Christopher M. Pravata
SMI Data Assessment

Question 1 Meta:

Meta callouts: Z-Score threshold (1.25), Winsorization, time series and box plots, MLM feedback.

Year: 2018 Month: 1

Year: 2018 Month: 2

Year: 2018 Month: 11

Year: 2020 Month: 10

Year: 2020 Month: 12

Year: 2021 Month: 6

Year: 2021 Month: 7

Year: 2021 Month: 8

Year: 2021 Month: 9

Additional insights:

- 90% Winsorization of revenue (caps numeric outliers) highlights 2020 month 7. ● Gaussian Process, Random forest, and Additive regression report highest MAPE (Mean absolute percentage error) when forecasting month 7 for the transposed dataset (uncertainty). ● Plotly Box and Whisker Plot identifies 2020 month 12 as an anomaly outside of the upper-fence 607.89M.
- Additive regression model (chart attached) forecasts Month 11 revenue overtaking Month 12 for the first time in next year (first occurrence in dataset).
- While all metrics periods stated above are noteworthy, 2020 Month 12 and 2021 Month 7 are the most confident callouts due to synergistic signals from plots, Z-score threshold, winsorization and machine learning feedback.

Question 1 Twitter:

Twitter callouts: Simultaneous Z-Score threshold (1.25) and Winsorization failures.

Year: 2018 Month: 11

Year: 2019 Month: 2
Year: 2020 Month: 3

Year: 2020 Month: 9

Year: 2021 Month: 1

Year: 2021 Month: 4

Year: 2021 Month: 5

Year: 2021 Month: 6

Additional insights:

- Every month in Q2 2021 is highlighted for noteworthy positive YOY trends. ● While all metrics periods stated above are noteworthy, 2020 Month 9 and 2021 Month 1 are the most confident callouts due to synergistic signals from plots, Z-score threshold, and Winsorization.

Question 2

Months 1- 9 ~0%. (ASSUMPTION: > .45% or < -.45%)

Months 10-12 last three months low single digits (ASSUMPTION > or equal to 5%, < or equal to -5%)

Amazon Callouts: 2021 Month 2. 2021 Month 3. 2021 Month 8. 2021 Month 9.

MONTH

1 0.00% 0.00% 0.14% 0.29%

2 0.00% 0.00% 0.12% 0.44%

3 0.00% 0.03% 0.12% 0.49%

4 0.00% 0.16% 0.32% 0.20%

5 0.00% 0.14% 0.23% 0.21%

6 0.00% 0.06% 0.19% 0.15%

7 0.00% 0.19% 0.24% 0.18%

8 0.00% 0.18% 0.22% -0.99%

9 0.00% 0.10% 0.23% -0.99%

10 0.00% 0.10% 0.28% -1.96%

11 0.00% 0.13% 0.35%
12 0.00% 0.17% 0.58%

Google Callouts: 2021 Month 9 and 2021 Month 10.

MONTH

1 0.03% 0.05% 0.07% 0.18%

2 0.04% 0.07% 0.08% 0.10%

3 0.05% 0.12% 0.14% 0.14%

4 0.06% 0.17% 0.04% 0.17%

5 0.10% 0.16% 0.10% 0.22%

6 0.09% 0.16% 0.08% 0.11%

7 0.19% 0.22% 0.15% 0.05%

8 0.21% 0.20% 0.11% 0.06%

9 0.13% 0.15% 0.10% 1.00%

10 0.08% 0.18% 0.09% 150.00%

11 0.10% 0.11% 0.08%

12 0.08% 0.06% 0.10%

Roku Callouts: All of 2018 and 2019. 2021 Month 1. 2021 Month 2. 2021 Month 5.

MONTH

1 -5.00% -5.00% 0.00% -0.45%

2 -5.00% -5.00% 0.00% -1.63%

3 -5.00% -5.00% 0.00% -0.20%

4 -5.00% -5.00% 0.00% -0.26%

5 -5.00% -5.00% 0.00% -0.90%

6 -5.00% -5.00% 0.00% -0.44%

7 -5.00% -5.00% 0.00% -0.04%

8 -5.00% -5.00% 0.00% -0.27%

9 -5.00% -5.00% 0.00% -0.03%

10 -5.00% -5.00% 0.00% 0.00%

11 -5.00% -5.00% -0.44%
12 -5.00% -5.00% -0.60%

Pinterest Callouts: All of 2021

MONTH

1 0.00% 0.00% 0.00% 20.00%

2 0.00% 0.00% 0.00% 20.00%

3 0.00% 0.00% 0.00% 20.00%

4 0.00% 0.00% 0.00% 20.00%

5 0.00% 0.00% 0.00% 20.00%

6 0.00% 0.00% 0.00% 20.00%

7 0.00% 0.00% 0.00% 20.00%

8 0.00% 0.00% 0.00% 20.00%

9 0.00% 0.00% 0.00% 21.00%

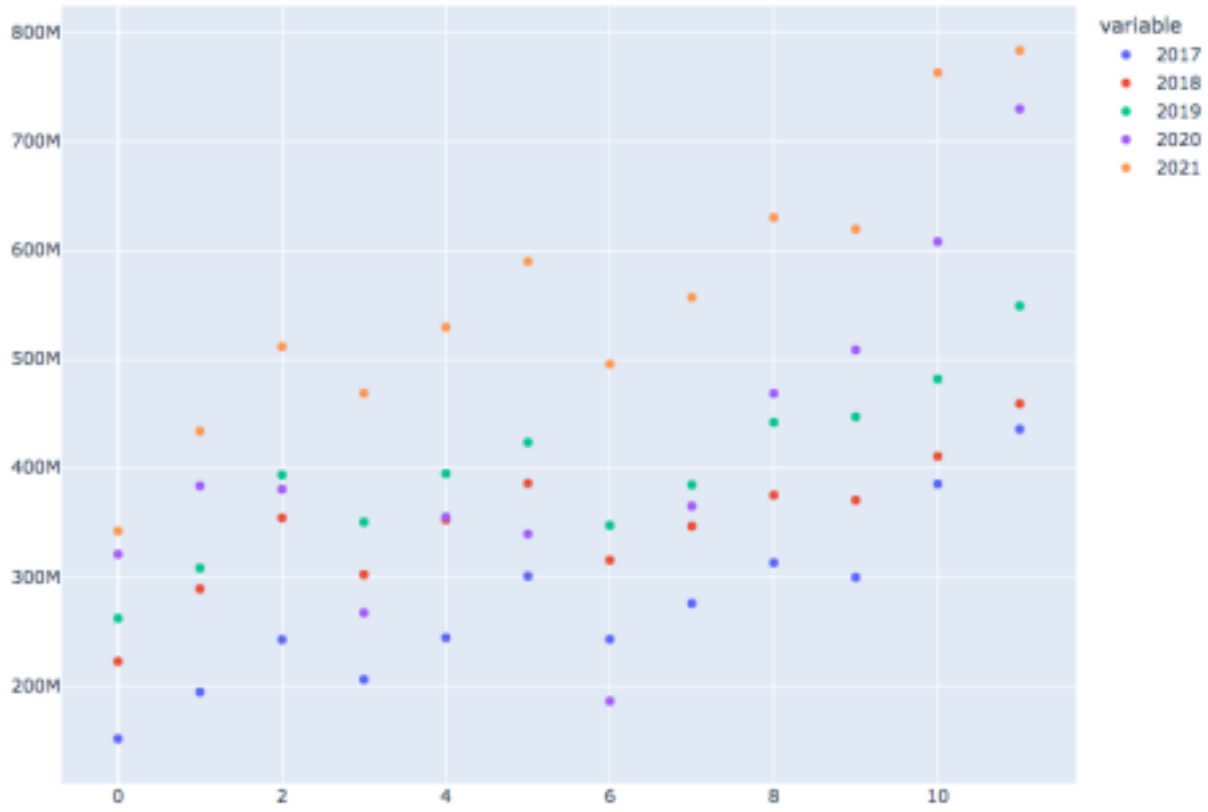
10 0.00% 0.00% 0.00% 29.00%

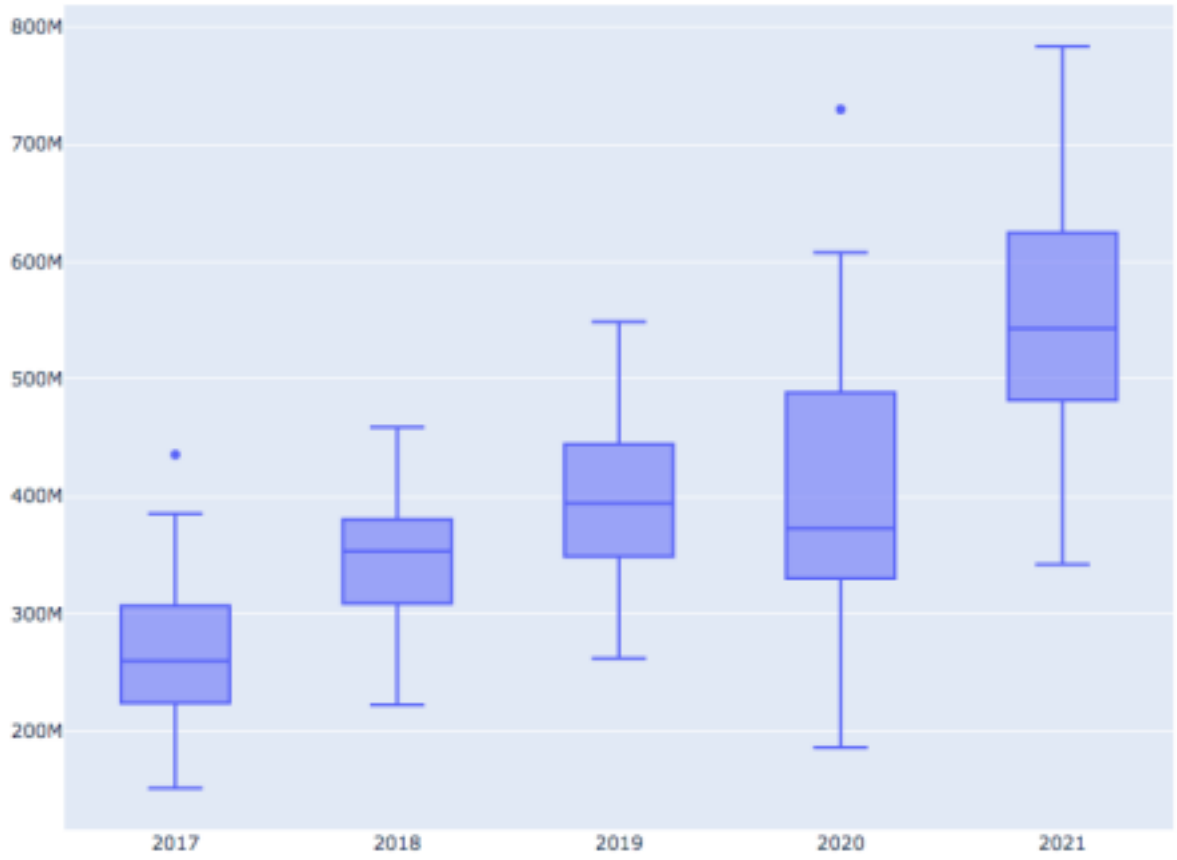
11 0.00% 0.00% 0.00%

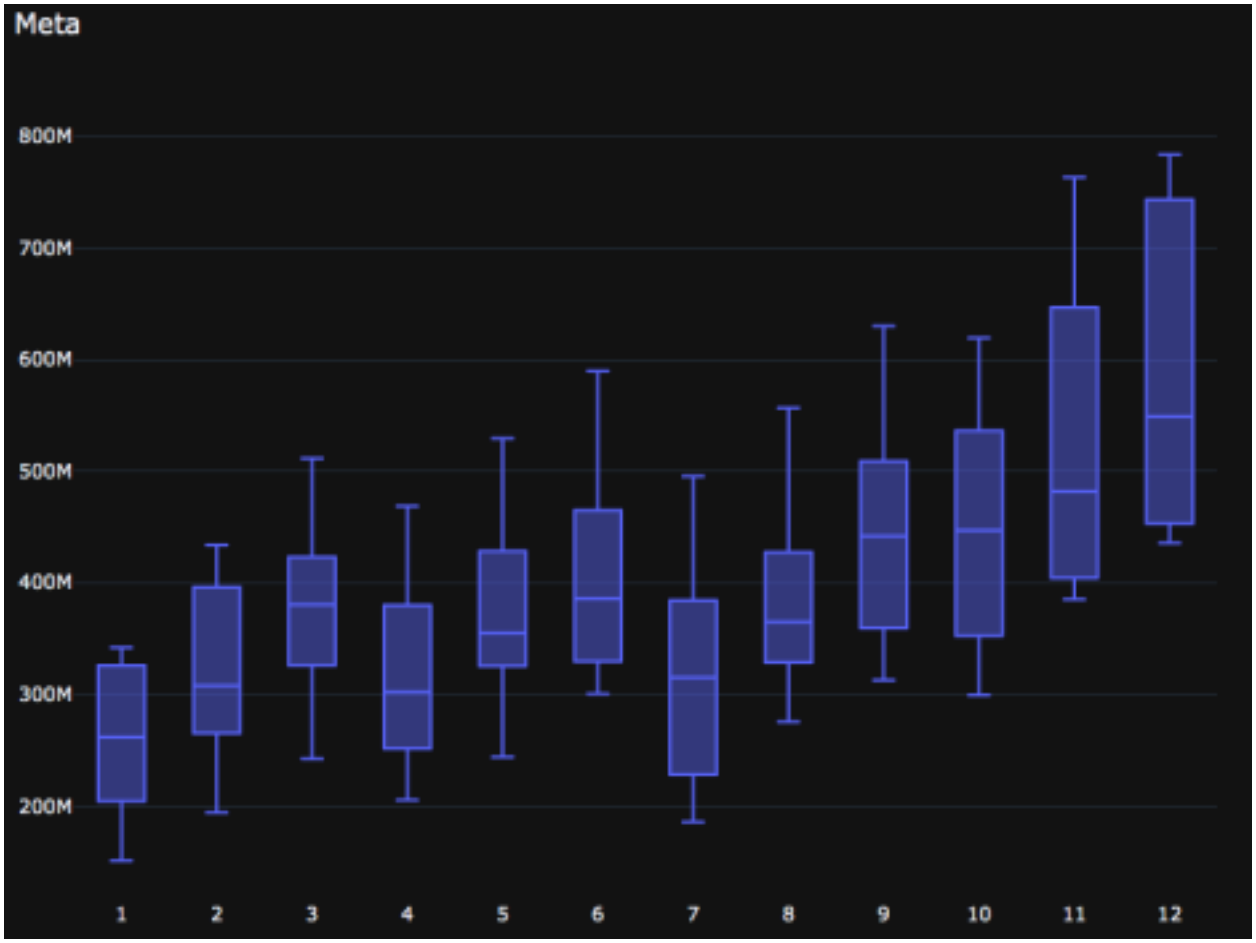
12 0.00% 0.00% 0.00%

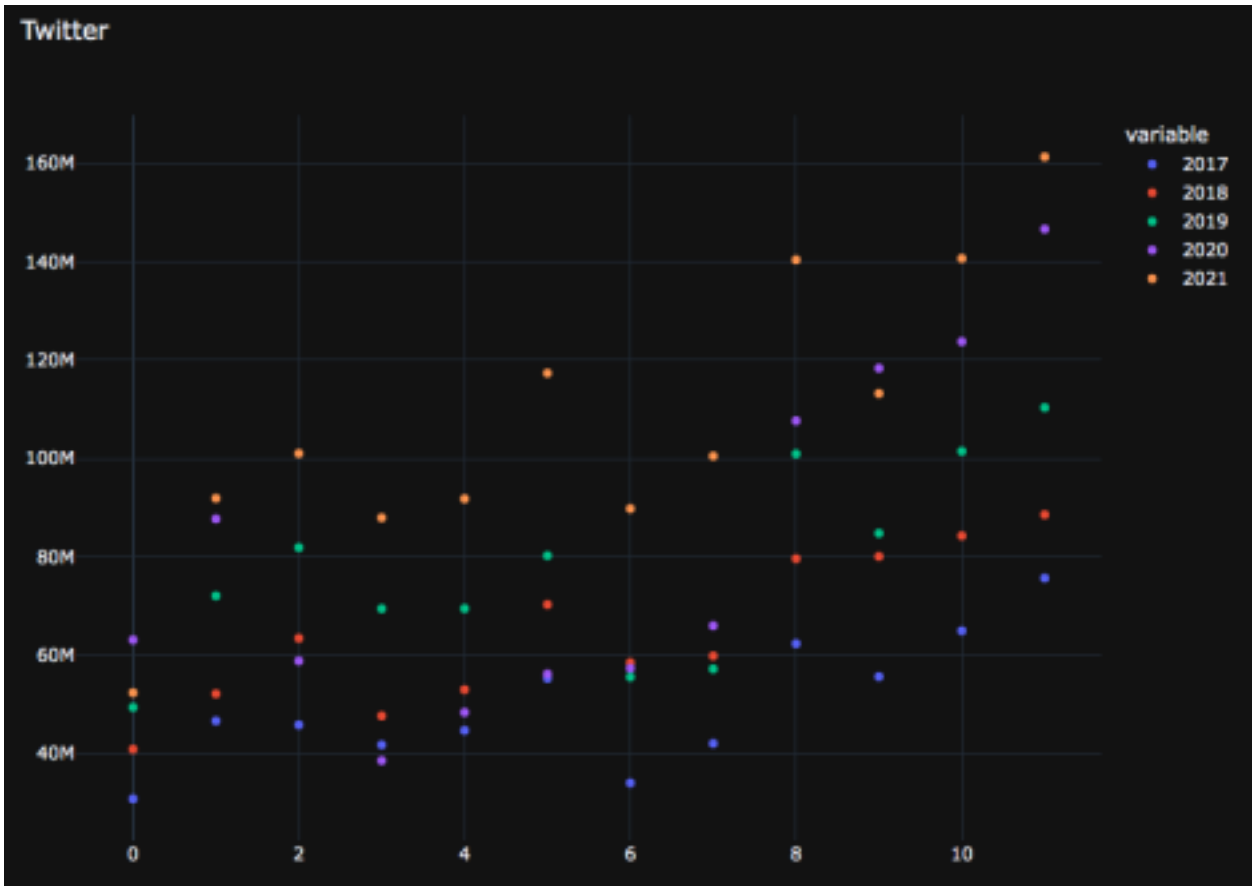
Additional:

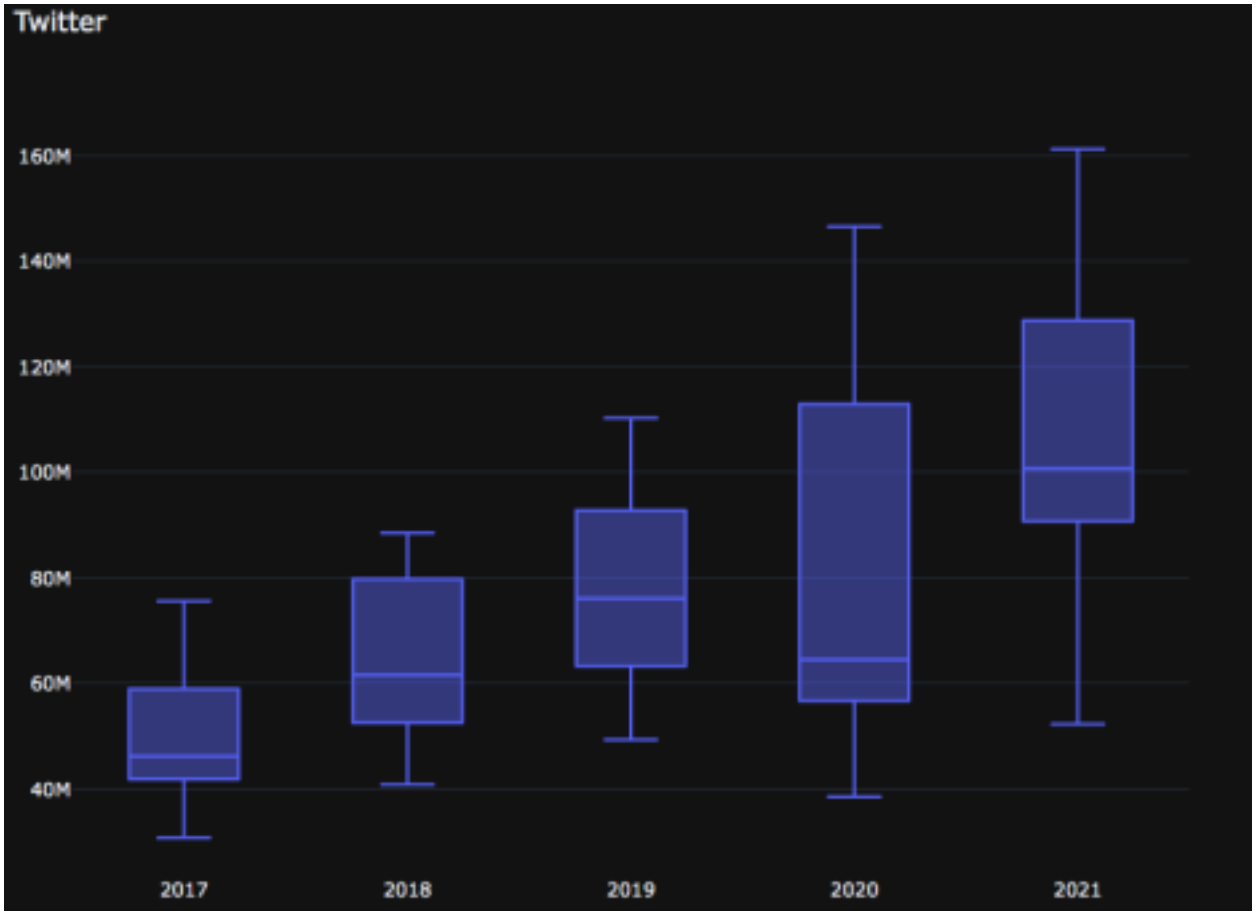
```
import pandas as pd

data = pd.read_csv('file:///Users/chrispravata/Downloads/metasmitransposed.csv') df
= pd.read_csv('file:///Users/chrispravata/Downloads/MetaREVSMI.csv') twitter_data =
pd.read_csv('file:///Users/chrispravata/Downloads/TwitterSMItable.csv')
twitter_data_transposed =
pd.read_csv('file:///Users/chrispravata/Downloads/TwitterSMITransposed.csv')
data.head()
print(data)
# plot
import plotly.express as px
fig = px.scatter(df)
fig.show()
fig = px.box(df)
fig.show()
fig = px.box(data, y=['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'], title='Meta',
             template='plotly_dark')
fig.show()
fig = px.scatter(
    data, y=['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'], title='Meta', template='plotly_dark')
fig.show()
fig = px.scatter(twitter_data, title='Twitter', template='plotly_dark')
fig.show()
fig = px.box(twitter_data, title='Twitter', template='plotly_dark')
fig.show()
fig = px.scatter(twitter_data_transposed, y=['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'],
title='Twitter', template='plotly_dark')
fig.show()
fig = px.box(twitter_data_transposed, y=['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'],
title='Twitter', template='plotly_dark')
fig.show()
```

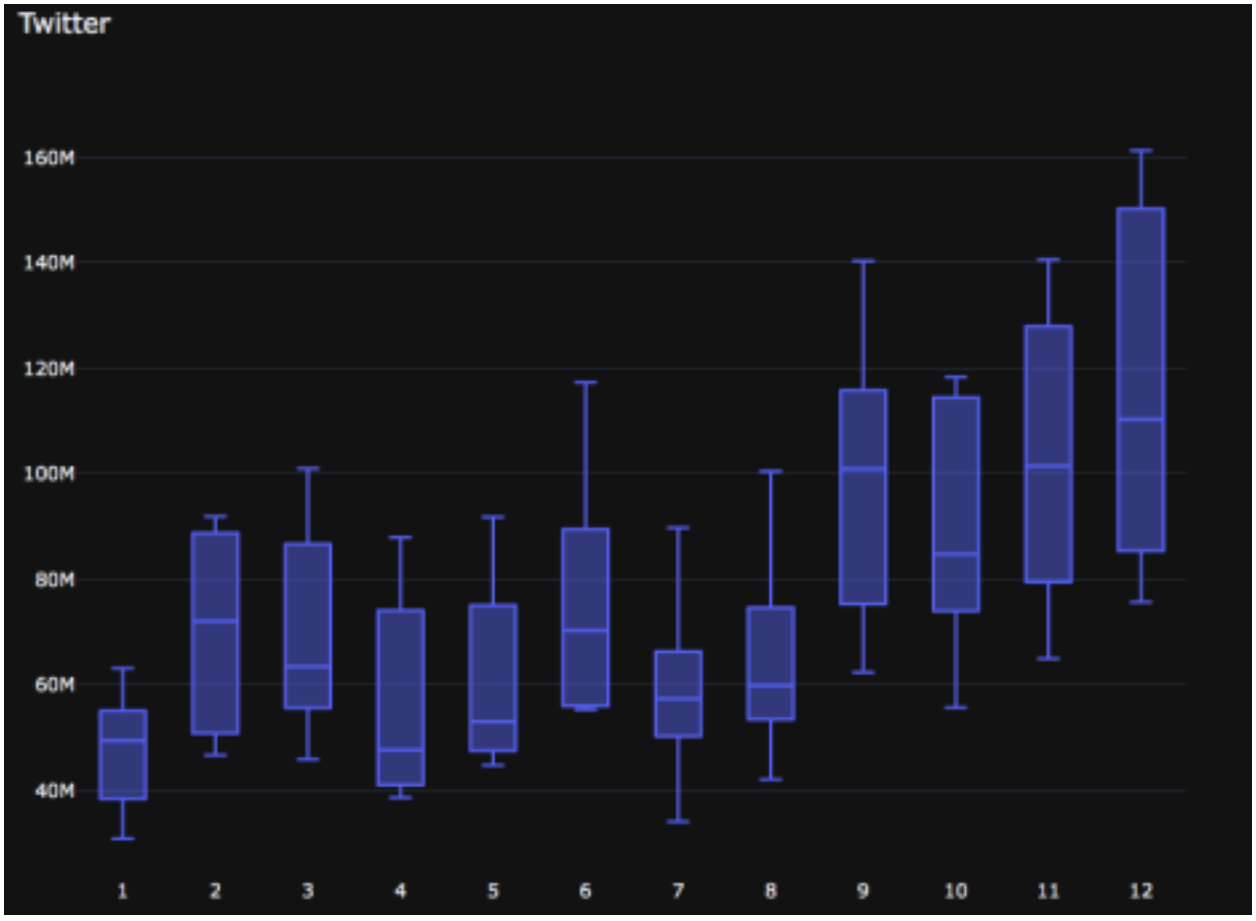

variable
2017
2018
2019
2020
2021
800M
700M
600M
500M
400M
300M
200M
0
2
4
6
8
10

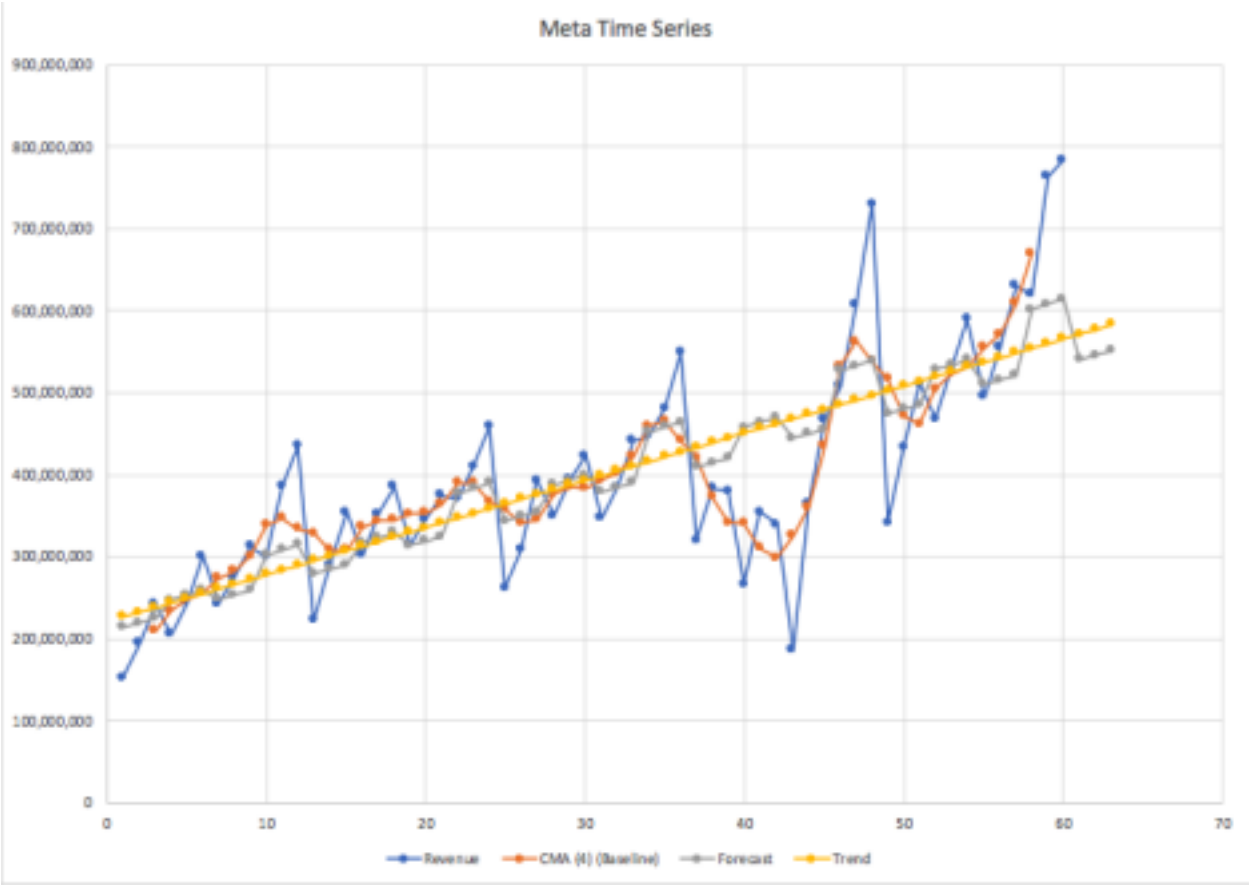

800M
700M
600M
500M
400M
300M
200M
2017
2018
2019
2020
2021

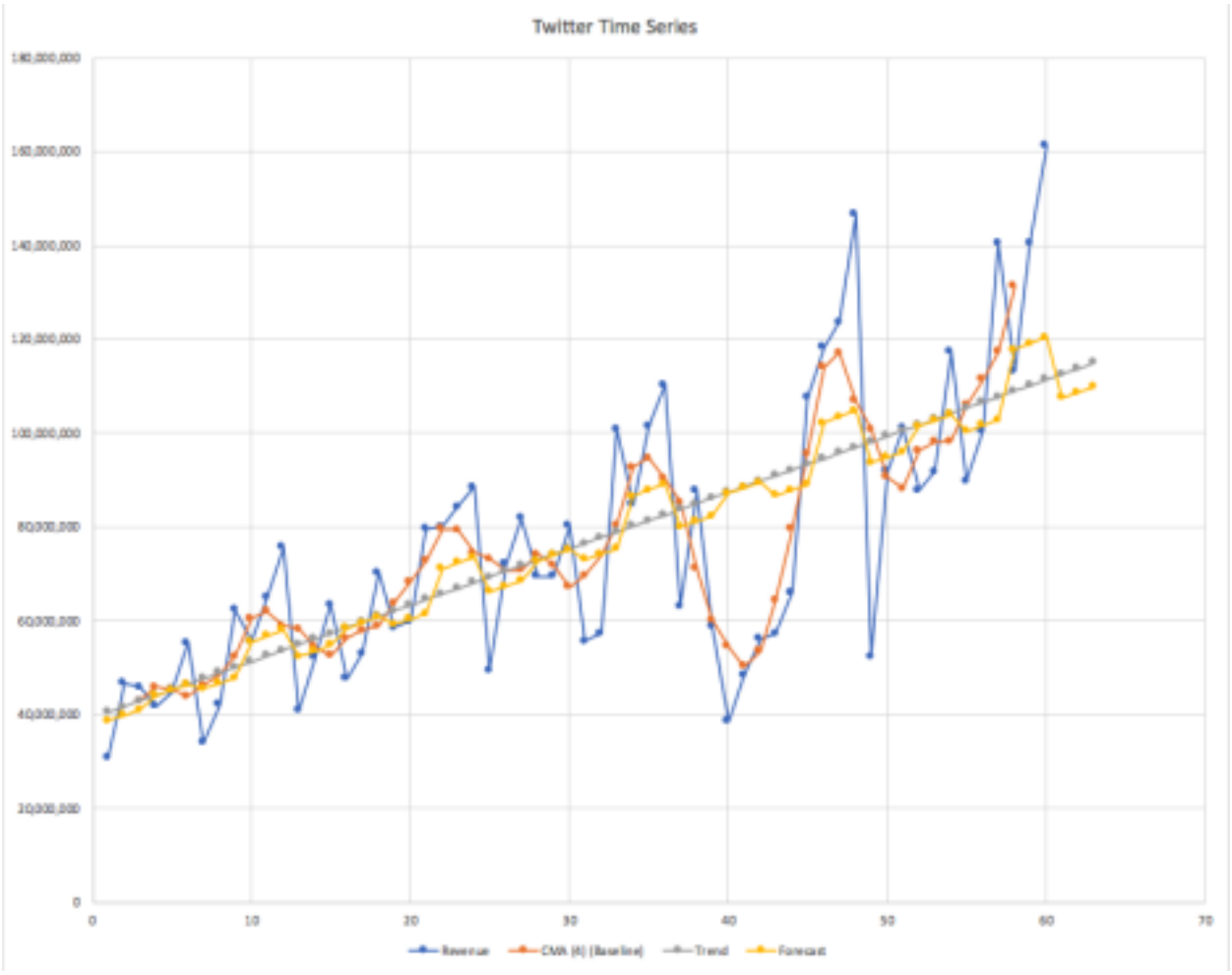

Meta
800M
700M
600M
500M
400M
300M
200M
1
2
3
4
5
6
7
8
9
10
11
12

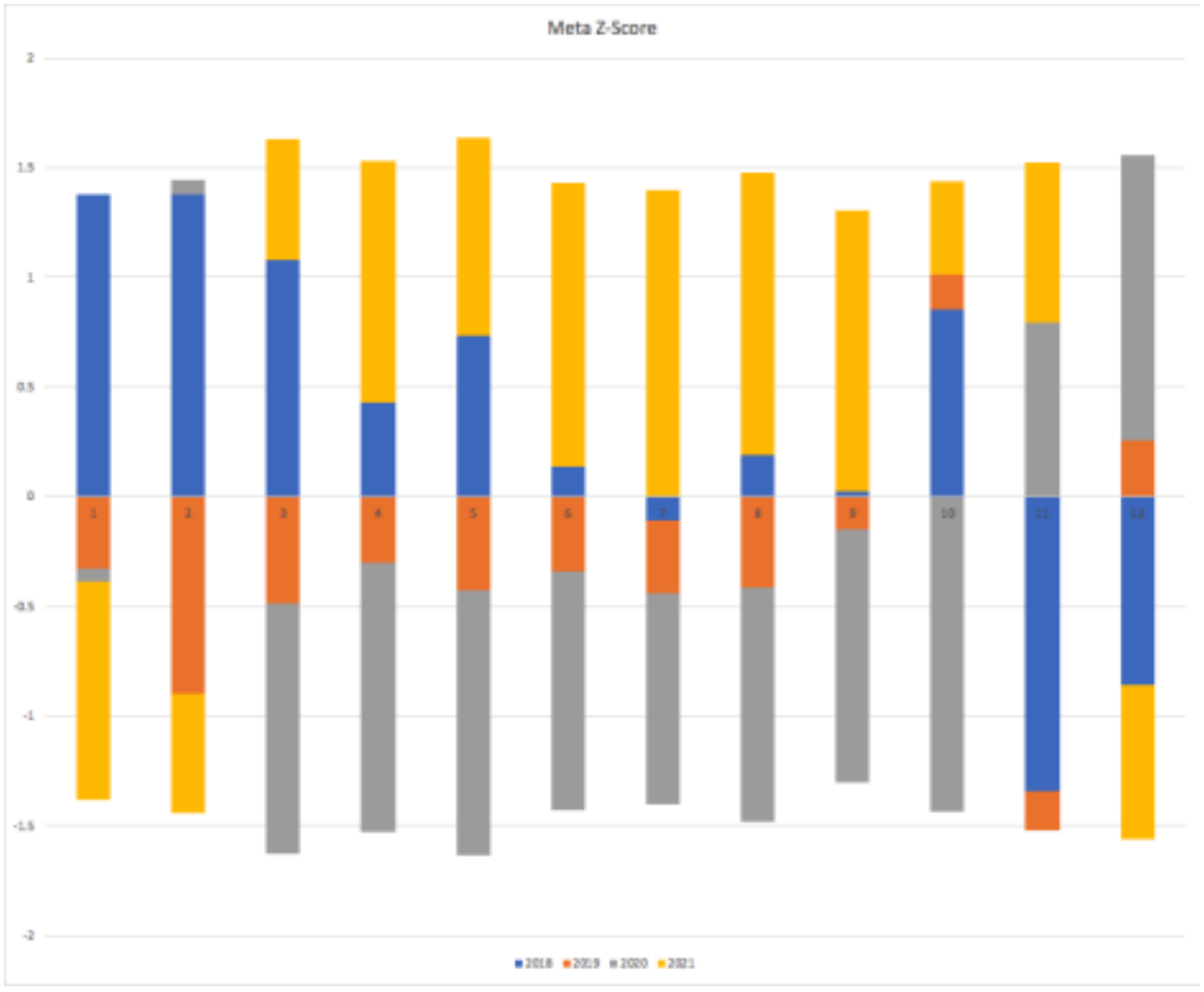

Twitter
160M
140M
120M
100M
80M
60M
40M
0
2
4
6
8
10
variable
2017
2018
2019
2020
2021

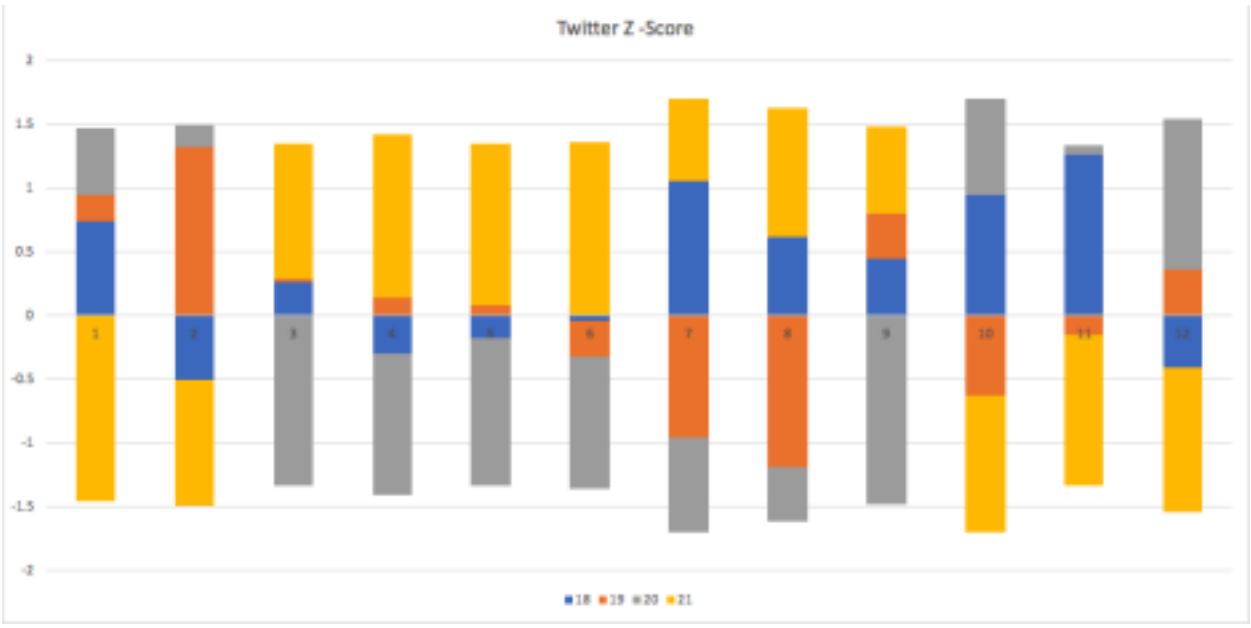

Twitter
160M
140M
120M
100M
80M
60M
40M
2017
2018
2019
2020
2021


Twitter
160M
140M
120M
100M
80M
60M
40M
1
2
3
4
5
6
7
8
9
10
11
12

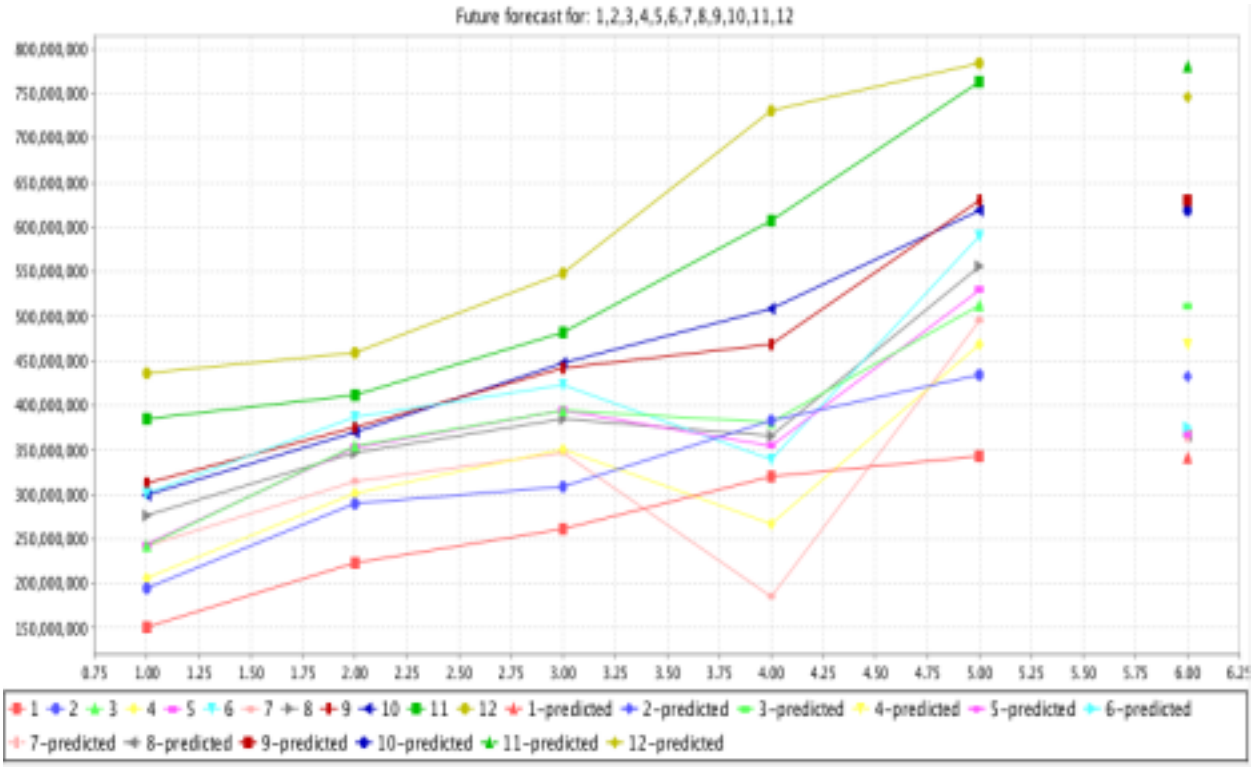

Meta Time Series
900,000,000
800,000,000
700,000,000
600,000,000
500,000,000
400,000,000
300,000,000
200,000,000
100,000,000
0
0
10
20
30
40
50
60
70
Revenue
CMA (4) (Baseline)
Forecast
Trend


Twitter Time Series
180,000,000
160,000,000
140,000,000
120,000,000
100,000,000
80,000,000
60,000,000
40,000,000
20,000,000
0
0
10
20
30
40
50
60
70
Revenue
CMA (6) (Baseline)
Trend
Forecast

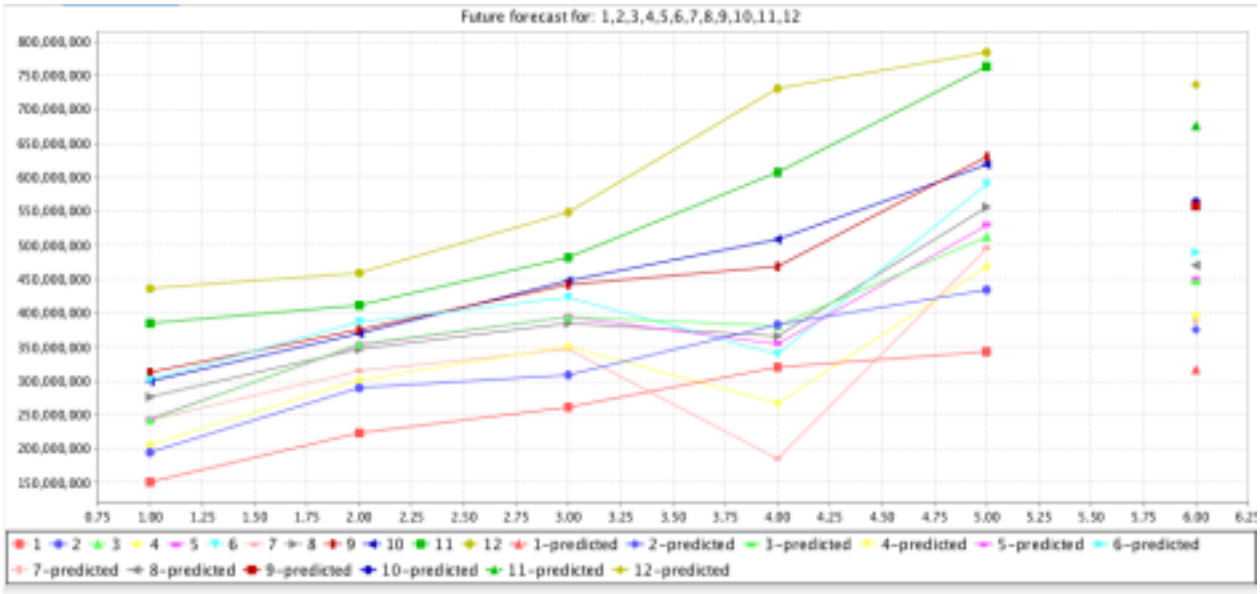




AdditiveRegression



GaussianProcess



GaussianProcess Run info:
=== Run information ===

Scheme:
GaussianProcesses -L 1.0 -N 0 -K "PolyKernel -E 1.0 -C 250007" -S 1

Lagged and derived variable options:
-F 1,2,3,4,5,6,7,8,9,10,11,12 -L 1 -M 2

Relation: metasmitransposed-weka.filters.unsupervised.attribute.Remove-R13
Instances: 5
Attributes: 12

1
2
3
4
5
6
7
8
9
10
11
12

Transformed training data:

1
2
3
4
5
6
7
8
9
10
11
12
ArtificialTimeIndex
Lag_1-1
Lag_1-2
Lag_2-1
Lag_2-2
Lag_3-1
Lag_3-2
Lag_4-1
Lag_4-2
Lag_5-1
Lag_5-2
Lag_6-1
Lag_6-2
Lag_7-1
Lag_7-2
Lag_8-1
Lag_8-2
Lag_9-1
Lag_9-2
Lag_10-1
Lag_10-2
Lag_11-1

Lag_11-2
Lag_12-1
Lag_12-2
ArtificialTimeIndex^2
ArtificialTimeIndex^3
ArtificialTimeIndex*Lag_1-1
ArtificialTimeIndex*Lag_1-2
ArtificialTimeIndex*Lag_2-1
ArtificialTimeIndex*Lag_2-2
ArtificialTimeIndex*Lag_3-1
ArtificialTimeIndex*Lag_3-2
ArtificialTimeIndex*Lag_4-1
ArtificialTimeIndex*Lag_4-2
ArtificialTimeIndex*Lag_5-1
ArtificialTimeIndex*Lag_5-2
ArtificialTimeIndex*Lag_6-1
ArtificialTimeIndex*Lag_6-2
ArtificialTimeIndex*Lag_7-1
ArtificialTimeIndex*Lag_7-2
ArtificialTimeIndex*Lag_8-1
ArtificialTimeIndex*Lag_8-2
ArtificialTimeIndex*Lag_9-1
ArtificialTimeIndex*Lag_9-2
ArtificialTimeIndex*Lag_10-1
ArtificialTimeIndex*Lag_10-2
ArtificialTimeIndex*Lag_11-1
ArtificialTimeIndex*Lag_11-2
ArtificialTimeIndex*Lag_12-1
ArtificialTimeIndex*Lag_12-2

1:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.5682959450446419
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.37823154229903044
       Highest Value = 0.19868720884967123

2:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>
All values shown based on: Normalize training data

Average Target Value : 0.532071671119202
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
  Vector: Lowest Value = -0.3556505721828951
       Highest Value = 0.17930706667120802

3:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.4982612703258071
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
Inverted Covariance Matrix * Target-value
    Vector: Lowest Value =
    -0.30607257310947517 Highest Value =
    0.1272551100468562

4:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.4298794309404264
Inverted Covariance Matrix:

Lowest Value = -0.2800630325975725
Highest Value = 0.5120433458111273
Inverted Covariance Matrix * Target-value
Vector: Lowest Value = -0.24343915709497765
Highest Value = 0.1506911491180384

5:
Gaussian Processes

Kernel used:
Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.4594746068352098
Inverted Covariance Matrix:
Lowest Value = -0.2800630325975725
Highest Value = 0.5120433458111273
Inverted Covariance Matrix * Target-value
Vector: Lowest Value = -0.2754033207425896
Highest Value = 0.13654398570356296

6:
Gaussian Processes

Kernel used:
Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.3708599470166714
Inverted Covariance Matrix:
Lowest Value = -0.2800630325975725
Highest Value = 0.5120433458111273
Inverted Covariance Matrix * Target-value
Vector: Lowest Value = -0.20962392630101295
Highest Value = 0.15416696067593932

7:
Gaussian Processes

Kernel used:
Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.4245331574069189
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.20774766837903286
       Highest Value = 0.19459907409773658

8:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.39150415000417865
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.24038330978139083
       Highest Value = 0.13870688885424876

9:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.41860320852628047
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.27011625706059533
       Highest Value = 0.12776505939794947

10:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.4672000249085178
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.30846716872500424
       Highest Value = 0.12541322348267142

11:
Gaussian Processes

Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.38237824002621523
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725
    Highest Value = 0.5120433458111273
      Inverted Covariance Matrix * Target-value
 Vector: Lowest Value = -0.26884906410645076
       Highest Value = 0.14890587693679574

12:
Gaussian Processes
Kernel used:
  Linear Kernel: K(x,y) = <x,y>

All values shown based on: Normalize training data

Average Target Value : 0.44765614561625455
Inverted Covariance Matrix:
    Lowest Value = -0.2800630325975725

Highest Value = 0.5120433458111273
Inverted Covariance Matrix * Target-value Vector:
Lowest Value = -0.3241171712230022
Highest Value = 0.24620902982921714

=== Future predictions from end of training data ===
inst# 1 2 3 4 5 6 7 8 9 10 11 12
1 151240574 194215644 242283550 205639456 244038718 300633299 242587707
275613895 312967168 299480705 385284528 435644000 2 222276110 288943657
354097343 302049922 352558188 385913139 315333193 346493353 375058837 370424429
410760935 459037308 3 261951567 308051199 393683361 350272687 394822442
423568441 347284603 384475694 441896274 447016703 481726167 548835480 4
320782354 383543248 380431042 266925127 354982504 339344769 185857094 364980459
468468101 508460645 607897228 729881781 5 342004721 433857388 511417451
468672444 529420771 589672897 495507771 556660359 629999759 619435512 763104542
783563950 6* 315431954.135 376060127.6803 444126087.2384 393751531.3827
450558741.0957 490204482.7883 389863546.0946 470165307.1102 557658889.0074
563238074.521 675117951.8558 736116945.9182

=== Evaluation on training data ===
Target 1-step-ahead
==========
1
N 3
Mean absolute percentage error 7.9977
2
N 3
Mean absolute percentage error 8.0538
3
N 3
Mean absolute percentage error 4.0634 4
N 3
Mean absolute percentage error 9.0029 5
N 3
Mean absolute percentage error 4.4723 6
N 3
Mean absolute percentage error 7.8743 7
N 3
Mean absolute percentage error 20.4273 8
N 3
Mean absolute percentage error 3.1231 9
N 3
Mean absolute percentage error 4.5114 10
N 3
Mean absolute percentage error 6.3261 11
N 3

Mean absolute percentage error 8.8711 12
N 3
Mean absolute percentage error 9.8903

Total number of instances: 5