EnergyForecaster_iv

June 6, 2025

[1]:
```
from EnergyForecaster import EnergyForecaster
import matplotlib.pyplot as plt
import shutil
import numpy as np
```

[2]:
```
reset = True
```

[3]:
```
if reset:
    try:
        shutil.rmtree('E:\example3')
    except FileNotFoundError:
        pass
```

[4]:
```
ef = EnergyForecaster('E:\example3')
```

[5]:
```
if reset:
    ef.data_controller.import_csv('E:/data2/Original Data/
 ↪renewables_ninja_country_GR_cloud-cover_merra-2_land-wtd.csv',␣
 ↪h5_name='cloud-cover', skip=4, quotes='"', all_float=True)
    ef.data_controller.import_csv('E:/data2/Original Data/
 ↪renewables_ninja_country_GR_irradiance-surface_merra-2_land-wtd.csv',␣
 ↪h5_name='irradiance-surface', skip=4, quotes='"', all_float=True)
    ef.data_controller.import_csv('E:/data2/Original Data/
 ↪renewables_ninja_country_GR_precipitation_merra-2_land-wtd.csv',␣
 ↪h5_name='precipitation', skip=4, quotes='"', all_float=True)
    ef.data_controller.import_csv('E:/data2/Original Data/
 ↪renewables_ninja_country_GR_temperature_merra-2_land-wtd.csv',␣
 ↪h5_name='temperature', skip=4, quotes='"', all_float=True)
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201401010000-201501010000.csv', h5_name='total-load-2014',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201501010000-201601010000.csv', h5_name='total-load-2015',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201601010000-201701010000.csv', h5_name='total-load-2016',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
```

```
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201701010000-201801010000.csv', h5_name='total-load-2017',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201801010000-201901010000.csv', h5_name='total-load-2018',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
    ef.data_controller.import_csv('E:/data2/Original Data/Total Load - Day␣
 ↪Ahead _ Actual_201901010000-202001010000.csv', h5_name='total-load-2019',␣
 ↪quotes='"', all_float=True, str_to_nan=['N/A', ''])
```

[6]:
```
ef.data_controller.get_all_datasets()
```

[7]:
```
ef.data_controller.merge_datasets(['total-load-2014', 'total-load-2015',␣
 ↪'total-load-2016', 'total-load-2017', 'total-load-2018', 'total-load-2019'],␣
 ↪'load', True)
```

[8]:
```
ef.data_controller.get_dataset_names()
```

[8]:
```
['cloud-cover',
 'irradiance-surface',
 'load',
 'precipitation',
 'temperature',
 'total-load-2014',
 'total-load-2015',
 'total-load-2016',
 'total-load-2017',
 'total-load-2018',
 'total-load-2019']
```

[9]:
```
print(ef.data_controller.data_summary('temperature'))
```

```
name: time
----------
units: units
length: 376944
nans: 0
zeros: 0

name: GR
--------
units: units
length: 376944
min-max: -8.65400 -> 39.88700
mean: 15.04267
std dev: 8.13157
z-score:  0.00001
nans: 0
```

```
zeros: 2

name: EL30
----------
units: units
length: 376944
min-max: -1.54100 -> 40.79900
mean: 17.65586
std dev: 7.25441
z-score:  0.00010
nans: 0
zeros: 0

name: EL41
----------
units: units
length: 376944
min-max: -1.13700 -> 34.47400
mean: 17.36873
std dev: 6.17184
z-score:  0.00000
nans: 0
zeros: 0

name: EL42
----------
units: units
length: 376944
min-max: 2.98000 -> 31.89000
mean: 19.02152
std dev: 5.04412
z-score:  0.00000
nans: 0
zeros: 0

name: EL43
----------
units: units
length: 376944
min-max: 3.37000 -> 35.21000
mean: 18.67098
std dev: 5.62125
z-score:  0.00000
nans: 0
zeros: 0

name: EL51
----------
```

```
units: units
length: 376944
min-max: -13.10800 -> 41.05500
mean: 13.38709
std dev: 9.27375
z-score:  0.00000
nans: 0
zeros: 5

name: EL52
----------
units: units
length: 376944
min-max: -16.76200 -> 42.37500
mean: 13.98951
std dev: 9.34419
z-score:  0.00002
nans: 0
zeros: 12

name: EL53
----------
units: units
length: 376944
min-max: -21.79200 -> 41.21700
mean: 11.21398
std dev: 9.64060
z-score:  0.00002
nans: 0
zeros: 12

name: EL54
----------
units: units
length: 376944
min-max: -14.13900 -> 41.64000
mean: 13.27123
std dev: 8.63094
z-score:  0.00008
nans: 0
zeros: 5

name: EL61
----------
units: units
length: 376944
min-max: -15.73500 -> 43.16500
mean: 14.21535
```

```
std dev: 8.98596
z-score:  0.00005
nans: 0
zeros: 6

name: EL62
----------
units: units
length: 376944
min-max: 1.53200 -> 36.47700
mean: 18.31841
std dev: 5.74515
z-score:  0.00002
nans: 0
zeros: 0

name: EL63
----------
units: units
length: 376944
min-max: -6.68900 -> 41.77100
mean: 15.79267
std dev: 8.07831
z-score:  0.00006
nans: 0
zeros: 3

name: EL64
----------
units: units
length: 376944
min-max: -8.70400 -> 41.78600
mean: 15.34710
std dev: 8.28681
z-score:  0.00010
nans: 0
zeros: 1

name: EL65
----------
units: units
length: 376944
min-max: -5.58800 -> 40.49500
mean: 16.08546
std dev: 7.80284
z-score:  0.00007
nans: 0
zeros: 0
```

[10]:
```
print(ef.data_controller.data_summary('load'))
```

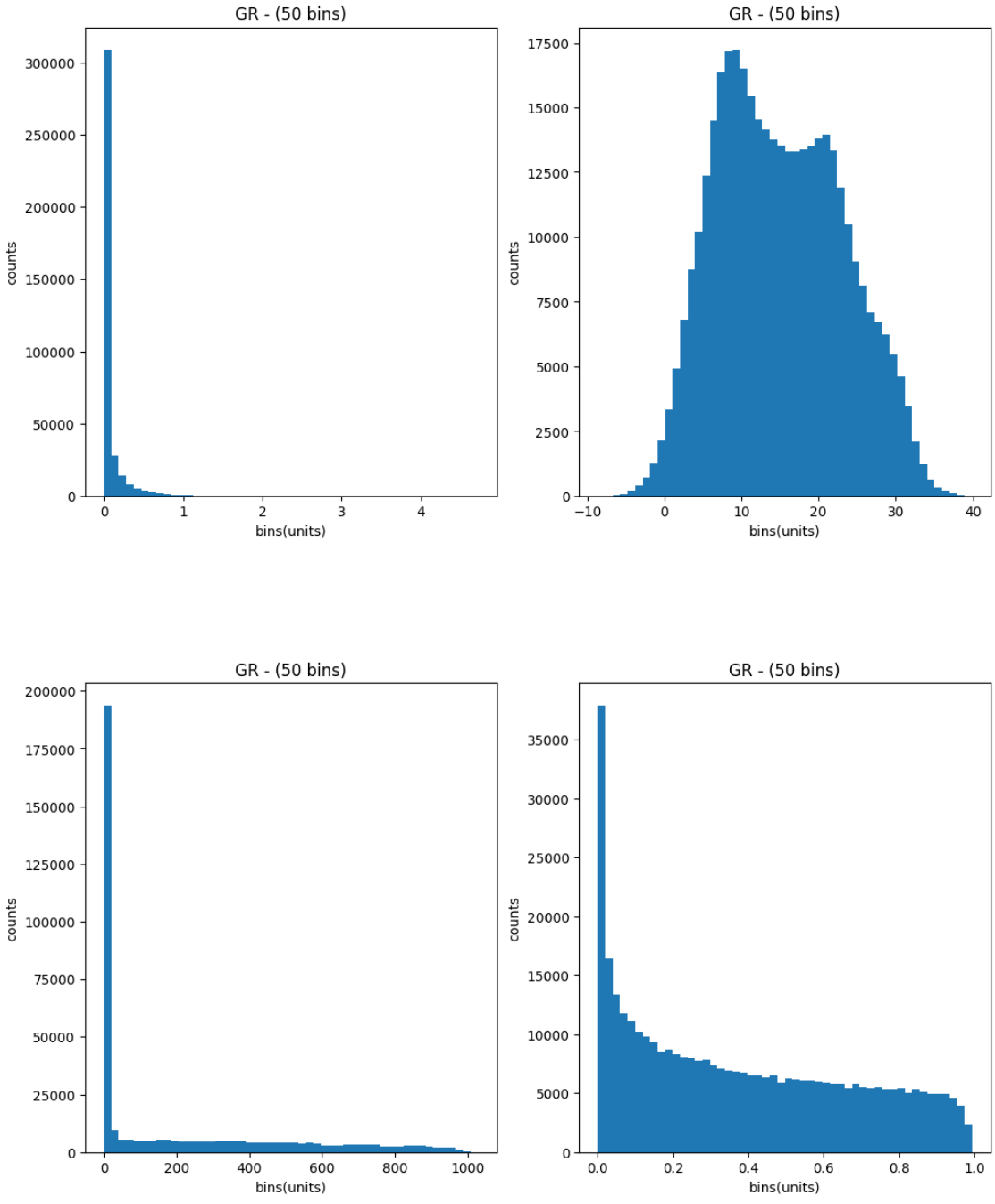
```
name: Time (UTC)
----------------
units: units
length: 52584
nans: 0
zeros: 0

name: Day-ahead Total Load Forecast [MW] - BZN|GR
-------------------------------------------------
units: units
length: 52584
min-max: 3151.00000 -> 9722.00000
mean: 5911.24030
std dev: 1133.98104
z-score:  0.00040
nans: 8710
zeros: 0

name: Actual Total Load [MW] - BZN|GR
-------------------------------------
units: units
length: 52584
min-max: 3182.00000 -> 9749.00000
mean: 5845.83430
std dev: 1119.69755
z-score:  0.00133
nans: 8759
zeros: 0
```

[11]:
```
print(ef.data_controller.data_summary('load', ['Day-ahead Total Load Forecast␣
↪[MW] - BZN|GR']))
```

```
name: Day-ahead Total Load Forecast [MW] - BZN|GR
-------------------------------------------------
units: units
length: 52584
min-max: 3151.00000 -> 9722.00000
mean: 5911.24030
std dev: 1133.98104
z-score:  0.00040
nans: 8710
zeros: 0
```

[12]:
```
columns = ['precipitation', 'temperature', 'irradiance-surface', 'cloud-cover']
```

[13]:
```
fig2 = plt.figure('histograms',figsize=(12,15))
fig2.subplots(nrows=2, ncols=2)
plt.subplots_adjust(hspace=0.4)

for i in range(len(columns)):
    ef.data_controller.datasets[columns[i]].hist(column='GR', bins=50,␣
 ↪axes=fig2.axes[i])
```



[14]:
```
if reset:
    ef.process_controller.run_process_script('E:\data2\script_1')
```

[15]:
```
ef.data_controller.datasets['temperature']
```

[15]:
```
array([(11.863, 15.538, 16.77 , 15.489, 9.992, 5.94 , 1.415, 3.134, 5.416,
11.038, 7.386, 7.632, 8.57 , 3.1553280e+08, 0.34348283, (1, 0, 0, 0, 0, 0), (1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0), ( 2.0129852e-01, 0.97952994), (0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)),
       (11.246, 15.467, 16.66 , 15.433, 8.897, 5.423, 0.896, 2.629, 5.145,
10.403, 6.943, 7.195, 8.184, 3.1553640e+08, 0.33384149, (1, 0, 0, 0, 0, 0), (1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0), ( 2.0129852e-01, 0.97952994), (1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)),
       (11.059, 15.233, 16.471, 15.194, 8.275, 5.068, 0.246, 2.206, 4.739,
9.621, 6.274, 6.811, 7.805, 3.1554000e+08, 0.3249212 , (1, 0, 0, 0, 0, 0), (1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0), ( 2.0129852e-01, 0.97952994), (0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)),
       …,
       (11.15 , 13.337, 15.544, 13.725, 7.008, 6.366, 4.076, 6.106, 6.284,
14.843, 7.599, 7.783, 8.113, 1.6725204e+09, 0.34414207, (0, 0, 0, 0, 1, 0), (0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (-2.4492936e-16, 1.        ), (0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)),
       (10.786, 13.166, 15.399, 13.586, 6.576, 6.049, 3.85 , 5.843, 5.983,
14.784, 7.188, 7.337, 7.744, 1.6725240e+09, 0.3375291 , (0, 0, 0, 0, 1, 0), (0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0), (-2.4492936e-16, 1.        ), (0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)),
       (10.377, 12.923, 15.269, 13.484, 6.225, 5.8  , 3.618, 5.621, 5.836, 14.68
, 6.825, 7.006, 7.433, 1.6725276e+09, 0.3321522 , (0, 0, 0, 0, 1, 0), (0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0), (-2.4492936e-16, 1.        ), (0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1))],
      dtype=[('EL30', '<f8'), ('EL41', '<f8'), ('EL42', '<f8'), ('EL43', '<f8'),
('EL51', '<f8'), ('EL52', '<f8'), ('EL53', '<f8'), ('EL54', '<f8'), ('EL61',
'<f8'), ('EL62', '<f8'), ('EL63', '<f8'), ('EL64', '<f8'), ('EL65', '<f8'),
('time', '<f8'), ('GR', '<f8'), ('weekday_one_hot', [('weekday_1', 'u1'),
('weekday_2', 'u1'), ('weekday_3', 'u1'), ('weekday_4', 'u1'), ('weekday_5',
'u1'), ('weekday_6', 'u1')]), ('month_one_hot', [('year_month_1', 'u1'),
('year_month_2', 'u1'), ('year_month_3', 'u1'), ('year_month_4', 'u1'),
('year_month_5', 'u1'), ('year_month_6', 'u1'), ('year_month_7', 'u1'),
('year_month_8', 'u1'), ('year_month_9', 'u1'), ('year_month_10', 'u1'),
('year_month_11', 'u1')]), ('monthday_cos_sin', [('monthday_sin', '<f8'),
('monthday_cos', '<f8')]), ('day_hour_one_hot', [('day_hour_1', 'u1'),
('day_hour_2', 'u1'), ('day_hour_3', 'u1'), ('day_hour_4', 'u1'), ('day_hour_5',
'u1'), ('day_hour_6', 'u1'), ('day_hour_7', 'u1'), ('day_hour_8', 'u1'),
('day_hour_9', 'u1'), ('day_hour_10', 'u1'), ('day_hour_11', 'u1'),
('day_hour_12', 'u1'), ('day_hour_13', 'u1'), ('day_hour_14', 'u1'),
```

```
('day_hour_15', 'u1'), ('day_hour_16', 'u1'), ('day_hour_17', 'u1'),
('day_hour_18', 'u1'), ('day_hour_19', 'u1'), ('day_hour_20', 'u1'),
('day_hour_21', 'u1'), ('day_hour_22', 'u1'), ('day_hour_23', 'u1')])])
```

[16]:
```
if reset:
    ef.process_controller.set_process('process_1', lags=72, black_lags=24,␣
 ↪measure_period=24, update_file=True)
```

[17]:
```
ef.process_controller.get_process('process_1')
```

[18]:
```
# ef.process_controller.close_process()
# ef.process_controller.remove_process('process_1')
```

[19]:
```
if reset:
    ef.process_controller.run_process_script('E:\data2\script_2')
```

[20]:
```
if reset:
    ef.process_controller.set_model(ef.Model.SARIMAX(exog=ef.process_controller.
 ↪process.get_data(),
                                                     endog=ef.
 ↪process_controller.process.get_target(),
                                                     order=(0, 0, 0)),␣
 ↪'arima_000', add_to_process=True)

    ef.process_controller.set_model(ef.Model.RandomForestRegressor(),␣
 ↪'random_forest', add_to_process=True)

    ef.process_controller.set_model(ef.Model.MLPRegressor(), 'mlp_100',␣
 ↪add_to_process=True)

    ef.process_controller.set_model(
        name='linear',
        fit_params={'batch_size': 64, 'func': ef.results_statistics.mape},
        add_to_process=True,
        model=ef.Model.TorchModel(
            input_size=330,
            device='cuda',
            components=[
                'linear', {'out_features': 1024},
                'mish', {},
                'linear', {'out_features': 1024},
                'relu', {},
                'linear', {'out_features': 1},
                'adam', {'lr': 0.005},
                'mse', {}
            ]
        )
```

```
    )

    ef.process_controller.set_model(
        name='lstm',
        fit_params={'batch_size': 64, 'func': ef.results_statistics.mape},
        add_to_process=True,
        model=ef.Model.TorchModel(
            input_size=330,
            device='cuda',
            components=[
                'lstm', {'input_size': 330, 'hidden_size': 512, 'num_layers':␣
↪1},
                'linear', {'out_features': 64},
                'linear', {'out_features': 1},
                'adam', {'lr': 0.005},
                'mse', {}
            ]
        )
    )

    ef.process_controller.set_model(
        name='conv1d',
        fit_params={'batch_size': 64, 'func': ef.results_statistics.mape},
        add_to_process=True,
        model=ef.Model.TorchModel(
            input_size=330,
            device='cuda',
            components=[
                'conv1d', {'out_channels': 16, 'kernel_size': 15, 'stride': 7},
                'mish', {},
                'linear', {'out_features': 1024},
                'relu', {},
                'linear', {'out_features': 1},
                'adam', {'lr': 0.005},
                'mse', {}
            ]
        )
    )

    ef.process_controller.update_process()

print(ef.process_controller.process.models)
```

```
['arima_000', 'random_forest', 'mlp_100', 'linear', 'lstm', 'conv1d']
```

[21]:
```
if reset:
    ef.process_controller.process.fit_models(n_epochs=300,␣
 ↪use_torch_validation=True)
```

```
Fitting model "arima_000"…
Fitting model "random_forest"…
Fitting model "mlp_100"…

E:\EnergyForecaster\venv\Lib\site-
packages\sklearn\neural_network\_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(

Fitting model "linear"…
Epoch: 300/300 (100.00%), loss: 172290.24448, best_loss: 122801.85705 (epoch
207), elapsed time: 0d 00:00:00, validation (mape): 0.10822, best validation:
0.06441 (epoch 217)
Fitting model "lstm"…
Epoch: 300/300 (100.00%), loss: 137458.52863, best_loss: 137065.66985 (epoch
300), elapsed time: 0d 00:00:00, validation (mape): 0.09161, best validation:
0.06333 (epoch 56)
Fitting model "conv1d"…
Epoch: 300/300 (100.00%), loss: 71337.40825, best_loss: 71337.40825 (epoch 301),
elapsed time: 0d 00:00:00, validation (mape): 0.08103, best validation: 0.05678
(epoch 63)
```

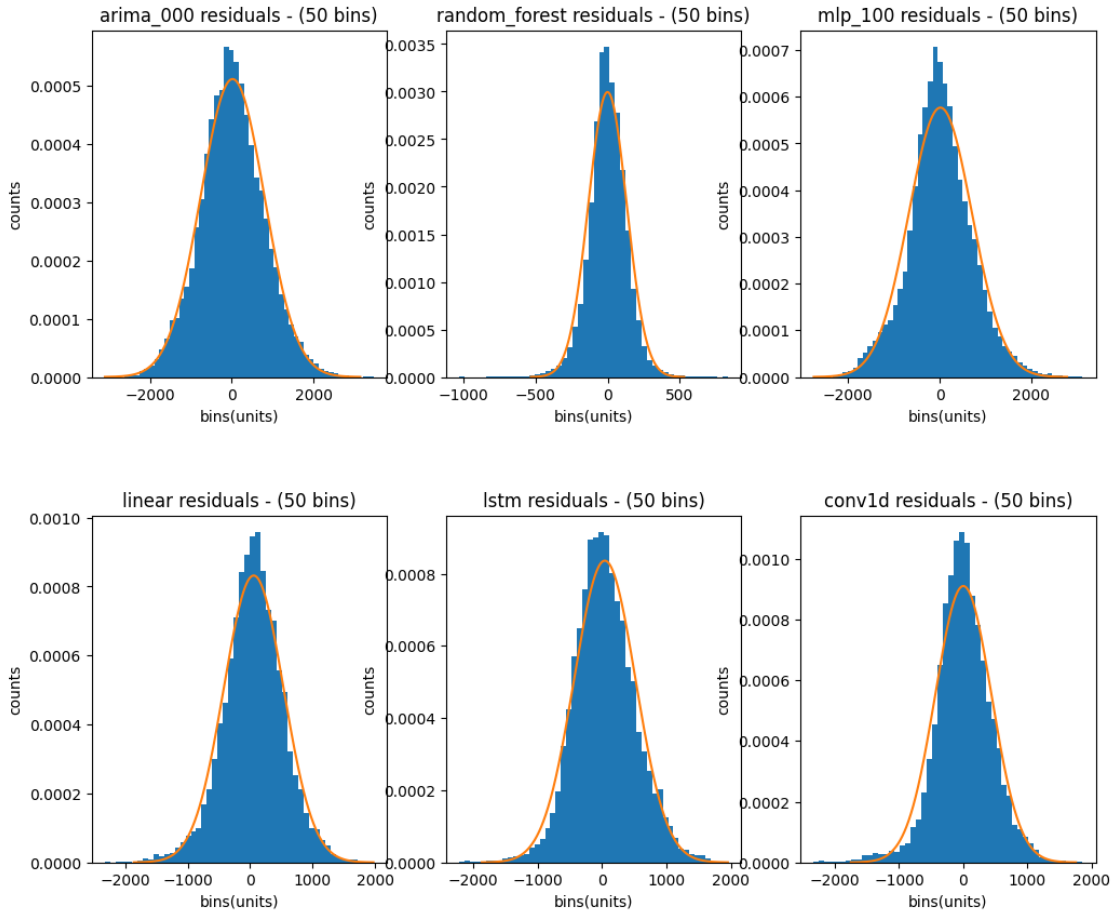
[22]:
```
print(ef.process_controller.process.evaluation_summary(data_part='test',␣
 ↪torch_best_valid=True))
```

```
              | MAPE    | wMAPE   | MAE     | RMSE    | MSE       | R2
--------------|---------|---------|---------|---------|-----------|--------
arima_000     | 0.10063 | 0.10216 | 602.381 | 764.793 | 584908.53 | 0.61870
random_forest | 0.07749 | 0.08073 | 476.019 | 644.811 | 415781.76 | 0.76599
mlp_100       | 0.08109 | 0.08528 | 502.845 | 662.361 | 438721.68 | 0.77642
linear        | 0.08088 | 0.07908 | 466.260 | 596.664 | 356007.53 | 0.73687
lstm          | 0.08329 | 0.08194 | 483.138 | 612.370 | 374997.10 | 0.71995
conv1d        | 0.06406 | 0.06309 | 372.005 | 489.802 | 239905.54 | 0.82130
```

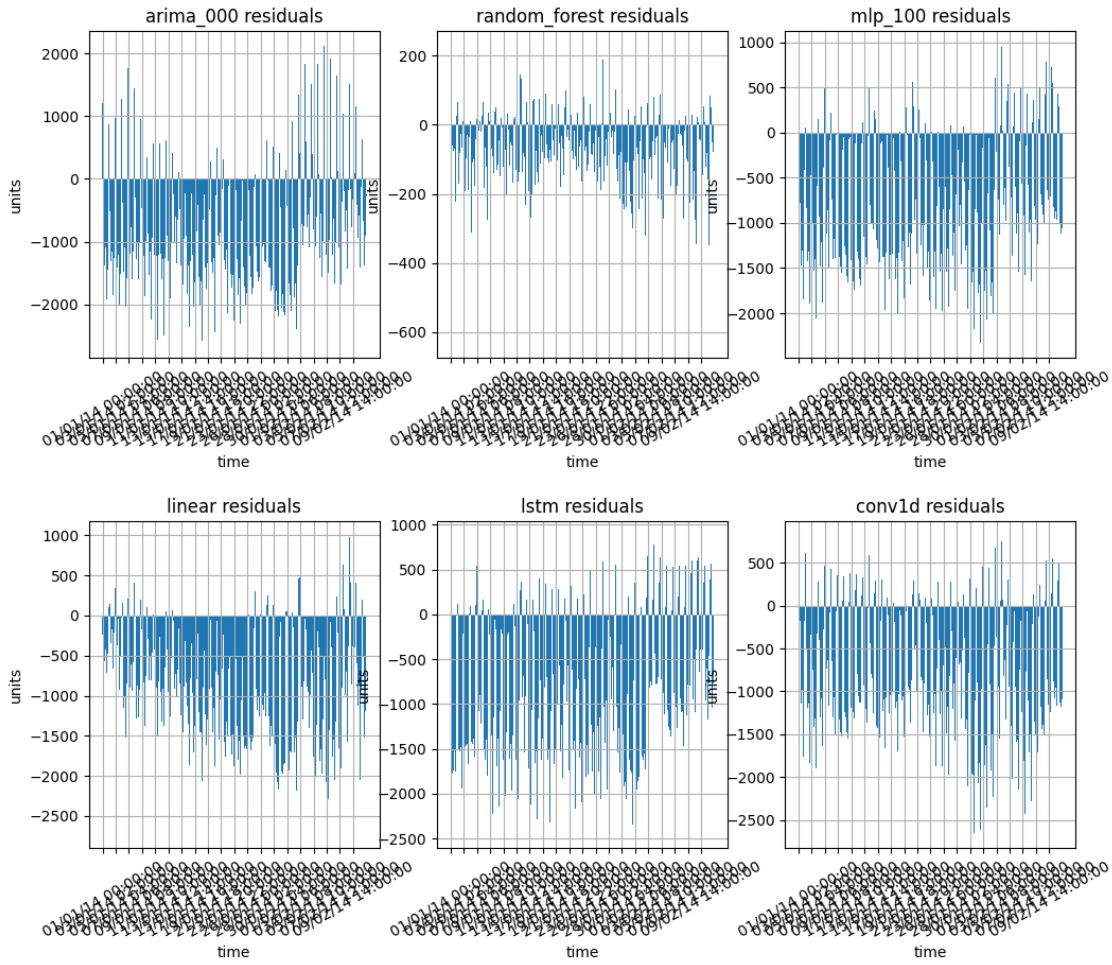
[23]:
```
for model in ef.process_controller.process.models:
    print(f"{model:<20}: {ef.process_controller.process.box_pierce(name=model,␣
 ↪torch_best_valid=False)}")
```

```
arima_000           : (119635.78932996457, 0.0)
random_forest       : (65008.37022421969, 0.0)
mlp_100             : (135310.88300064625, 0.0)
linear              : (129014.21113262541, 0.0)
lstm                : (127552.61916559406, 0.0)
conv1d              : (116283.08822030776, 0.0)
```

[24]:
```
fig3 = plt.figure('histograms',figsize=(12,10))
fig3.subplots(nrows=2, ncols=3)
plt.subplots_adjust(hspace=0.4)

for i, model in enumerate(ef.process_controller.process.models):
    ef.process_controller.process.hist_residuals(name=model, bins=50,␣
 ↪density=True, plot_norm=True, axes=fig3.axes[i])
```



[25]:
```
fig4 = plt.figure('histograms', figsize=(12, 10))
fig4.subplots(nrows=2, ncols=3)
plt.subplots_adjust(hspace=0.5)

for i, model in enumerate(ef.process_controller.process.models):
    ef.process_controller.process.plot_residuals(name=model, start=0,␣
 ↪steps=1000, axes=fig4.axes[i],
                                                 torch_best_valid=False)
```

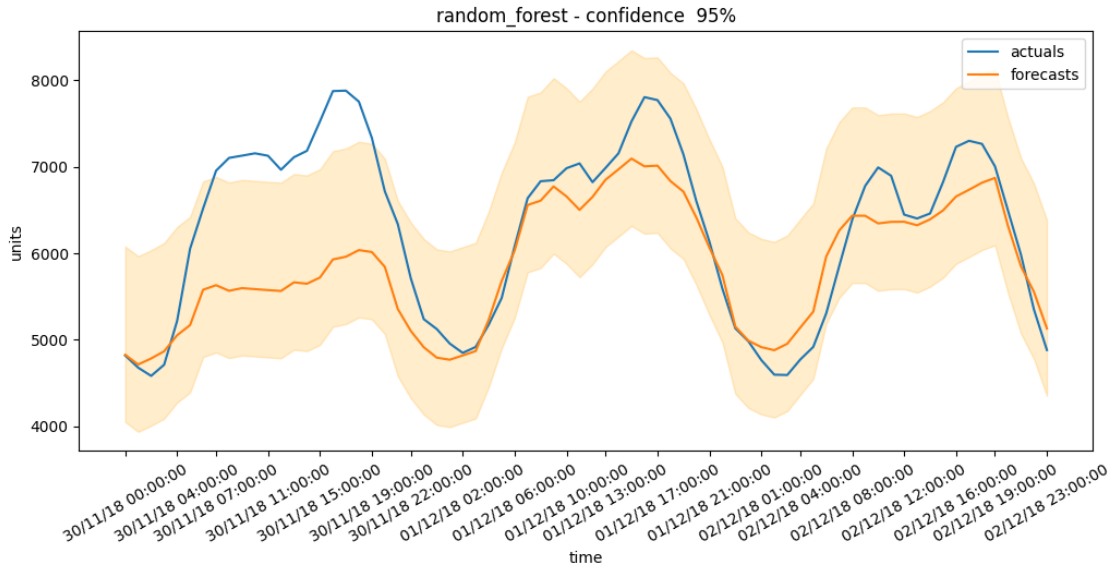


[26]:
```
ef.process_controller.process.get_forecasts('linear', data_part='validation',␣
↪torch_best_valid=True, alpha=.05, start=989, steps=24)
```

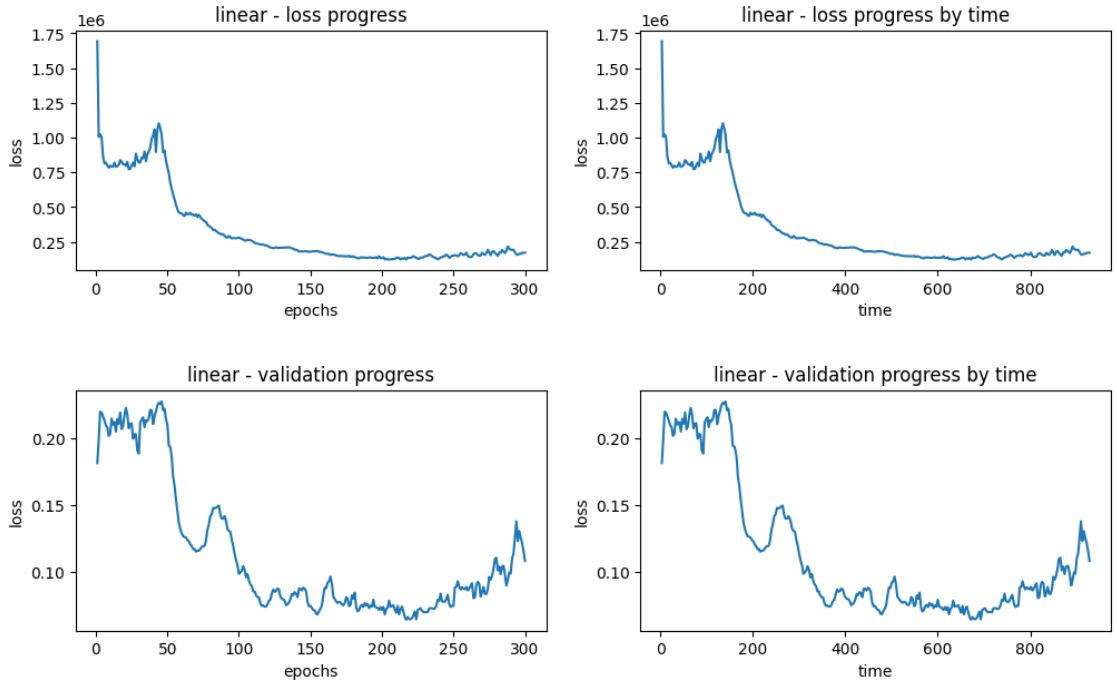
[26]:
```
{'forecast': array([[5119.7153],
        [4868.7783],
        [4681.9565],
        [4290.2373],
        [3976.9683],
        [4136.295 ],
        [3960.2432],
        [3984.277 ],
        [4265.731 ],
        [4749.907 ],
        [5576.8467],
        [6097.9272],
        [6264.846 ],
        [6408.453 ],
```

```
        [6494.9663],
        [6505.113 ],
        [6381.751 ],
        [6015.284 ],
        [5836.078 ],
        [5839.1953],
        [6093.5503],
        [6292.835 ],
        [6413.6226],
        [6094.117 ]], dtype=float32),
 'start': 989,
 'steps': 24,
 'alpha': 0.05,
 'conf_int': [(4237.170122111676, 6117.14650272044),
  (3986.233110392926, 5866.20949100169),
  (3799.411333049176, 5679.38771365794),
  (3407.692094767926, 5287.66847537669),
  (3094.423051799176, 4974.39943240794),
  (3253.749711955426, 5133.72609256419),
  (3077.697954142926, 4957.67433475169),
  (3101.731889689801, 4981.708270298565),
  (3383.185747111676, 5263.16212772044),
  (3867.362016642926, 5747.33839725169),
  (4694.301469767926, 6574.27785037669),
  (5215.382036174176, 7095.35841678294),
  (5382.300981486676, 7262.27736209544),
  (5525.907915080426, 7405.88429568919),
  (5612.421098674176, 7492.39747928294),
  (5622.567583049176, 7502.54396365794),
  (5499.205766642926, 7379.18214725169),
  (5132.738969767926, 7012.71535037669),
  (4953.532915080426, 6833.50929568919),
  (4956.650102580426, 6836.62648318919),
  (5211.005083049176, 7090.98146365794),
  (5410.289751017926, 7290.26613162669),
  (5531.077348674176, 7411.05372928294),
  (5211.571977580426, 7091.54835818919)]}
```

[27]:
```
plt.rcParams["figure.figsize"] = (12, 5)
ef.process_controller.process.plot_forecasts('random_forest', alpha=.05,␣
 ↪data_part='test', start=989, steps=72, torch_best_valid=True)
```

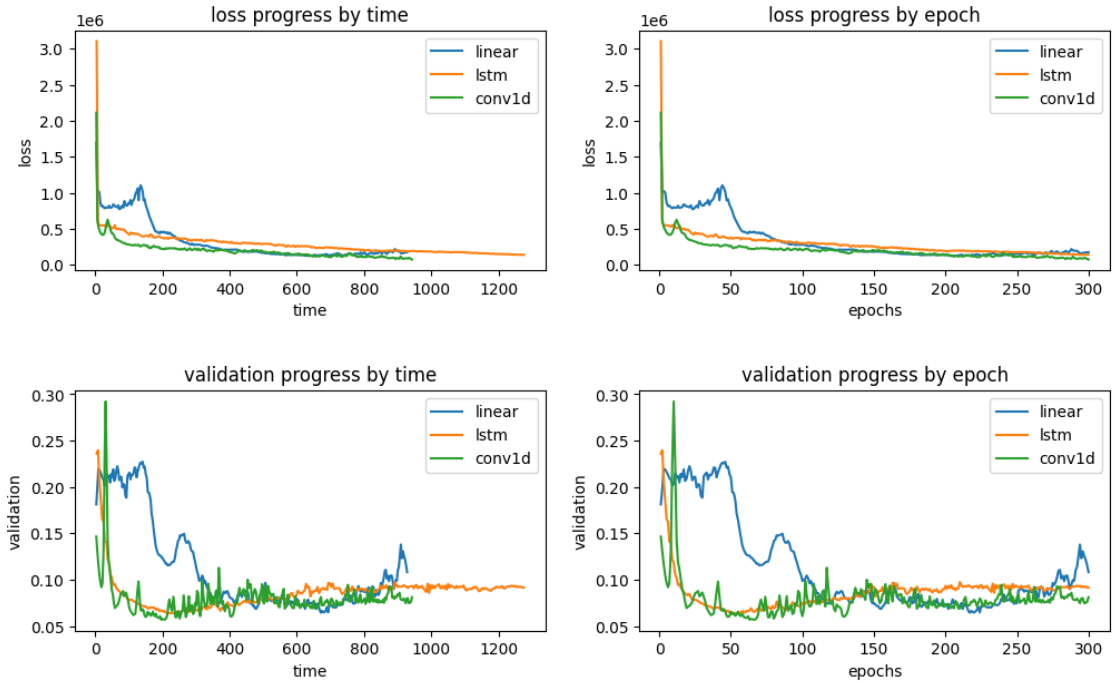


[28]:
```
fig5 = plt.figure('loss_progress',figsize=(12,7))
fig5.subplots(nrows=2, ncols=2)
plt.subplots_adjust(hspace=0.5)

ef.process_controller.process.plot_loss_by_epoch(name='linear', axes=fig5.
 ↪axes[0])
ef.process_controller.process.plot_loss_by_time(name='linear', axes=fig5.
 ↪axes[1])
ef.process_controller.process.plot_validation_by_epoch(name='linear', axes=fig5.
 ↪axes[2])
ef.process_controller.process.plot_validation_by_time(name='linear', axes=fig5.
 ↪axes[3])
```



[29]:
```
fig4 = plt.figure('loss_progress',figsize=(12,7))
fig4.subplots(nrows=2, ncols=2)
plt.subplots_adjust(hspace=0.5)

ef.process_controller.process.plot_compare_models_loss(names=['linear', 'lstm',␣
 ↪'conv1d'], time=True, use_validation=False, axes=fig4.axes[0])
ef.process_controller.process.plot_compare_models_loss(names=['linear', 'lstm',␣
 ↪'conv1d'], time=False, use_validation=False, axes=fig4.axes[1])
ef.process_controller.process.plot_compare_models_loss(names=['linear', 'lstm',␣
 ↪'conv1d'], time=True, use_validation=True, axes=fig4.axes[2])
ef.process_controller.process.plot_compare_models_loss(names=['linear', 'lstm',␣
 ↪'conv1d'], time=False, use_validation=True, axes=fig4.axes[3])
```

[29]:
```
<Axes: title={'center': 'validation progress by epoch'}, xlabel='epochs',
ylabel='validation'>
```



[30]:
```
if reset:
    ef.process_controller.set_voting_model(name='v_lstm_rf',␣
 ↪model_names=['lstm', 'random_forest'], add_to_process=True)
    ef.process_controller.set_voting_model(name='v_conv1d_rf',␣
 ↪model_names=['conv1d', 'random_forest'], add_to_process=True)
    ef.process_controller.set_voting_model(name='v_conv1d_lstm',␣
 ↪model_names=['conv1d', 'lstm'], add_to_process=True)
    ef.process_controller.set_voting_model(name='v_rf_conv1d_lstm',␣
 ↪model_names=['random_forest', 'conv1d', 'lstm'], add_to_process=True)
    ef.process_controller.update_process()
```

[31]:
```
print(ef.process_controller.process.evaluation_summary(names=['v_lstm_rf',␣
 ↪'v_conv1d_rf', 'v_conv1d_lstm', 'v_rf_conv1d_lstm'], torch_best_valid=True,␣
 ↪data_part='test'))
```

```
                 | MAPE    | wMAPE   | MAE     | RMSE    | MSE       | R2
-----------------|---------|---------|---------|---------|-----------|--------
v_lstm_rf        | 0.07140 | 0.07209 | 425.082 | 554.032 | 306951.10 | 0.81891
v_conv1d_rf      | 0.06263 | 0.06359 | 374.937 | 505.345 | 255373.95 | 0.84637
v_conv1d_lstm    | 0.06987 | 0.06874 | 405.289 | 523.092 | 273625.07 | 0.79959
v_rf_conv1d_lstm | 0.06551 | 0.06560 | 386.798 | 506.964 | 257012.90 | 0.83803
```

[32]:
```
if reset:
    ef.process_controller.process.extend_fit(name='linear', n_epochs=200,␣
 ↪use_torch_validation=True)
```

```
    ef.process_controller.process.extend_fit(name='lstm', n_epochs=200,␣
↪use_torch_validation=True)
    ef.process_controller.process.extend_fit(name='conv1d', n_epochs=200,␣
↪use_torch_validation=True)
```

```
Epoch: 500/500 (100.00%), loss: 67853.76620, best_loss: 65128.86654 (epoch 500),
elapsed time: 0d 00:00:00, validation (mape): 0.09502, best validation: 0.06441
(epoch 217)
Epoch: 500/500 (100.00%), loss: 74568.49403, best_loss: 71678.13259 (epoch 500),
elapsed time: 0d 00:00:00, validation (mape): 0.08722, best validation: 0.06333
(epoch 56)
Epoch: 500/500 (100.00%), loss: 31964.06541, best_loss: 24772.83524 (epoch 489),
elapsed time: 0d 00:00:00, validation (mape): 0.08022, best validation: 0.05678
(epoch 63)
```

[33]:
```
print(ef.process_controller.process.evaluation_summary(torch_best_valid=True,␣
↪data_part='test'))
```

```
                 | MAPE    | wMAPE   | MAE     | RMSE    | MSE       | R2
-----------------|---------|---------|---------|---------|-----------|--------
arima_000        | 0.10063 | 0.10216 | 602.381 | 764.793 | 584908.53 | 0.61870
random_forest    | 0.07749 | 0.08073 | 476.019 | 644.811 | 415781.76 | 0.76599
mlp_100          | 0.08109 | 0.08528 | 502.845 | 662.361 | 438721.68 | 0.77642
linear           | 0.08088 | 0.07908 | 466.260 | 596.664 | 356007.53 | 0.73687
lstm             | 0.08329 | 0.08194 | 483.138 | 612.370 | 374997.10 | 0.71995
conv1d           | 0.06406 | 0.06309 | 372.005 | 489.802 | 239905.54 | 0.82130
v_lstm_rf        | 0.07140 | 0.07209 | 425.082 | 554.032 | 306951.10 | 0.81891
v_conv1d_rf      | 0.06263 | 0.06359 | 374.937 | 505.345 | 255373.95 | 0.84637
v_conv1d_lstm    | 0.06987 | 0.06874 | 405.289 | 523.092 | 273625.07 | 0.79959
v_rf_conv1d_lstm | 0.06551 | 0.06560 | 386.798 | 506.964 | 257012.90 | 0.83803
```