

Time Series Forecasting: Machine Learning and Deep Learning with R & Python

Marco Zanotti

Part I

Time Series
Forecasting in
2023

Business Forecasting is Changing

1. Higher frequencies.
2. Scale is increasing.
3. Automation is needed.

Time Series is Changing

To meet today's demands

**Forecasting
in 2015**

Monthly Series

Scale: <10 Time Series

Simple ARIMA OK

This transformation is
challenging us to learn
new skills

**Forecasting
in 2020**

Daily Series

Scale: 10,000+ Time Series

New Technologies Needed: ML & DL





Your Organization:

"I need a data scientist that knows time series, can improve forecasting, and I need it done for **10,000+ Time Series every day.**"

What are they asking for?

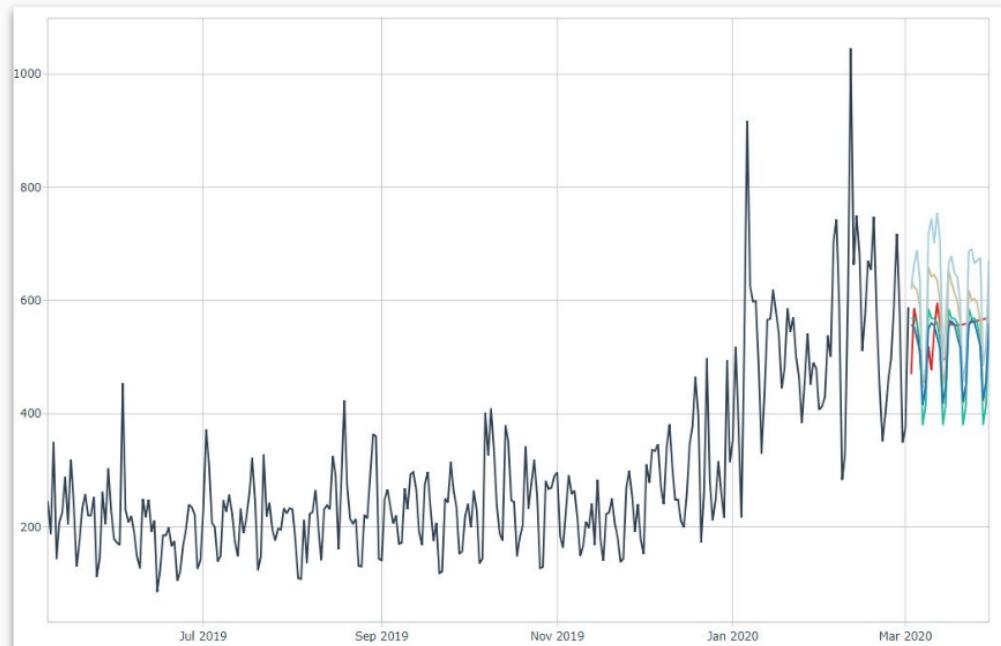
"I need a data scientist that knows time series, can improve forecasting, and I need it done for **10,000+ Time Series every day.**"

What I really meant to say is, "**I need you to make a High-Performance Forecasting System (HPFS).**"

HPFS

=

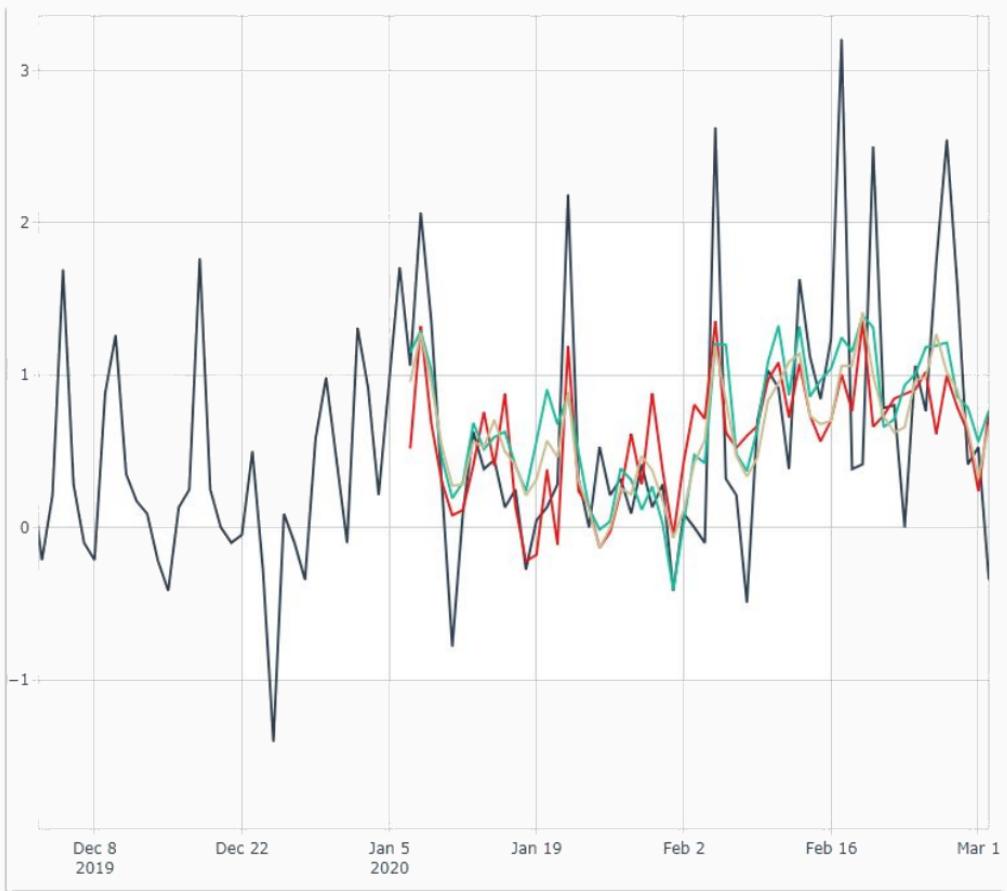
**This...
x 10,000**



3 Properties of

High-Performance Forecasting Systems

What companies want



1. Improves Results

Improvements of 10%
save organizations
Millions of \$\$\$

This is why the
organization **invests in**
you.

Property #1: Delivers the Best Possible Results Given Your Resource Constraints

.model_id	.model_desc ↑	.type ↓	↓ mae	↓ mape	↓ mase	↓ smape	↑ rmse	↓ rsq
21	PROPHET W/ XGBOOST ERRORS	Test	0.46	1519.18	0.62	85.54	0.63	0.44
14	XGBOOST - Lag	Test	0.48	891.48	0.66	90.59	0.65	0.4
20	PROPHET W/ REGRESSORS	Test	0.49	1435.07	0.66	87.04	0.66	0.39
19	NNAR(2,1,10)[7]	Test	0.51	1377.19	0.69	96.09	0.69	0.33
2	GLMNET - Lag	Test	0.52	1286.65	0.7	98.93	0.7	0.36
16	CUBIST - Lag	Test	0.53	1364.64	0.71	96.51	0.7	0.32
7	KERNLAB - Spline	Test	0.52	1180.45	0.7	97.06	0.71	0.34
8	KERNLAB - Lag	Test	0.53	1200.72	0.72	100.76	0.71	0.34
12	RANDOMFOREST - Lag	Test	0.52	843	0.7	96.48	0.71	0.37
28	SEASONAL DECOMP: REGRESSION WITH ARIMA(2,1,3) ERRORS	Test	0.53	1058.11	0.71	101.56	0.71	0.32
3	EARTH - Spline	Test	0.52	1051.52	0.7	91.72	0.72	0.28
18	NNET - Lag	Test	0.59	1884.91	0.8	92.29	0.72	0.35
22	REGRESSION WITH ARIMA(2,1,1) (0,0,2)[7] ERRORS	Test	0.56	2206.12	0.76	86.77	0.72	0.31
1	GLMNET - Spline	Test	0.54	1222.27	0.73	102.76	0.73	0.3
4	EARTH - Lag	Test	0.52	570.12	0.71	105.52	0.73	0.36
23	ARIMA(2,1,2) W/ XGBOOST ERRORS	Test	0.52	1473.81	0.71	102.71	0.73	0.34
25	TBATS(1, {0,2}, -, {<7,2>, <30,6>, <365,8>})	Test	0.52	990.87	0.71	93.43	0.74	0.3
9	KKNN - Spline	Test	0.56	1279.32	0.76	98.45	0.75	0.25
17	NNET - Spline	Test	0.55	1838.88	0.75	90.6	0.75	0.19
27	SEASONAL DECOMP: ARIMA(3,1,3)	Test	0.57	946.13	0.77	109.56	0.76	0.26

2. Report Accuracy

Your organization needs to know if there is an issue with the system.

Which time series?
Which models?

Property #2: Reports accuracy & Alerts if an issue is detected



Property #3: Scalable from One to Many Time Series

Part II

Competitions

Competition Review

Competitions have the exact properties we are looking for:

- Scalability
- Maximize Results
- Report Accuracy

Learn from competitions.

The figure displays four screenshots of forecasting competition websites:

- Web Traffic Time Series Forecasting**: A competition to forecast future traffic to Wikipedia pages. It shows a leaderboards page with three entries: Arthur Sutin, CPMP, and thousandvoices.
- Walmart Recruiting - Store Sales Forecasting**: A competition to predict store sales. It shows a leaderboards page with three entries: David Thaler, Srihari Jagannathan, and James King.
- MOFC**: The Machine Forecasting Competition homepage, featuring links for About, Competitions, Conferences, Publications, Courses, and Contact.
- M4 Competition**: A competition to estimate unit sales of Walmart retail products. It shows a leaderboards page with four entries: BigLeader YJ_STU, Matthias, mf, and monsarada.

Forecasting Competitions

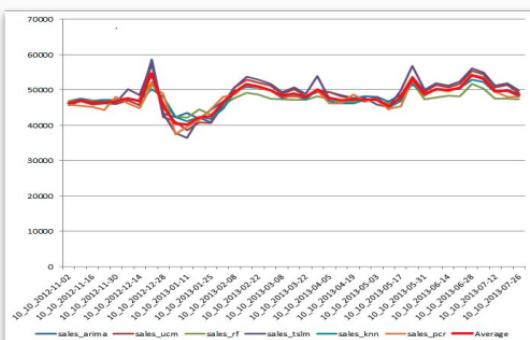
Competition	Year Held
Walmart Recruiting - Store Sales Forecasting	2014
M4 Competition - 100,000 Univariate Time Series	2018
Wikipedia Website Traffic Forecast Competition	2018
M5 Competition - Walmart	2020

Walmart Sales (2014): 2nd Place Solution - Machine Learning & Ensembling 6 Bad Models Make 1 Good One!

Ensembling Different ML Approaches using Calendar (Date) Features

- ARIMA (2891 / 2999)
- Random Forest (2783 / 2868)
- KNN Regression (2657 / 2858)
- +3 More ML / Univariate Models

Simple Average (2371)



Srihari Jaganathan
2nd place

6 bad models make 1 good model: Power of Ensemble Learning

posted in [Walmart Recruiting - Store Sales Forecasting](#) 6 years ago



Below are my Public/Private Leader board scores for individual models.

1. ARIMA (2891/2999)
2. Unobserved Components Model (2817/2949)
3. Random Forest (2783/2868)
4. KNN Regression (2657/2711)
5. Linear Regression (TSLM) (2858/2986)
6. Principle Component Regression (3131/3190)

Simple average of the above 6 models yielded a score (2421/2501). with some little tweaking of how I average greatly improved my scores.

The key thing that I learnt early on was that diversity is very important in ensemble learning. The correlation among models were not good. See below for Department 10, Store 10.

More on why this worked to come ...

Pros:

Easy to automate using Classical + ML + Date Features + Ensembling (Model Averaging)

Cons: Didn't detect the Key Event (Shift in Holiday Spending)

Website Traffic Forecasting: 1st Place Solution - Deep Learning RNN

RNN using features:

- Yearly Seasonality
- Holidays

LSTM can accept multiple features, which is very powerful for learning using autocorrelation.

“...[RNN] a natural extension of ARIMA models, but much more flexible & expressive.”

- Arthur Suilin

Overview Data Notebooks Discussion Leaderboard Rules New Topic


Arthur Suilin
1st place

1st place solution

posted in [web-traffic-time-series-forecasting](#) 3 years ago

369

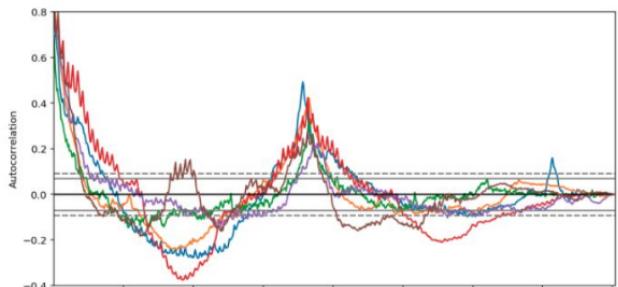
TL;DR this is seq2seq model with some additions to utilize year-to-year and quarter-to-quarter seasonality in data.

Model code: <https://github.com/Arturus/kaggle-web-traffic>

There are two main information sources for prediction:

1. Local features. If we see a trend, we expect that it will continue (AutoRegressive model), if we see a traffic spike, it will gradually decay (Moving Average model), if we see more traffic on holidays, we expect to have more traffic on holidays in the future (seasonal model).
2. Global features. If we look at autocorrelation plot, we'll notice strong year-to-year autocorrelation and some quarter-to-quarter autocorrelation.

The good model should use both global and local features, combining them in an intelligent way.



M4 Competition: 1st Place - Deep Learning Hybrid

Key to the victory

Combining Deep Learning with
Exponential Smoothing

Won by 3%
(big margin vs 2nd Place)

In more recent news, **N-Beats RNN algorithm** is reported to have improved an additional 3% over ES-RNN Hybrid model.

Author(s)	Affiliation	sMAPE MASE OWA Rank				% improvement method over the benchmark			
		sMAPE	MASE	OWA	Rank	sMAPE	MASE	OWA	sMAPE
Smyl	Uber Technologies	11.374	1.536	0.821	1	1	1	9.4	7.7
Montero-Manso, et al.	University of A Coruña & Monash University	11.720	1.551	0.838	3	3	2	6.6	6.7
Pawlakowski, et al.	ProLogistica Soft	11.845	1.547	0.841	5	2	3	5.7	6.9

The screenshot shows a scatter plot titled "M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model". The plot compares forecasted values (blue dots) against actual values (red dots) for various time series. The axes represent time steps from 1 to 100. Below the plot, a caption reads: "Uber's business depends on accurate forecasting. For instance, we use forecasting to predict the reported supply of drivers and demands of riders in the 650+ cities we operate in, to identify when our systems are facing outages, to ensure we always have enough customer coverage." There are also sections for "Popular Articles" and "Sign up for User Engineering updates".

M5 Competition: 1st Place - Feature Engineering

Feature Engineering

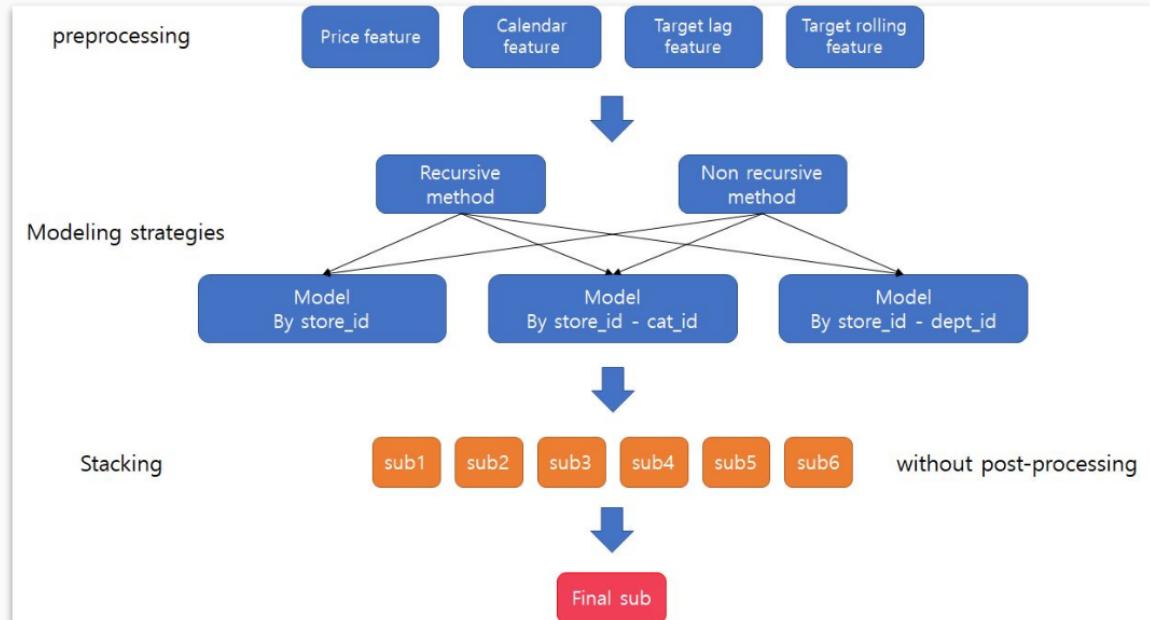
- Price
- Calendar Features
- Lag Features
- Rolling Lags

Machine Learning:

LightGBM

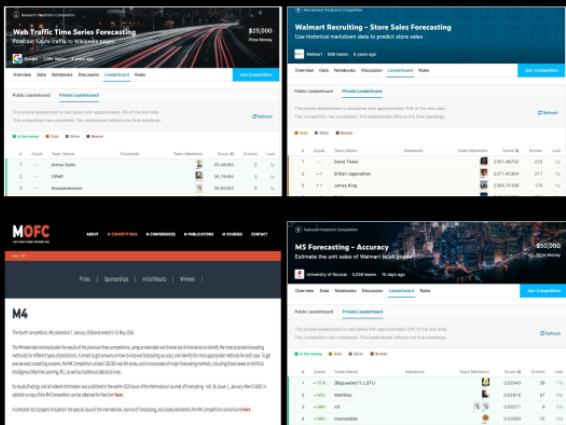
Ensembling (Stacking):

Led to robustness



5 Competition Takeaways

The techniques that will lead to a **high-performance forecasting system.**



1 Feature engineering. 1st Place & 3rd Place in 2020 M5 Walmart Challenge. Calendar-Features, Lags.

2 Experimentation and ensembling. 1st Place in 2020 M5 Walmart Competition. 2nd Place in 2014 Walmart Challenge (almost won it).

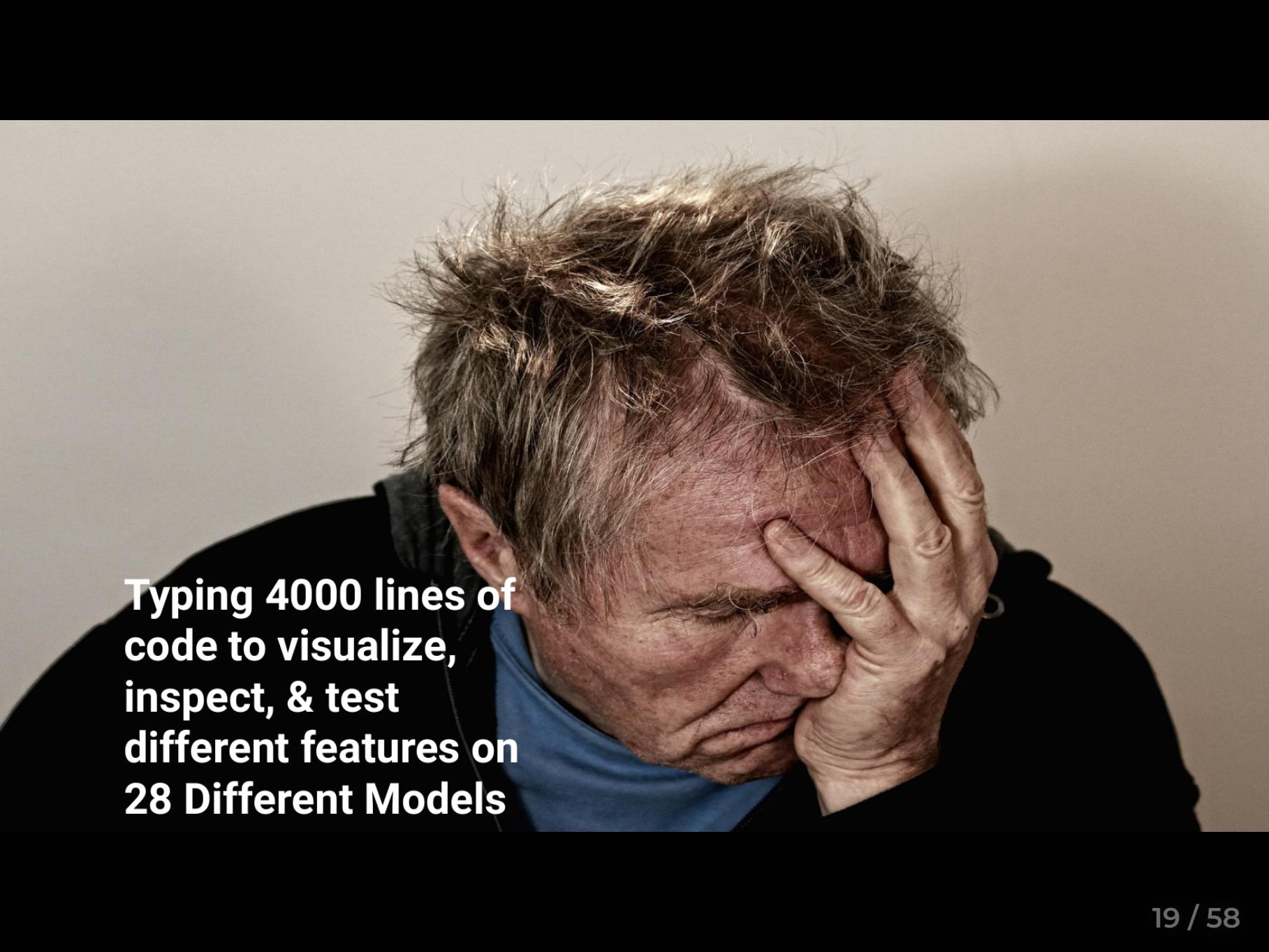
3 Knowledge of key events. Shift in holiday sales improved accuracy 12% in 2014 Walmart Challenge, landing 1st place spot.

4 Machine Learning. Machine Learning claimed 1st Place in 2020 M5 Competition. 2nd Place in 2014 Walmart Challenge.

5 Deep Learning. Recurrent Neural Networks (RNNs) won 2018 M4 & Wikipedia Competitions. Also performed well in 2020 M5 Competition.

Part III

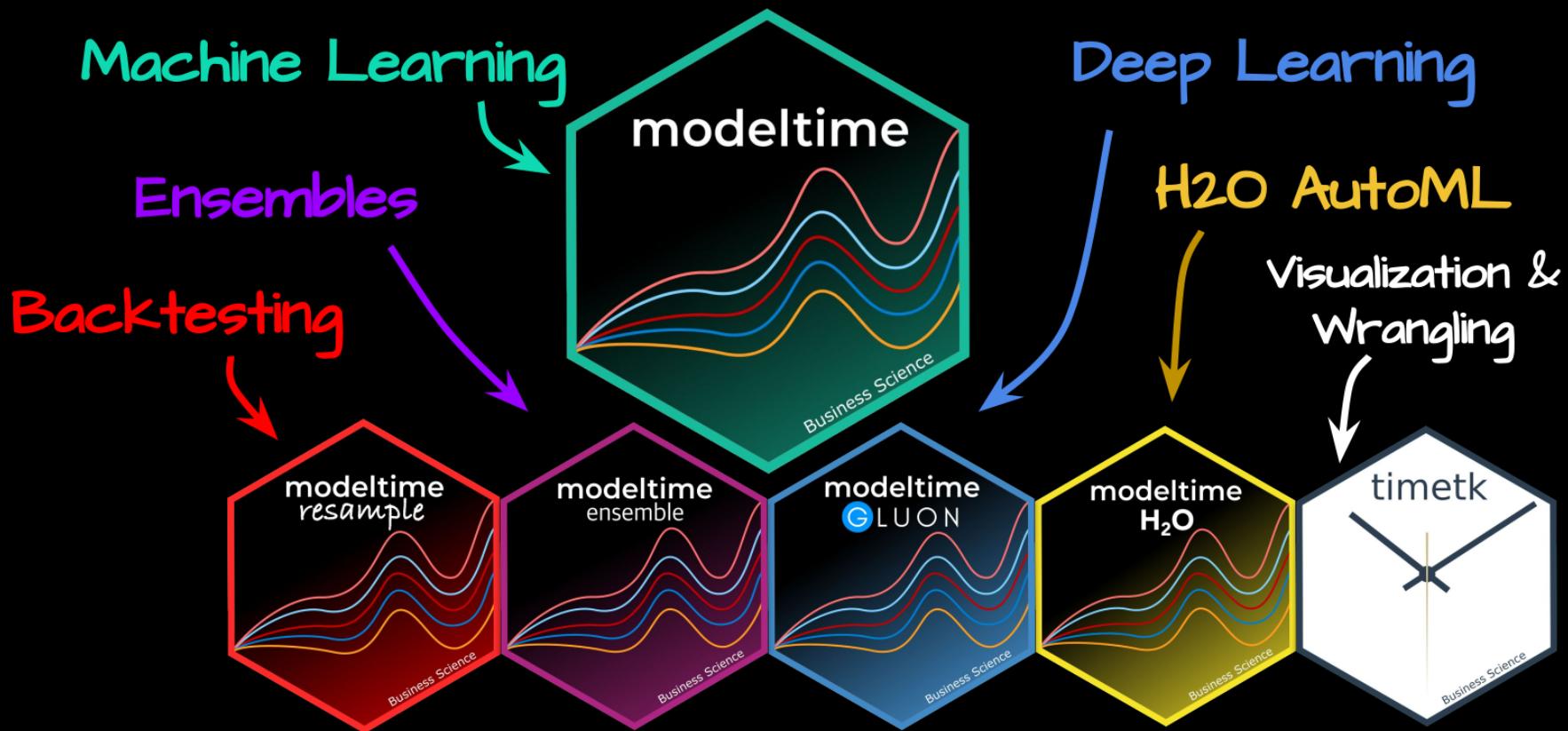
Modeltime

A close-up photograph of a man with light brown hair, wearing a dark jacket over a blue shirt. He is holding his head in his hands, with his fingers resting on his forehead and eyes closed, conveying a sense of stress or exhaustion.

**Typing 4000 lines of
code to visualize,
inspect, & test
different features on
28 Different Models**

A close-up photograph of a man with light brown hair, wearing a dark jacket over a blue shirt. He is holding his right hand to his forehead, with his fingers resting against his temple and brow, suggesting stress or exhaustion. His eyes are partially closed, and he has a weary expression. The background is a plain, light-colored wall.

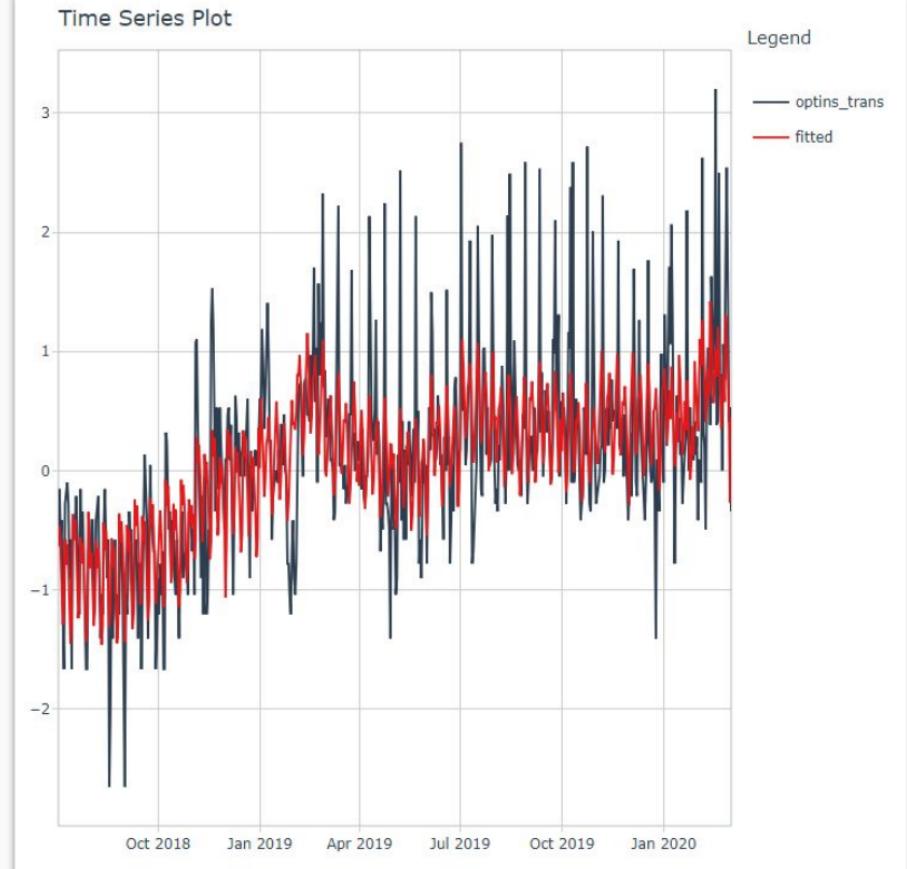
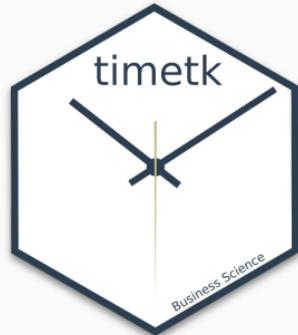
**Typing 2000 lines of
code to make a
multi-model forecast**



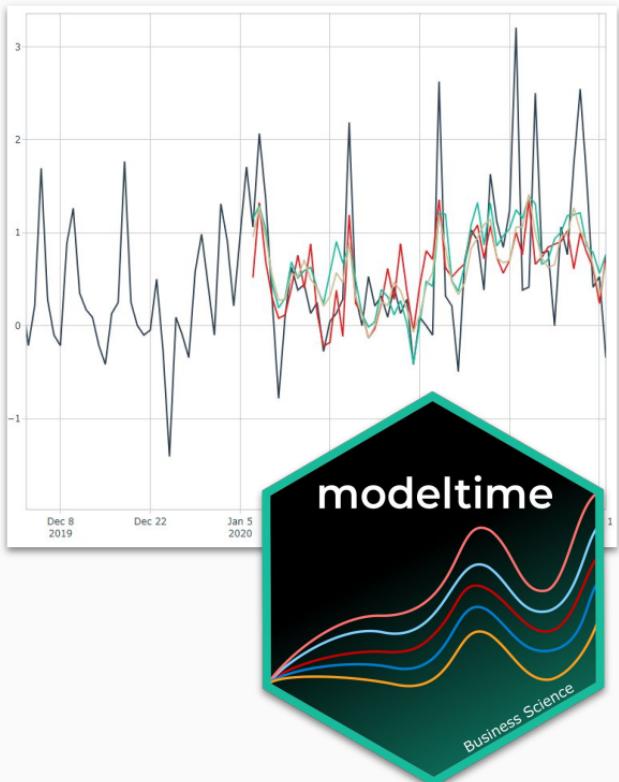
Feature Engineering

```
134 data_prep_signature_tbl %>%
135   plot_time_series_regression(
136     optin_time,
137     .formula = model_formula_seasonality,
138     .show_summary = TRUE
139   )
```

```
Residual standard error: 0.6023 on 580 degrees of freedom
Multiple R-squared:  0.4869,    Adjusted R-squared:  0.4621
F-statistic: 19.65 on 28 and 580 DF,  p-value: < 2.2e-16
```

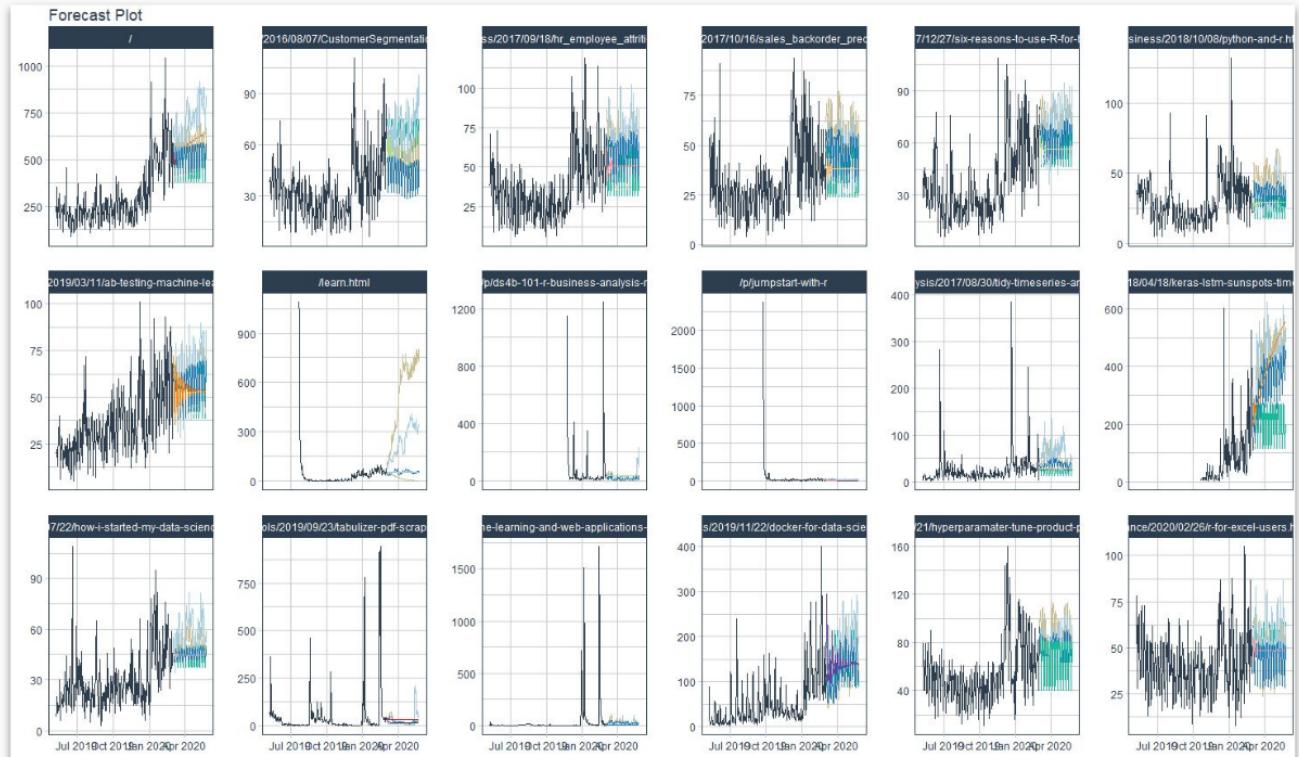
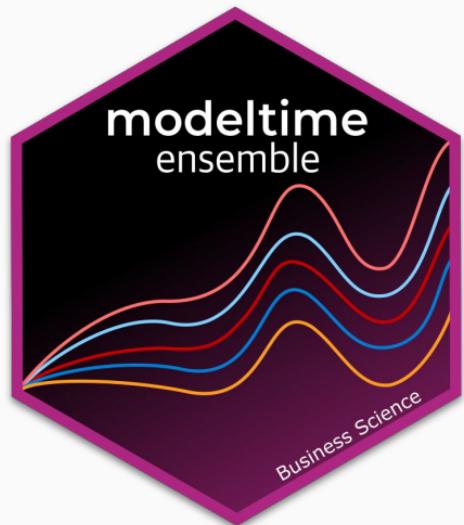


Multi-Model Experiments



.model_id	.model_desc ↑	.type	↓ mae	↑ mape	↓ mase	↓ smape	↑ rmse	↓ rsq
21	PROPHET W/ XGBOOST ERRORS	Test	0.46	1519.18	0.62	85.54	0.63	0.44
14	XGBOOST - Lag	Test	0.48	891.48	0.66	90.59	0.65	0.4
20	PROPHET W/ REGRESSORS	Test	0.49	1435.07	0.66	87.04	0.66	0.39
19	NNAR(2,1,10)[7]	Test	0.51	1377.19	0.69	96.09	0.69	0.33
2	GLMNET - Lag	Test	0.52	1286.65	0.7	98.93	0.7	0.36
16	CUBIST - Lag	Test	0.53	1364.64	0.71	96.51	0.7	0.32
7	KERNLAB - Spline	Test	0.52	1180.45	0.7	97.06	0.71	0.34
8	KERNLAB - Lag	Test	0.53	1200.72	0.72	100.76	0.71	0.34
12	RANDOMFOREST - Lag	Test	0.52	843	0.7	96.48	0.71	0.37
28	SEASONAL DECOMP: REGRESSION WITH ARIMA(2,1,3) ERRORS	Test	0.53	1058.11	0.71	101.56	0.71	0.32
3	EARTH - Spline	Test	0.52	1051.52	0.7	91.72	0.72	0.28
18	NNET - Lag	Test	0.59	1884.91	0.8	92.29	0.72	0.35
22	REGRESSION WITH ARIMA(2,1,1) (0,0,2)[7] ERRORS	Test	0.56	2206.12	0.76	86.77	0.72	0.31
1	GLMNET - Spline	Test	0.54	1222.27	0.73	102.76	0.73	0.3
4	EARTH - Lag	Test	0.52	570.12	0.71	105.52	0.73	0.36
23	ARIMA(2,1,2) W/ XGBOOST ERRORS	Test	0.52	1473.81	0.71	102.71	0.73	0.34
25	TBATS(1, {0,2}, -, {<7,2>, <30,6>, <365,8>})	Test	0.52	990.87	0.71	93.43	0.74	0.3
9	KKNN - Spline	Test	0.56	1279.32	0.76	98.45	0.75	0.25
17	NNET - Spline	Test	0.55	1838.88	0.75	90.6	0.75	0.19
27	SEASONAL DECOMP: ARIMA(3,1,3)	Test	0.57	946.13	0.77	109.56	0.76	0.26

Scalable Forecasting with Ensembles

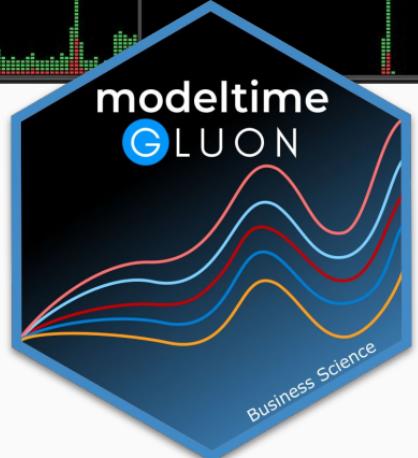


TRUE

```
[25]: t0 <- Sys.time()
model_fit_deepar <- deep_ar(
  id = "id",
  freq = "M",
  prediction_length = 24,
  lookback_length = 36,
  epochs = 1,
  num_batches_per_epoch = 500,
  learn_rate = 0.001,
  num_layers = 3,
  num_cells = 80,
  dropout = 0.10
) %>%
  set_engine("gluonts_deepar") %>%
  fit(value ~ date + id, training(m750_splits))

t1 <- Sys.time()
t1-t0

Execution started at 2021-03-05 17:31:47
Time difference of 44.16673 secs
```

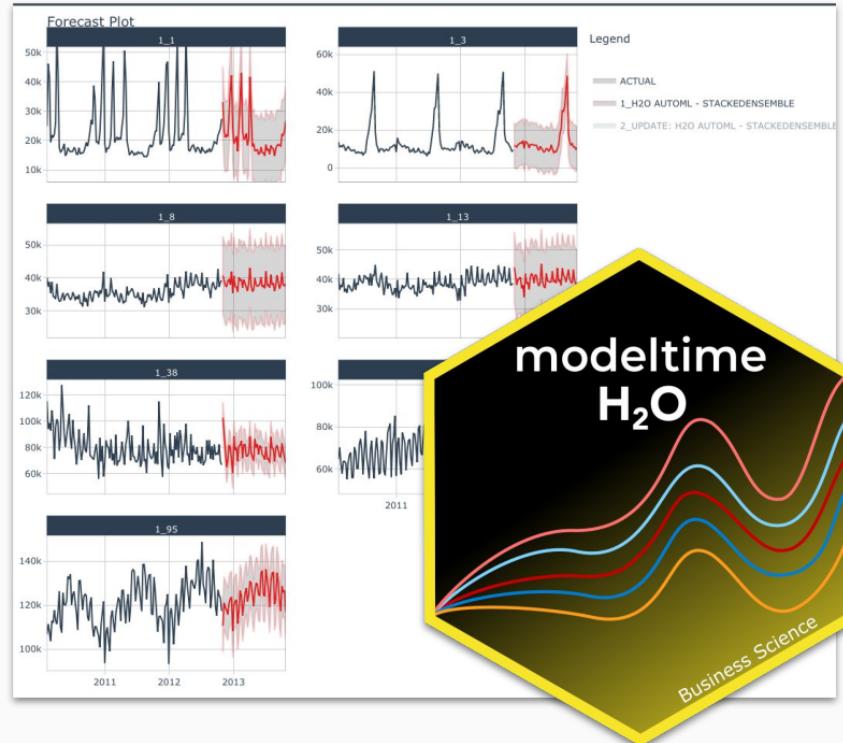


GluonTS Deep Learning for Forecasting

```
# A tibble: 32 x 6
  model_id      mean_residual_devi...    rmse     mse   mae rmsle
  <chr>          <dbl> <dbl>    <dbl> <dbl> <dbl>
1 StackedEnsemble_AllModels_A... 36291288. 6024. 3.63e7 3626. 0.147
2 StackedEnsemble_BestOfFamil... 37802329. 6148. 3.78e7 3709. 0.149
3 XGBoost_grid__1_AutoML_2021... 38139582. 6176. 3.81e7 3767. 0.153
4 XGBoost_grid__1_AutoML_2021... 38195759. 6180. 3.82e7 3738. 0.157
5 XGBoost_grid__1_AutoML_2021... 39154490. 6257. 3.92e7 3959. 0.156
6 XGBoost_grid__1_AutoML_2021... 39757275. 6305. 3.98e7 3953. 0.156
7 XGBoost_3_AutoML_20210406_1... 40117557. 6334. 4.01e7 3841. 0.154
8 XGBoost_grid__1_AutoML_2021... 40380426. 6355. 4.04e7 3915. 0.160
9 GBM_grid__1_AutoML_20210406... 41239637. 6422. 4.12e7 3801. 0.161
10 GBM_1_AutoML_20210406_140637 41610511. 6451. 4.16e7 3994. 0.172
# ... with 22 more rows
```

H₂O.ai

Unlocks the power of
H2O AutoML inside of
Modeltime



Lot's of models available

PROPHET



PyTorch

GLUON

H2O.ai

Lot's of models available

modeltime::prophet_reg()

- Prophet

modeltime::arima_reg()

- Auto ARIMA
- ARIMA

modeltime::exp_smoothing()

- ETS
- CROSTON
- Theta

modeltime::seasonal_reg()

- TBATS
- STLM

modeltime::nnetar()

- NNETAR

modeltime::prophet_boost()

- Prophet + XGBoost

modeltime::arima_boost()

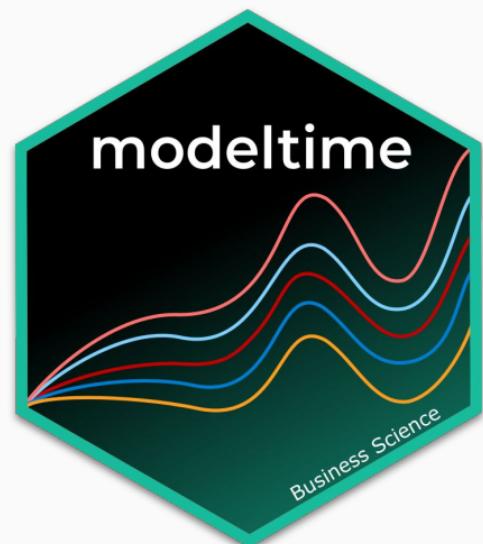
- Auto ARIMA + XGBoost
- ARIMA + XGBoost

modeltime::window_reg()

- Window Function Models

modeltime::naive_reg()

- Naive
- Seasonal Naive



[Modltime Algorithm List](#)

Lot's of models available

`modeltimes.gluonts::deep_ar()`

- GluonTS Deep AR
- Torch Deep AR

`modeltimes.gluonts::nbeats()`

- GluonTS N-BEATS
- GluonTS N-BEATS Ensemble

`modeltimes.gluonts::deep_state()`

- GluonTS Deep State

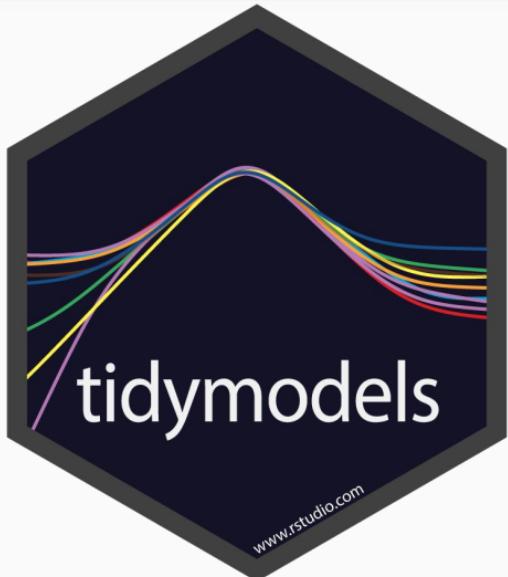
`modeltimes::gp_forecaster()`

- GluonTS GP Forecaster



[Modeltimes Gluonts Algorithm List](#)

Lot's of models available



[Parsnip Models List](#)

XGBoost

Neural Networks

Decision Trees

K-Nearest Neighbors

GAMS

Random Forest

Linear Regression

Support Vector Machines

MARS

& more!

Community contributed Modeltime Extensions!



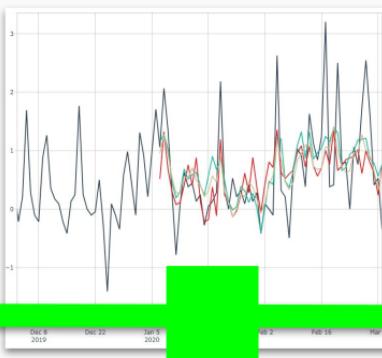
A lot to learn to Time Series

Feature Engineering

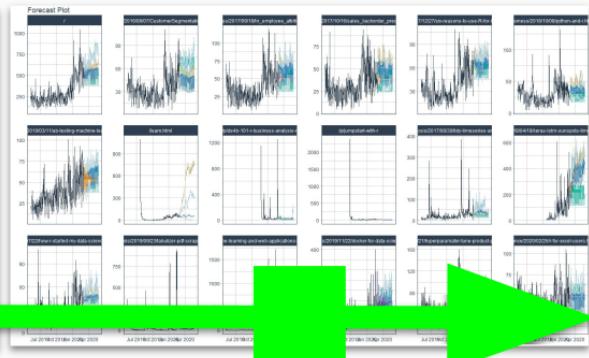
Modeling & Experimentation

model_id	model_desc	type	max	mean	min	range
21	PREDICT W/NGHOOST ERRORS	Test	0.46	1518.18	0.0	85.14
14	SGNOOOT - Lag	Test	0.48	891.48	0.0	90.19
30	PREDICT W/ REGRESSORS	Test	0.49	1405.07	0.06	67.04
18	SNARBLI,1.1(0)	Test	0.51	1717.18	0.09	96.09
1	GLMNNT - Lag	Test	0.52	1286.65	0.7	88.83
14	CUBIST - Lag	Test	0.53	1364.64	0.7	86.11
7	KEDGAS - Spline	Test	0.53	1189.41	0.7	97.06
8	KEDGAS - Lag	Test	0.53	1268.72	0.7	106.76
12	RANDOMFOREST - Lag	Test	0.52	843.07	0.7	96.44
28	SEASONAL, DECOMP REGRESSION WITH ARIMA(2,1,3) ERRORS	Test	0.53	1031.11	0.71	101.56
3	EARTH - Spline	Test	0.52	1031.52	0.7	91.72
14	SGNOOOT - Spline	Test	0.59	1884.91	0.8	92.29
22	REGRESSION WITH ARIMA(2,1,3) 0.0(2,1) ERRORS	Test	0.56	1298.12	0.76	86.77
1	GLMNNT - Spline	Test	0.54	1222.77	0.7	102.76
4	EARTH - Lag	Test	0.57	579.12	0.71	105.52
23	ARIMA(2,1,3) W/ NGHOOST ERRORS	Test	0.52	1473.81	0.71	102.71
27	TRACIN, (0,2), <1.2,>,<35.0>	Test	0.57	1031.77	0.7	93.43
3	KNNNN - Spline	Test	0.56	1189.77	0.7	88.45
SEASONAL, DECOMP ARIMA(2,1,3)				0	100.00	

Tuning & Cross Validation

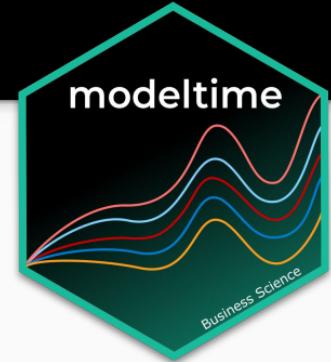


Scaling to 1000s of Time Series



Streamlined Forecasting Workflow

MODELTIME Workflow



Create Modeltime Table modeltime_table()

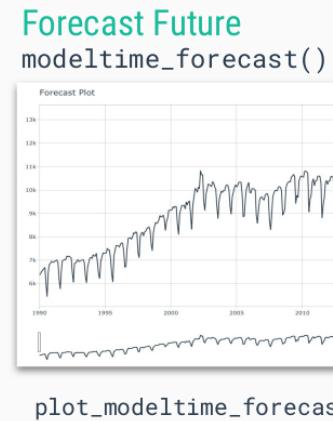
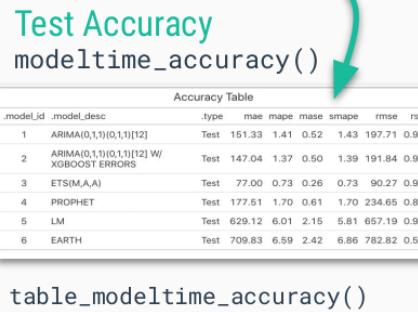
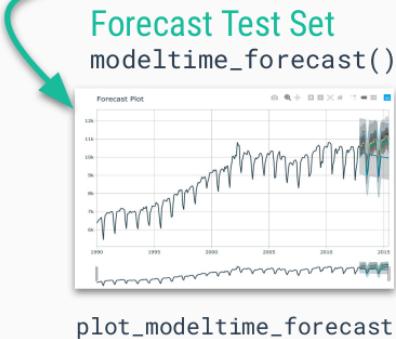
```
# Modeltime Table
# A tibble: 5 x 3
  .model_id .model   .model_desc
  <int> <list> <chr>
1 1 <fit[+]> ARIMA(0,1,1)(0,1,1)[12]
2 2 <fit[+]> ARIMA(0,1,1)(0,1,1)[12] W/ XGBOOST ERRORS
3 3 <fit[+]> ETS(M,A,A)
4 4 <fit[+}> PROPHET
5 5 <fit[+}> LM
5 5 <fit[+}> EARTH
```

Calibrate modeltime_calibrate()

```
# Modeltime Table
# A tibble: 6 x 5
  .model_id .model   .model_desc   .type .calibration_data
  <int> <list> <chr> <list>
1 1 <fit[+}> ARIMA(0,1,1)(0,1,1)[12] Test <tibble [31 x 4]>
2 2 <fit[+}> ARIMA(0,1,1)(0,1,1)[12] W/ XGBOOST ERRORS Test <tibble [31 x 4]>
3 3 <fit[+}> ETS(M,A,A) Test <tibble [31 x 4]>
4 4 <fit[+}> PROPHET Test <tibble [31 x 4]>
5 5 <fit[+}> LM Test <tibble [31 x 4]>
6 6 <fit[+}> EARTH Test <tibble [31 x 4]>
```

Refit modeltime_refit()

```
# Modeltime Table
# A tibble: 6 x 5
  .model_id .model   .model_desc   .type .calibration_data
  <int> <list> <chr> <list>
1 1 <fit[+}> ARIMA(0,1,1)(0,1,1)[12] Test <tibble [31 x 4]>
2 2 <fit[+}> ARIMA(0,1,1)(0,1,1)[12] W/ XGBOOST ERRORS Test <tibble [31 x 4]>
3 3 <fit[+}> ETS(M,A,A) Test <tibble [31 x 4]>
4 4 <fit[+}> PROPHET Test <tibble [31 x 4]>
5 5 <fit[+}> LM Test <tibble [31 x 4]>
6 6 <fit[+}> EARTH Test <tibble [31 x 4]>
```



Part IV

Recursivity

Why Recursive Forecasting? It started with ARIMA

Topics

Autocorrelation & Lags

AR & MA Models (ARIMA)

Differencing (ARIMA)

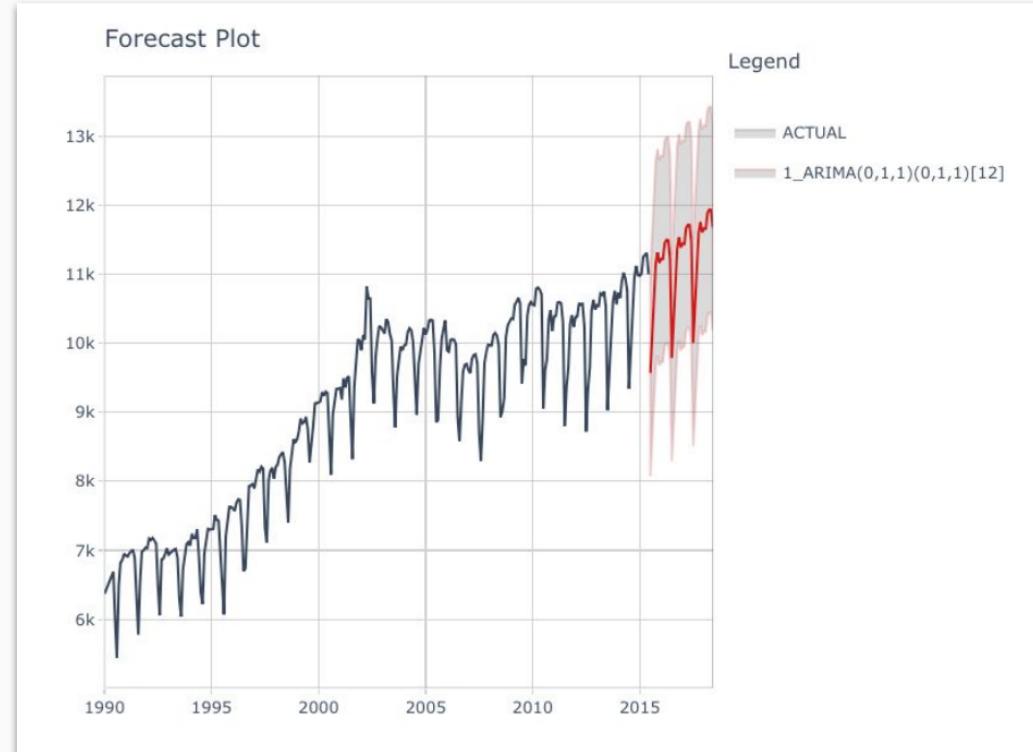
Seasonal ARIMA (SARIMA)

X-Regs (ARIMAX)

Automated Model Selection

Strengths & Weaknesses

What is ARIMA?



Autocorrelation and Lags

Time Series often have relationship to what happened previously. This relationship can be exploited.

Lagged correlations can be used to iteratively predict steps forward. And lagged errors as well.

Auto ARIMA automates the process of finding the optimal model that represents the structure using both Lagged Features and Lagged Error terms.

Date	Value	Value _{t-1}	Value _{t-2}
1/1/2017	200	NA	NA
1/2/2017	220	200	NA
1/3/2017	215	220	200
1/4/2017	230	215	220
1/5/2017	235	230	215
1/6/2017	225	235	230
1/7/2017	220	225	235
1/8/2017	225	220	225
1/9/2017	240	225	220
1/10/2017	245	240	225

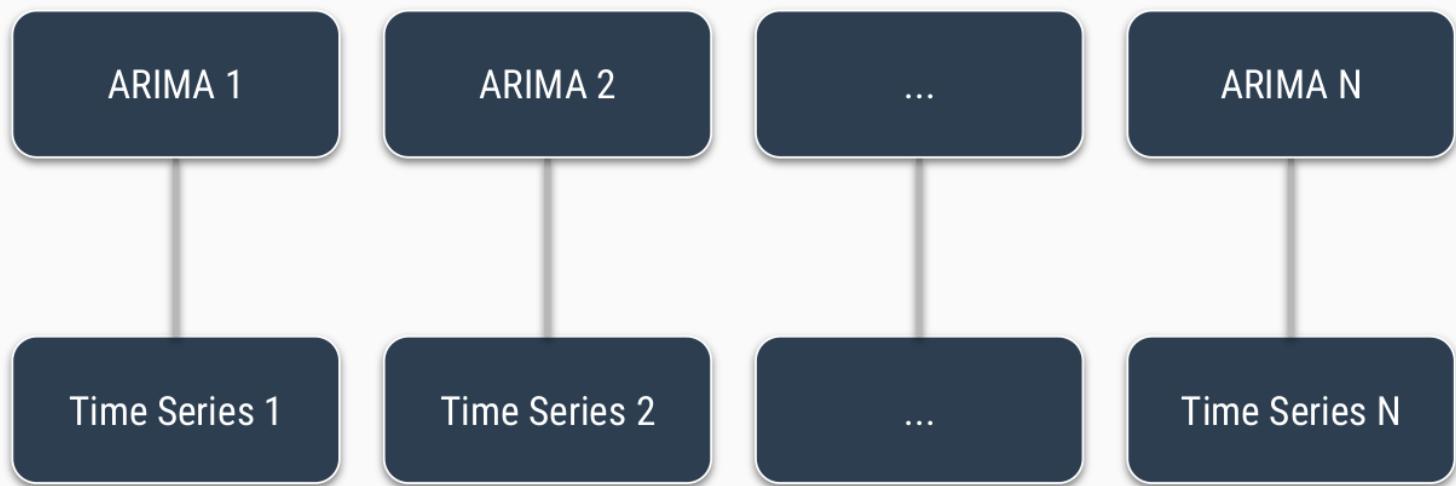


$$\text{value}_t \sim b_1 \text{value}_{t-1} + b_2 \text{value}_{t-2}$$

Problem

ARIMA has a **big problem**

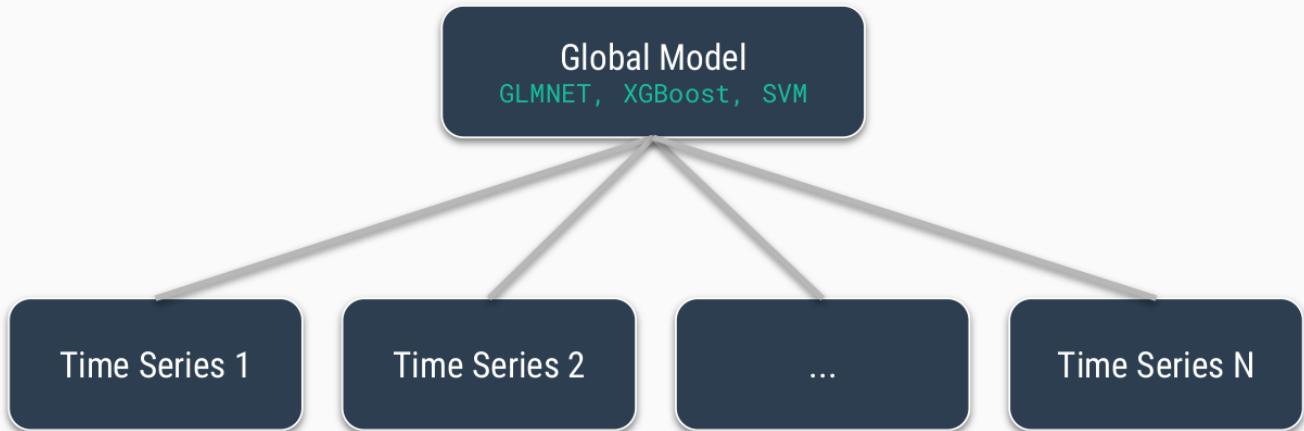
18 Time Series requires 18 different ARIMA Models



A photograph of a woman with dark hair, wearing a black top, sitting at a wooden desk. She has her left hand resting on her chin in a contemplative pose. In the foreground, a silver-colored alarm clock sits on the desk, its hands pointing to approximately 10:10. The background is softly blurred.

ARIMA can leave you
waiting for days for your
forecast job to finish...

Solution: Global Models

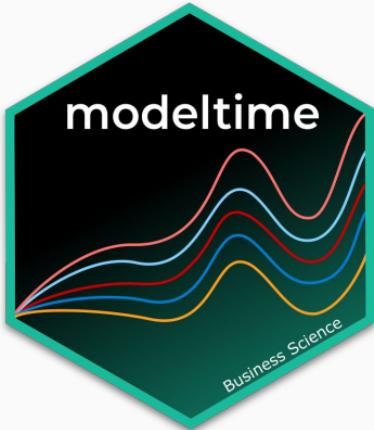


This sounds great until you realize, your future data has Missing Values

```
> data_future_tbl %>% select(source_key, date, value, starts_with("value_lag"))
# A tibble: 432 x 27
   source_key      date    value value_lag1 value_lag2 value_lag3 value_lag4 value_lag5 value_lag6 value_lag7 value_lag8
   <chr>        <date>   <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
1 ELEC.GEN.ALL... 2021-02-01     NA 350815 344970 302702 313910 334270 399504 414243 352766
2 ELEC.GEN.ALL... 2021-03-01     NA     NA 350815 344970 302702 313910 334270 399504 414243
3 ELEC.GEN.ALL... 2021-04-01     NA     NA     NA 350815 344970 302702 313910 334270 399504
4 ELEC.GEN.ALL... 2021-05-01     NA     NA     NA     NA 350815 344970 302702 313910 334270
5 ELEC.GEN.ALL... 2021-06-01     NA     NA     NA     NA     NA 350815 344970 302702 313910
6 ELEC.GEN.ALL... 2021-07-01     NA     NA     NA     NA     NA     NA 350815 344970 302702
7 ELEC.GEN.ALL... 2021-08-01     NA     NA     NA     NA     NA     NA     NA 350815 344970
8 ELEC.GEN.ALL... 2021-09-01     NA     NA     NA     NA     NA     NA     NA     NA 350815
9 ELEC.GEN.ALL... 2021-10-01     NA     NA     NA     NA     NA     NA     NA     NA     NA
10 ELEC.GEN.ALL... 2021-11-01    NA     NA     NA     NA     NA     NA     NA     NA     NA
# ... with 422 more rows, and 16 more variables: value_lag9 <dbl>, value_lag10 <dbl>, value_lag11 <dbl>, value_lag12 <dbl>,
#   value_lag13 <dbl>, value_lag14 <dbl>, value_lag15 <dbl>, value_lag16 <dbl>, value_lag17 <dbl>, value_lag18 <dbl>,
#   value_lag19 <dbl>, value_lag20 <dbl>, value_lag21 <dbl>, value_lag22 <dbl>, value_lag23 <dbl>, value_lag24 <dbl>
>
```

`modeltime::recursive()`

Handles the **iterative updating** of lags with predictions.



```
> data_future_tbl %>% select(source_key, date, value, starts_with("value_lag"))
# A tibble: 432 x 27
   source_key     date    value value_lag1 value_lag2 value_lag3 value_lag4 value_lag5 value_lag6 value_lag7 value_lag8
   <chr>       <date>   <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 ELEC.GEN.ALL... 2021-02-01    NA  350815  344970  302702  313910  334270  399504  414243  352766
2 ELEC.GEN.ALL... 2021-03-01    NA    NA  350815  344970  302702  313910  334270  399504  414243
3 ELEC.GEN.ALL... 2021-04-01    NA    NA    NA  350815  344970  302702  313910  334270  399504
4 ELEC.GEN.ALL... 2021-05-01    NA    NA    NA    NA  350815  344970  302702  313910  334270
5 ELEC.GEN.ALL... 2021-06-01    NA    NA    NA    NA    NA  350815  344970  302702  313910
6 ELEC.GEN.ALL... 2021-07-01    NA    NA    NA    NA    NA    NA  350815  344970  302702
7 ELEC.GEN.ALL... 2021-08-01    NA    NA    NA    NA    NA    NA    NA  350815  344970
8 ELEC.GEN.ALL... 2021-09-01    NA    NA    NA    NA    NA    NA    NA    NA  350815
9 ELEC.GEN.ALL... 2021-10-01    NA    NA    NA    NA    NA    NA    NA    NA    NA
10 ELEC.GEN.ALL... 2021-11-01   NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
# ... with 422 more rows, and 16 more variables: value_lag9 <dbl>, value_lag10 <dbl>, value_lag11 <dbl>, value_lag12 <dbl>,
#   value_lag13 <dbl>, value_lag14 <dbl>, value_lag15 <dbl>, value_lag16 <dbl>, value_lag17 <dbl>, value_lag18 <dbl>,
#   value_lag19 <dbl>, value_lag20 <dbl>, value_lag21 <dbl>, value_lag22 <dbl>, value_lag23 <dbl>, value_lag24 <dbl>
>
```

Part V

Scalability

Customer ID	Date	Revenue	Event 1
1	2021-01-01	0	New Years Day
1	2021-01-02	400	
1	2021-01-03	250	
1	2021-01-04	300	
...
1000	2021-01-01	100	New Years Day
1000	2021-01-02	2050	
1000	2021-01-03	3575	
1000	2021-01-04	4035	

Panel 1

x1000

Panel 1000

3-Pro Tips

When dealing with **large scale.**

Pro-Tip 1: Start with 1 Time Series

Forecasting 1000's of time series is hard.

- Many Time Series
- Many Models

Simplify your problem.



Pro-Tip 2: Scale Efficiently

Time is your greatest resource. Use it wisely.

Strategy:

- **Test Many Models.** Hurts Time,
Increases Accuracy
- **Parallel Processing.** Helps Time.



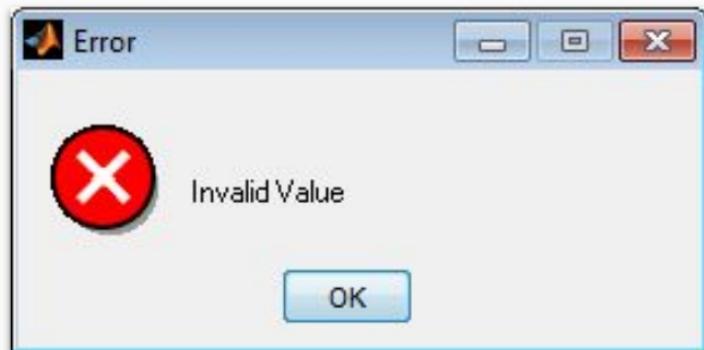
Pro-Tip 3: Anticipate errors.

People don't plan to fail. They fail to plan.

Anticipate that something will go wrong.

Have a plan.

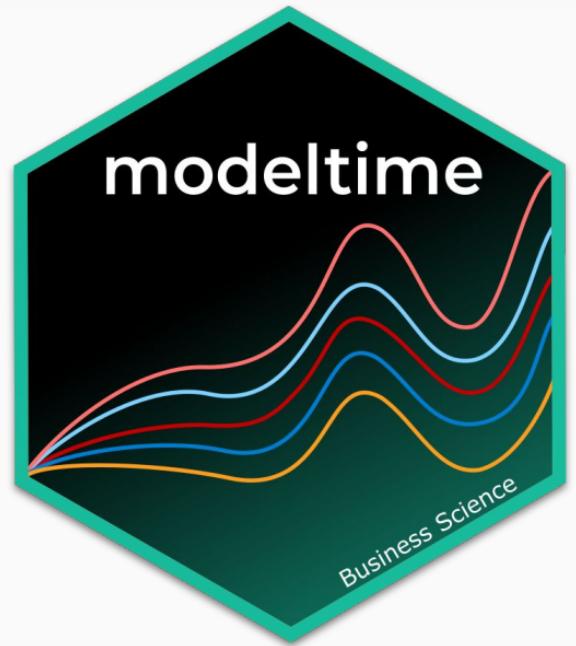
- Implement error reports
- Review all errors



Developed for Scale

We have 2 Systems to Scale:

1. **Global Modeling:** 100X faster
2. **Nested Modeling:** Slower, More Accurate





Panel 1

x1000

Panel 1 000

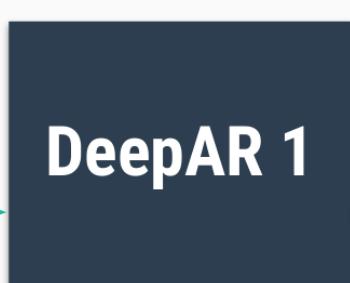
Customer ID	Date	Revenue	Event 1
1	2021-01-01	0	New Years Day
1	2021-01-02	400	
1	2021-01-03	250	
1	2021-01-04	300	
...
1000	2021-01-01	100	New Years Day
1000	2021-01-02	2050	
1000	2021-01-03	3575	
1000	2021-01-04	4035	

XGBOOST 1

Global Models

Customer ID	Date	Revenue	Event 1
1	2021-01-01	0	New Years Day
1	2021-01-02	400	
1	2021-01-03	250	
1	2021-01-04	300	
...
1000	2021-01-01	100	New Years Day
1000	2021-01-02	2050	
1000	2021-01-03	3575	
1000	2021-01-04	4035	

1000 Time Series



1 Global Model



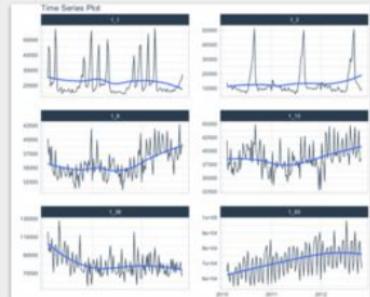
1000 Forecasts

Nested Forecasting

- Iterative Analysis
- Built for Scale
(Parallel Processing)
- Handles Errors
Gracefully
- Key Concepts: Logging

Nested Forecasting

Many Time Series



Many Models



Best Forecasts

Iterative
forecasting
using Nested
Modeltime
Tables



Panel 1

Customer ID	Date	Revenue	Event 1
1	2021-01-01	0	New Years Day
1	2021-01-02	400	
1	2021-01-03	250	
1	2021-01-04	300	
...
1000	2021-01-01	100	New Years Day
1000	2021-01-02	2050	
1000	2021-01-03	3575	
1000	2021-01-04	4035	

ARIMA 1

x1000

Panel 1000

ARIMA 1000

x1000

Extremely easy to **run in parallel**.

Parallel Processing

```
111 # REFIT ----  
112  
113 parallel_start(2) ← Setup Clusters  
114  
115 nested_modeltimes_refit_tbl <- best_nested_modeltimes_tbl %>%  
116   modeltimes_nested_refit(  
117     control = control_nested_refit(  
118       verbose    = TRUE,  
119       allow_par = TRUE  
120     )  
121   ) ← Control Processing  
122
```

Logging & Error Handling

Pushes **everything** you need to the most expensive operation. Proceeds even when errors occur.

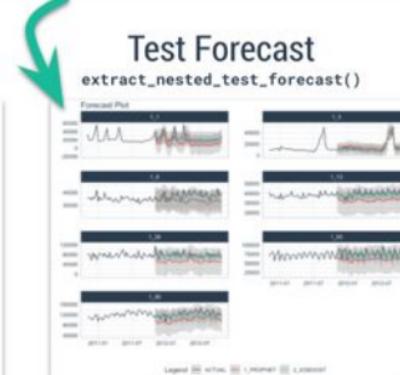
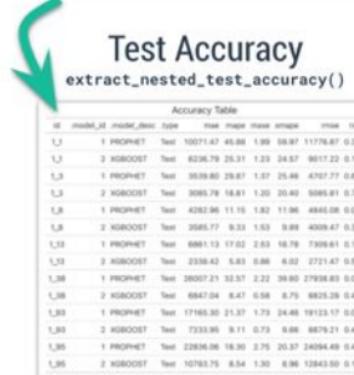
Benefit 1: Can retrieve key information instantly.

Benefit 2: Can check error logs & isolate edge cases where your analysis fails

Extracting Nested Modeltime Table Logs

Nested Modeltime Table

```
# Nested Modeltime Table
# Trained on: .actual_data | Model Errors: [0]
# A tibble: 7 x 5
#   id   .actual_data     .future_data     .splits     .modeltime_tables
#   <fct> <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 1 1_1   <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 2 1_3   <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 3 1_8   <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 4 1_13  <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 5 1_38  <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 6 1_93  <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
# 7 1_95  <tibble [104 x 2]> <tibble [52 x 2]> <split [52/52]> <mdl_time_tbl [1 x 5]>
```



Error Reporting

```
extract_nested_error_report()
```

```
# A tibble: 4 x 4
#   id   .model_id .model_desc .error_desc
#   <fct> <int> <chr>      <chr>
# 1 1_1    2 BOOST_TREE  "data" has class 'character' and length 52.\n  'data...
# 2 1_3    2 BOOST_TREE  "data" has class 'character' and length 52.\n  'data...
# 3 1_8    1 BOOST_TREE  "'x' should be an 'rsplit' object"
# 4 1_8    2 BOOST_TREE  "'x' should be an 'rsplit' object"
```

Contains Logged Attributes for fast extraction

Thank you!