In [1]:
```
import statistics
import pprint
import pandas as pd
import numpy as np
from random import uniform
from tslearn.utils import to_time_series_dataset
from tslearn.metrics import dtw#, gak
import plotly.express as px
import scipy.stats as st
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import seaborn as sns; sns.set()
#ToDo: Threading
```

In [2]:
```
def get_best_distribution(data):

    dist_names = ["gamma", "gumbel_l", "cauchy", "dgamma", "beta", "bet
aprime", "exponweib", "rayleigh", "fisk",
                  "gausshyper", "invweibull", "pareto", "alpha", "expo
n", "hypsecant", "mielke", "loggamma",
                  "rdist", "rice"] ## Agregar más a voluntad
    dist_results = []
    params = {}
    for dist_name in dist_names:
        dist = getattr(st, dist_name)
        param = dist.fit(data)
        params[dist_name] = param
        # Applying the Kolmogorov-Smirnov test
        D, p = st.kstest(data, dist_name, args=param)
        print("p value for "+dist_name+" = "+str(p))
        dist_results.append((dist_name, p))

    # select the best fitted distribution
    best_dist, best_p = (max(dist_results, key=lambda item: item[1]))
    # store the name of the best fit and its p value
    print("Best fitting distribution: "+str(best_dist))
    print("Best p value: "+ str(best_p))

    parms = params[best_dist]
    #print("Parameters for the best fit: "+ str(parms))

    map_parms = {}
    dist = getattr(st, best_dist)
    try:
        counter_wrong_chars = 0                                    #To
solve a bug
        for position, shape_parameter in enumerate(dist.shapes):
            #print(position, shape_parameter)
            if shape_parameter not in [' ', ',']:
                map_parms[shape_parameter] = parms[position-counter_wro
ng_chars]
            else:
                counter_wrong_chars += 1
    except:
        pass
    finally:
        map_parms["loc"] = parms[-2]
        map_parms["scale"] = parms[-1]
    print("Parameters for the best fit: "+ str(map_parms))

    return best_dist, best_p, parms, map_parms
```

In [3]:
```
def get_optimal_curves(df_curves, example_curves, ts_example_curves, di
ct_probability_distrs, prob_distrs,
                min_count_generated_curves, a, b, E_min, min_f_load, roo
f_dtw_distance, min_corr):

    I = 5000                          #5000
    acum_generated_curves = 0

    while acum_generated_curves < min_count_generated_curves:
        for i in range(1,I+1):
            C_i = [None] * 24
            h_max = int(round(uniform(19, 21),0))
            C_i[h_max] = 1

            for h, none in enumerate(C_i):
                if h != h_max:
                    function = dict_probability_distrs[prob_distrs
[h][0]]
                    parms = prob_distrs[h][1]
                    was_random_number_found = False
                    while was_random_number_found is False:
                        E = function.rvs(**parms, size=1)[0]
                        if (E>=E_min and E<1):
                            was_random_number_found = True
                    C_i[h] = E
            E_acum = sum(C_i)
            if (E_acum>=a and E_acum<=b):
                #print(C_i, type(C_i))
                f_load = statistics.mean(C_i) / max(C_i)
                if f_load >= min_f_load:
                    ts_C_i = to_time_series_dataset(C_i)[0]
                    dtw_distances = []

                    for k, curve in enumerate(ts_example_curves):
                        dtw_distance = dtw(ts_C_i, curve)
                        dtw_distances.append(dtw_distance)
                    average_dtw = statistics.mean(dtw_distances)
                    if average_dtw < roof_dtw_distance:
                        corrs = []

                        for example_curve in example_curves:
                            corr = np.corrcoef(C_i, example_curve)
                            corrs.append(corr[0][1])
                        average_corr = statistics.mean(corrs)
                        if average_corr>=min_corr:
                            print(i, f_load, E_acum, average_dtw, avera
ge_corr)
                            df_curves = df_curves.append(
                                    { '0': C_i[0], '1': C_i[1], '2': C_
i[2],
                                     '3': C_i[3], '4': C_i[4], '5': C_i
[5],
                                     '6': C_i[6], '7': C_i[7], '8': C_i
[8],
                                     '9': C_i[9], '10': C_i[10], '11':
```

```
C_i[11],
                                        '12': C_i[12], '13': C_i[13], '14
': C_i[14],
                                        '15': C_i[15], '16': C_i[16], '17
': C_i[17],
                                        '18': C_i[18], '19': C_i[19], '20
': C_i[20],
                                        '21': C_i[21], '22': C_i[22], '23
': C_i[23],
                                        'FC': f_load, 'Sum': E_acum,
                                        'DTW_avg_distance': average_dtw, '
Avg_correlation': average_corr },
                                    ignore_index=True
                                )
                                acum_generated_curves += 1
                                if acum_generated_curves>=min_count_generat
ed_curves:

                                    return (df_curves)
```

In [4]:
```
df_example_curves = pd.read_excel (r'Curvas.xlsx')
df_example_curves.drop(
    df_example_curves.columns[
        df_example_curves.columns.str.contains('unnamed', case = False,
na=False)
    ],
    axis = 1,
    inplace = True
)

a = df_example_curves['Sum'].min()
b = df_example_curves['Sum'].max()

df_example_curves = df_example_curves.drop(['FC', 'Sum', 'Comentario'],
axis=1)

print("a: ", a, " b: ", b)
print(df_example_curves)
```

```
a:  15.161736140999999  b:  19.249227906976746
          0         1         2         3         4         5
6   \
0  0.465685  0.397058  0.367646  0.372548  0.382352  0.421568  0.5686
26
1  0.637209  0.506977  0.469767  0.469767  0.488372  0.548837  0.7255
81
2  0.637209  0.506977  0.469767  0.469767  0.488372  0.548837  0.7055
81
3  0.617209  0.486977  0.449767  0.449767  0.468372  0.528837  0.6855
81
4  0.589328  0.474497  0.439237  0.440463  0.456867  0.512020  0.6713
43
5  0.539052  0.436891  0.403702  0.403582  0.420730  0.470823  0.6210
96

          7         8         9   ...        14        15        16
17  \
0  0.622548  0.784315  0.779408  ...  0.647058  0.504901  0.495097
0.460784
1  0.800000  0.990698  0.981395  ...  0.818605  0.865116  0.893023
0.734884
2  0.800000  0.901320  0.891395  ...  0.818605  0.865116  0.893023
0.767884
3  0.800000  0.901320  0.891395  ...  0.818605  0.865116  0.873023
0.747884
4  0.755637  0.894413  0.885899  ...  0.775718  0.775062  0.788542
0.677859
5  0.702376  0.844583  0.829497  ...  0.728547  0.718564  0.731319
0.641747

          18        19        20        21        22        23
0  0.593136  0.931371  1.000000  0.936270  0.838232  0.622548
1  0.758140  0.995349  1.000000  0.999256  0.989507  0.809302
2  0.758140  0.965742  0.994320  1.000000  0.932314  0.809302
3  0.758140  0.925742  1.000000  0.956440  0.932314  0.759302
4  0.716889  0.954551  0.999999  1.000000  0.923092  0.750114
5  0.694820  1.000000  0.975000  0.978047  0.900141  0.709570

[6 rows x 24 columns]
```

In [5]:
```
prob_distrs = []
plots = []
for (columnName, columnData) in df_example_curves.iteritems():
    ## Maximizar el p-value ##
    print('Colunm Name : ', columnName)
    #print('Column Contents : ', columnData.values, type(columnData.values), columnData.values.shape)
    best_dist, best_p, parms, map_parms = get_best_distribution(columnData.values)
    prob_distrs.append([best_dist, map_parms])
    #if columnName == 12:
    #    ax = sns.distplot(columnData.values, kde=False)
    #ax = sns.distplot(columnData.values, kde=False)
print("prob_distrs: ")
pprint.pprint(prob_distrs)
```

```
Colunm Name :  0
p value for gamma = 0.9153858662603467
p value for gumbel_l = 0.9611650805916115
p value for cauchy = 0.5828338337557992
p value for dgamma = 0.7103295891934879
p value for beta = 0.4258226481275949
p value for betaprime = 0.9240764030877295

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_continuous_di
stns.py:708: RuntimeWarning: divide by zero encountered in true_divid
e
  a/(b-1.0),
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_continuous_di
stns.py:712: RuntimeWarning: divide by zero encountered in true_divid
e
  a*(a+1.0)/((b-2.0)*(b-1.0)),
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_distn_infrast
ructure.py:1063: RuntimeWarning: invalid value encountered in subtrac
t
  mu2 = mu2p - mu * mu
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_distn_infrast
ructure.py:2407: RuntimeWarning: invalid value encountered in double_
scalars
  Lhat = muhat - Shat*mu

p value for exponweib = 0.09960496852055273
p value for rayleigh = 0.6757859930739376
p value for fisk = 0.9528344588739082

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_continuous_di
stns.py:3373: RuntimeWarning: divide by zero encountered in power
  return 1.0/Cinv * x**(a-1.0) * (1.0-x)**(b-1.0) / (1.0+z*x)**c

p value for gausshyper = 0.30576640871424515
p value for invweibull = 0.8417519047290813
p value for pareto = 0.5336665364053506
p value for alpha = 0.9067330652747674
p value for expon = 0.4587312555913199
p value for hypsecant = 0.9223079906180552
p value for mielke = 0.47566663473773685
p value for loggamma = 0.0037722908093278493

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_distn_infrast
ructure.py:1702: RuntimeWarning: divide by zero encountered in log
  return log(self._pdf(x, *args))
```

```
p value for rdist = 0.0
p value for rice = 0.9112020080835465
Best fitting distribution: gumbel_l
Best p value: 0.9611650805916115
Parameters for the best fit: {'loc': 0.6085684047258331, 'scale': 0.0
42804155481658675}
Colunm Name :  1
p value for gamma = 0.8889958053792573
p value for gumbel_l = 0.9646057727711407
p value for cauchy = 0.8261402488613532
p value for dgamma = 0.9029286229445405
p value for beta = 0.42576280633747704
p value for betaprime = 0.8990403693904233
p value for exponweib = 0.14434269055250448
p value for rayleigh = 0.6539560891521108
p value for fisk = 0.9763835373511555
p value for gausshyper = 0.4257544581462159
p value for invweibull = 0.8033963410294281
p value for pareto = 0.43577568017068163
p value for alpha = 0.8662969549571008
p value for expon = 0.438748236767811
p value for hypsecant = 0.9617653431217729
p value for mielke = 0.9338345346805417
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.8828955905498191
Best fitting distribution: fisk
Best p value: 0.9763835373511555
Parameters for the best fit: {'c': 110700791.6736241, 'loc': -267925
2.0299123884, 'scale': 2679252.5025851782}
Colunm Name :  2
p value for gamma = 0.8880133797117553
p value for gumbel_l = 0.9603542298049117
p value for cauchy = 0.8503807156937999
p value for dgamma = 0.925443406240686
p value for beta = 0.4257580548841562
p value for betaprime = 0.9055074478771753
p value for exponweib = 0.018042221288103823
p value for rayleigh = 0.6534853437425535
p value for fisk = 0.9853322716409895
p value for gausshyper = 0.32739697804479817
p value for invweibull = 0.6862142836462212
p value for pareto = 0.5191456709040805
p value for alpha = 0.877983073622411
p value for expon = 0.43608645826686726
p value for hypsecant = 0.9685694883456961
p value for mielke = 0.458096562321279
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.8789205538846989
Best fitting distribution: fisk
Best p value: 0.9853322716409895
Parameters for the best fit: {'c': 489772.7331588231, 'loc': -10700.8
54448774731, 'scale': 10701.291971158724}
Colunm Name :  3
p value for gamma = 0.8646588801686683
```

```
p value for gumbel_l = 0.9594013468474849
p value for cauchy = 0.8572999478325235
p value for dgamma = 0.9334887453165497
p value for beta = 0.4257627549572284
p value for betaprime = 0.9051923058931672
p value for exponweib = 0.021779940761754053
p value for rayleigh = 0.6402162044840866
p value for fisk = 0.9780979288657342
p value for gausshyper = 0.3470643413439783
p value for invweibull = 0.7158332576038476
p value for pareto = 0.3931039060807708
p value for alpha = 0.8661149486735855
p value for expon = 0.4246686745275836
p value for hypsecant = 0.9698762755260467
p value for mielke = 0.935808156047545
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.8605570869653416
Best fitting distribution: fisk
Best p value: 0.9780979288657342
Parameters for the best fit: {'c': 116592818.89739853, 'loc': -246097
1.9200478606, 'scale': 2460972.3582348386}
Colunm Name :  4
p value for gamma = 0.8965748044479315
p value for gumbel_l = 0.9626779479601569
p value for cauchy = 0.8394204068060477
p value for dgamma = 0.9152319685565558
p value for beta = 0.4257613743918177
p value for betaprime = 0.8066199059659821
p value for exponweib = 0.11344321021270526
p value for rayleigh = 0.6543915959226873
p value for fisk = 0.986143784521361
p value for gausshyper = 0.3906556726115891
p value for invweibull = 0.8035296188722569
p value for pareto = 0.4312838610891643
p value for alpha = 0.8755954409799727
p value for expon = 0.43891407443899394
p value for hypsecant = 0.9653808682887519
p value for mielke = 0.517711153776423
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.8832069624216988
Best fitting distribution: fisk
Best p value: 0.986143784521361
Parameters for the best fit: {'c': 89049207.38371027, 'loc': -198658
3.0182634608, 'scale': 1986583.4736009042}
Colunm Name :  5
p value for gamma = 0.8897459584800032
p value for gumbel_l = 0.970927442906684
p value for cauchy = 0.7117286422697515
p value for dgamma = 0.8579798425377843
p value for beta = 0.4257809008556552
p value for betaprime = 0.9230155289172822
p value for exponweib = 0.0019271563280967017
p value for rayleigh = 0.6630777639028033
p value for fisk = 0.9758261422843311
```

```
p value for gausshyper = 0.42575445814207347
p value for invweibull = 0.8038553334747851
p value for pareto = 0.4695851665864335
p value for alpha = 0.8960460193519899
p value for expon = 0.4463011938337734
p value for hypsecant = 0.9486089600196449
p value for mielke = 0.48699425897357856
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.8940037370173213
Best fitting distribution: fisk
Best p value: 0.9758261422843311
Parameters for the best fit: {'c': 129895838.77416685, 'loc': -348898
6.76553393, 'scale': 3488987.276380528}
Colunm Name :  6
p value for gamma = 0.8980711629535092
p value for gumbel_l = 0.9981493181130431
p value for cauchy = 0.9695410392634496
p value for dgamma = 0.9687455462436659
p value for beta = 0.996255197908629
p value for betaprime = 0.8546792905999896
p value for exponweib = 0.08578575673502599
p value for rayleigh = 0.6593123351515688
p value for fisk = 0.9831150580280652
p value for gausshyper = 0.8185259288766672
p value for invweibull = 0.8004204115288965
p value for pareto = 0.5103204911545219
p value for alpha = 0.8692031022654653
p value for expon = 0.4378665272165728
p value for hypsecant = 0.991131412052655
p value for mielke = 0.9998358673660042
p value for loggamma = 4.2866941015089106e-05
p value for rdist = 0.0
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_continuous_di
stns.py:1608: RuntimeWarning: invalid value encountered in add
  negxc + sc.xlogy(c - 1.0, x))
```

```
p value for exponweib = 0.03883655304054355
p value for rayleigh = 0.6362219194693021
p value for fisk = 0.81077888034992
p value for gausshyper = 0.06558641975271512
p value for invweibull = 0.8432989693608792
p value for pareto = 0.4909984342470688
p value for alpha = 0.6187465093875405
p value for expon = 0.46038765914758417
p value for hypsecant = 0.8010978605201914
p value for mielke = 0.08650095941200725
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.6150327026762274
Best fitting distribution: invweibull
Best p value: 0.8432989693608792
Parameters for the best fit: {'c': 89762340.98886724, 'loc': -641827
1.597604012, 'scale': 6418272.308923408}
Colunm Name :  8
p value for gamma = 0.8909274136772382
p value for gumbel_l = 0.5668489340967714
p value for cauchy = 0.8089051243416464
p value for dgamma = 0.4257526467691534
p value for beta = 0.2072037795466098
p value for betaprime = 0.902588161776341
p value for exponweib = 0.30383723296705595
p value for rayleigh = 0.7341108707671156
p value for fisk = 0.9071054886509802
p value for gausshyper = 0.7417524962010064
p value for invweibull = 0.8143495044364858
p value for pareto = 0.44999428134931413
p value for alpha = 0.8976045948696798
p value for expon = 0.44702201335536357
p value for hypsecant = 0.9049309439911588
p value for mielke = 0.03779365280092401
p value for loggamma = 0.8813912827002307
p value for rdist = 0.0
p value for rice = 0.8950602943009982
Best fitting distribution: fisk
Best p value: 0.9071054886509802
Parameters for the best fit: {'c': 24349073.45796358, 'loc': -868246.
0790497053, 'scale': 868246.9655063269}
Colunm Name :  9
p value for gamma = 0.8951485153215494
p value for gumbel_l = 0.5616206928215003
p value for cauchy = 0.7059692741400958
p value for dgamma = 0.4257544567064905
p value for beta = 0.27331963201966203
p value for betaprime = 0.8789583370699603
p value for exponweib = 0.22512866129595388
p value for rayleigh = 0.713728862407169
p value for fisk = 0.8925500777526092

C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\_continuous_di
stns.py:547: RuntimeWarning: invalid value encountered in sqrt
  sk = 2*(b-a)*np.sqrt(a + b + 1) / (a + b + 2) / np.sqrt(a*b)
```

```
p value for beta = 0.42575445968738856
p value for betaprime = 0.9340419420995283
p value for exponweib = 0.1912368964165128
p value for rayleigh = 0.686326276822022
p value for fisk = 0.9864105748028513
p value for gausshyper = 0.42575445815486407
p value for invweibull = 0.8237504309089143
p value for pareto = 0.4552939665002624
p value for alpha = 0.9020986421326999
p value for expon = 0.4531596049126275
p value for hypsecant = 0.994839115540973
p value for mielke = 0.6286174246263803
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.9050530696799255
Best fitting distribution: hypsecant
Best p value: 0.994839115540973
Parameters for the best fit: {'loc': 0.8937249916078858, 'scale': 0.0
49857587343792586}
Colunm Name :  11
p value for gamma = 0.6303708391173221
p value for gumbel_l = 0.47652857952026006
p value for cauchy = 0.06644746581252656
p value for dgamma = 0.44063818285977574
p value for beta = 0.06563832891444513
p value for betaprime = 0.626898464523127
p value for exponweib = 0.09450109823857221
p value for rayleigh = 0.6459534776456458
p value for fisk = 0.8289207231424871
p value for gausshyper = 0.06558641975240037
p value for invweibull = 0.7216347757775902
p value for pareto = 0.47739242750312594
p value for alpha = 0.6400028113061278
p value for expon = 0.4583372179696159
p value for hypsecant = 0.8010501963083594
p value for mielke = 0.09747005894242201
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.6230603972351948
Best fitting distribution: fisk
Best p value: 0.8289207231424871
Parameters for the best fit: {'c': 25622.28074224032, 'loc': -1046.21
63111149446, 'scale': 1047.100475108277}
Colunm Name :  12
p value for gamma = 0.9000339397648278
p value for gumbel_l = 0.986932609089354
p value for cauchy = 0.8957499243633867
p value for dgamma = 0.9195317886793581
p value for beta = 0.8940135607369734
p value for betaprime = 0.9287012530031191
p value for exponweib = 0.04355403087344065
p value for rayleigh = 0.6879614830362372
p value for fisk = 0.9857300781258278
p value for gausshyper = 0.5054685602639493
p value for invweibull = 0.8389559984013213
p value for pareto = 0.441526390547176
```

```
p value for alpha = 0.8957920729322958
p value for expon = 0.45404035313855945
p value for hypsecant = 0.9906679185278626
p value for mielke = 0.027657469673356166
p value for loggamma = 0.9773037695743986
p value for rdist = 0.0
p value for rice = 0.9063696546221549
Best fitting distribution: hypsecant
Best p value: 0.9906679185278626
Parameters for the best fit: {'loc': 0.9085276840337329, 'scale': 0.0
4491738502602223}
Colunm Name :  13
p value for gamma = 0.67254065485954
p value for gumbel_l = 0.4784651973451519
p value for cauchy = 0.0666273244413018
p value for dgamma = 0.44015264573054164
p value for beta = 0.06563459245560707
p value for betaprime = 0.6125997822792638
p value for exponweib = 0.2123716410391767
p value for rayleigh = 0.6456940664506213
p value for fisk = 0.8146419010081765
p value for gausshyper = 0.06558641975290405
p value for invweibull = 0.731514735668143
p value for pareto = 0.4332775007958757
p value for alpha = 0.6213550165874666
p value for expon = 0.4638525796760709
p value for hypsecant = 0.8005592515132045
p value for mielke = 0.1482351006331736
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.6223209184066696
Best fitting distribution: fisk
Best p value: 0.8146419010081765
Parameters for the best fit: {'c': 85052737.61507161, 'loc': -343575
0.954842735, 'scale': 3435751.8014597762}
Colunm Name :  14
p value for gamma = 0.6206817462694757
p value for gumbel_l = 0.47880227739560177
p value for cauchy = 0.0672871386969308
p value for dgamma = 0.4403974802933155
p value for beta = 0.06563889648993398
p value for betaprime = 0.6858479000485802
p value for exponweib = 0.0160556769080774
p value for rayleigh = 0.6471355258324951
p value for fisk = 0.8155182840071702
p value for gausshyper = 0.06558641975238105
p value for invweibull = 0.8250735954059218
p value for pareto = 0.44703962550681037
p value for alpha = 0.6169213128999123
p value for expon = 0.4611413063265274
p value for hypsecant = 0.800325282066208
p value for mielke = 0.06751804375302277
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.6226851101537889
Best fitting distribution: invweibull
```

```
Best p value: 0.8250735954059218
Parameters for the best fit: {'c': 13054.131467397729, 'loc': -897.43
35864004695, 'scale': 898.1653702212532}
Colunm Name :  15
p value for gamma = 0.3762901077683317
p value for gumbel_l = 0.4835523102628958
p value for cauchy = 0.06643961993005029
p value for dgamma = 0.47166772459391676
p value for beta = 0.0657906271561995
p value for betaprime = 0.6799738276128438
p value for exponweib = 0.05624535993535162
p value for rayleigh = 0.5881505067483151
p value for fisk = 0.8437465799007223
p value for gausshyper = 0.06558641975067768
p value for invweibull = 0.7096558751003664
p value for pareto = 0.2593679484386502
p value for alpha = 0.680366148734665
p value for expon = 0.24238426958265474
p value for hypsecant = 0.7375412878028462
p value for mielke = 0.15430418101226823
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.0
Best fitting distribution: fisk
Best p value: 0.8437465799007223
Parameters for the best fit: {'c': 115488161.99855241, 'loc': -812132
9.627112514, 'scale': 8121330.414561536}
Colunm Name :  16
p value for gamma = 0.2919129621354061
p value for gumbel_l = 0.7242884012085219
p value for cauchy = 0.36107777114150025
p value for dgamma = 0.5825763310552533
p value for beta = 0.4262741701229195
p value for betaprime = 0.8486320957033308
p value for exponweib = 0.2102342061577885
p value for rayleigh = 0.5492950827260542
p value for fisk = 0.9045892179461279
p value for gausshyper = 0.14429030542277665
p value for invweibull = 0.6331440522769305
p value for pareto = 0.2609791485979628
p value for alpha = 0.8399127050950357
p value for expon = 0.22856676012098903
p value for hypsecant = 0.8551896110840269
p value for mielke = 0.6843350154384503
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.0
Best fitting distribution: fisk
Best p value: 0.9045892179461279
Parameters for the best fit: {'c': 67096.92213942789, 'loc': -5107.48
7236737725, 'scale': 5108.28892342235}
Colunm Name :  17
p value for gamma = 0.019772857946657573
p value for gumbel_l = 0.9004192015853957
p value for cauchy = 0.7279729407390465
p value for dgamma = 0.8874087117014774
```

```
p value for beta = 0.6756769142638492
p value for betaprime = 0.9150416640420204
p value for exponweib = 0.06691649705021503
p value for rayleigh = 0.47726793222145436
p value for fisk = 0.974231510527606
p value for gausshyper = 0.13159242251700254
p value for invweibull = 0.6541472070216855
p value for pareto = 0.20352211747404736
p value for alpha = 0.9077311275010344
p value for expon = 0.2029074493410869
p value for hypsecant = 0.9609063276729631
p value for mielke = 0.9848303581998389
p value for loggamma = 4.2866941015089106e-05
p value for rdist = 0.0
p value for rice = 0.0
Best fitting distribution: mielke
Best p value: 0.9848303581998389
Parameters for the best fit: {'k': 73326215.23085016, 's': 171141943
9.8636742, 'loc': -7400304.56267521, 'scale': 7400305.33546778}
Colunm Name :  18
p value for gamma = 0.08240545537395466
p value for gumbel_l = 0.48706557118831884
p value for cauchy = 0.06696354454564168
p value for dgamma = 0.47851016292457665
p value for beta = 0.06581480326322475
p value for betaprime = 0.7028046121273227
p value for exponweib = 0.017275901989393102
p value for rayleigh = 0.5056750049202393
p value for fisk = 0.8338576230706877
p value for gausshyper = 0.06558641974695038
p value for invweibull = 0.573645939526151
p value for pareto = 0.23770983603904716
p value for alpha = 0.6936920266648501
p value for expon = 0.21326094427435005
p value for hypsecant = 0.7307888334887699
p value for mielke = 0.41078769849771235
p value for loggamma = 0.06558641975308642
p value for rdist = 0.0
p value for rice = 0.0
Best fitting distribution: fisk
Best p value: 0.8338576230706877
Parameters for the best fit: {'c': 294993556.1385083, 'loc': -905152
8.952159427, 'scale': 9051529.67594156}
Colunm Name :  19
p value for gamma = 0.9516363575303212
p value for gumbel_l = 0.953853831982619
p value for cauchy = 0.9888000151876468
p value for dgamma = 0.9376793491764674
p value for beta = 0.9770414499693317
p value for betaprime = 0.958820968010875
p value for exponweib = 0.057868731964175586
p value for rayleigh = 0.940404850100629
p value for fisk = 0.28887485288320547
p value for gausshyper = 0.0392041605139932
p value for invweibull = 0.9718769002363463
p value for pareto = 0.9466782426944375
```

```
p value for alpha = 0.9521724839190224
p value for expon = 0.9470509015881015
p value for hypsecant = 0.9623269532515218
p value for mielke = 0.9663380662901748
p value for loggamma = 0.9531845732660127
p value for rdist = 0.9952823413036828
p value for rice = 0.9388879901589855
Best fitting distribution: rdist
Best p value: 0.9952823413036828
Parameters for the best fit: {'c': 1.0609993438745944, 'loc': 0.96465
94345073922, 'scale': 0.03891703450739226}
Colunm Name :  20
p value for gamma = 0.008100258685190079
p value for gumbel_l = 0.18311500709503842
p value for cauchy = 0.0677097913134959
p value for dgamma = 0.06558645565109904
p value for beta = 0.31653555007241735
p value for betaprime = 0.287260453320707
p value for exponweib = 0.013147581956943096
p value for rayleigh = 0.22954751123061612
p value for fisk = 0.4050575975137472
p value for gausshyper = 0.003778954425048489
p value for invweibull = 0.36996067841006147
p value for pareto = 0.11970590838192162
p value for alpha = 0.26116644583157766
p value for expon = 0.11810363540671524
p value for hypsecant = 0.32089554628644035
p value for mielke = 0.13929221701829408
p value for loggamma = 0.0037913779475018975
p value for rdist = 0.0
p value for rice = 0.0
Best fitting distribution: fisk
Best p value: 0.4050575975137472
Parameters for the best fit: {'c': 341675704.2338467, 'loc': -150853
4.8998198714, 'scale': 1508535.8968744436}
Colunm Name :  21
p value for gamma = 0.5500989648653845
p value for gumbel_l = 0.4836560296376879
p value for cauchy = 0.14448092094878107
p value for dgamma = 0.48684156078535634
p value for beta = 0.4259367161814764
p value for betaprime = 0.5395174190829946
p value for exponweib = 0.02373109072598405
p value for rayleigh = 0.5700710901669934
p value for fisk = 0.6964541271509157
p value for gausshyper = 0.09657091802714728
p value for invweibull = 0.6998147866412076
p value for pareto = 0.5756244624394926
p value for alpha = 0.5464521333791471
p value for expon = 0.5816161203604638
p value for hypsecant = 0.8419260444022465
p value for mielke = 0.12758783491252562
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.5586562416572148
Best fitting distribution: hypsecant
```

```
Best p value: 0.8419260444022465
Parameters for the best fit: {'loc': 0.9832313307890309, 'scale': 0.0
18033385915142468}
Colunm Name :  22
p value for gamma = 0.9495739946393842
p value for gumbel_l = 0.655404848825679
p value for cauchy = 0.917781871334807
p value for dgamma = 0.4258475165907629
p value for beta = 0.19517346652265957
p value for betaprime = 0.9376465814834966
p value for exponweib = 0.0476178667203046
p value for rayleigh = 0.8208938196368079
p value for fisk = 0.9160532591991455
p value for gausshyper = 0.8851016266536179
p value for invweibull = 0.8715110297563478
p value for pareto = 0.2981828931909373
p value for alpha = 0.9475405143362311
p value for expon = 0.3123403992562756
p value for hypsecant = 0.9171445549717535
p value for mielke = 0.26191951270452685
p value for loggamma = 0.8692540949219563
p value for rdist = 0.0
p value for rice = 0.9497100431378136
Best fitting distribution: rice
Best p value: 0.9497100431378136
Parameters for the best fit: {'b': 1.943763232844457, 'loc': 0.809480
4453402771, 'scale': 0.04938076437684877}
Colunm Name :  23
p value for gamma = 0.9424008887894284
p value for gumbel_l = 0.9387185867241562
p value for cauchy = 0.9770143223733181
p value for dgamma = 0.9961169634109586
p value for beta = 0.42575446119808
p value for betaprime = 0.9459986495638795
p value for exponweib = 0.0311407553304246
p value for rayleigh = 0.724291674513198
p value for fisk = 0.9959848159529379
p value for gausshyper = 0.42575445815478546
p value for invweibull = 0.8724804774158588
p value for pareto = 0.3929785009162604
p value for alpha = 0.9386427609716709
p value for expon = 0.37884648563088963
p value for hypsecant = 0.9941053555538182
p value for mielke = 0.8145674189358512
p value for loggamma = 0.0037722908093278493
p value for rdist = 0.0
p value for rice = 0.9400473552026078
Best fitting distribution: dgamma
Best p value: 0.9961169634109586
Parameters for the best fit: {'a': 0.9150410513721037, 'loc': 0.75930
23255813959, 'scale': 0.05044580639008252}
prob_distrs:
[['gumbel_l', {'loc': 0.6085684047258331, 'scale': 0.0428041554816586
75}],
 ['fisk',
  {'c': 110700791.6736241,
```

```
   'loc': -2679252.0299123884,
   'scale': 2679252.5025851782}],
 ['fisk',
  {'c': 489772.7331588231,
   'loc': -10700.854448774731,
   'scale': 10701.291971158724}],
 ['fisk',
  {'c': 116592818.89739853,
   'loc': -2460971.9200478606,
   'scale': 2460972.3582348386}],
 ['fisk',
  {'c': 89049207.38371027,
   'loc': -1986583.0182634608,
   'scale': 1986583.4736009042}],
 ['fisk',
  {'c': 129895838.77416685,
   'loc': -3488986.76553393,
   'scale': 3488987.276380528}],
 ['mielke',
  {'k': 60661223.74721834,
   'loc': -3921399.169850569,
   's': 728375815.5223036,
   'scale': 3921399.896689184}],
 ['invweibull',
  {'c': 89762340.98886724,
   'loc': -6418271.597604012,
   'scale': 6418272.308923408}],
 ['fisk',
  {'c': 24349073.45796358,
   'loc': -868246.0790497053,
   'scale': 868246.9655063269}],
 ['hypsecant', {'loc': 0.8778802262868111, 'scale': 0.041483206783774
65}],
 ['hypsecant', {'loc': 0.8937249916078858, 'scale': 0.049857587343792
586}],
 ['fisk',
  {'c': 25622.28074224032,
   'loc': -1046.2163111149446,
   'scale': 1047.100475108277}],
 ['hypsecant', {'loc': 0.9085276840337329, 'scale': 0.044917385026022
23}],
 ['fisk',
  {'c': 85052737.61507161,
   'loc': -3435750.954842735,
   'scale': 3435751.8014597762}],
 ['invweibull',
  {'c': 13054.131467397729,
   'loc': -897.4335864004695,
   'scale': 898.1653702212532}],
 ['fisk',
  {'c': 115488161.99855241,
   'loc': -8121329.627112514,
   'scale': 8121330.414561536}],
 ['fisk',
  {'c': 67096.92213942789,
   'loc': -5107.487236737725,
```

```
   'scale': 5108.28892342235}],
  ['mielke',
   {'k': 73326215.23085016,
    'loc': -7400304.56267521,
    's': 1711419439.8636742,
    'scale': 7400305.33546778}],
  ['fisk',
   {'c': 294993556.1385083,
    'loc': -9051528.952159427,
    'scale': 9051529.67594156}],
  ['rdist',
   {'c': 1.0609993438745944,
    'loc': 0.9646594345073922,
    'scale': 0.038917034507392 26}],
  ['fisk',
   {'c': 341675704.2338467,
    'loc': -1508534.8998198714,
    'scale': 1508535.8968744436}],
  ['hypsecant', {'loc': 0.9832313307890309, 'scale': 0.018033385915142
468}],
  ['rice',
   {'b': 1.943763232844457,
    'loc': 0.8094804453402771,
    'scale': 0.04938076437684877}],
```

In [6]:
```
dict_probability_distrs = { "gamma": st.gamma, "gumbel_l": st.gumbel_l,
"cauchy": st.cauchy, "dgamma": st.dgamma,
                            "beta": st.beta, "betaprime": st.betaprime,
"exponweib": st.exponweib, "rayleigh": st.rayleigh,
                            "fisk": st.fisk, "gausshyper": st.gausshype
r, "invweibull": st.invweibull, "pareto": st.pareto,
                            "alpha": st.alpha, "expon": st.expon, "hypse
cant": st.hypsecant, "mielke": st.mielke,
                            "loggamma": st.loggamma, "rdist": st.rdist,
"rice": st.rice }
```

In [7]:
```
example_curves = df_example_curves.values.tolist()
ts_example_curves = to_time_series_dataset(example_curves)
#pprint.pprint(ts_example_curves)
```

In [8]:
```
df_curves = pd.DataFrame(
    columns=[
        '0','1','2','3','4','5','6','7','8','9','10','11','12','13','14
','15','16','17','18','19','20','21','22','23',
        'FC','Sum','DTW_avg_distance','Avg_correlation'
    ]
)
print(df_curves)
```

```
Empty DataFrame
Columns: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1
7, 18, 19, 20, 21, 22, 23, FC, Sum, DTW_avg_distance, Avg_correlatio
n]
Index: []

[0 rows x 28 columns]
```

In [9]:
```
E_min = 0.375
min_f_load = 0.7625
min_count_generated_curves = 25
roof_dtw_distance = 0.25        #0.25
min_corr = 0.95                 #0.95

df_curves = get_optimal_curves(df_curves, example_curves, ts_example_cu
rves, dict_probability_distrs, prob_distrs,
                               min_count_generated_curves, a, b, E_min,
min_f_load, roof_dtw_distance, min_corr)
```

171 0.7647251331973163 18.35340319673559 0.24503839754008055 0.961931
004890237
3323 0.7643839578333489 18.34521498800037 0.24547113782050675 0.95460
63977452295
3563 0.7688067637255614 18.451362329413477 0.24813106456230546 0.9656
95644938727
4503 0.765315690300572 18.367576567213728 0.24941423267433088 0.95759
25135551389
4656 0.765117687890543 18.362824509373034 0.2465874505082063 0.950532
602590968
303 0.7638784961744066 18.33308390818576 0.23952572194898983 0.959238
945217764
1338 0.7636822787411582 18.328374689787797 0.24931849291311164 0.9608
439128899408
2178 0.7644288616935668 18.346292680645604 0.24485351528679425 0.9586
88497511137
930 0.7625874552448444 18.302098925876262 0.2495956201062087 0.952529
7815128697
1528 0.7665429814088554 18.397031553812532 0.24951533093937722 0.9594
38828470824
1975 0.7627562723310513 18.30615053594523 0.246796913882674 0.9597233
601560198
2287 0.7635390850453465 18.324938041088316 0.24536118350853603 0.9683
438606387573
3971 0.7641261700839537 18.339028082014885 0.23196229888748496 0.9662
829522957982
562 0.7679573461710194 18.430976308104466 0.24090494728437123 0.95596
13347979116
4225 0.7645447944666002 18.349075067198406 0.24741318068746676 0.9530
021427690418
2596 0.766774974340985 18.40259938418364 0.23757411804463513 0.959454
3074079265
4091 0.7687534210134895 18.450082104323748 0.24806832111244398 0.9574
828187183869
1743 0.7646185735020942 18.350845764050263 0.2492214675928359 0.95629
40913260015
671 0.7629166230037233 18.30999895208936 0.2456570931394796 0.9611989
573514303
1232 0.7627742357374949 18.306581657699876 0.24759312590266563 0.9612
802669225594
3992 0.7645138801278767 18.34833312306904 0.23946324325089557 0.96677
06943315902
683 0.7724547856448182 18.538914855475635 0.24375775634485272 0.96252
65340767684
2204 0.7630014296179811 18.312034310831546 0.2498586534397405 0.97160
69227668427
3421 0.7626335124546738 18.303204298912167 0.24985713485508393 0.9640
309536101712
4669 0.7657127801281483 18.377106723075556 0.2358868906109151 0.97005
43312036126

In [10]:
```
print(df_curves)
```

	0	1	2	3	4	5	6
0	0.610331	0.434236	0.449072	0.401362	0.469460	0.527173	0.721520
1	0.584646	0.446702	0.425810	0.459555	0.451049	0.591894	0.708186
2	0.601119	0.486388	0.416625	0.486396	0.474938	0.542921	0.613405
3	0.631015	0.481942	0.445273	0.430063	0.459355	0.515438	0.690438
4	0.597480	0.468423	0.495378	0.478195	0.481154	0.464275	0.686589
5	0.569603	0.464617	0.444821	0.425164	0.471543	0.528657	0.700729
6	0.582364	0.488758	0.444639	0.411429	0.432149	0.587136	0.750045
7	0.617605	0.498322	0.432600	0.459844	0.511347	0.508999	0.726855
8	0.565926	0.485179	0.439380	0.405086	0.515203	0.525995	0.716504
9	0.551098	0.507204	0.489339	0.437384	0.461882	0.560855	0.686293
10	0.575701	0.505688	0.479513	0.435272	0.493991	0.500583	0.668671
11	0.601050	0.461377	0.428182	0.426886	0.446890	0.503203	0.701188
12	0.567889	0.475363	0.469553	0.459329	0.464381	0.507080	0.706435
13	0.606832	0.558500	0.446838	0.425723	0.452666	0.489967	0.677048
14	0.608089	0.456735	0.426914	0.469663	0.463503	0.484841	0.681581
15	0.593597	0.512707	0.452895	0.431816	0.472401	0.549037	0.692828
16	0.579149	0.479544	0.462354	0.448638	0.529496	0.525509	0.710774
17	0.621202	0.517476	0.395390	0.483414	0.523765	0.511839	0.704509
18	0.576396	0.454978	0.464117	0.483288	0.501596	0.562533	0.667869
19	0.550749	0.442094	0.408348	0.464953	0.438253	0.587730	0.719126
20	0.583954	0.487172	0.466046	0.449141	0.512082	0.586618	0.700041
21	0.582889	0.519084	0.445525	0.446502	0.472892	0.480493	0.693889
22	0.595415	0.468229	0.486110	0.407924	0.476947	0.528138	0.736204
23	0.573570	0.499195	0.437844	0.448753	0.476909	0.569435	0.716191
24	0.590578	0.495606	0.457482	0.511778	0.466572	0.523478	0.719049

	7	8	9	...	18	19	20	21
0	0.805128	0.922059	0.854495	...	0.730580	0.929617	0.995966	

```
1.000000
1   0.868968  0.872311  0.898259  ...  0.680803  0.993300  1.000000
0.973175
2   0.787211  0.957635  0.878689  ...  0.735049  1.000000  0.993491
0.979745
3   0.696887  0.843409  0.861639  ...  0.771561  0.940681  1.000000
0.973746
4   0.812116  0.775915  0.888854  ...  0.679053  0.961869  1.000000
0.952062
5   0.712695  0.808513  0.869409  ...  0.755403  0.962396  1.000000
0.990879
6   0.810974  0.934806  0.891414  ...  0.719578  0.974938  1.000000
0.993948
7   0.852907  0.904960  0.872153  ...  0.786658  1.000000  0.998248
0.994051
8   0.807198  0.865226  0.855095  ...  0.642927  0.966234  0.998750
1.000000
9   0.818304  0.890324  0.872878  ...  0.719487  1.000000  0.991797
0.994524
10  0.865534  0.897513  0.931538  ...  0.753769  0.981755  1.000000
0.955048
11  0.828686  0.926019  0.892286  ...  0.713970  1.000000  0.995353
0.971315
12  0.810038  0.852903  0.908589  ...  0.703210  1.000000  0.995361
0.976136
13  0.784099  0.937545  0.824590  ...  0.690498  0.997580  0.994436
1.000000
14  0.757003  0.903793  0.885075  ...  0.711228  0.932500  0.993643
1.000000
15  0.798460  0.865597  0.932701  ...  0.786419  1.000000  0.985692
0.992387
16  0.800766  0.870810  0.885892  ...  0.749812  0.964640  0.990651
1.000000
17  0.847926  0.925868  0.932277  ...  0.708537  0.976736  1.000000
0.972218
18  0.726720  0.951390  0.863404  ...  0.722370  0.963382  0.995184
1.000000
19  0.781916  0.848816  0.857951  ...  0.729142  0.989391  1.000000
0.956537
20  0.795496  0.909213  0.879520  ...  0.698039  0.954411  0.998974
1.000000
21  0.806863  0.911280  0.902454  ...  0.782325  0.976283  1.000000
0.948558
22  0.752517  0.907268  0.865346  ...  0.719541  0.991781  1.000000
0.949757
23  0.682673  0.890427  0.878648  ...  0.769428  1.000000  0.991502
0.986843
24  0.783704  0.925827  0.887782  ...  0.677566  0.958990  0.990497
1.000000

          22        23        FC        Sum  DTW_avg_distance  Avg_co
rrelation
0   0.917261  0.760741  0.764725  18.353403          0.245038
0.961931
1   0.944064  0.787006  0.764384  18.345215          0.245471
0.954606
```

```
2   0.957858  0.770301  0.768807  18.451362          0.248131
0.965696
3   0.894379  0.815266  0.765316  18.367577          0.249414
0.957593
4   0.944624  0.781170  0.765118  18.362825          0.246587
0.950533
5   0.955520  0.744181  0.763878  18.333084          0.239526
0.959239
6   0.909428  0.758223  0.763682  18.328375          0.249318
0.960844
7   0.955644  0.772123  0.764429  18.346293          0.244854
0.958688
8   0.932136  0.795833  0.762587  18.302099          0.249596
0.952530
9   0.972552  0.746250  0.766543  18.397032          0.249515
0.959439
10  0.993480  0.732835  0.762756  18.306151          0.246797
0.959723
11  0.904928  0.775960  0.763539  18.324938          0.245361
0.968344
12  0.941367  0.775846  0.764126  18.339028          0.231962
0.966283
13  0.925390  0.759007  0.767957  18.430976          0.240905
0.955961
14  0.896448  0.757157  0.764545  18.349075          0.247413
0.953002
15  0.936814  0.777048  0.766775  18.402599          0.237574
0.959454
16  0.946515  0.743536  0.768753  18.450082          0.248068
0.957483
17  0.946179  0.744502  0.764619  18.350846          0.249221
0.956294
18  0.932184  0.769242  0.762917  18.309999          0.245657
0.961199
19  0.937225  0.769230  0.762774  18.306582          0.247593
0.961280
20  0.987470  0.758447  0.764514  18.348333          0.239463
0.966771
21  0.921090  0.773489  0.772455  18.538915          0.243758
0.962527
22  0.923676  0.782096  0.763001  18.312034          0.249859
0.971607
23  0.903781  0.750081  0.762634  18.303204          0.249857
0.964031
24  0.941060  0.770900  0.765713  18.377107          0.235887
0.970054

[25 rows x 28 columns]
```

In [11]:
```
for index, row in df_curves.loc[:, "0":"23"].iterrows():
    fig = px.line(row, width=600, height=300)
    fig.show()
```


variable
0
value
index
1
0.8
0.6
0.4
0
5
10
15
20


variable
1
value
index
1
0.8
0.6
0.4
0
5
10
15
20


variable
2
value
index
1
0.8
0.6
0.4
0
5
10
15
20


1
0.8
0.6
value
0
5
10
15
20
index
variable
4


1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
5

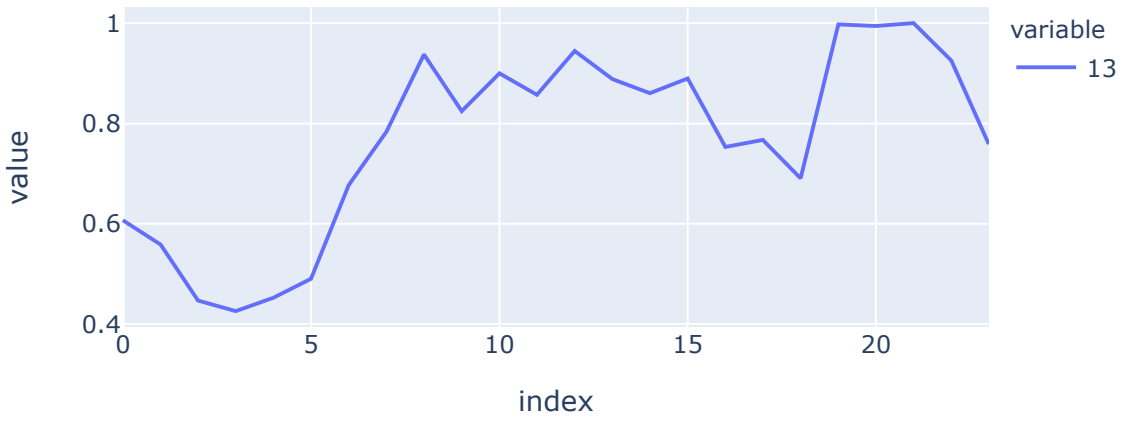

1
0.8
0.6
value
0
5
10
15
20
index
variable
7

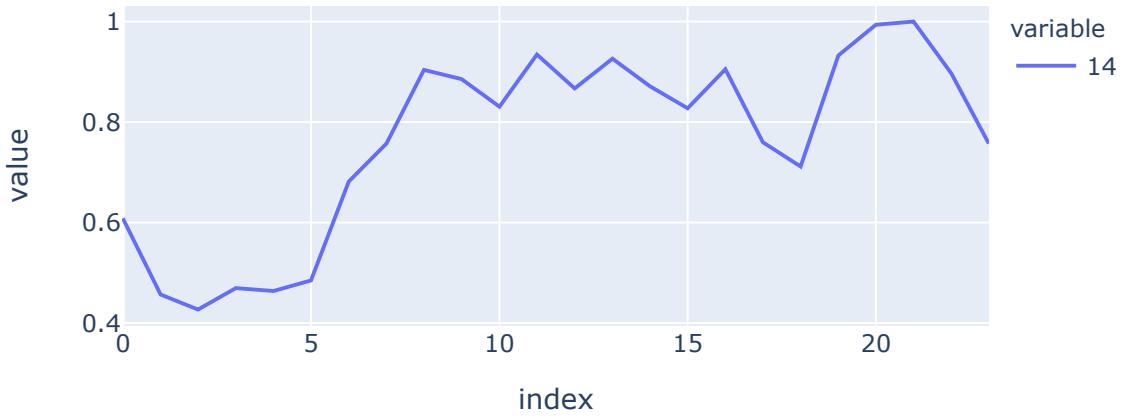

1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
8


1
0.8
0.6
value
0
5
10
15
20
index
variable
10


1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
11


1
variable

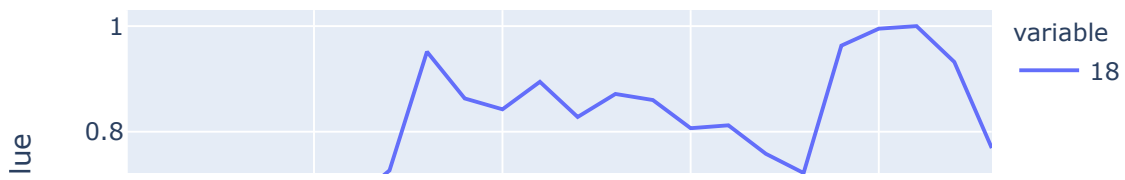

1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
13


1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
14


1
variable
15

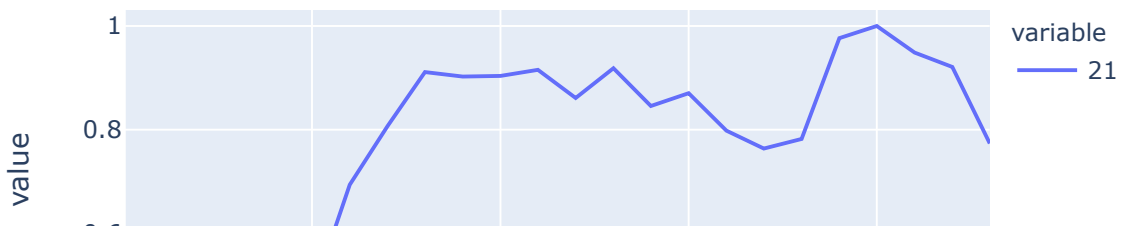

1
0.8
0.6
value
0
5
10
15
20
index
variable
16


1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
17


1
0.8
lue
variable
18

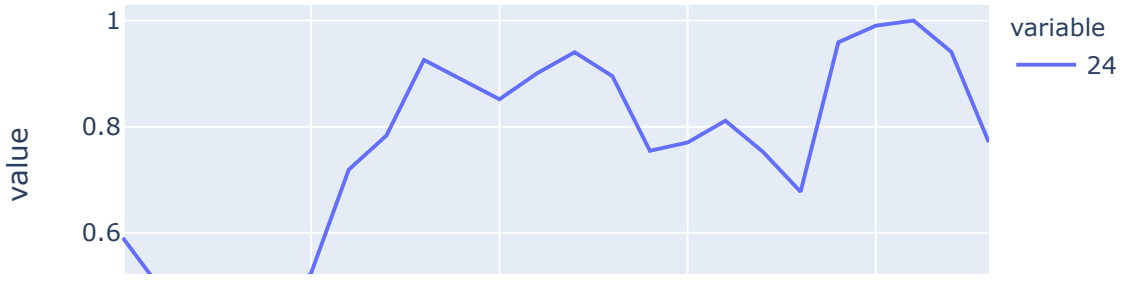

1
0.8
0.6
0.4
value
0
5
10
15
20
index
variable
19


1
0.8
0.6
value
0
5
10
15
20
index
variable
20


1
0.8
value
variable
21


value
1
0.8
0.6
0.4
0
5
10
15
20
index
variable
22

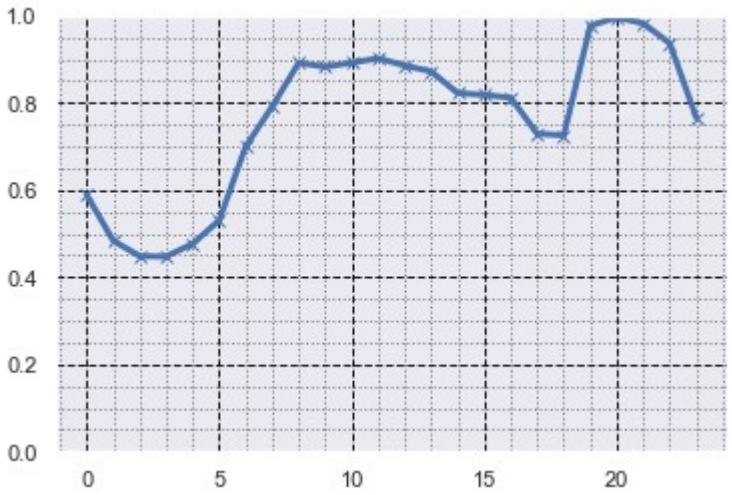

value
1
0.8
0.6
0
5
10
15
20
index
variable
23


value
1
0.8
0.6
variable
24

In [12]:
```
average_optimal_curve = df_curves.loc[:, "0":"23"].mean(axis=0)

print(average_optimal_curve, type(average_optimal_curve))
print("Load Factor: ", )
```

```
0     0.588730
1     0.483821
2     0.448402
3     0.447502
4     0.476817
5     0.530553
6     0.699839
7     0.791792
8     0.891977
9     0.882838
10    0.892188
11    0.902291
12    0.885800
13    0.872095
14    0.822360
15    0.818517
16    0.811809
17    0.729538
18    0.725478
19    0.976659
20    0.996382
21    0.982437
22    0.936843
23    0.766819
dtype: float64 <class 'pandas.core.series.Series'>
Load Factor:
```

In [47]:
```
average_optimal_curve.plot(linewidth=3.0, marker='x', ms=6.5)
plt.axis((None,None,0,1))
plt.grid(b=True, which='major', color='k', linestyle='--')
plt.minorticks_on()
plt.grid(b=True, which='minor', color='grey', linestyle=':')
plt.show()

final_load_factor = average_optimal_curve.mean() / average_optimal_curv
e.max()
print("final_load_factor: ", final_load_factor)

final_energy_sum = average_optimal_curve.sum()
print("final_energy_sum: ", final_energy_sum)
```




```
final_load_factor:  0.7678401436126037
final_energy_sum:  18.36148530228425
```

In [ ]: