# Meta-learning for financial forecasting

Thomas J. Delaney

## Contents

# 1 Introduction

The point of this document is to review our experiments with the meta-learning forecasting method developed by Pablo Montero-Manso et al (2018) [1] for the M4 competition [2]. The method was written in the programming language `R`, and all the programming and analysis outlined in this document was also performed in `R`.

First we will outline the objectives of these experiments. Then we will give some background on the M4 competition and the meta-learning forecasting method itself. Then we will outline the experiments performed, and the results obtained. The document will conclude with a description of the limitations of our experiments, and suggestions for further research.

# 2 Objectives

Our overall objective was to assess how useful the meta-learning forecasting method is in the context of financial market forecasting. In order to do this, we did the following:

1. We replicated the findings reported by Montero-Manso et al in their submission to the M4 competition. We focussed on the financial time series provided for the M4 competition in particular.

2. We obtained time series from real financial contexts, equity and fixed income prices from the early $21^{\text{st}}$ century. We formatted these data to match the M4 competition data.

3. We applied our meta-learning forecasting method trained on financial data only to an augmented test set that included the real financial data.

4. We compared the performance of the meta-learning method to the performances of the simpler forecasting methods used in the meta-learning framework.

We also aimed to find out if an improvement in the model's performance on our real financial data could be achieved by training the model on data from a non-financial context (aka. *transfer learning*). The authors of the meta-learning method provided a github repository containing the meta-learning model trained on the full M4 competition data [1]. We applied this fully trained model to a test set consisting of just the real financial data. In order to achieve our secondary aim, we compared the performance of the fully trained meta-learning method to the performance of the financially trained meta-learning method on our financial data.

# 3 Background

The M4 competition is a time series forecasting competition in which a dataset of time series is provided, and researchers and developers are challenged to develop the most

[1] For details on downloading the trained model, see: `https://github.com/robjhyndman/M4metalearning/blob/master/docs/M4_reprod.md`

accurate forecasting method for these series. No driving series are provided for any of the time series, so all of the methods developed are univariate. The M4 competition is the fourth in a series of forecasting competitions, and took place in 2018. The dataset consisted of $100000$ time series from demographic, financial, industrial, macro and micro economic, and miscellaneous contexts. All of the time series were recorded either hourly, daily, weekly, monthly, quarterly, or yearly. The methods are judged by an error metric on their mean forecast, and another error metric on their prediction intervals. For more information on the M4 competition, see [2].

The meta-learning forecasting method that we use in these experiments came second in the M4 competition. The meta-learning forecasting method is an example of an *ensemble forecasting* method. An ensemble forecasting method combines multiple forecasts to produce another forecast more accurate than its constituents. The meta-learning forecasting method learns a set of weights for combining the forecasts and prediction intervals of simpler forecasting methods in a way that gives a minimal forecasting error across all of the time-series provided. For more details on the method see [3], section $8.3.2$.

# 4 Methods

## 4.1 Error Measures

For the M4 competition, a composite error measure is used. This is a combination of the symmetric mean absolute percentage error (sMAPE), and the mean absolute scaled error (MASE) for seasonal series.

Note that when comparing the performance of the meta-learning method to other methods, we just used the median absolute percentage error (MdAPE) as this measure is more robust to outliers.

### 4.1.1 Symmetric Mean Absolute Percentage Error (sMAPE)

The forumula for calculating the sMAPE is

$$sMAPE = \frac{1}{h}\sum_{t=1}^{h}\frac{2|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|} \tag{1}$$

where $h$ is the forecast horizon, $Y_t$ is the value of the time series during the forecasted period, and $\hat{Y}_t$ is the forecasted value for $Y_t$.

The sMAPE was invented to solve the problem of the standard MAPE penalising over-estimations in the forecast ($\hat{Y}_t > Y_t$) more severely than under-estimations.

### 4.1.2 Mean Absolute Scaled Error

The formula for calculating the MASE for season series is

$$MASE = \frac{1}{h}\frac{\sum_{t=1}^{h}|Y_t - \hat{Y}_t|}{\frac{1}{h-m}\sum_{t=m+1}^{h}|Y_t - Y_{t-m}|} \tag{2}$$

| Frequency | Number of Series |
|---|---|
| Hourly | 0 |
| Daily | 1559 |
| Weekly | 164 |
| Monthly | 10987 |
| Quarterly | 5305 |
| Yearly | 6519 |

Table 1: The number of series for each frequency in the financial time series.

where all variables are the same as in section 4.1.1, and $m$ is the frequency of the data (e.g. 12 for monthly). Note that the denominator is the mean absolute error of the one-step seasonal naive forecast

### 4.1.3 Overall weighted average (OWA)

The error measure used by the M4 competition to evaluate mean forecasts is an average of the *relative* sMAPE, and *relative* MASE, taken relative to the sMAPE or MASE of a seasonally adjusted naïve forecast. In other words, they divided the sMAPE of the given method by the sMAPE of the seasonally adjusted naïve method. They did the same for the MASE. Then they took an average of the two.

### 4.1.4 Mean Scaled Interval Score (MSIS)

To measure the accuracy of the prediction intervals, the M4 competition uses the MSIS:

$$MSIS = \frac{1}{h}\frac{\sum_{t=1}^{h}(U_t - L_t) + \frac{2}{a}(L_t - Y_t)\mathbf{1}\{Y_t < L_t\} + \frac{2}{a}(Y_t - U_t)\mathbf{1}\{Y_t > U_t\}}{\frac{1}{h-m}\sum_{t=m+1}^{h}|Y_t - Y_{t-m}|} \tag{3}$$

where $h$ is the forecast horizon, $U_t$ is the upper prediction interval, $L_t$ is the lower prediction interval, $Y_t$ is the value of the time series at time $t$, $a$ is a *significance level* and

$$\mathbf{1}\{\text{condition}\} = \begin{cases} 0 \text{ if condition is false,} \\ 1 \text{ if condition is true} \end{cases} \tag{4}$$

is the *indicator* function. The first term in the numerator of (3) rewards a tighter prediction interval with a smaller error. The second and third terms penalise any elements of the time series that fall outside of the prediction interval.

## 4.2 Finance only model

### 4.2.1 Training the model

To train a meta-learning model specificially for financial data, we extracted the financial time series from the M4 database. There were $24{,}534$ series in total. See table 1 for a breakdown of the frequencies of these series.

We divided this dataset up into a training set of $n_{training} = 20{,}000$ series and a test set of $n_{test} = 4534$ series. In order to train the meta-learning model we did the following:

1. We divided each individual series in the training set into training and holdout series, where the holdout series is taken from the end of the original series, and is the length of the forecast horizon.

2. We then calculated forecasts for each series in the training set, using each of the 9 forecasting methods included in the meta-learning framework. Each of these methods were implemented using the `forecast` library in `R`.

3. The error measures used by the M4 competition were calculated for each of the 9 forecasts for each series.

4. The 42 features used in the meta-learning model were extracted from each series.

5. By training multiple `xgboost` random forests on our dataset of features and series we found the ideal hyperparameters for an `xgboost` random forest.

6. Using these hyperparameters we trained an `xgboost` random forest model, using the M4 competition error measurement as an objective function to minimise. This `xgboost` random forest is the trained meta-learning model.

### 4.2.2 Using the model

In order to use the model, we did the following:

1. We divided each indivudual series in the test set into training and holdout series. Similarly to the set of training series, the holdout series is taken from the end of the original series, and is the length of the forecast horizon.

2. We calculated forecasts for each of the series in the test set, using each of the 9 forecasting methods included in the meta-learning framework.

3. The 42 features used in the meta-learning model were extracted from each series in the test set.

4. Given the test set of $n_{test}$ series and the features extracted from those test series, the trained meta-learning model returned $n_{test}$ vectors of 9 real numbers between $0$ and $1$, which sum to one. These 9 numbers are used to create a weighted sum of the 9 forecasts calculated in step 2. This weighted sum of forecasts is the forecast of the meta-learning forecasting method.

To evalute the performance of the model on our real financial data, we formatted the data similarly to the M4 data, and we simply concatenated our data with the test set. We had *instert figure here* weekly time series, and *insert figure here* daily series.

## 4.3 Fully trained model

To apply the fully trained meta-learning model to the test data, we downloaded the model from the repository mentioned in section 2. We then followed the same steps as those outlined in section 4.2.2, using the fully trained model instead of the finance only model.

| Forecasting Method | M4 Finance MdAPE | Real Finance MdAPE |
|---|---|---|
| Meta-learning | 0.093 | 0.016 |
| Meta-learning (finance only) | 0.123 | 0.015 |
| Random Walk with Drift | 0.147 | 0.013 |
| ARIMA | 0.137 | 0.014 |
| ETS | 0.138 | 0.015 |
| Neural Network | 0.176 | 0.024 |
| TBATS | 0.137 | 0.015 |
| STL | 0.232 | 0.326 |
| Theta | 0.138 | 0.014 |
| Naïve | 0.151 | 0.015 |
| Seasonal naïve | 0.163 | 0.015 |

Table 2: Median absolute percentage error of various forecasting methods on the M4 financial time series and the real financial time series. We used the meta-learning method trained using all of the M4 time series and another meta-learning method trained using only a subset of the financial time series from the M4 time series. Note that most of the forecasting methods perform better on the real financial dataset compared to the M4 financial dataset. But while the best performing method on the M4 financial dataset was the fully trained meta-learning method, the best performing method on the real financial dataset is the random walk with drift.

# 5 Results

## 5.1 Comparison to simpler methods

We first transformed our real financial data into the M4 competition format and applied the fully trained meta-learning model to these data. The forecast horizons were determined by the frequency of the data, 14 for daily, 13 for weekly. We compared the performance of the meta-learning method to all of the simpler methods in the meta-learning framework using the MdAPE. We found that the fully trained meta-learning model performed worse on this dataset than most of the simpler forecasting methods. The best performing method was the random walk with drift method. We then trained the finance only meta-learning model, using only a subset of the M4 financial data, and applied this model to the real financial data. This meta-learning model did perform better than the fully trained model, but the random walk with drift was still the best performing method (see table 2).

We then applied both of these meta-learning models to a test set of M4 financial data. Again, we measured the MdAPE of the forecasts of each meta-learning model and the nine simpler methods. The results obtained this time agreed with what might be expected. The best performing method was the fully trained meta-learning method, and the second best performing method was the finance only meta-learning method (see table 2 for full results).

| Meta-Learning Model | Dataset | Overall Weighted Average |
|---|---|---|
| Finance only | M4 financial | 0.716 |
| Finance only | Real financial | 0.956 |
| Fully trained | M4 financial | 0.583 |
| Fully trained | Real financial | 1.037 |
| Fully trained | Full M4 dataset | 0.838[4] |

Table 3: The overall weighted average error of the finance only and fully trained meta-learning model applied to both the M4 financial data and the real financial data. Note that by this measure the meta-learning model performs better on the M4 financial data. On the real data, the finance only model outperforms the fully trained model, similar to the findings in table 2.

## 5.2 Overall Weighted Averages

We also compared the overall weighted averages of the fully trained and finance only meta-learning models when applied to the M4 financial data and the real financial data. By this measurement, the fully trained meta-leaning method performs the best on the M4 financial data. This suggests that there is some transfer learning occurring when training on the M4 dataset. However, the finance only meta-learning model performs better on the real financial data. So there is no transfer learning happening in this context. See table 3 for full results and comparison to the result reported to the M4 competition.

## 5.3 Ideal `xgboost` hyperparameters

In our search for good hyperparamters for the `xgboost` random forest, we tried $106$ different combinations of these parameters. The parameters were

- the maximum depth of the trees,
- the learning rate $\eta$,
- the data subsampling rate,
- the feature subsampling rate.

During the searching process each combination of paramters and the resulting objective function value was recorded. We clustered these $106$ data points using a Gaussian mixture model. The optimal model was found by minimising the Bayesian information criterion (BIC). This gave a model consisting of two gaussian distributions. We examined the means and covariance matrices for each of these Gaussians. The only major difference between the two distributions was the mean value for $\eta$. It seemed like a lower learning rate resulted in a worse performance.

# 6 Conclusions

We applied the forecasting method that came second in the M4 time series forecasting competition to real financial time series and the M4 financial data, and compared the

results. The method itself is an ensemble forecasting method, trained using machine learning. We refer to it as the meta-learning method. Although the method performed well on the M4 financial data, it was outperformed by simpler statistical methods when applied to the real financial data. This implies that any information that can be used by any of the statistical models in the meta-learning framework to forecast the real financial time series has already be found, and aggregated away through arbitrage.

## 6.1 Limitations

The M4 competition provides a dataset of time series only, and requires the competitors to forecast each individual series. There are no driving series, and there is no other information other than the context of the series. Therefore each of the methods in the M4 competition, including the meta-learning method, are univariate. In a financial context, we would expect driving factors to be a vital part of any forecasting framework. Therefore the forecasting methods developed for the M4 competition may not be well suited for financial forecasting.

## 6.2 Further Research

The meta-learning forecasting method uses mostly statistical methods in its ensemble framework. It does use one neural network through the `nnetar` function in the `forecasting` package, but this is a feedforward neural network that uses a small number of time-lags as inputs. This is not the optimal implementation of neural networks for time series.

Recurrent neural networks are designed for processing time series. So implementation of the most up-to-date innovations in recurrent neural networks is a good starting point for further research. Convolutional recurrent neural networks, long-short term memory (LSTM) networks, and neural networks with gated recurrent units (GRU) are examples of these innovations. The theory is that these networks should be able to capture long-term effects better than statistical methods. In fact, the winner of the M4 competition was a forecasting method that combined exponential smoothing and various architectures of recurrent neural networks [2].

# References

[1] Pablo Montero-Manso, Thiyanga Talagala, Rob J Hyndman, George Athanasopoulos, *M4metalearning*. https://github.com/robjhyndman/M4metalearning/, M4 Competition, (2018)

[2] Spyros Makridakis, *M4 Competition*. https://www.m4.unic.ac.cy/about/, University of Nicosia, (2018)

[3] Thomas Delaney, *Forecasting Methods: An Overview*. https://bitbucket.org/thomasjdelaney/financial-forecasting/src/master/latex, (2018)

[4] Evangelos Spiliotis, Spyros Makridakis, Vassilios Assimakopoulos, *The M4 Competition in Progress*. https://www.m4.unic.ac.cy/wp-content/uploads/2018/06/Evangelos_Spiliotis_ISF2018.pdf, (2018)