BIGBRaiN

T+18 hours Energy Production Forecast Model for ile de France

Analysis on problem set and historical datasets

These are the stats we got for the energy production values for ile de france, which is the first column in our data set.

Mean= 17541

Median= 10500

Max= 89000

Min= 0

SD= 19133

Persistence: 0.44 (2d.p)

Analysis on problem set and historical datasets

These are the stats we got for the energy production values for ile de france, which is the first column in our data set.

Mean= 17541

Median= 10500

Max= 89000

Min= 0

SD= 19133

Persistence: 0.44 (2d.p)

Solution Development Methodology

The first technique we used is Normalization. Normalization was applied to all columns used in features.

Reasons:

- Normalisation changes the values to a common scale without distorting differences in range of values
- It ensures that the mean is about 0
- Makes sure that all data inputs are about the same size and have a magnitude of about 1
- All of which are crucial for neural networks

```
: normalize-A ( seq -- seq ) cols
  [: 17541 -. 19133 /. ;] map
;

: normalize-B ( seq -- seq ) cols
  [: 3.4715 -. 1.7733 /. ;] map
;

: normalize-C ( seq -- seq ) cols
  [: 3.3173 -. 1.6815 /. ;] map
;

: normalize-D ( seq -- seq ) cols
  [: 3.5678 -. 1.7827 /. ;] map
;
```

Solution Development Methodology

The first technique we used is Normalization. Normalization was applied to all columns used in features.

Reasons:

- Normalisation changes the values to a common scale without distorting differences in range of values
- It ensures that the mean is about 0
- Makes sure that all data inputs are about the same size and have a magnitude of about 1
- All of which are crucial for neural networks

```
: normalize-A ( seq -- seq ) cols
  [: 17541 -. 19133 /. ;] map
;

: normalize-B ( seq -- seq ) cols
  [: 3.4715 -. 1.7733 /. ;] map
;

: normalize-C ( seq -- seq ) cols
  [: 3.3173 -. 1.6815 /. ;] map
;

: normalize-D ( seq -- seq ) cols
  [: 3.5678 -. 1.7827 /. ;] map
;
```

Solution Development Methodology

Next, we established the persistence loss for the test dataset as the initial benchmark.

Reason:

- To compare the result with the model output.
- In our case, the persistence to beat is 0.44.

Solution Development Methodology

Next, we established the persistence loss for the test dataset as the initial benchmark.

Reason:

- To compare the result with the model output.
- In our case, the persistence to beat is 0.44.

Solution Development Methodology

Then, we used Transformation. We used MEAN, SD, DIFF, MAX features from range 0 to -12 for D and range 0:-20 for C; MEAN, SD, MAX from range 0:-4 for B; MEAN and SD over range 0:-8 as well as difference, momentum and force with lead time 18 over range 0:-32 for A.

```
\ T+0 - T-N
: difference ( seq n -- seq ) { n }
  dup
  n ['] tail times
  ['] -. zipwith
;

: momentum ( seq n -- seq )
  difference
;

: force ( seq n -- seq ) { n }
  n difference n difference
;

: transform
  \ --- ACTUALS ---
  A:18          \ Labels, T + 18hrs.
  A:0           \ y0

  \ -------- FEATURES -------
  A:0:-8 MEAN    \ feature 1
  A:0:-8 SD      \ feature 2
  A:0:-32 18 difference { xs }  \ T+0 - T-18 ; T-1 - T-19 ; ... T-14 - T-32
  xs                    \ Raw differences (XS)
  xs 18 momentum         \ 1st-order differences of XS
  xs 18 force            \ 2nd-order differences of XS
```

```
27    B:18
28
29    B:0:-4 MEAN    \ feature 1
30    B:0:-4 SD      \ feature 2
31    B:0:-4 MAX
32
33    C:18
34
35    C:0:-20 MEAN    \ feature 1
36    C:0:-20 SD      \ feature 2
37    C:0:-20 DIFF
38    C:0:-20 MAX
39
40    D:18
41
42    D:0:-12 MEAN    \ feature 1
43    D:0:-12 SD      \ feature 2
44    D:0:-12 DIFF
45    D:0:-12 MAX
46 ;
```

Solution Development Methodology

Then, we used Transformation. We used MEAN, SD, DIFF, MAX features from range 0 to -12 for D and range 0:-20 for C; MEAN, SD, MAX from range 0:-4 for B; MEAN and SD over range 0:-8 as well as difference, momentum and force with lead time 18 over range 0:-32 for A.

```
\ T+0 - T-N
: difference ( seq n -- seq ) { n }
  dup
  n ['] tail times
  ['] -. zipwith
;

: momentum ( seq n -- seq )
  difference
;

: force ( seq n -- seq ) { n }
  n difference n difference
;

: transform
  \ --- ACTUALS ---
  A:18          \ Labels, T + 18hrs.
  A:0           \ y0

  \ -------- FEATURES -------
  A:0:-8 MEAN    \ feature 1
  A:0:-8 SD      \ feature 2
  A:0:-32 18 difference { xs }  \ T+0 - T-18 ; T-1 - T-19 ; ... T-14 - T-32
  xs                    \ Raw differences (XS)
  xs 18 momentum         \ 1st-order differences of XS
  xs 18 force            \ 2nd-order differences of XS
```

```
27    B:18
28
29    B:0:-4 MEAN    \ feature 1
30    B:0:-4 SD      \ feature 2
31    B:0:-4 MAX
32
33    C:18
34
35    C:0:-20 MEAN    \ feature 1
36    C:0:-20 SD      \ feature 2
37    C:0:-20 DIFF
38    C:0:-20 MAX
39
40    D:18
41
42    D:0:-12 MEAN    \ feature 1
43    D:0:-12 SD      \ feature 2
44    D:0:-12 DIFF
45    D:0:-12 MAX
46 ;
```

Solution Development Methodology

Reason:

- Yielded lowest loss amongst selection of multiple combinations
- Momentum and force introduces rate of change, hence reducing lag
- Difference is used so that lead time is taken into consideration

Solution Development Methodology

Reason:

- Yielded lowest loss amongst selection of multiple combinations
- Momentum and force introduces rate of change, hence reducing lag
- Difference is used so that lead time is taken into consideration

Solution Development Methodology

For our network, we start off with input scaling to:

- Reduce noise
- Reduce the effects of outliers by using a clamp. (We used a tan h clamped on our inputs.)

However, we noticed that there wasn't much outliers in the dataset and clamping caused our test loss to increase by about 0.01.

```
\ Weight sharing
: share ( l0 l1 "name" -- l0 l1 ) { l0 l1 ns }
  l0 ns name-weights
  l1 ns name-weights
;

: diag ( xs -- l )
  1 1 1 scale     \ l0
  dup 1 1 1 scale \ l1
  "scaling" share \ ensures these two scaling layers are the same.
                  \ ie, weights are positive.
;
```

```
: scaling ( xs -- xs' )
  \ diagonal weights applies to each xi.
  \ weights are constrained to be positive.
  diag
  \ to remove outliers.
  ${clamp-on-scale?} -> tanh |.
;
```

Solution Development Methodology

For our network, we start off with input scaling to:

- Reduce noise
- Reduce the effects of outliers by using a clamp. (We used a tan h clamped on our inputs.)

However, we noticed that there wasn't much outliers in the dataset and clamping caused our test loss to increase by about 0.01.

```
\ Weight sharing
: share ( l0 l1 "name" -- l0 l1 ) { l0 l1 ns }
  l0 ns name-weights
  l1 ns name-weights
;

: diag ( xs -- l )
  1 1 1 scale     \ l0
  dup 1 1 1 scale \ l1
  "scaling" share \ ensures these two scaling layers are the same.
                  \ ie, weights are positive.
;
```

```
: scaling ( xs -- xs' )
  \ diagonal weights applies to each xi.
  \ weights are constrained to be positive.
  diag
  \ to remove outliers.
  ${clamp-on-scale?} -> tanh |.
;
```

Solution Development Methodology

We also used dimensional reduction to reduce dimensions of input without losing the necessary information. An autoencoder, a 3 layer subnetwork, is used and trained together with the main one.

We took a bottleneck size of 2 (reducing the size of the vector by 50% from input vector) although smaller bottlenecks have larger errors because it gave lower test loss compared to higher bottlenecks (as shown in later slides).

Relu perceptron was used since it was faster.

```
: autoencoder ( xs -- l ) { xs }
  xs
  ${input-size} perceptron    \ input layer
  ${input-size bottleneck-size /} perceptron dup { out } \ middle layer, tapped
  ${input-size} perceptron
  ${input-size} innerproduct
        xs euclidean asloss >> drloss \ losses plotted separately.
  out
;
```

Solution Development Methodology

We also used dimensional reduction to reduce dimensions of input without losing the necessary information. An autoencoder, a 3 layer subnetwork, is used and trained together with the main one.

We took a bottleneck size of 2 (reducing the size of the vector by 50% from input vector) although smaller bottlenecks have larger errors because it gave lower test loss compared to higher bottlenecks (as shown in later slides).

Relu perceptron was used since it was faster.

```
: autoencoder ( xs -- l ) { xs }
  xs
  ${input-size} perceptron    \ input layer
  ${input-size bottleneck-size /} perceptron dup { out } \ middle layer, tapped
  ${input-size} perceptron
  ${input-size} innerproduct
        xs euclidean asloss >> drloss \ losses plotted separately.
  out
;
```

Solution Development Methodology

We also used dimensional reduction to reduce dimensions of input without losing the necessary information. An autoencoder, a 3 layer subnetwork, is used and trained together with the main one.

We took a bottleneck size of 2 (reducing the size of the vector by 50% from input vector) although smaller bottlenecks have larger errors because it gave lower test loss compared to higher bottlenecks (as shown in later slides).

Relu perceptron was used since it was faster.

```
: autoencoder ( xs -- l ) { xs }
  xs
  ${input-size} perceptron    \ input layer
  ${input-size bottleneck-size /} perceptron dup { out } \ middle layer, tapped
  ${input-size} perceptron
  ${input-size} innerproduct
        xs euclidean asloss >> drloss \ losses plotted separately.
  out
;
```

Solution Development Methodology

Next, we used Early Stopping and Dropouts in our main network. Early stopping stops training when losses “begin to increase”. Dropouts temporarily and randomly disable neurons during training.

```
\ Sizes for the Neural Network
{{ 32 64 128  }} := nn-size

\ Adds a tanh after scaling (or not...)
{{ true false }} := clamp-on-scale?

\ shrinkage factor of the bottleneck
\ {{ 2 4 8 }} := bottleneck-size
{{ 2 4 }} := bottleneck-size

0.05 := dropout-prob

\ Max iterations for solver.
10000 := iters

\ Solver type.
\ Adam is highly optimized for gradient descent. (default)
\ SGD is Stochasic Gradient Descent. (can be better in some cases)
"Adam" := solver-type

\ the number of features you use.
input-cols 2 - := input-size
```

Reason:

- To prevent overfitting

```
: perceptron ( l n -- l )
  innerproduct relu
  ${dropout-prob} dropout
;
```

Solution Development Methodology

Next, we used Early Stopping and Dropouts in our main network. Early stopping stops training when losses “begin to increase”. Dropouts temporarily and randomly disable neurons during training.

```
\ Sizes for the Neural Network
{{ 32 64 128  }} := nn-size

\ Adds a tanh after scaling (or not...)
{{ true false }} := clamp-on-scale?

\ shrinkage factor of the bottleneck
\ {{ 2 4 8 }} := bottleneck-size
{{ 2 4 }} := bottleneck-size

0.05 := dropout-prob

\ Max iterations for solver.
10000 := iters

\ Solver type.
\ Adam is highly optimized for gradient descent. (default)
\ SGD is Stochasic Gradient Descent. (can be better in some cases)
"Adam" := solver-type

\ the number of features you use.
input-cols 2 - := input-size
```

Reason:

- To prevent overfitting

```
: perceptron ( l n -- l )
  innerproduct relu
  ${dropout-prob} dropout
;
```

Solution Development Methodology

Overall, we used a 4 layer network with a ⅔ reduction in network size at each subsequent layer.

Reason:

- Yielded the lowest loss after comparison with a double layer

```
: network ( 1 -- 1 )
   named P003
     scaling autoencoder
     ${nn-size} perceptron \ layer 1
     ${nn-size 2 * 3 /} perceptron   \ layer 2
     ${nn-size 4 * 9 /} perceptron   \ layer 3
     ${nn-size 8 * 27 /} perceptron  \ layer 4
     1 innerproduct
   end-named
;
```

Solution Development Methodology

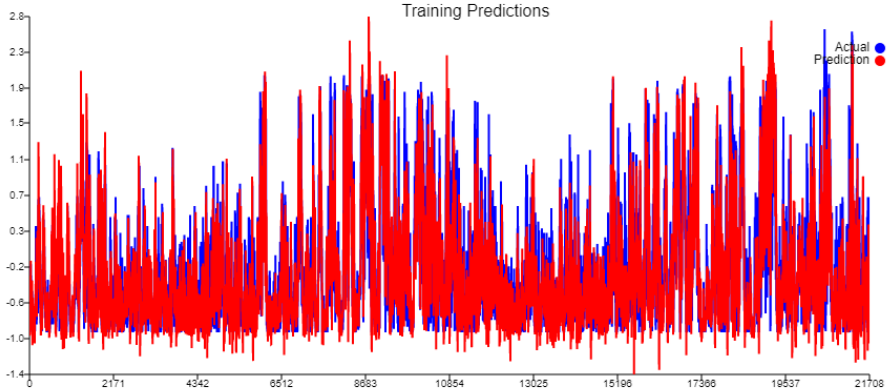
Overall, we used a 4 layer network with a ⅔ reduction in network size at each subsequent layer.

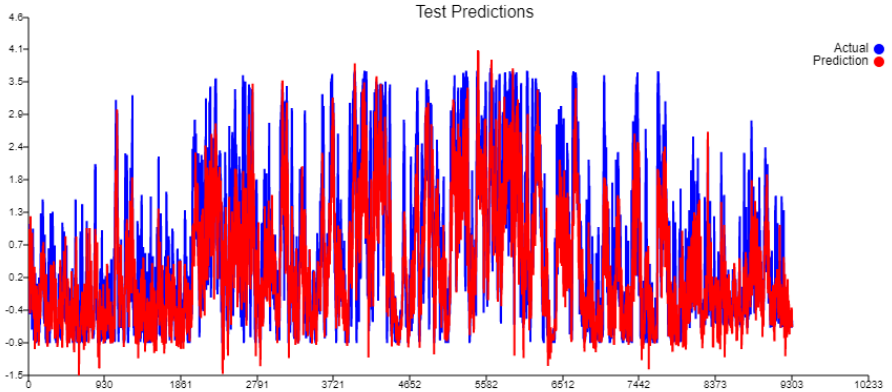
Reason:

- Yielded the lowest loss after comparison with a double layer

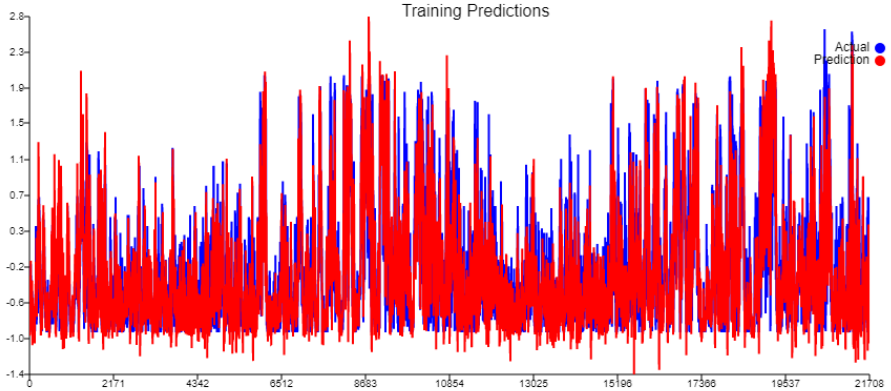
```
: network ( 1 -- 1 )
   named P003
     scaling autoencoder
     ${nn-size} perceptron \ layer 1
     ${nn-size 2 * 3 /} perceptron   \ layer 2
     ${nn-size 4 * 9 /} perceptron   \ layer 3
     ${nn-size 8 * 27 /} perceptron  \ layer 4
     1 innerproduct
   end-named
;
```

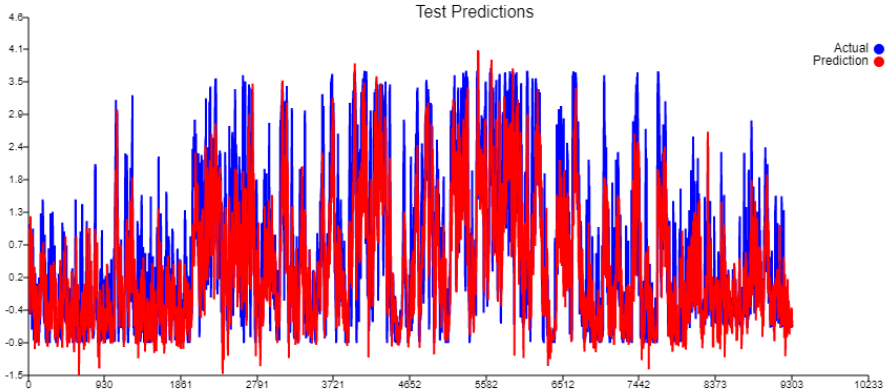
Result - Prediction Output





Result - Prediction Output





Result - Losses

You have 63 columns in your dataset.
Test losses statistics
Mean : 0.290919 Max : 0.304167
Median: 0.289298 Min : 0.283560
SD : 0.006637 Range: 0.020607
(Mean-Min)/SD: 1.108866

Config#	Test Loss	bottleneck-size	input-size	dropout-prob	iters	solver-type	nn-size	clamp-on-scale?
1	0.294664	2	61	0.05	10000	Adam	32	true
2	0.29258	2	61	0.05	10000	Adam	64	true
3	0.302301	2	61	0.05	10000	Adam	128	true
4	0.285938	2	61	0.05	10000	Adam	32	false
5	0.284059	2	61	0.05	10000	Adam	64	false
6	0.28356	2	61	0.05	10000	Adam	128	false
7	0.29013	4	61	0.05	10000	Adam	32	true
8	0.286462	4	61	0.05	10000	Adam	64	true
9	0.290971	4	61	0.05	10000	Adam	128	true
10	0.287734	4	61	0.05	10000	Adam	32	false
11	0.304167	4	61	0.05	10000	Adam	64	false
12	0.288466	4	61	0.05	10000	Adam	128	false

ok

Result - Losses

You have 63 columns in your dataset.
Test losses statistics
Mean : 0.290919 Max : 0.304167
Median: 0.289298 Min : 0.283560
SD : 0.006637 Range: 0.020607
(Mean-Min)/SD: 1.108866

Config#	Test Loss	bottleneck-size	input-size	dropout-prob	iters	solver-type	nn-size	clamp-on-scale?
1	0.294664	2	61	0.05	10000	Adam	32	true
2	0.29258	2	61	0.05	10000	Adam	64	true
3	0.302301	2	61	0.05	10000	Adam	128	true
4	0.285938	2	61	0.05	10000	Adam	32	false
5	0.284059	2	61	0.05	10000	Adam	64	false
6	0.28356	2	61	0.05	10000	Adam	128	false
7	0.29013	4	61	0.05	10000	Adam	32	true
8	0.286462	4	61	0.05	10000	Adam	64	true
9	0.290971	4	61	0.05	10000	Adam	128	true
10	0.287734	4	61	0.05	10000	Adam	32	false
11	0.304167	4	61	0.05	10000	Adam	64	false
12	0.288466	4	61	0.05	10000	Adam	128	false

ok

Result - Loss Graph

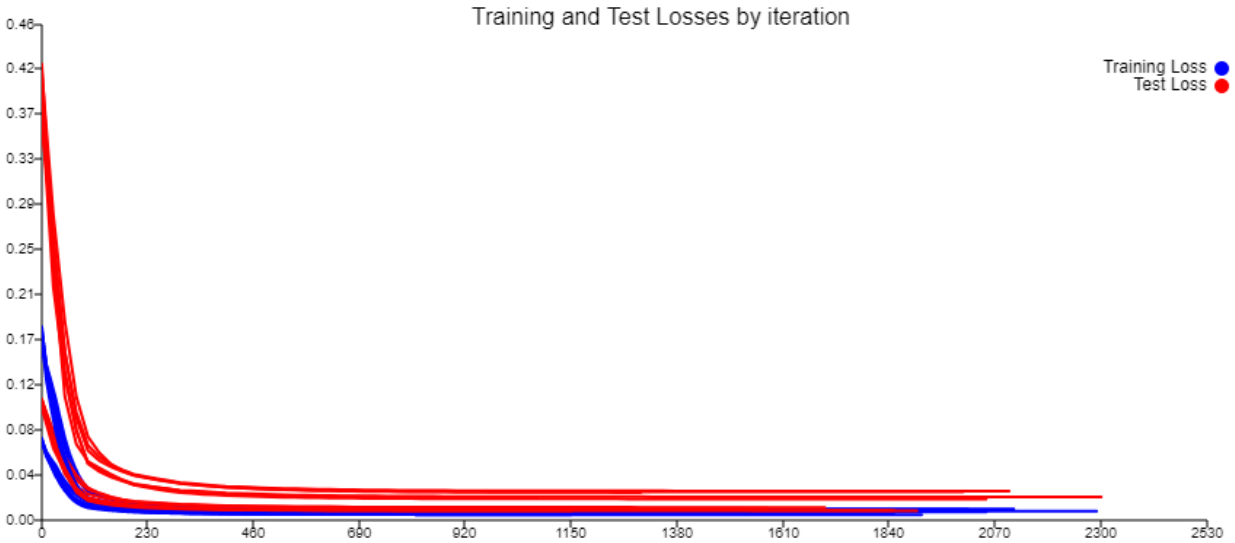
Test loss: 0.284 (3s.f)

Result - Loss Graph

Test loss: 0.284 (3s.f)

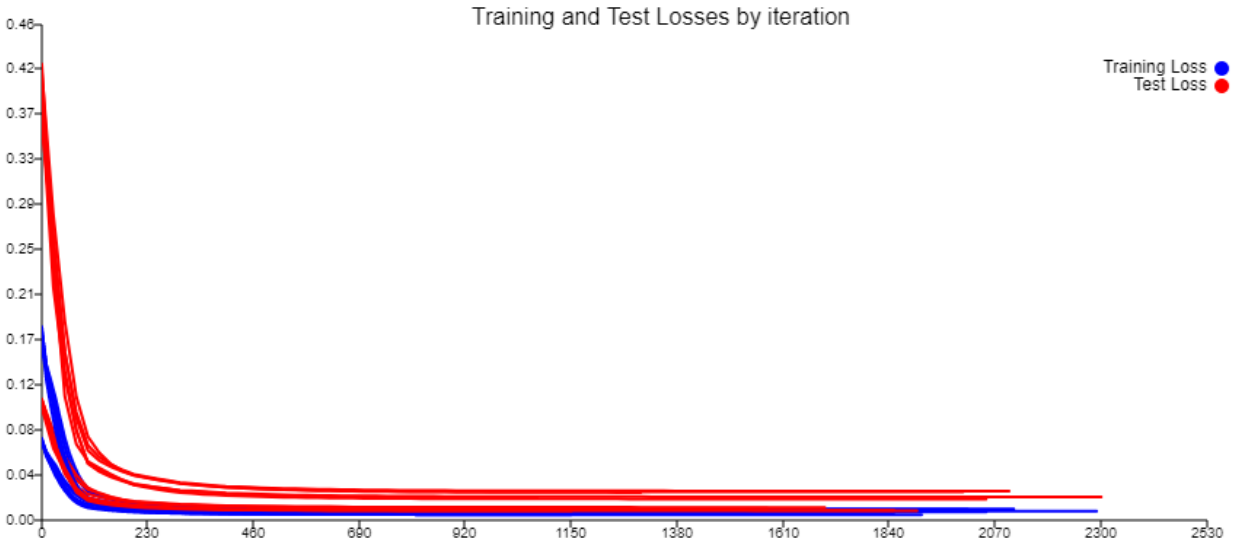
Result - Autoencoder Loss Graph

We plotted the autoencoder loss graph separately and observe that there is good reconstruction as shown by the close proximity of the training and test loss curves to the x-axis.



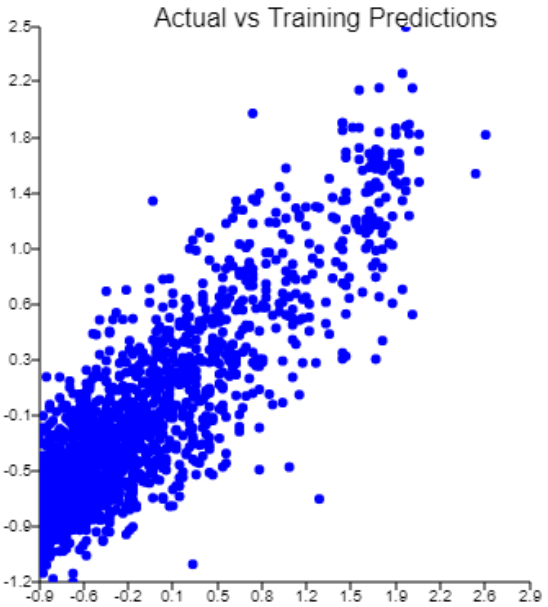
Result - Autoencoder Loss Graph

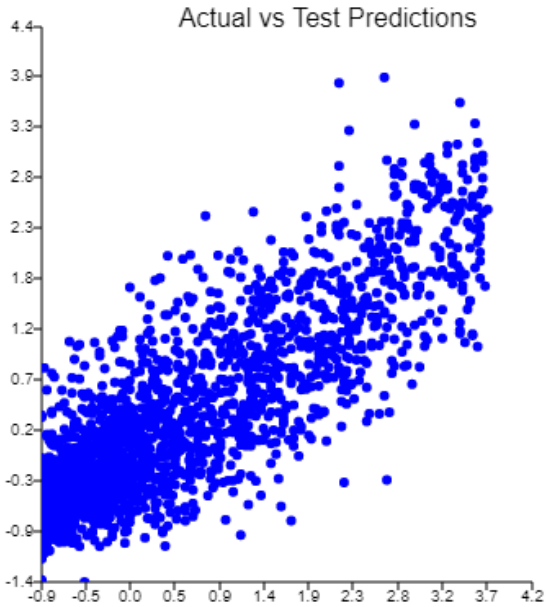
We plotted the autoencoder loss graph separately and observe that there is good reconstruction as shown by the close proximity of the training and test loss curves to the x-axis.



Result - Prediction Output

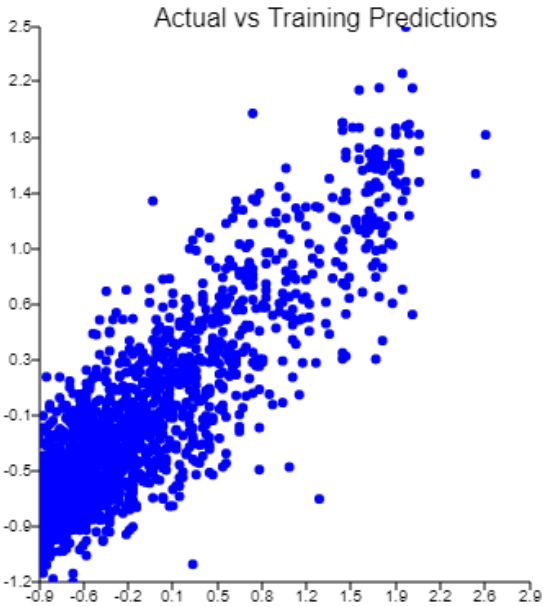
High correlation between actual and test predictions

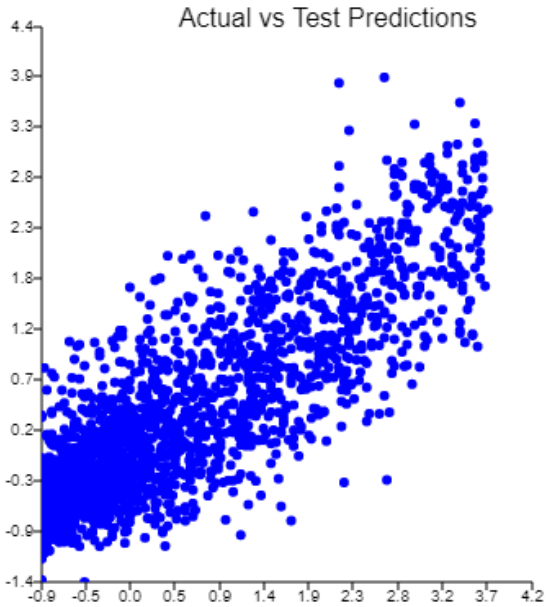




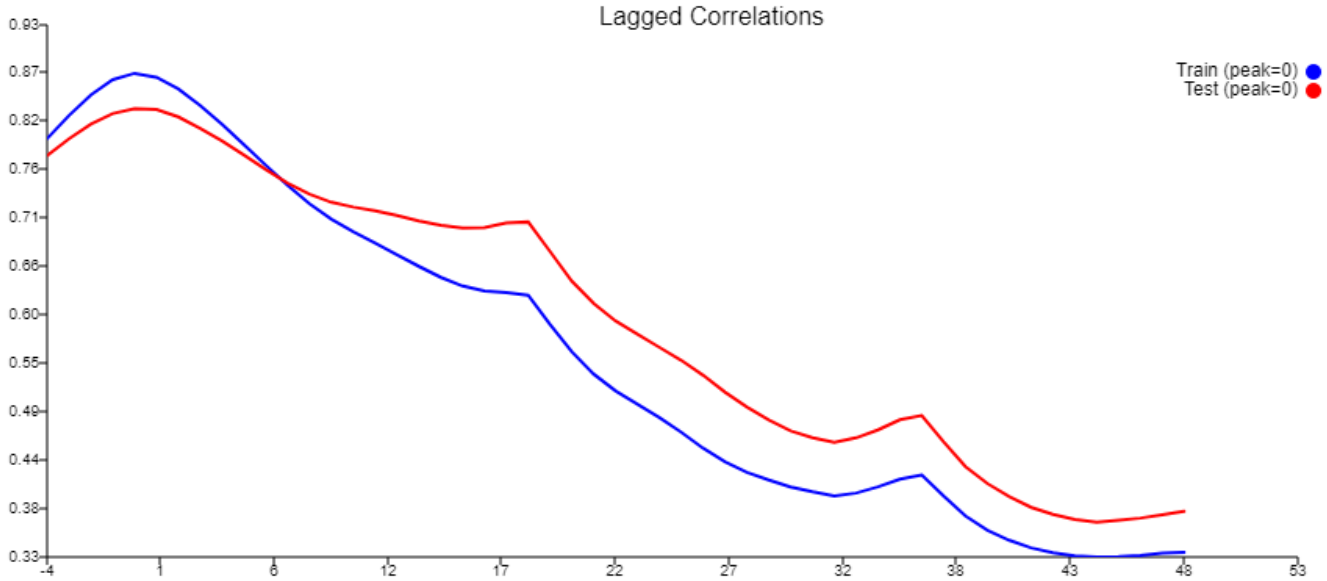
Result - Prediction Output

High correlation between actual and test predictions



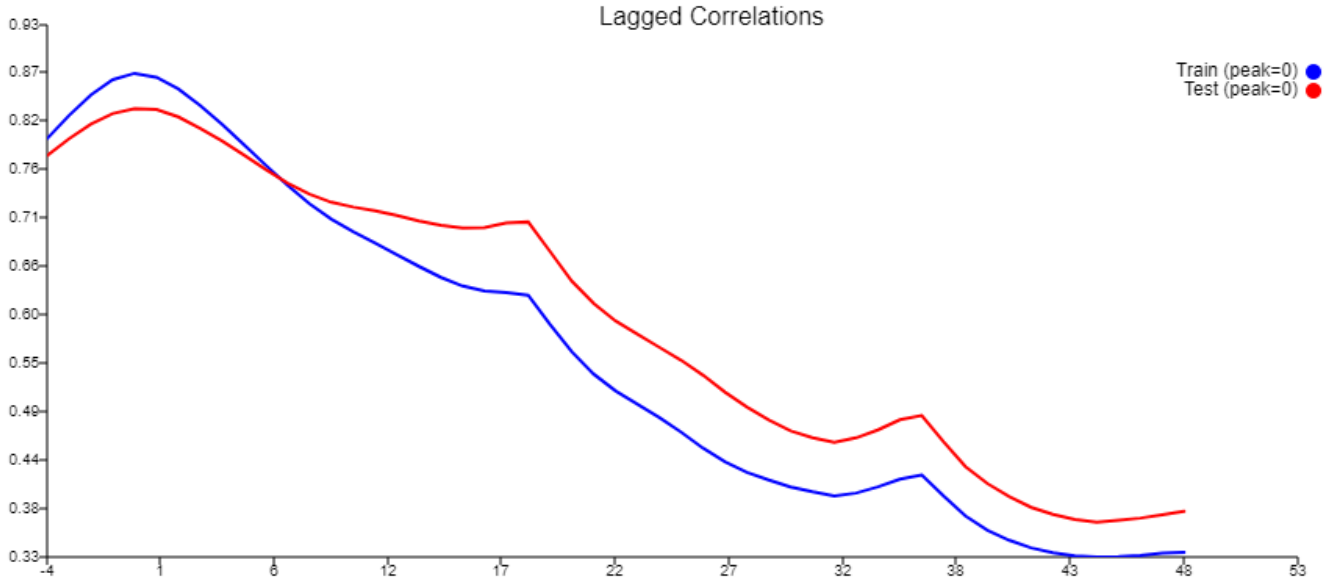


Result - Lag plots



Lag = 0

Result - Lag plots



Lag = 0

Results

Our test loss was 0.284 (3s.f) which is lower than 0.44. Hence, we managed to beat persistence.

After we first beat persistence, we tried multiple combinations of features, nn-size, dropout probabilities, clamping and bottleneck sizes to beat test loss and lag. Finally, we arrived at our lowest test loss of 0.284.

We also managed to achieve a lag of zero.

Results

Our test loss was 0.284 (3s.f) which is lower than 0.44. Hence, we managed to beat persistence.

After we first beat persistence, we tried multiple combinations of features, nn-size, dropout probabilities, clamping and bottleneck sizes to beat test loss and lag. Finally, we arrived at our lowest test loss of 0.284.

We also managed to achieve a lag of zero.

Trades

```
51842.3894,0.0000,0.0000,3157.6106,1351422770.0000
59046.5379,0.0000,0.0000,953.4621,1351481816.0000
56712.3119,0.0000,0.0000,787.6881,1351538528.0000
45884.1819,0.0000,0.0000,4115.8181,1351584412.0000
42123.7821,0.0000,0.0000,10376.2179,1351626535.0000
49790.3752,0.0000,0.0000,209.6248,1351676325.0000
43631.4625,0.0000,0.0000,8868.5375,1351719956.0000
47950.3546,0.0000,0.0000,4549.6454,1351767906.0000
49412.6898,0.0000,0.0000,587.3102,1351817318.0000
52699.3565,0.0000,0.0000,4800.6435,1351870017.0000
56989.9318,0.0000,0.0000,5510.0682,1351927006.0000
62500.0000,21613.4910,0.0000,0.0000,1351967893.0000
57500.0000,12696.3112,0.0000,0.0000,1352012697.0000
64832.9311,0.0000,0.0000,12667.0689,1352077529.0000
79254.4299,0.0000,0.0000,38245.5701,1352156783.0000
71005.4283,0.0000,0.0000,91494.5717,1352227788.0000
63860.2094,0.0000,0.0000,103639.7906,1352291648.0000
40510.4875,0.0000,0.0000,246989.5125,1352332158.0000
51580.4587,0.0000,0.0000,225919.5413,1352383738.0000
51597.1044,0.0000,0.0000,158402.8956,1352435335.0000
53825.7162,0.0000,0.0000,96174.2838,1352489160.0000
42633.4852,0.0000,0.0000,114866.5148,1352531793.0000
41527.0238,0.0000,0.0000,125972.9762,1352573320.0000
38384.0459,0.0000,0.0000,51615.9541,1352611704.0000
41900.8826,0.0000,0.0000,8099.1174,1352653604.0000
Net Profit: 1.342653604E9
```

Using the trade function, we got

Net profit : 1,342,653,604 euro cents

Trades

```
51842.3894,0.0000,0.0000,3157.6106,1351422770.0000
59046.5379,0.0000,0.0000,953.4621,1351481816.0000
56712.3119,0.0000,0.0000,787.6881,1351538528.0000
45884.1819,0.0000,0.0000,4115.8181,1351584412.0000
42123.7821,0.0000,0.0000,10376.2179,1351626535.0000
49790.3752,0.0000,0.0000,209.6248,1351676325.0000
43631.4625,0.0000,0.0000,8868.5375,1351719956.0000
47950.3546,0.0000,0.0000,4549.6454,1351767906.0000
49412.6898,0.0000,0.0000,587.3102,1351817318.0000
52699.3565,0.0000,0.0000,4800.6435,1351870017.0000
56989.9318,0.0000,0.0000,5510.0682,1351927006.0000
62500.0000,21613.4910,0.0000,0.0000,1351967893.0000
57500.0000,12696.3112,0.0000,0.0000,1352012697.0000
64832.9311,0.0000,0.0000,12667.0689,1352077529.0000
79254.4299,0.0000,0.0000,38245.5701,1352156783.0000
71005.4283,0.0000,0.0000,91494.5717,1352227788.0000
63860.2094,0.0000,0.0000,103639.7906,1352291648.0000
40510.4875,0.0000,0.0000,246989.5125,1352332158.0000
51580.4587,0.0000,0.0000,225919.5413,1352383738.0000
51597.1044,0.0000,0.0000,158402.8956,1352435335.0000
53825.7162,0.0000,0.0000,96174.2838,1352489160.0000
42633.4852,0.0000,0.0000,114866.5148,1352531793.0000
41527.0238,0.0000,0.0000,125972.9762,1352573320.0000
38384.0459,0.0000,0.0000,51615.9541,1352611704.0000
41900.8826,0.0000,0.0000,8099.1174,1352653604.0000
Net Profit: 1.342653604E9
```

Using the trade function, we got

Net profit : 1,342,653,604 euro cents

Potential Improvements

- We could have analysed the correlation between the wind speeds and directions for the different wind farms to determine which are the more significant features that should be included in our model.

- More of the available data could have been used in features that affect energy demand in our model. Increased bottleneck values would be implemented with this larger dataset, potentially reducing error.

- Results could have been rerun more times to ensure reproducibility.

Potential Improvements

- We could have analysed the correlation between the wind speeds and directions for the different wind farms to determine which are the more significant features that should be included in our model.

- More of the available data could have been used in features that affect energy demand in our model. Increased bottleneck values would be implemented with this larger dataset, potentially reducing error.

- Results could have been rerun more times to ensure reproducibility.

Conclusion

Since we have a positive net profit, 0 lag, beat persistence and also a high correlation between actual and test predictions, we conclude that our model is able to predict the forecasts relatively well.

However, we believe that our model can still be further improved as mentioned under “Potential Improvements”.

Conclusion

Since we have a positive net profit, 0 lag, beat persistence and also a high correlation between actual and test predictions, we conclude that our model is able to predict the forecasts relatively well.

However, we believe that our model can still be further improved as mentioned under “Potential Improvements”.

The End

Presented by BIGBRaiN