Ensemble forecasting and the M4 competition

Lars Lien Ankile and Kjartan Krange

Brief agenda

- Forecasting background

- Our model
  - Long Short Term Memory Auto-Encoder for feature extraction
  - Optimization as a means for understanding ensemble methods

- Results

Some forecasting history

History of theoretical forecasting goes back to the 70s

- Reid, Newbold, and Granger conducted the first studies to assess quality and accuracy of forecasting methods early 1970s
- The first indications that ensembles are good for forecasts was found by Newbold and Granger in 1974
- Back then, they only combined 2 or 3 very simple methods
- Over the years, this finding has been confirmed and reinforced by numerous papers (among which are M1, M2, and M3)
- Both (1) complex ways of weighting ensembles and (2) using advanced ML methods have been tried, but up until recently have been unsuccessful
- This is most likely due to (1) the amount of available data and compute has grown immensely and (2) the field of ML has progressed very far
- In the latest literature and M-competitions, more complex methods outperform simpler methods—making interpretation harder for non-experts

The M-competitions have played a central role in the field

M1, 1982	M2, 1993	M3, 2000
What accuracy measure used can change the ranking of the different methods The performance of the various methods depend upon the length of the forecasting horizon	Good and robust performance of ets methods.	**Statistical and sophisticated methods does not necessarily produce more accurate forecast than simple ones**
Accuracy of combinations of methods outperforms on average the single methods alone and does well in comparison with other methods	Less randomness => better relative accuracy of the more sophisticated methods	**Accuracy of combinations of methods outperforms on average the single methods alone and does well in comparison with other methods**
Given an ensemble of methods, the simple average over them outperformed the more complex average based on covariates, though that still did well	**The greatest improvement in forecasting accuracy came from measurement and extrapolating the seasonality of the series**	

Our project: Forecast the M4 competition data

- 100,000 normalized time series
- 6 time interval categories, 6 domains of series
- Importantly a lot of approaches and papers to build on

Table 1
Number of M4 series per data frequency and domain.

Time interval between successive observations	Micro	Industry	Macro	Finance	Demographic	Other	Total
Yearly	6,538	3,716	3,903	6,519	1,088	1,236	**23,000**
Quarterly	6,020	4,637	5,315	5,305	1,858	865	**24,000**
Monthly	10,975	10,017	10,016	10,987	5,728	277	**48,000**
Weekly	112	6	41	164	24	12	**359**
Daily	1,476	422	127	1,559	10	633	**4,227**
Hourly	0	0	0	0	0	414	**414**
Total	**25,121**	**18,798**	**19,402**	**24,534**	**8,708**	**3,437**	**100,000**

Source: Makradakis et. al, The M4 competition: 100,000 time series and 61 forecasting methods

We found the first two winning papers especially relevant

A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting

Slawek Smyl
Uber Technologies, 555 Market St, 94104, San Francisco, CA, USA

1. place M4

ARTICLE INFO

Keywords:
Forecasting competitions
M4
Dynamic computational graphs
Automatic differentiation
Long short term memory (LSTM) networks
Exponential smoothing

ABSTRACT

This paper presents the winning submission of the M4 forecasting competition. The submission utilizes a dynamic computational graph neural network system that enables a standard exponential smoothing model to be mixed with advanced long short term memory networks into a common framework. The result is a hybrid and hierarchical forecasting method.

FFORMA: Feature-based forecast model averaging

Pablo Montero-Manso *, George Athanasopoulos, Rob J. Hyndman, Thiyanga S. Talagala
Department of Econometrics and Business Statistics, Monash University, Australia

2. place M4

ARTICLE INFO

Keywords:
Time series features
Forecast combination
XGBoost
M4 competition
Meta-learning

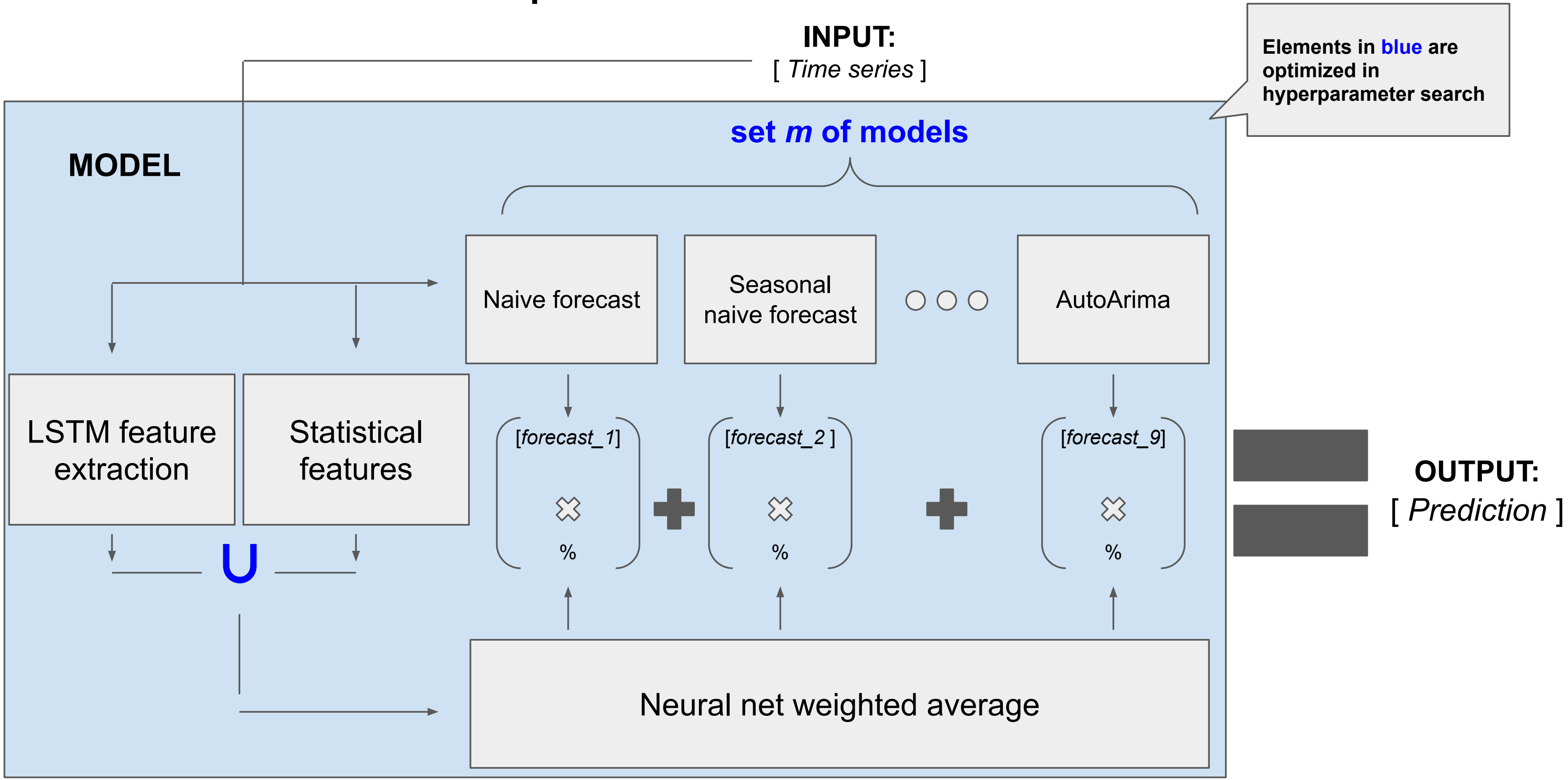
ABSTRACT

We propose an automated method for obtaining weighted forecast combinations using time series features. The proposed approach involves two phases. First, we use a collection of time series to train a meta-model for assigning weights to various possible forecasting methods with the goal of minimizing the average forecasting loss obtained from a weighted forecast combination. The inputs to the meta-model are features that are extracted from each series. Then, in the second phase, we forecast new series using a weighted forecast combination, where the weights are obtained from our previously trained meta-model. Our method outperforms a simple forecast combination, as well as all of the most popular individual methods in the time series forecasting literature. The approach achieved second position in the M4 competition.

	1. Smyl (Uber tech.)	2. Montero-Manta Et al.
Metadata	✓	✗
Training on feature extraction	✗	✓
Ensemble	✗	✓

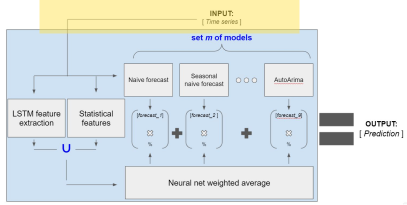
Our model

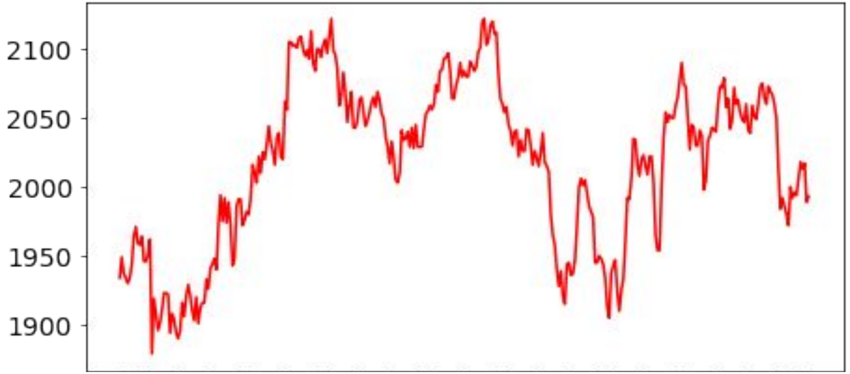
Essence of approach: Reduce assumptions about important characteristics and best fit models to predict time series

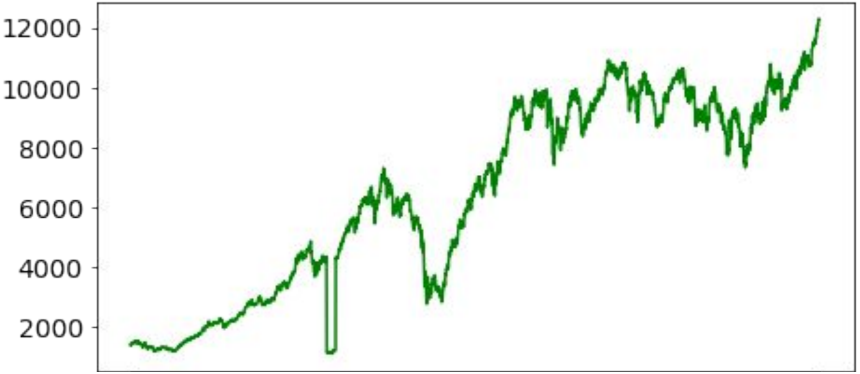


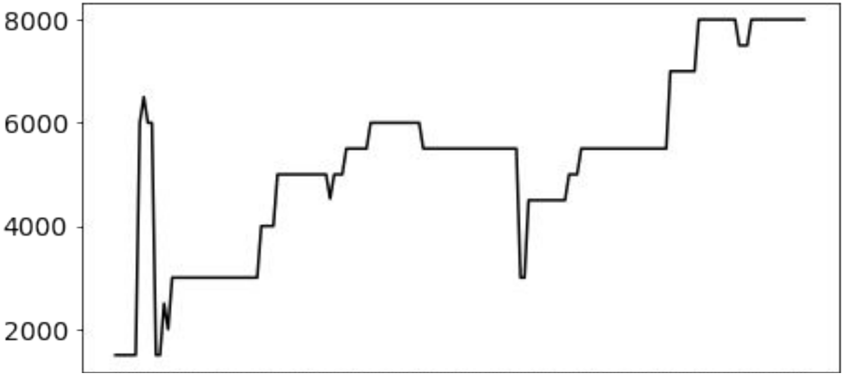
What does a prediction look like

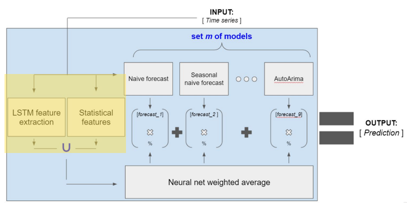
1) Start with time series











2) Get features

We combine 42 features montero-manso et.al

With our own novel approach for feature extraction discussed later

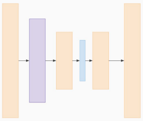
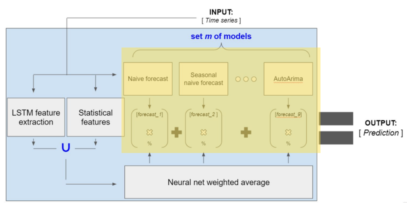
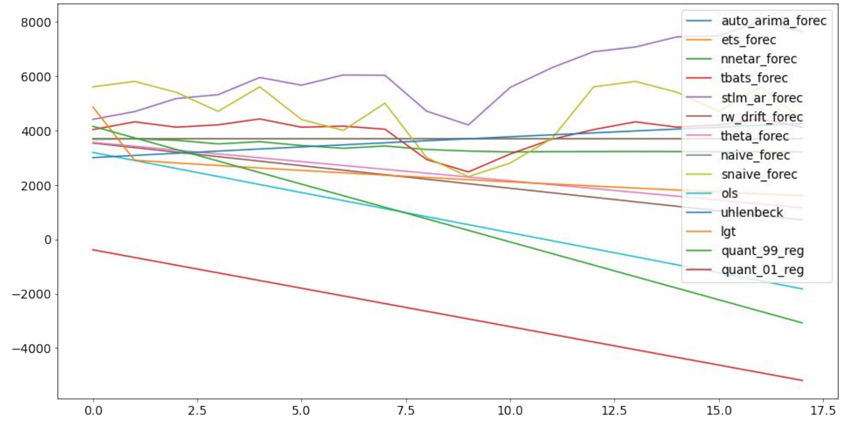


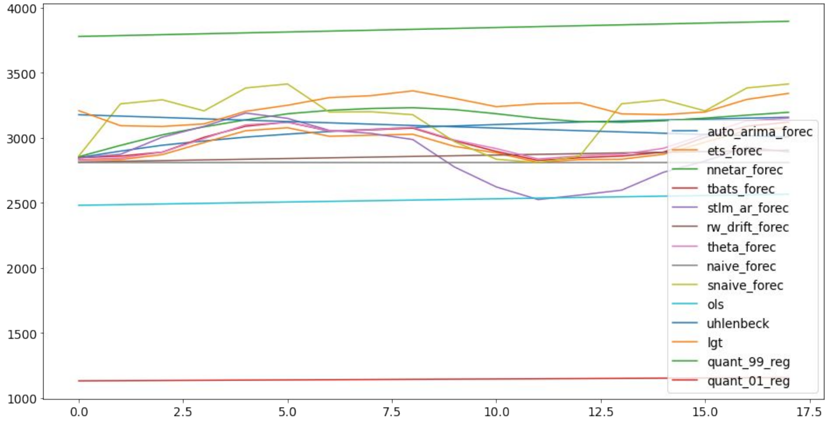
Table 1
Features used in the FFORMA framework.

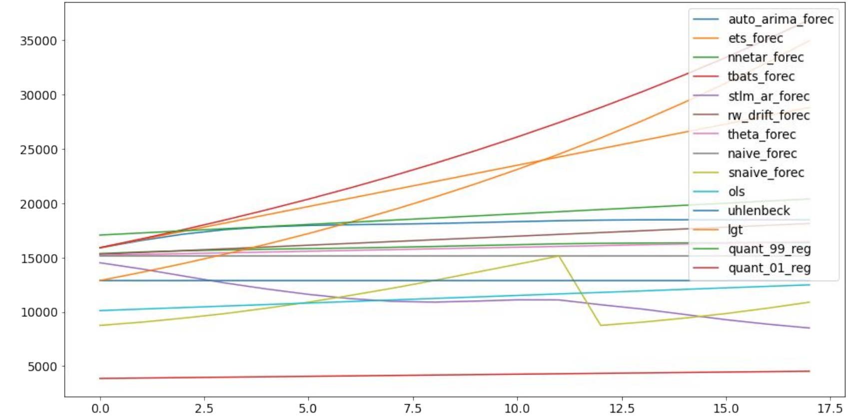
Feature		Description	Non-seasonal	Seasonal
1	T	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonality	strength of seasonality	–	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	e_acf10	sum of squares of first 10 ACF values of remainder series	✓	✓
9	stability	stability	✓	✓
10	lumpiness	lumpiness	✓	✓
11	entropy	spectral entropy	✓	✓
12	hurst	Hurst exponent	✓	✓
13	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	–	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	–	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	–	✓
18	ur_pp	test statistic based on Phillips–Perron test	✓	✓
19	ur_kpss	test statistic based on KPSS test	✓	✓
20	y_acf1	first ACF value of the original series	✓	✓
21	diff1y_acf1	first ACF value of the differenced series	✓	✓
22	diff2y_acf1	first ACF value of the twice-differenced series	✓	✓
23	y_acf10	sum of squares of first 10 ACF values of original series	✓	✓
24	diff1y_acf10	sum of squares of first 10 ACF values of differenced series	✓	✓
25	diff2y_acf10	sum of squares of first 10 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first seasonal lag	–	✓
27	sediff_acf1	first ACF value of seasonally differenced series	–	✓
28	y_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
29	diff1y_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
30	diff2y_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓
31	seas_pacf	partial autocorrelation coefficient at first seasonal lag	✓	✓
32	crossing_point	number of times the time series crosses the median	✓	✓
33	flat_spots	number of flat spots, calculated by discretizing the series into 10 equal-sized intervals and counting the maximum run length within any single interval	✓	✓
34	nperiods	number of seasonal periods in the series	–	✓
35	seasonal_period	length of seasonal period	–	✓
36	peak	strength of peak	✓	✓
37	trough	strength of trough	✓	✓
38	ARCH.LM	ARCH LM statistic	✓	✓
39	arch_acf	sum of squares of the first 12 autocorrelations of z^2	✓	✓
40	garch_acf	sum of squares of the first 12 autocorrelations of r^2	✓	✓
41	arch_r2	R^2 value of an AR model applied to z^2	✓	✓
42	garch_r2	R^2 value of an AR model applied to r^2	✓	✓

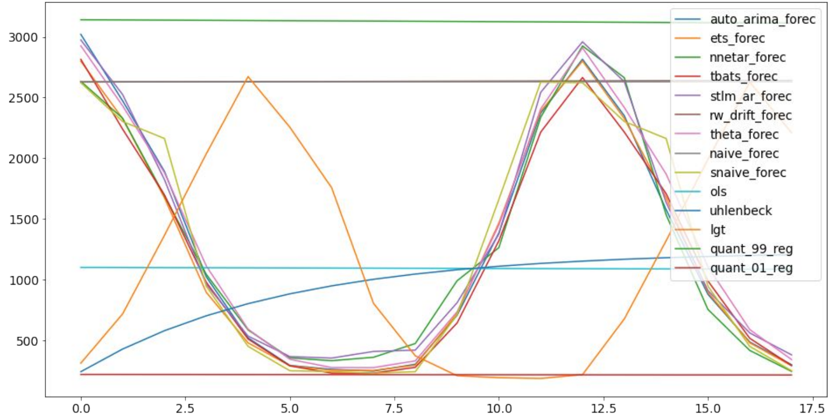
3) Make m predictions



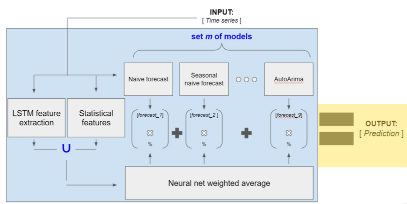


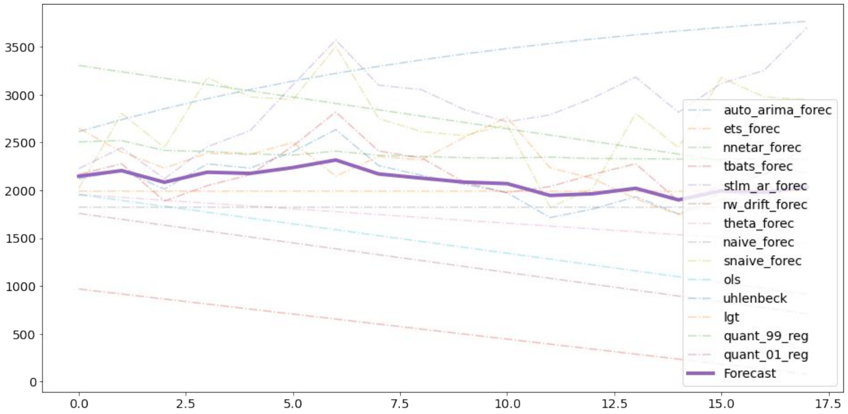


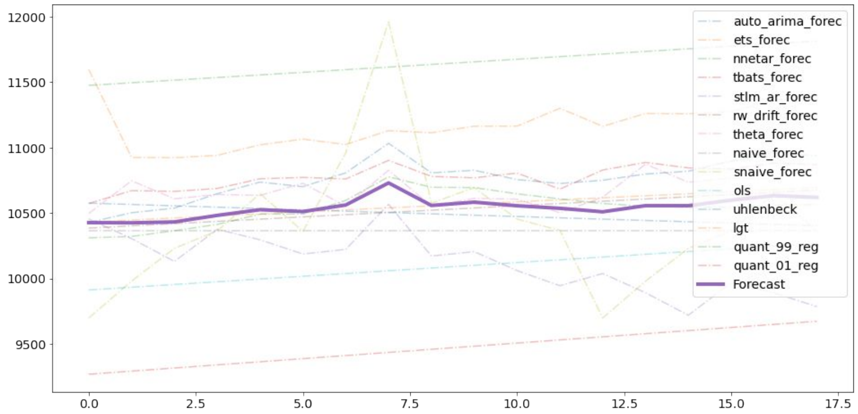


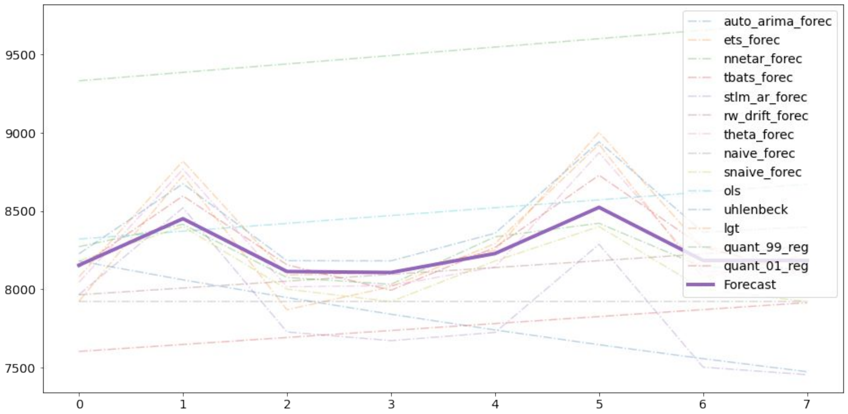


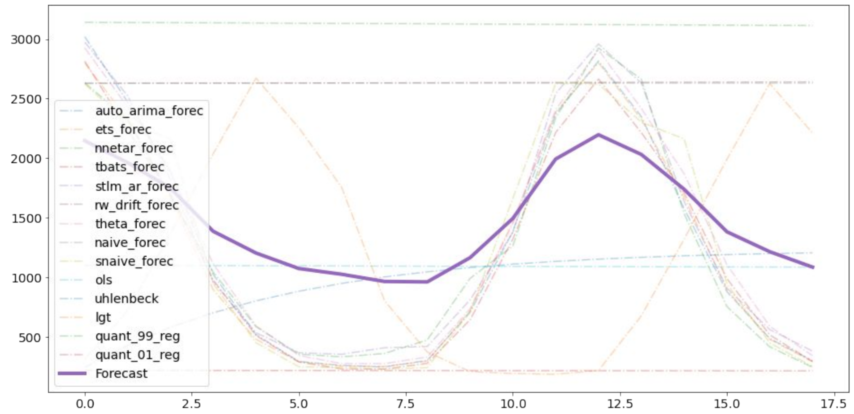
4) Combine to one forecast



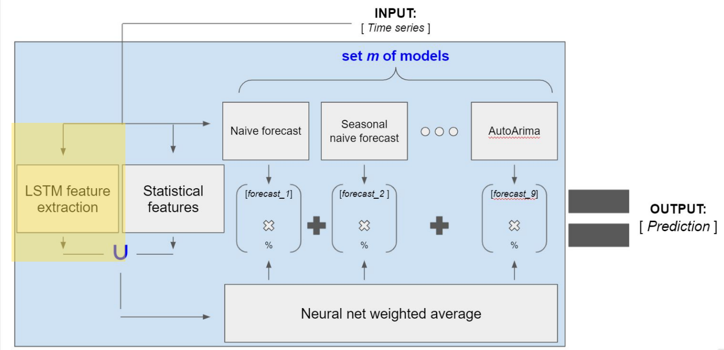




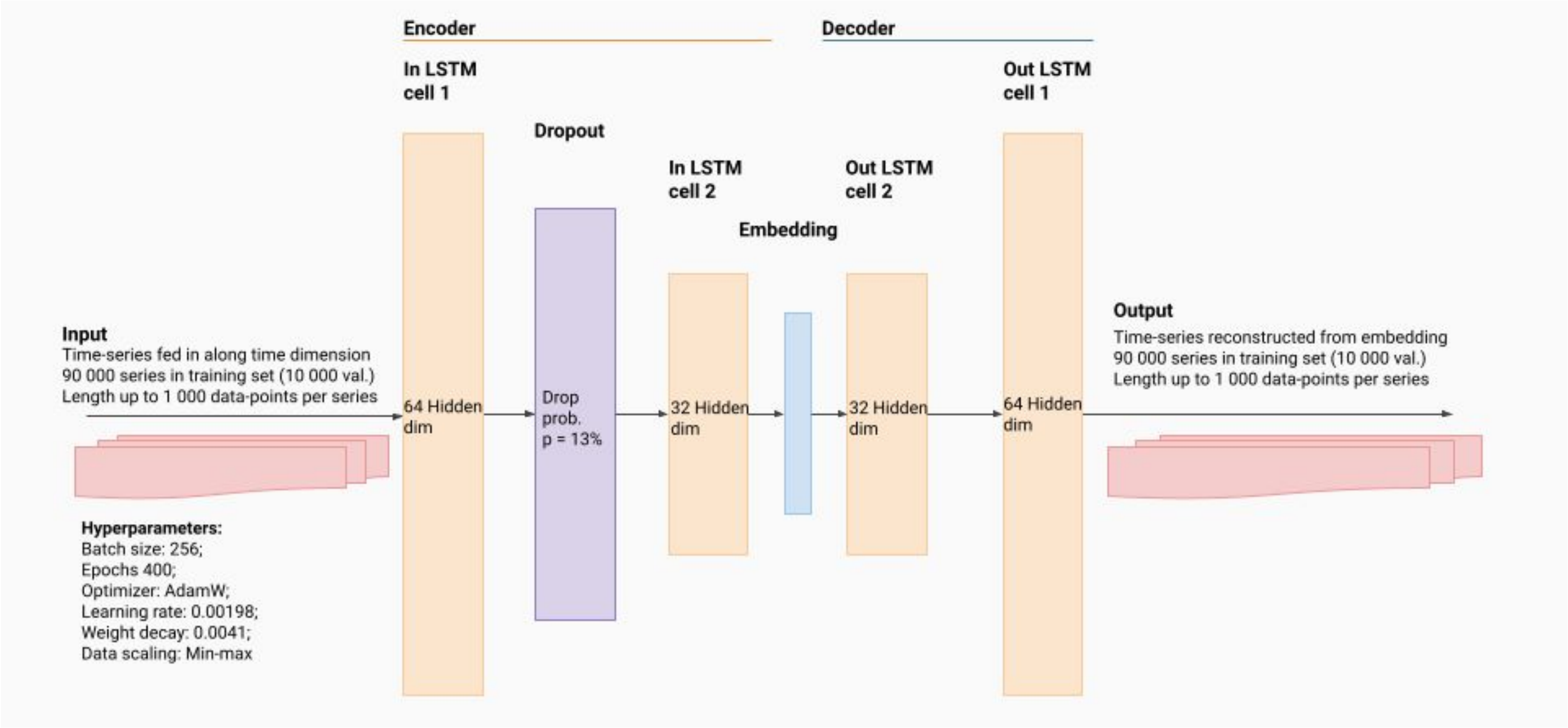




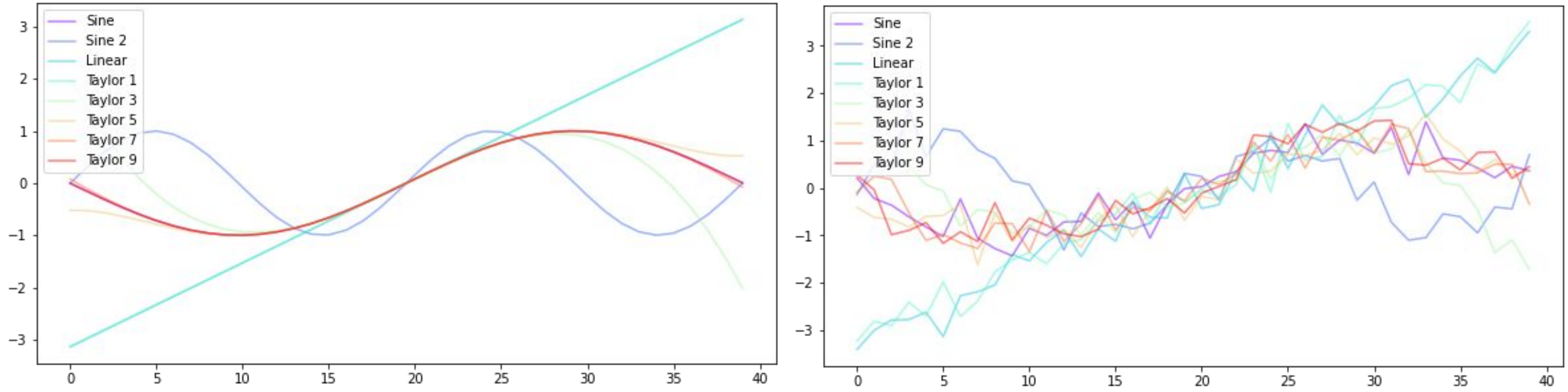
Automatic time-series feature extraction using LSTM auto-encoder



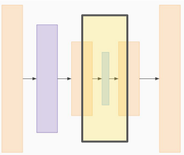
The LSTM auto encoder compresses time series information, keeping the information necessary to recreate (noiseless) series

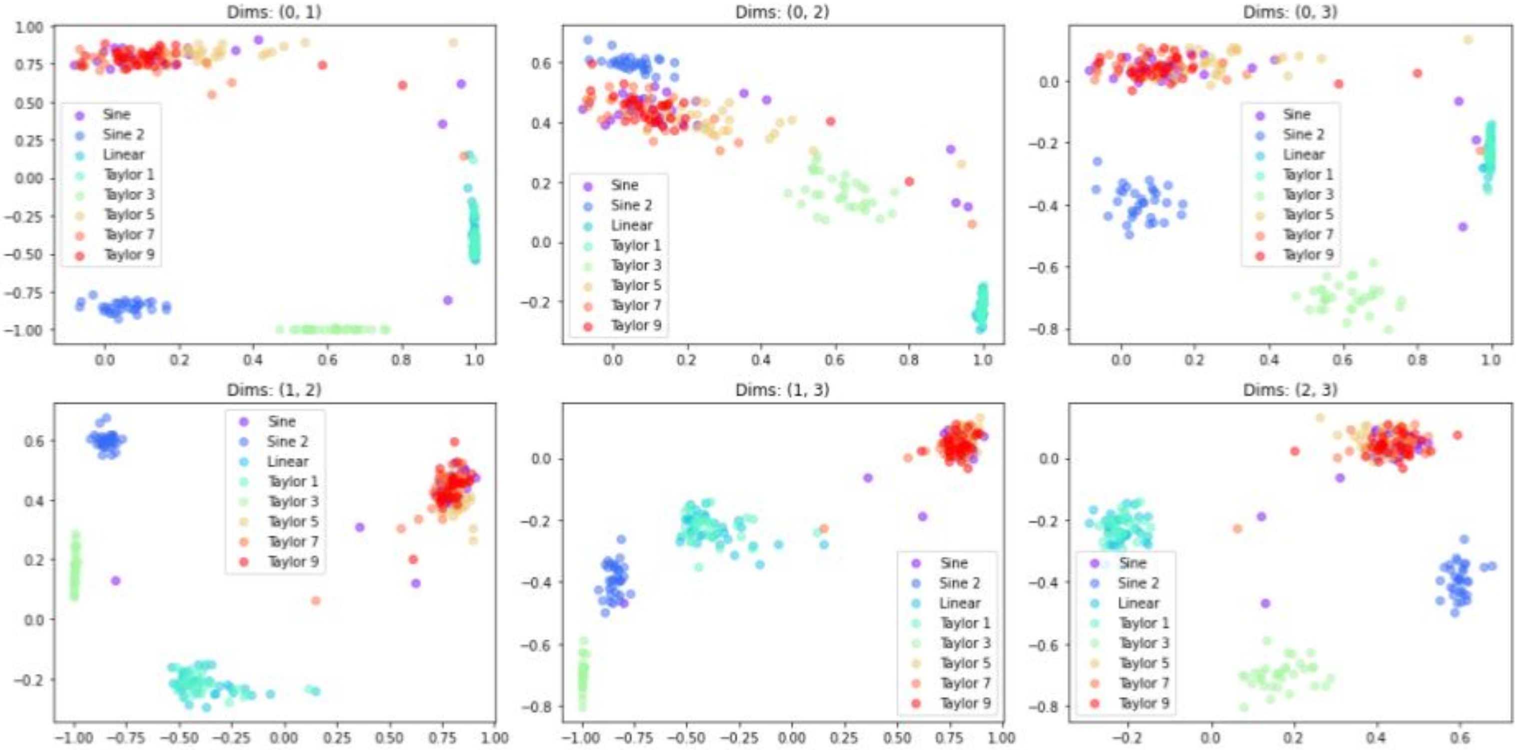


To enable interpretation of the learned feature embeddings, we create a simple, synthetic data set, here composed of a line, 2 sines, and taylor approximations to the slowest sine

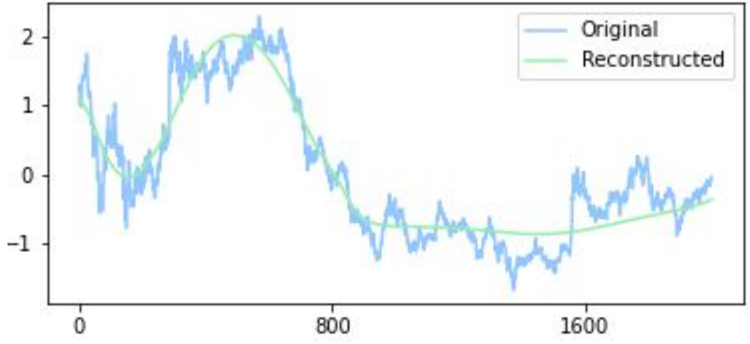


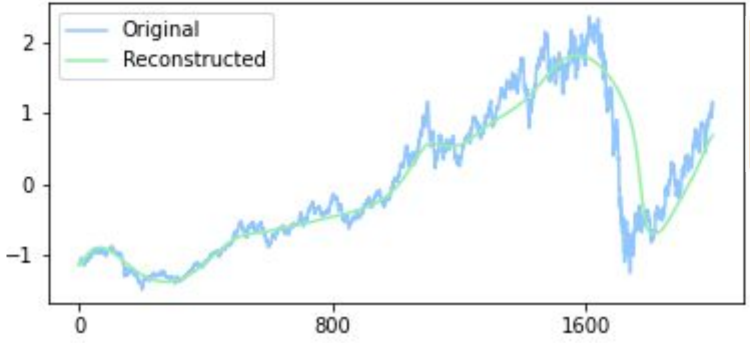
The plot below shows every combination of two dimensions (out of four) in the feature space, the locations of the dots can have several interesting meanings

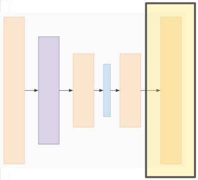


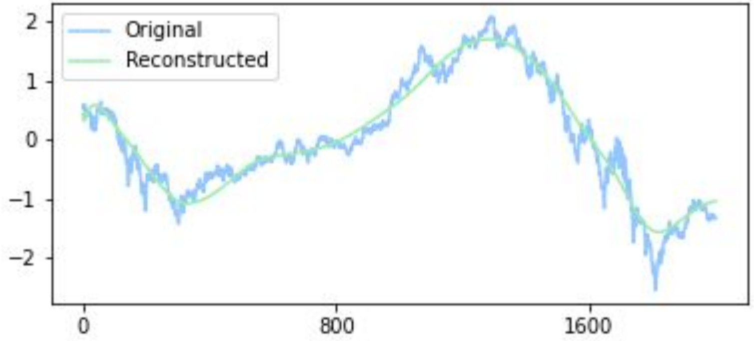


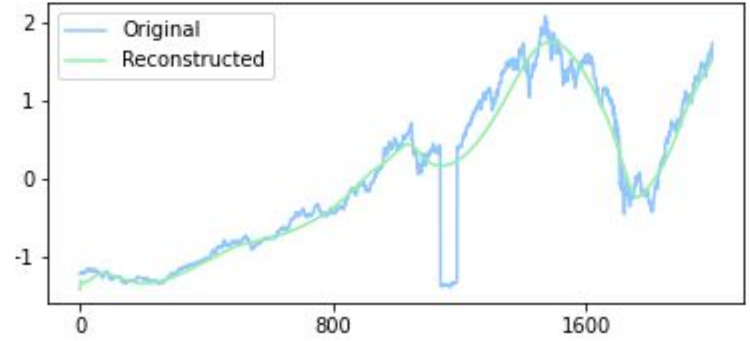
Autoencoder seems to be able to capture and recreate out-of-sample series from the M4 dataset with an embedding dimension of 32



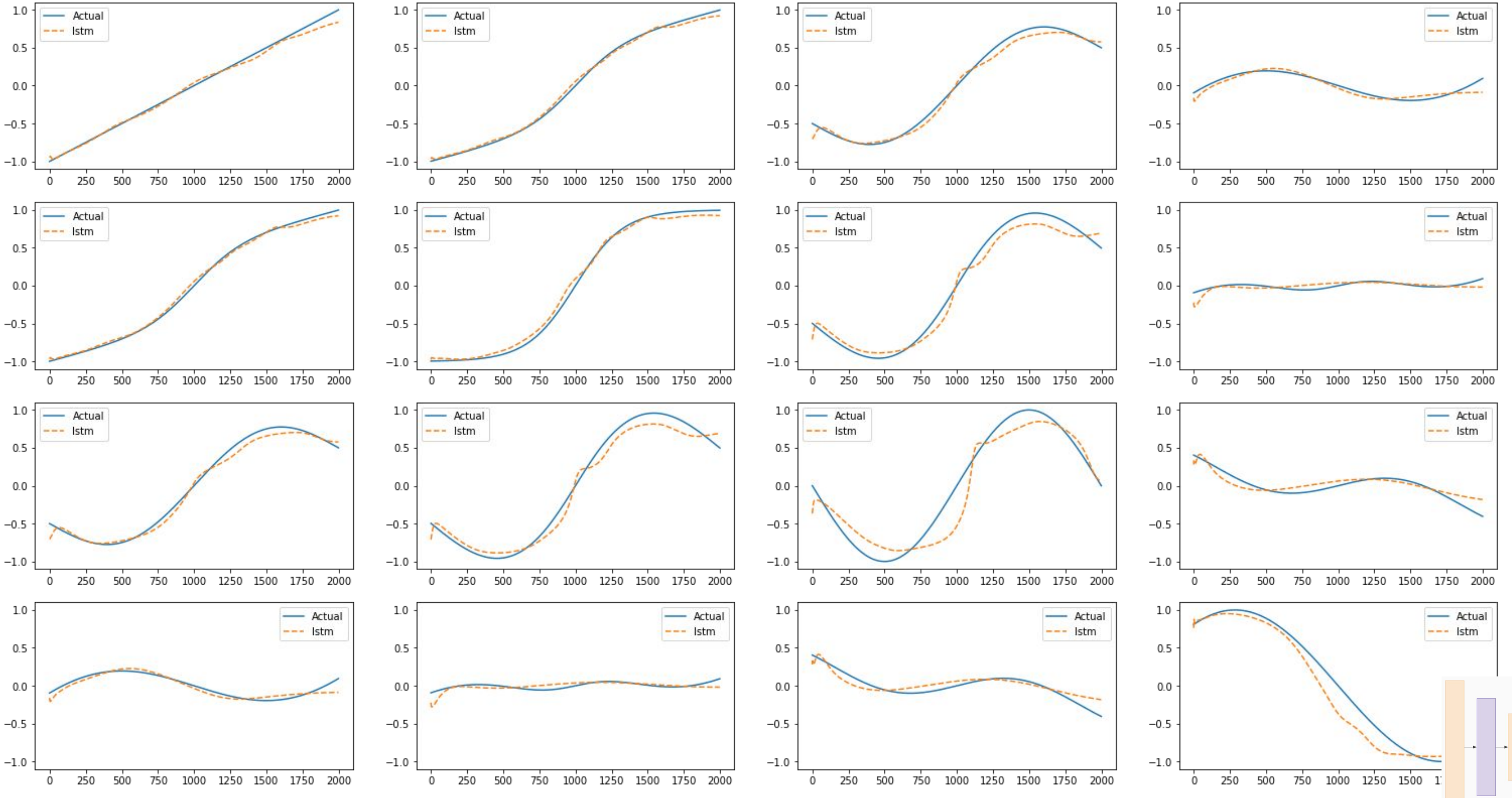


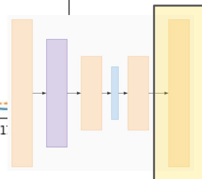




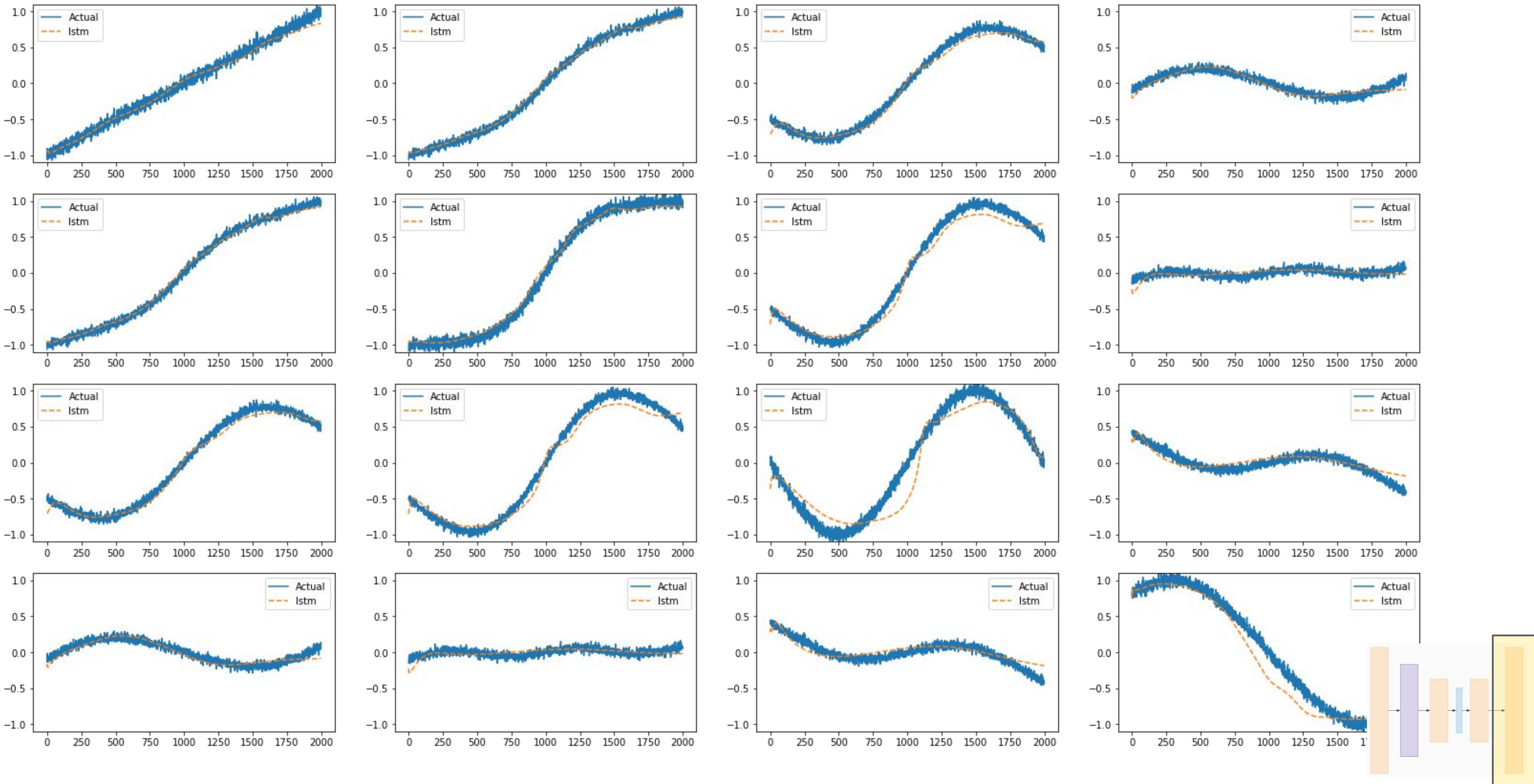


With a latent space of length 32, the autoencoder must create an efficient representation of a time-series of length 2000 to successfully recreate new series

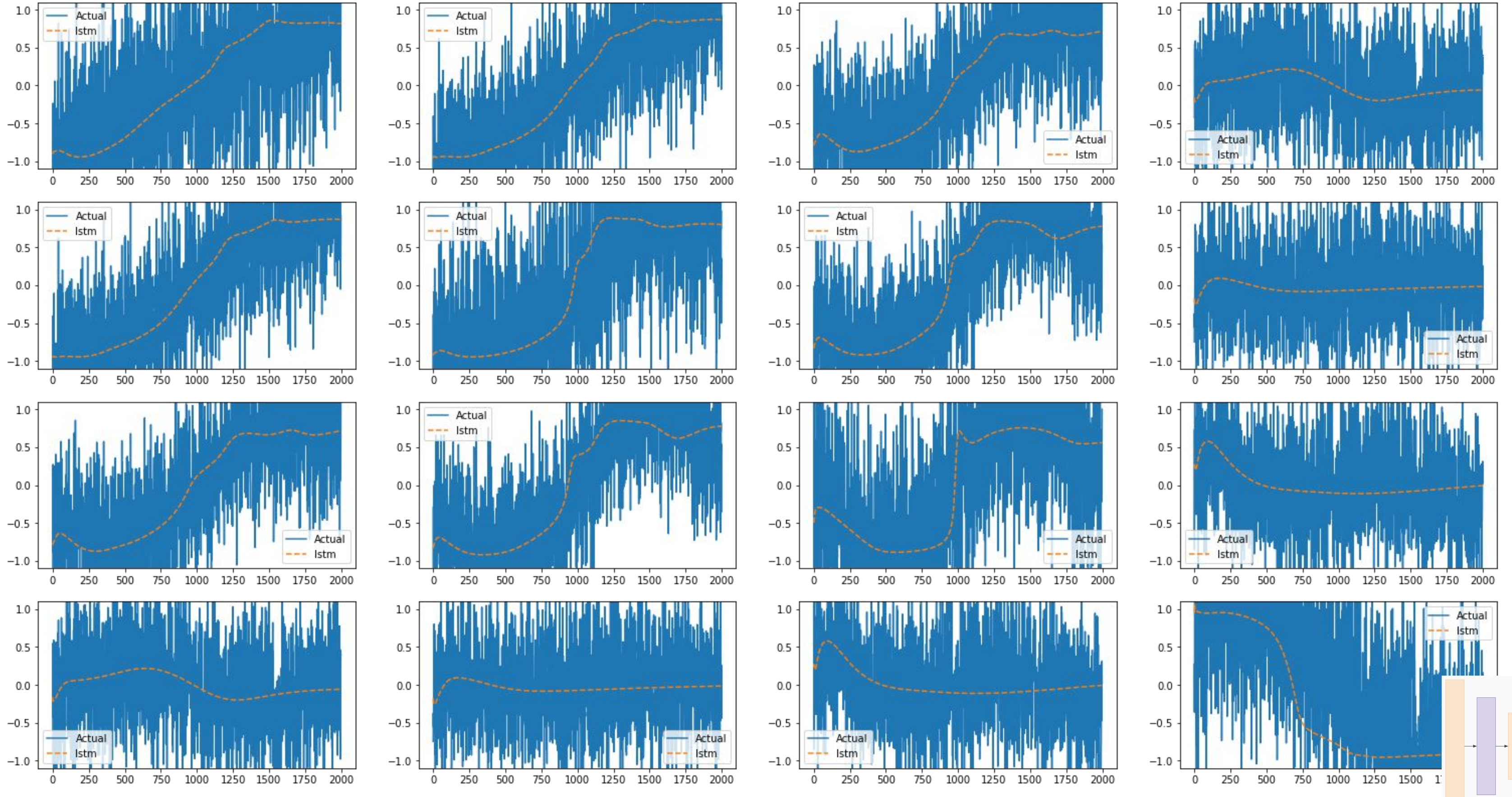


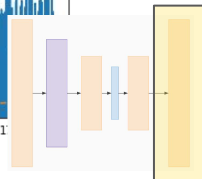


When random noise is added to the input, the autoencoder efficiently filters out the noise and reproduces more or less only the signal itself

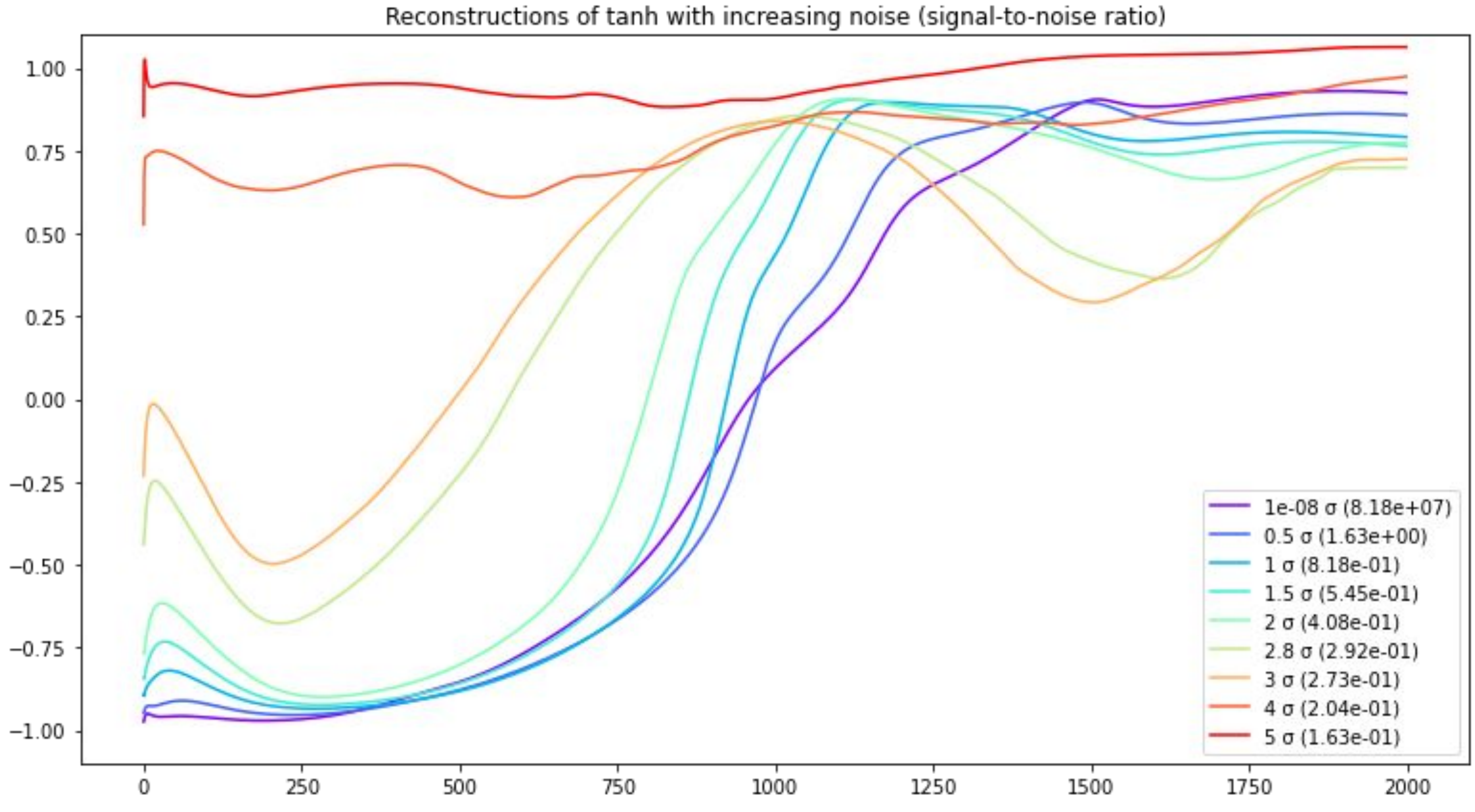


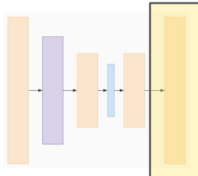
Even under conditions of extreme noise, it is able to find a signal to a significant degree



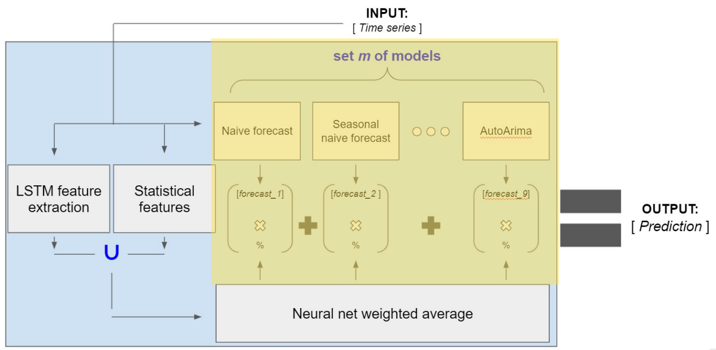


Even though it is very robust to noise, it will still break down at some point–in this example, this happens when signal-to-noise ratio drops below ca. 0.3





Theoretical ensemble accuracy through optimization methods



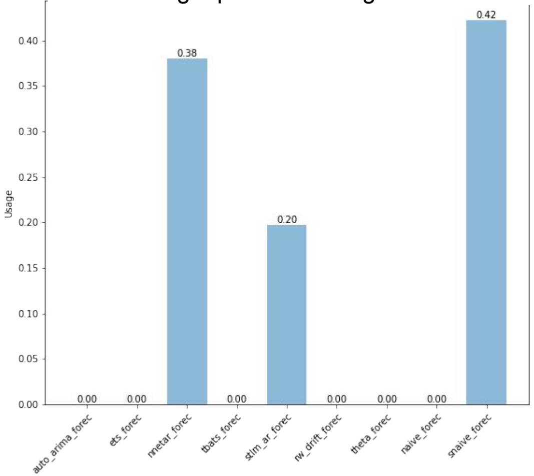
Through a posteriori optimizations using a linear program, we found 4 interesting observations

Sets	M	Models in ensemble
	T	Forecast horizon
Parameters	$b_{ij}, (i,t) \in M \times T$	Matrix parameter representing model i forecast at t
	$y_t, t \in T$	Matrix parameter representing
Variables	$x_i, i \in M$	The weight of each model
	$z_t^+, z_t^-, t \in T$	Variable for modeling absolute value distance
Minimize	$\sum_{t \in T} z_t^+ + z_t^-$	Minimize total distance from actual
Subject to:	$y_t - \sum_{i \in M} x_i b_{it} \leq z_t^+, \forall t \in T$	The actual value is larger than the forecast
	$-y_t + \sum_{i \in M} x_i b_{it} \leq z_t^-, \forall t \in T$	The actual value is smaller than the forecast
	$\sum_{i \in M} x_i = 1$	Weighted sum equals 1
	$0 \leq x_i, \forall (i) \in M$	Each weight non-negative

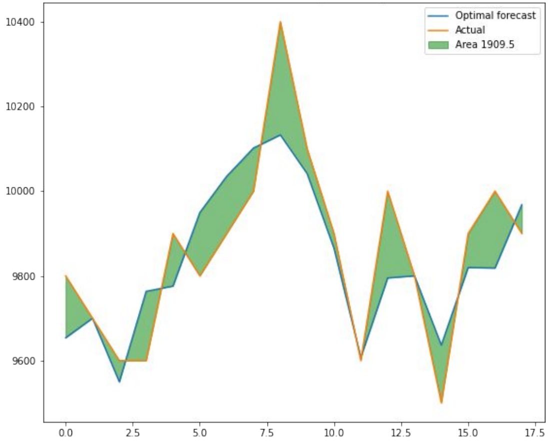
In short, the program minimizes MASE loss given a set of models *m* for forecasts, and a given forecast horizon, with respect to the actual values of the time-series, y_t

1) Sometimes your ensemble would never be able to forecast accurately

Weight per forecasting model



Actual vs forecast at optimal weight



Left ensemble:

3 models used.

Optimal forecast is not far from actual values.

In this scenario a net would theoretically be able to train away a significant error.

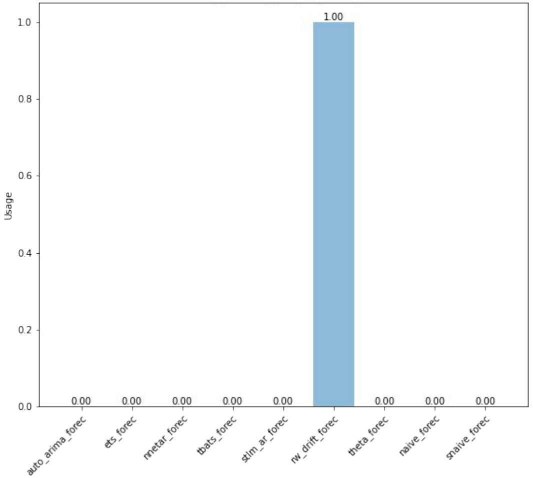
Right ensemble:

1 model used.

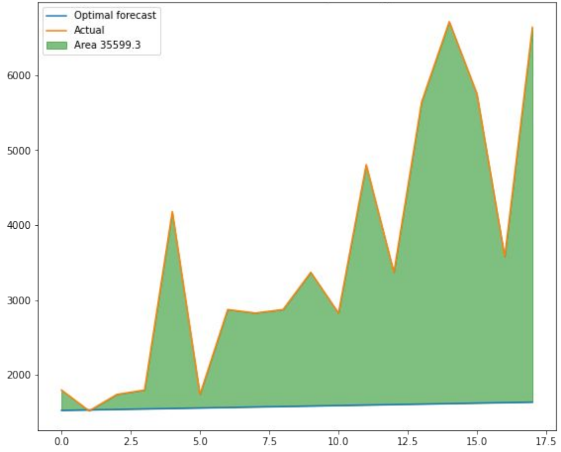
Optimal forecast deviates a lot from the actual values.

In this scenario a net is not able to train away a significant amount of error.

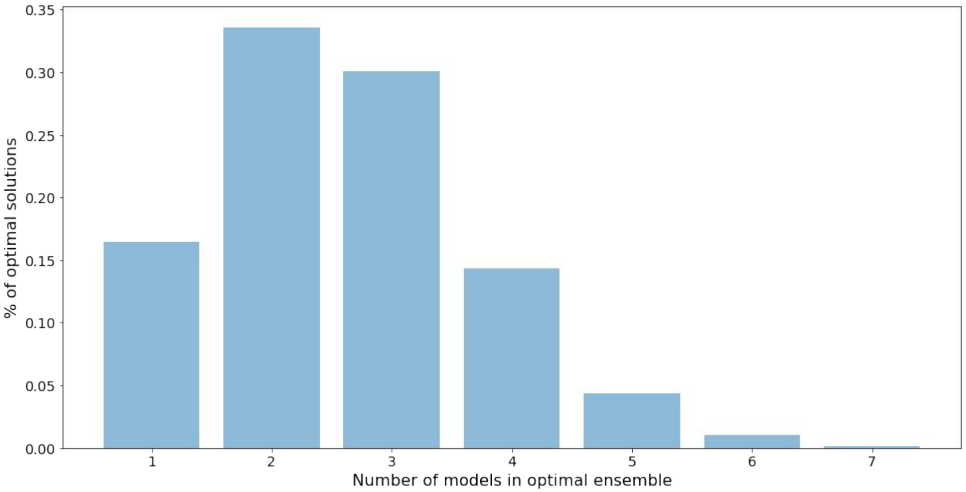
Weight per forecasting model

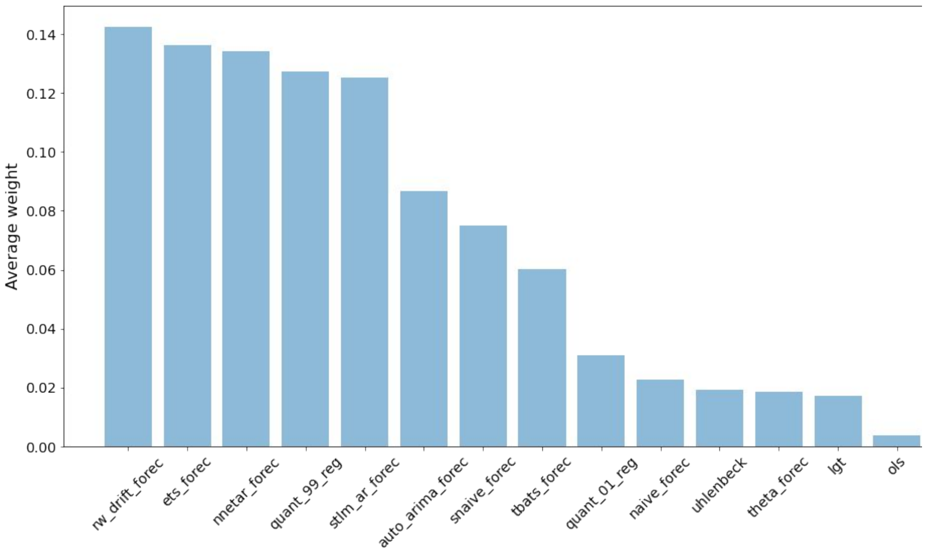


Actual vs forecast at optimal weight

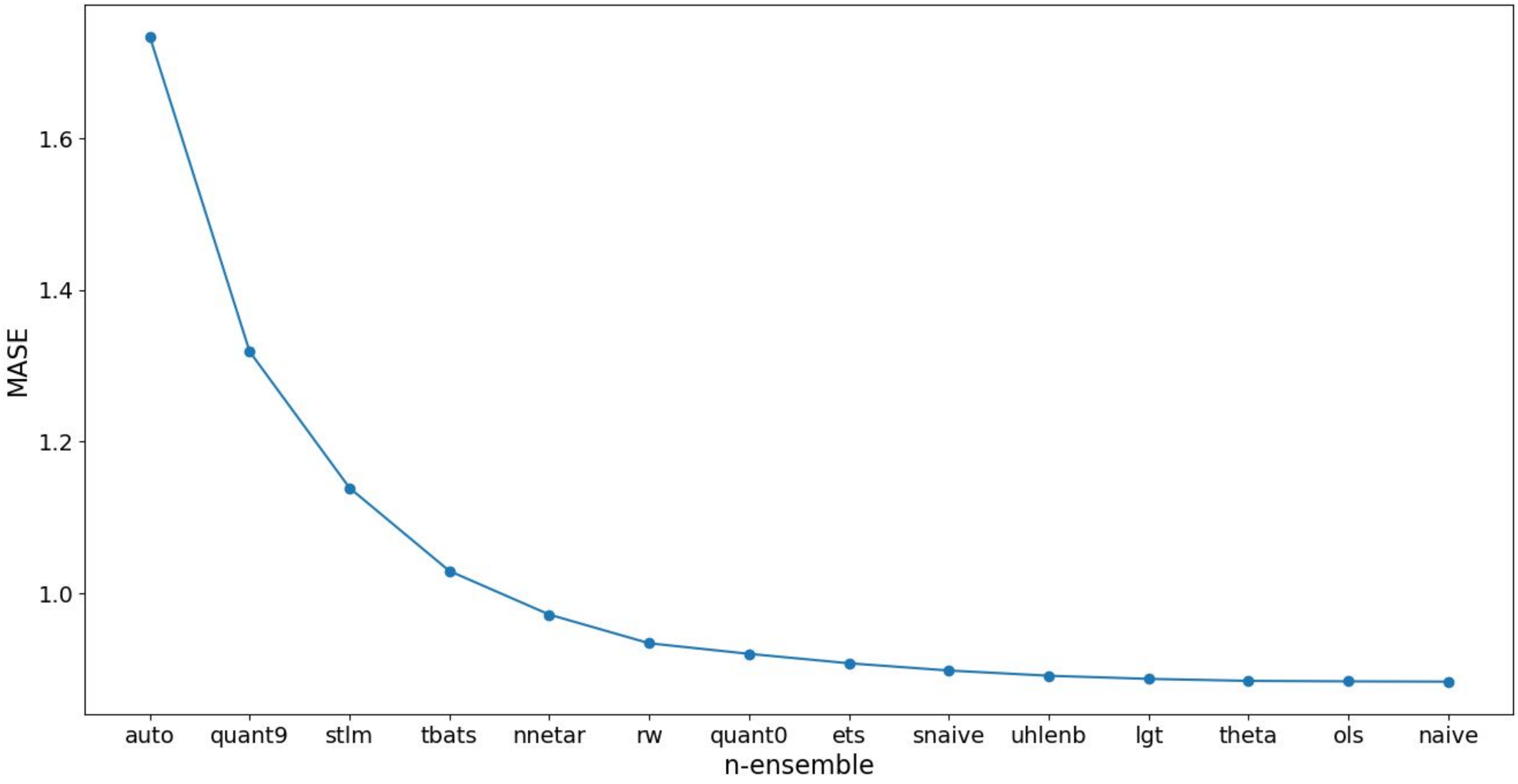


2) Of optimal validation ensembles 3/4s consists of 2 or more models
3) The mean weight of some models is way larger than others





4) The optimal error decreases with larger ensembles with a diminishing marginal effect



Lastly, we now think of total loss, a sum of two orthogonal parts:

*Total_loss = weight_loss(m, **weights**) + latent_ensemble_loss(m)*

In our case:

m = set of 14 models

weights = net(stat_lstm_features, m, training_params)

Results

Performing at par with top placements on out-of-sample data

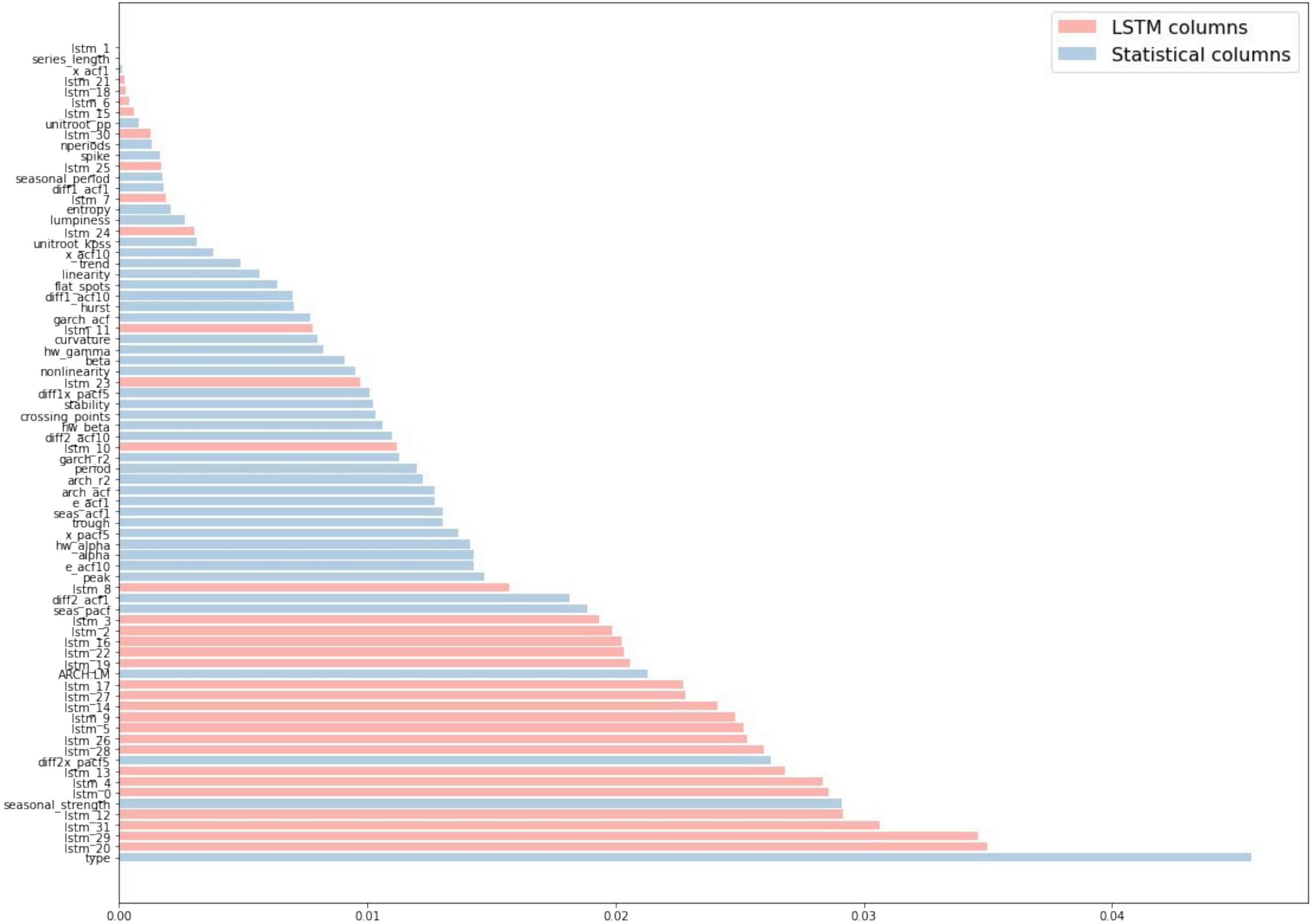
	Jolly Sweep (as of last night)	Smyl	Montero-Manso et. al
MASE	1.531	1.536	1.551

How much does loss increase when randomizing a column

Key takeaways:

- The *type* of a series is most important for predicting capabilities
- 17 of the top 20 inputs are a lstm features automatically gathered from the LSTM

Note: Of course, this is only applicable
to our ensemble and net



Hyperparameter search finds that only using traditional statistical features while training correlates positively with loss

Historic findings	**Our findings**
M4: Using statistical features as input to weighting an ensemble is the second best forecasting method	In our approach, new auto-encoded features seems more important in determining the right weights
M2: Seasonality is important for prediction	We find *type* and *seasonal strength* to be most important statistical features
M3: Ensembling forecasts do better than the constituent models single-handedly	The optimization and latent_ensemble_loss in part explain this