of 12 has been introduced for NN3 dataset which consists of monthly readings, and partial auto-correlation of lag 7 for the NN5, which consists of daily transactions. The Meta-features are summarized in Table 3.

Table 3. List of Meta-features and their importance

Meta-features		NN3		NN5		NN3+NN5	
Feature	Description	Features	Imp.	Features	Imp.	Features	Imp.
std	Std dev of de-trended series	season	1.00	kurt	1.00	season	1.00
trend	std(series) / std(de-trended series)	turn	0.85	season	0.96	turn	0.76
skew	Skewness of series	acf2	0.79	trend	0.90	trend	0.76
turn	Turning points	trend	0.72	step	0.89	pacf2	0.73
kurt	Kurtosis of series	pacf2	0.72	pacf2	0.83	acf2	0.72
step	Step changes	kurt	0.70	nonlin	0.81	pacf1	0.71
length	Length of series	skew	0.70	turn	0.79	acf1	0.70
non-linear	Non-linearity measure	pacf1	0.70	maxSpec	0.79	kurt	0.68
maxSpec	Power spectrum: maximal value	acf1	0.67	std	0.79	nonlin	0.68
ff	No. of peaks not lower than 60% of the max	step	0.66	ff	0.76	skew	0.67
acf[1, 2] and pacf[1, 2]	Auto- and partial correlations at lags one and two	std	0.65	skew	0.76	std	0.66
season	Seasonality: pacf[12] for NN3, pacf[7] for NN5	maxSpec	0.65	acf1	0.74	step	0.62

4.4. *Meta-knowledge Preparation*

Meta-learning requires an extensive and diverse set of time-series to build a reliable Meta-database on a given problem domain. The Meta-knowledge is composed of Meta-features mapped with the performance measures of respective EoD. The performance measures used to evaluate five base-models is SMAPE, whereas the classifier with lowest SMAPE against every EoD is selected as target variable (class label) of the Meta-learner. The size of the Meta-database is varied for different experiments ranging from 111 to 288 instances with 16 Meta-features. In our experiments we have used three sets of features i.e. (1) the top three most important features, (2) features with importance greater than the mean importance, and (3) all 16 features, exploiting a Random Forest based feature scoring approach as described in [13].

The Meta-knowledge for both NN3 and NN5 are biased towards MA and ARIMA respectively, which leads to imbalanced dataset problem. In current scenario the imbalanced Meta-knowledge is producing biased classifiers that have a higher estimation accuracy for the majority classes, i.e., MA and ARIMA, but lower accuracy for the minority classes. This problem has been solved using Synthetic Minority Over-sampling TEchnique (SMOTE) which balances the dataset by over-sampling the minority classes. SMOTE synthetically generates more instances of the minority class hence broadening their decision regions [9].

4.5. Meta-Learning

The Meta-database contains inputs (Meta-features) and class labels (the best forecasting algorithm for each time-series). Three supervised learning algorithms have been used as Meta-learners: feed-forward Neural Network, Decision Tree and Support Vector Machine. At this stage, leave-one-out cross validation strategy has been adopted i.e. the Meta-learners are trained on all but one time-series from the respective dataset (NN3, NN5) and tested on the remaining one. The methods have been first evaluated and compared using Classification Accuracy:

1. Neural Network (NN) is used with six different number of neurons in the hidden layer $\in \{10, 15, 20, 25, 30, 35, 40\}$ and weight decay of 0.01.
2. Decision Tree C5.0 (DT) with trials and number of boosting iterations ranging from 1 to 100.
3. Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel and with width $\sigma \in \{0.05, 0.01, 0.1\}$ and $C \in \{30, 35, 40, 45, 50, 55, 60, 65, 70\}$.

From the above experiments the Meta-learner that outperformed all others have been chosen to predict the best forecasting method for datasets from different domains.

Table 4. Accuracies and SMAPES of various Meta-learners

Method	NN3	NN5	NN3+NN5
Top three variables			
NN	49.54 (20.98)	72.83 (8.63)	61.26 (14.06)
DT	52.25 (20.97)	72.97 (7.84)	**63.06 (13.18)**
SVM	54.05 (21.52)	74.77 (7.74)	56.76 (16.05)
Above the mean score			
NN	52.25 (20.06)	60.36 (12.46)	57.20 (16.61)
DT	53.15 (20.79)	74.77 (7.48)	62.61 (13.96)
SVM	49.02 (21.85)	**76.58 (7.38)**	59.00 (16.90)
All variables			
NN	**56.76 (17.47)**	71.17 (9.40)	62.16 (14.74)
DT	51.35 (21.87)	71.87 (9.13)	59.46 (15.87)
SVM	52.25 (22.43)	75.68 (7.82)	62.16 (15.70)

The overall accuracies and standard deviation of above Meta-learners are shown in Table 4. These accuracy estimates have been compared between predicted method recommended by Meta-learner and the best algorithm out of five candidate forecasting methods against each time-series. There were three different experiments at the Meta-level:

1. Using top three features as shown in Table 3. The SVM Meta-learner performed slightly better than Decision Tree on NN3 and NN5. Whereas Neural Network outperformed the remaining two Meta-learners on NN-GC1.
2. Using the features whose importance was greater than the mean of overall feature importance score. Decision Tree was found to be consistently dominating Meta-learner for all datasets.
3. Using all the features. SVM performed well for simpler time-series, NN3 and GC, while Decision Tree outperformed both SVM and NN for complex datasets including NN5 and Combined.

From the above experiments the Meta-learner that outperformed all others have been chosen to predict the best forecasting method for datasets from different domains.

4.6. Cluster Analysis

Hierarchical clustering has been performed on the Meta-database to further investigate whether any correlation exists between Meta-learner's accuracy and cluster structure. Specifically, various combinations of clustering methods and distance measures have been tested, including Ward [19], single, complete and average linkage paired with Euclidean and Manhattan distance against two cluster counts: 10 and 20.

5. Results

The Meta-models described in Section 4.5 have been applied to 66 time-series of the NN-GC1 dataset [11] (Table 1). The results are summarized in Table 5. The best forecasting method is Moving Average with average SMAPE of 16.0 whereas the Meta-learner achieved a small improvement. Also, the paired t-test on the results of 6 NN-GC1 series indicates little significant difference in SMAPE and standard deviation of Base- versus Meta-learner.

Table 5. SMAPE and standard deviation of NN-GC1 series

Dataset	**Base-learning**		**Meta-learning**	
NN-GC1	Method	SMAPE (StD.)	Method	SMAPE (StD.)
NNGC-A	Structural	**7.8 (4.7)**	SVM MLL(NN5)→GC [All features]	**7.8 (4.7)**
NNGC-B	Moving Average	**7.2 (3.6)**	DT MLL(NN5)→GC [Top 3 features]	7.2 (4.8)
NNGC-C	Neural Network	13.4 (9.5)	NN MLL(NN3)→GC [Above Average features]	**12.5 (9.8)**
NNGC-D	Moving Average	**9.5 (9.1)**	SVM MLL(NN3)→GC [All features]	10.0 (9.0)
NNGC-E	Moving Average	26.9 (20.8)	NN MLL(NN3)→GC [All features]	**26.6 (20.7)**
NNGC-F	Moving Average	60.1 (5.8)	NN MLL(NN3)→GC [All features]	**59.8 (5.6)**
Average	Moving Average	16.0	NN MLL(NN3)→GC [Above Average features]	**15.9**

The detailed results of NN-GC1 datasets on various combination of Meta-models as well as three different sets of predictors (based on feature importance) are given in Table 6. The average SMAPE of the best possible base-level forecasting algorithm has been compared with different combinations of Meta-models' average SMAPE that came out from various experimentations. In six out of nine cases Meta-learning(NN3)→GC (NN3 Meta-model on NN-GC1) came out as the best Meta-model whereas in remaining four cases combination of both NN3 and NN5 datasets Meta-learning(NN3+NN5)→GC outperformed the rest. However, by analyzing the average SMAPE of different experiments, it can be seen that NN Meta-learner performed reasonably well on the set of features whose importance is above average, followed by SVM and DT. At deeper level the six NN-GC1 showed that apart from a few cases, there was no significant difference between the best possible base-learning and Meta-learning SMAPE.

The individual NN-GC1 dataset was further analyzed for those series which are unable to show the minimum error at Meta-level. The analysis showed that 44% of the series, estimated by the Meta-learner, were ranked as second best followed by 24% series ranked as third best, whereas only 2% were ranked as the worst. Overall for 70% time-series the model achieved better than the average SMAPE. Small difference between the best overall base-learner (i.e. Moving Average) and the best possible base-learner, left very little room for improvement, contributing to a challenge of showing significant improvement of Meta-learning over base-learning. However, Meta-learning falls between the best possible and average base-learner and can hence be used for recommendation of a predictive algorithm effectively minimizing the probability that a bad predictor will be selected.

Figure 2 shows histograms with the number of times a particular method performed best at base and Meta-level for NN-GC1 time-series. The NNGC-A, NNGC-B, NNGC-C and NNGC-F are showing the mixed base-level class distribution where NNGC-D and NNGC-E are biased towards MA and NN respectively. However, for NNGC-D and NNGC-E datasets Meta-learning has recommended ARIMA for most of the time-series.

6. Analysis

The performance of the Meta-learning system is further investigated for three subsets of Meta-features, formed based on their importance as described in Section 4.4. There was not much accuracy variation among these three

Table 6. SMAPE (and standard deviation) of NN-GC1 series

Method	NNGC-A	NNGC-B	NNGC-C	NNGC-D	NNGC-E	NNGC-F	Average
Moving Average	12.8 (9.4)	**7.2 (3.6)**	14.6 (11.9)	**9.5 (9.1)**	**26.9 (20.8)**	**60.1 (5.8)**	**16.0**
Arima	10.1 (6.9)	8.6 (4.7)	16.2 (19.1)	14.0 (9.7)	42.4 (52.0)	62.2 (6.5)	21.0
Structural	**7.8 (4.7)**	7.6 (4.7)	24.1 (25.8)	14.1 (9.5)	30.4 (23.3)	84.3 (16.1)	21.0
Neural Network	13.1 (6.7)	14.9 (6.9)	**13.4 (9.5)**	15.8 (12.6)	39.2 (30.1)	61.4 (5.7)	19.1
Base Learning (Best Possible)	6.5 (4.6)	6.0 (3.6)	11.9 (9.8)	9.4 (8.9)	25.5 (20.0)	59.1 (5.8)	14.3
Meta-learning on top 3 Meta-features							
Neural Network Meta-Model							
MLL(NN3)→GC	11.3 (8.0)	8.1 (3.9)	14.3 (11.6)	12.4 (9.9)	28.6 (23.0)	61.8 (6.5)	16.6
MLL(NN5)→GC	9.9 (6.7)	8.3 (5.4)	15.3 (12.3)	13.3 (8.9)	43.6 (51.2)	75.1 (12.4)	21.9
MLL(NN3+NN5)→GC	11.3 (8.3)	7.8 (4.5)	14.8 (12.2)	12.6 (9.9)	40.5 (47.9)	63.7 (9.6)	20.3
Decision Trees Meta-Model							
MLL(NN3)→GC	12.6 (8.1)	7.4 (3.8)	18.4 (21.4)	10.4 (9.3)	28.0 (22.2)	63.5 (9.3)	17.9
MLL(NN5)→GC	10.1 (7.2)	**7.2 (4.8)**	15.2 (12.4)	10.2 (9.0)	36.8 (47.3)	61.8 (6.6)	19.0
MLL(NN3+NN5)→GC	11.5 (7.5)	7.9 (4.4)	17.5 (19.1)	10.8 (9.5)	28.4 (23.0)	63.9 (9.5)	17.8
Support Vector Machines Meta-Model							
MLL(NN3)→GC	12.6 (8.1)	8.3 (5.7)	14.4 (11.6)	**10.0 (9.0)**	27.8 (21.5)	60.4 (5.9)	16.3
MLL(NN5)→GC	12.0 (8.2)	8.1 (4.7)	16.8 (19.2)	11.6 (9.3)	41.6 (51.8)	61.9 (6.9)	21.0
MLL(NN3+NN5)→GC	11.1 (8.3)	7.9 (4.4)	14.0 (11.7)	10.3 (9.2)	27.3 (22.4)	60.5 (6.2)	16.1
Meta-learning on Meta-features whose importance is greater than mean							
Neural Network Meta-Model							
MLL(NN3)→GC	10.0 (6.1)	7.7 (4.6)	**12.5 (9.8)**	12.2 (9.4)	28.9 (22.8)	60.9 (5.8)	**15.9**
MLL(NN5)→GC	9.0 (7.6)	8.4 (5.4)	23.6 (26.1)	14.3 (10.3)	30.2 (23.4)	81.5 (16.8)	21.4
MLL(NN3+NN5)→GC	12.5 (8.1)	9.1 (7.3)	13.0 (10.0)	12.0 (12.1)	42.1 (47.9)	64.1 (8.3)	20.5
Decision Trees Meta-Model							
MLL(NN3)→GC	12.1 (7.8)	8.5 (5.8)	13.4 (10.4)	10.4 (9.3)	27.5 (21.8)	61.2 (6.0)	16.2
MLL(NN5)→GC	10.1 (7.2)	7.4 (4.7)	15.2 (12.4)	10.2 (9.0)	36.8 (47.3)	61.8 (6.6)	19.1
MLL(NN3+NN5)→GC	11.8 (8.3)	8.0 (4.5)	13.2 (10.1)	12.1 (12.9)	32.8 (25.0)	61.1 (6.3)	17.2
Support Vector Machines Meta-Model							
MLL(NN3)→GC	12.9 (7.8)	8.3 (5.7)	13.4 (10.4)	10.0 (9.0)	27.7 (22.2)	60.4 (5.9)	16.1
MLL(NN5)→GC	10.6 (7.0)	7.7 (4.7)	16.4 (19.0)	14.0 (9.7)	42.4 (52.0)	62.2 (6.5)	21.0
MLL(NN3+NN5)→GC	12.8 (9.4)	7.9 (4.4)	16.1 (18.5)	10.7 (9.1)	36.6 (47.6)	61.0 (6.3)	20.0
Meta-learning on all the Meta-features							
Neural Network Meta-Model							
MLL(NN3)→GC	10.5 (7.3)	7.6 (3.9)	17.0 (19.2)	12.1 (10.7)	**26.6 (20.7)**	60.3 (5.8)	16.8
MLL(NN5)→GC	8.4 (5.3)	9.2 (5.1)	23.7 (26.0)	14.4 (10.3)	30.6 (23.2)	84.3 (16.1)	21.4
MLL(NN3+NN5)→GC	10.6 (7.9)	7.4 (4.4)	17.1 (19.0)	11.6 (10.7)	31.8 (25.0)	61.6 (6.7)	17.8
Decision Trees Meta-Model							
MLL(NN3)→GC	12.7 (8.0)	8.4 (5.7)	13.7 (11.3)	11.2 (10.6)	36.7 (47.4)	61.2 (6.0)	19.4
MLL(NN5)→GC	10.3 (7.8)	8.3 (4.4)	15.0 (12.6)	10.2 (8.5)	36.8 (47.3)	61.2 (6.5)	19.1
MLL(NN3+NN5)→GC	11.3 (8.0)	8.5 (4.5)	14.0 (11.3)	10.3 (9.3)	36.9 (47.4)	61.0 (6.3)	19.1
Support Vector Machines Meta-Model							
MLL(NN3)→GC	12.5 (8.3)	8.1 (5.0)	14.8 (11.8)	**10.0 (9.0)**	26.9 (20.8)	**59.8 (5.6)**	16.0
MLL(NN5)→GC	**7.8 (4.7)**	7.6 (4.7)	24.1 (25.8)	14.1 (9.5)	30.4 (23.3)	84.3 (16.1)	21.0
MLL(NN3+NN5)→GC	11.0 (7.3)	7.6 (4.4)	16.3 (19.1)	10.5 (9.5)	36.3 (47.4)	61.3 (5.9)	19.7

subsets, which indicates that not all the Meta-features are contributing, as shown in Table 6. The Meta-knowledge can be enhanced by increasing the number of examples as one of the challenges in this work is scarce input data with large variations within different time-series of each dataset. In particular, the NN5 dataset contains only 111 time-series, and even within this small number there are a few subsets of series representing different trends and patterns, contributing to the difficulty of building a stable Meta-model as shown in Figure 3 (a).

In another experiment, the Meta-learner was built using both NN3 and NN5 datasets to assess the impact of additional Meta-knowledge instances. The improvement was however very modest which can be observed from MLL(NN3+NN5) combined results reported in Table 6. The reason is found in their cluster analysis where very few time-series are clustered together based on the similarity of features. Hence the Meta-model was unable to learn patterns from cross-domain time-series.

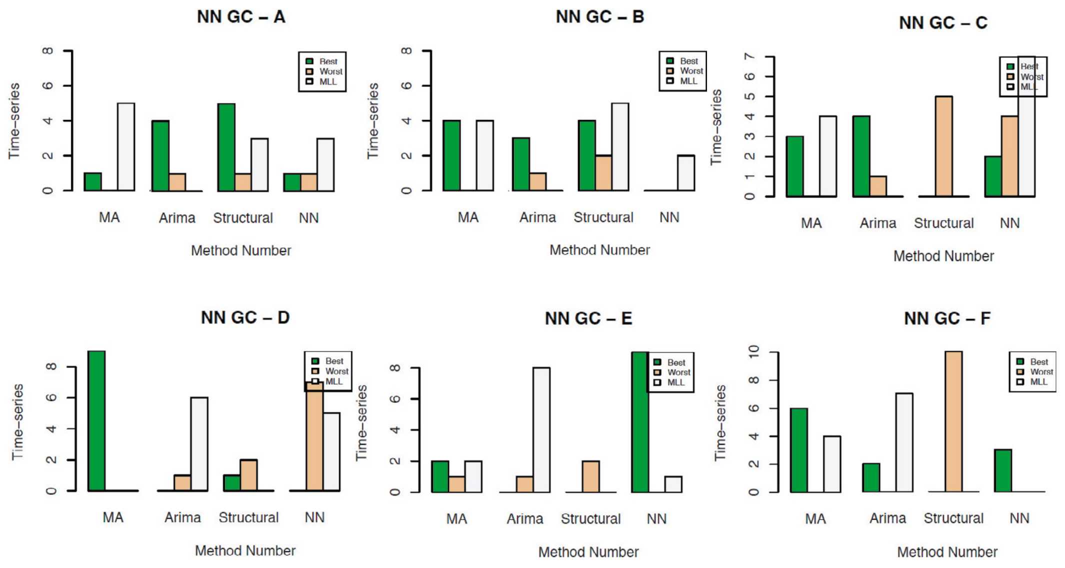

Fig. 2. Histogram showing number of times a particular method performs best for NN-GC1

Meta-learning performed reasonably well even in presence of imbalanced class distribution (for NN5 single base-level forecasting method was performing best for 72% of time-series) while applying Meta-model on NN-GC1. Even though NN3 dataset is found to be simpler (smooth, less noisy) than NN5, the overall Meta-level accuracy of NN5 is higher than NN3. The reason is that ARIMA came out as the best base-learning algorithm for 72% of the time-series which made this dataset imbalanced for Meta-learning. However, for NN3 dataset MA is the best algorithm for 41% of the time-series followed by the NN and Structural methods with more than 20% each. This suggests that Meta-learner worked well for imbalanced class distribution where, in most of the cases, Meta-learner selected majority class.

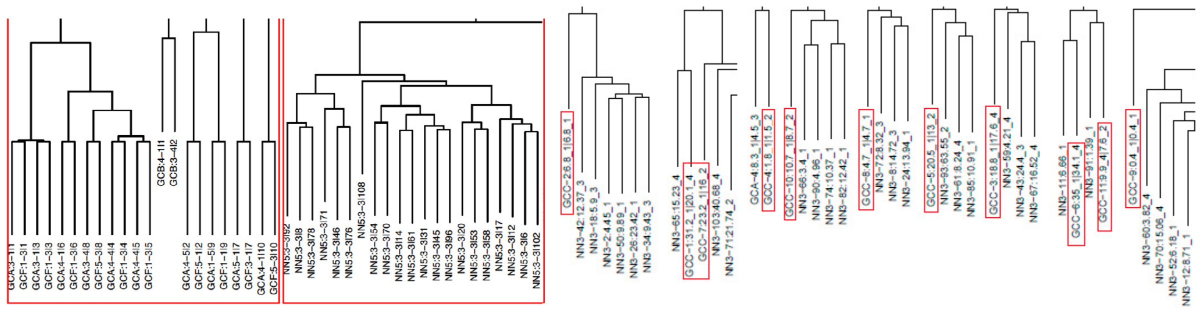

Fig. 3. (a) NN5 clustered together with NNGC dataset where the cluster cut over is at k = 10. The cluster labels can be interpreted as 'Dataset-TimeSeries:MLLSMAPE_MLLMethod—BestMethodSMAPE_BestMethod'; (b) NN3 clustered together with NNGC-C dataset where the cluster cut over is at k = 10. The cluster labels can be interpreted as 'Dataset:TimeSeries-MLLMethod—BestMethod'.

The final experiment was performed to analyse whether any correlation exists between high Meta-learning accuracies versus homogeneous clusters of Meta-examples. There were three different combinations of Meta-examples clustered with the NN-GC1 Meta-examples which included NN3, NN5 and combined NN3+NN5 data. The clusters of NN3 and NN5 are found to be heterogeneous since there were very few time-series clustered together. In Figure 3 (b) it can be observed that the base- and Meta-learner are the same in most of the clusters for NN3 and NNGC-C series. On contrary to this NN5 is not clustered with NNGC-E in most of the cases.

7. Conclusions

The key focus of this study was to investigate whether the use of additional training data from a different domain is beneficial for improving performance of the Meta-learner. The experiments confirmed a number of issues potentially hampering the performance, including a small number of EoDs available from each domain for Meta-modelling relative to the number of Meta-features, making the input space sparse, and very few instances in NN-GC1 with huge variations in the trends and distribution of its six different sub-datasets.

In consideration of the above challenges the Meta-learning demonstrated a performance level which can still be considered sufficient on cross-domain problem. In more than 90% of the cases, the recommended base-learner was the second or third best option, and there were only 2% of cases where the Meta-learning recommended the worst performing base methods for the respective time-series. Overall, for 70% of the time-series the SMAPE of the recommended algorithm is better than the average SMAPE of all base-learners. Hence while using Meta-learning in this setting might not always give the highest possible performance, it helps to alleviate the risk of selecting the worst model and tends to be a robust approach.

There are few key observations suggesting that the additional data was likely not the reason of better Meta-learning performance observed in NN3+NN5 Meta-model. In several instances the NN3 Meta-model performed better than the combined NN3+NN5 model. One of the reasons found from cluster analysis is that both NN3 and NN5 have very few Meta-examples that are similar to each other. Additionally, NN-GC1 is relatively more similar to NN3 than NN5 because based on the Meta-model recommendations NN3 time-series end up in different clusters in these two cases. Both NNGC-C and NNGC-E are clustered with NN3 whereas unexpectedly there were very few instances of NNGC-E and NN5 found in the same clusters. This is confirmed by the results of the Meta-learning experiments, where MLL(NN3) consistently performed well for NNGC-E. One potential reason for better Meta-level performance is the similar frequency of observation recording.

In considering several data related challenges, the performance of Meta-learning on cross-domain problem was satisfactory. Even though there was not much room for improvement for Meta-learner, it made its place between the best possible base-learner and MA (the best overall base-learner).

References

[1] D. O. Afolabi and S. Guan and K. L. Man and P. W. H. Wong and X. Zhao. Hierarchical Meta-Learning in Time Series Forecasting for Improved Interference-Less Machine Learning. *Symmetry*, vol. 9, pages 283, 2017.

[2] T. Chen, I. Goodfellow, and J. Shlens. Net2Net: Accelerating Learning via Knowledge Transfer. *ICLR 2016*, 2016.

[3] R. Fonseca and P. Gómez-Gil. Temporal validated meta-learning for long-term forecasting of chaotic time series using monte carlo cross-validation. *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, pages 353–367, Springer, 2014.

[4] S. Taieb, G. Bontempi, A. Atiya and A. Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert systems with applications*, 398:7067–7083, Elsevier, 2012.

[5] S. Crone, M. Hibon and K. Nikolopoulos. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting*, 273:635–660, Elsevier, 2011.

[6] C. M. Bishop. Neural networks for pattern recognition. *Oxford university press*, 1995.

[7] A. R. Ali and P. Lin. Demand-Driven Asset Reutilization Analytics. *ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference*, Stanford University, USA, 2014.

[8] G. E. P. Box and G. M. Jenkins. *Time series analysis*. 1970.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[10] S. Crone. *NN3, NN5 and NN-GC1 forecasting competitions [online]*, 2006-2008-2010.

[11] S. Crone. *NN-GC1 forecasting competition [online]* 2010.

[12] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. *In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[13] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recogn. Lett.*, 31(14):2225–2236, 10 2010.

[14] C. Giraud-Carrier. Meta-learning - a tutorial. *In Proceedings of the Seventh International Conference on Machine Learning and Applications, ICMLA '08*, San Diego, CA, USA, 2008.

[15] R. J. Hyndman and Y. Kh. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software*, 2008.

[16] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, pages 1–14, 2013.

[17] C. Lemke and B. Gabrys. Meta-learning for time-series forecasting and forecast combination. *Journal of Neurocomputing*, 73(10-12), 2010.

[18] C. Lemke and B. Gabrys. Meta-learning for time-series forecasting in the NN-GC1 competition. *Fuzzy Systems (FUZZ)*, pages 1–5, 7 2010.

[19] F. Murtagh and P. Legendre. Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of Classification*, 31(3):274–295, 2014.

[20] G. Petris and S. Petrone. *State space models in R. Journal of Statistical Software*, 41, 2011.

[21] R Development Core Team. R: *A language and environment for statistical computing*, 2008.

[22] C. Shah. Model selection in univariate time-series forecasting using discriminant analysis. *Intl. Journal of Forecasting*, 13(4):489–500, 1997.

[23] F. Tusell. Kalman Filtering in R. *Journal of Statistical Software*, 39, 2011.