In [1]:
```
# Created by: Sergiu Iatco / 2021.10.01
# https://github.com/itsergiu/Predict-S-P-500-correction-with-Shiller-PE-Ratio
# How to predict S&P 500 correction

# Nadeem Walayat - The Market Oracle
# http://www.marketoracle.co.uk/Article69423.html

# Vitaliy Katsenelson
# https://contrarianedge.com/sideways-market/

# https://www.marketwatch.com/story/market-analysts-cant-agree-on-where-stocks-are-going-next-so-double-check-the-data-before-you-buy-

# Purpose: to build a machine learning model to predict S&P 500 correction within next 6 months
```

In [2]:
```
# !pip install pandas
# !pip install dateparser
# !pip install xgboost
# !pip install sklearn
# !pip install numpy
# !pip install plotly_express
```

In [3]:
```
import pandas as pd
# import dateparser
import xgboost as xgb
from sklearn import model_selection
from sklearn.metrics import r2_score
import numpy as np
from fredapi import Fred
import plotly.express as px
from IPython.display import display
```

In [4]:
```
url_per='https://www.multpl.com/shiller-pe/table/by-month'
```

In [5]:
```
ls_tables = pd.read_html(url_per)
```

In [6]:
```
type(ls_tables)
```

Out[6]:
```
list
```

In [7]:
```
len(ls_tables)
```

Out[7]:
```
1
```

In [8]:
```
ls_tables[0]
```

Out[8]:

	Date	Value Value
0	Jan 13, 2023	29.19
1	Jan 1, 2023	27.96
2	Dec 1, 2022	28.65
3	Nov 1, 2022	28.74
4	Oct 1, 2022	27.35
...	...	...
1820	Jun 1, 1871	12.59
1821	May 1, 1871	12.59
1822	Apr 1, 1871	12.05
1823	Mar 1, 1871	11.19
1824	Feb 1, 1871	10.92

1825 rows × 2 columns

In [9]:
```
df_per=ls_tables[0]
```

In [10]:
```
df_per.dtypes
```

Out[10]:
```
Date            object
Value Value    float64
dtype: object
```

In [11]:
```
cols = df_per.columns.tolist()
cols
```

Out[11]:
```
['Date', 'Value Value']
```

In [12]:
```
df_per.rename(
    columns={{ cols[1]: 'PER'}),
    inplace=True,)
```

In [13]:
```
df_per.head()
```

Out[13]:

	Date	PER
0	Jan 13, 2023	29.19
1	Jan 1, 2023	27.96
2	Dec 1, 2022	28.65
3	Nov 1, 2022	28.74
4	Oct 1, 2022	27.35

In [14]:
```
url_sp='https://www.multpl.com/s-p-500-historical-prices/table/by-month'
```

In [15]:
```
ls_tables = pd.read_html(url_sp)
```

In [16]:
```
len(ls_tables)
```

Out[16]:
```
1
```

In [17]:
```
df_sp=ls_tables[0]
df_sp.head()
```

Out[17]:

	Date	Price Value
0	Jan 13, 2023	3991.94
1	Jan 1, 2023	3824.14
2	Dec 1, 2022	3912.38
3	Nov 1, 2022	3917.49
4	Oct 1, 2022	3726.05

In [18]:
```
cols = df_sp.columns.tolist()
cols
```

Out[18]:
```
['Date', 'Price Value']
```

In [19]:
```
df_sp.rename(
    columns={{ cols[1]: 'Price'}),
    inplace=True,)
```

In [20]:
```
df_sp.head()
```

Out[20]:

	Date	Price
0	Jan 13, 2023	3991.94
1	Jan 1, 2023	3824.14
2	Dec 1, 2022	3912.38
3	Nov 1, 2022	3917.49
4	Oct 1, 2022	3726.05

In [21]:
```
df_per.index, df_sp.index
```

Out[21]:
```
(RangeIndex(start=0, stop=1825, step=1),
 RangeIndex(start=0, stop=1826, step=1))
```

In [22]:
```
df = df_per.merge(df_sp, left_on='Date', right_on='Date')
df
```

Out[22]:

	Date	PER	Price
0	Jan 13, 2023	29.19	3991.94
1	Jan 1, 2023	27.96	3824.14
2	Dec 1, 2022	28.65	3912.38
3	Nov 1, 2022	28.74	3917.49

	Date	PER	Price
4	Oct 1, 2022	27.35	3726.05
...	...	...	...
1820	Jun 1, 1871	12.59	4.82
1821	May 1, 1871	12.59	4.86
1822	Apr 1, 1871	12.05	4.74
1823	Mar 1, 1871	11.19	4.61
1824	Feb 1, 1871	10.92	4.50

1825 rows × 3 columns

In [23]:
```
df.dtypes
```

Out[23]:
```
Date      object
PER      float64
Price    float64
dtype: object
```

In [24]:
```
df['Date']=pd.to_datetime(df['Date'])
df
```

Out[24]:

	Date	PER	Price
0	2023-01-13	29.19	3991.94
1	2023-01-01	27.96	3824.14
2	2022-12-01	28.65	3912.38
3	2022-11-01	28.74	3917.49
4	2022-10-01	27.35	3726.05
...	...	...	...
1820	1871-06-01	12.59	4.82
1821	1871-05-01	12.59	4.86
1822	1871-04-01	12.05	4.74
1823	1871-03-01	11.19	4.61
1824	1871-02-01	10.92	4.50

1825 rows × 3 columns

In [25]:
```
df.head()
```

Out[25]:

	Date	PER	Price
0	2023-01-13	29.19	3991.94
1	2023-01-01	27.96	3824.14
2	2022-12-01	28.65	3912.38
3	2022-11-01	28.74	3917.49
4	2022-10-01	27.35	3726.05

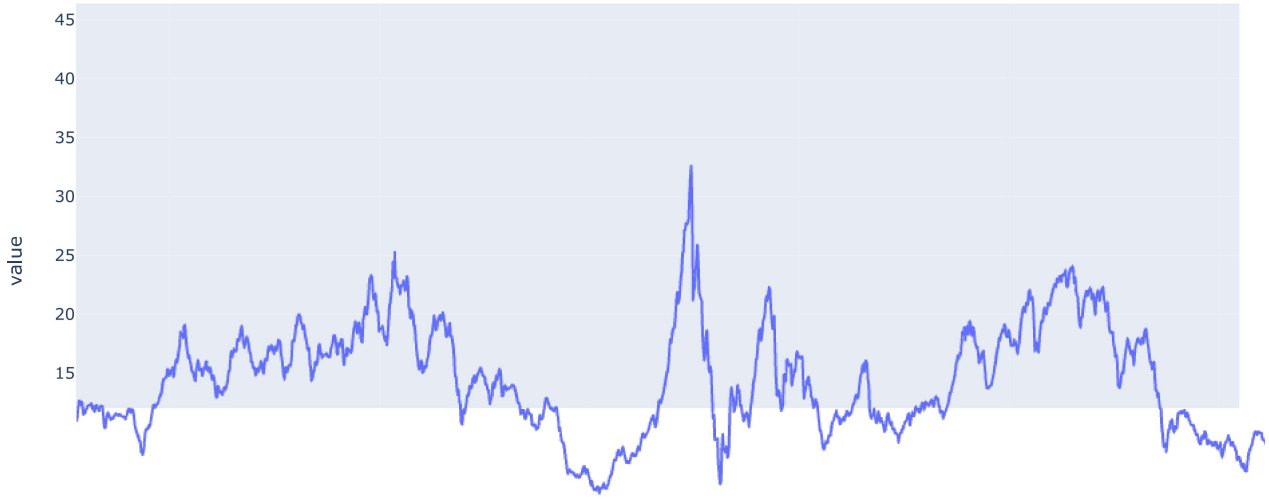
In [26]:
```
df.set_index(['Date'], inplace=True)
df['Date']=df.index
df
```

Out[26]:

	PER	Price	Date
Date			
2023-01-13	29.19	3991.94	2023-01-13
2023-01-01	27.96	3824.14	2023-01-01
2022-12-01	28.65	3912.38	2022-12-01
2022-11-01	28.74	3917.49	2022-11-01
2022-10-01	27.35	3726.05	2022-10-01
...	...	...	...
1871-06-01	12.59	4.82	1871-06-01
1871-05-01	12.59	4.86	1871-05-01
1871-04-01	12.05	4.74	1871-04-01
1871-03-01	11.19	4.61	1871-03-01
1871-02-01	10.92	4.50	1871-02-01

1825 rows × 3 columns

In [27]:
```
# https://pandas.pydata.org/docs/reference/api/pandas.melt.html
df_per = pd.melt(df, id_vars=['Date'], value_vars=['PER'])
px.line(df_per, x='Date', y='value', color='variable')
```



In [28]:
```
df.describe()
```

Out[28]:

	PER	Price
count	1825.000000	1825.000000
mean	17.005299	362.025249
std	7.065774	771.981468
min	4.780000	2.730000
25%	11.700000	7.930000
50%	15.910000	17.920000
75%	20.570000	179.400000
max	44.190000	4674.770000

In [29]:
```
df.dtypes
```

Out[29]:
```
PER             float64
Price           float64
Date     datetime64[ns]
dtype: object
```

In [30]:
```
ls_price_p = ['Price'+str(i)+'F' for i in range(7)]
ls_price_p
```

Out[30]:
```
['Price0F', 'Price1F', 'Price2F', 'Price3F', 'Price4F', 'Price5F', 'Price6F']
```

In [31]:
```
ls_price_p[1:]
```

Out[31]:
```
['Price1F', 'Price2F', 'Price3F', 'Price4F', 'Price5F', 'Price6F']
```

In [32]:
```
ls_price_p[0]
```

Out[32]:
```
'Price0F'
```

In [33]:
```
df.columns.tolist()
```

Out[33]:
```
['PER', 'Price', 'Date']
```

In [34]:
```
df.head(10)
```

Out[34]:

Date	PER	Price	Date
2023-01-13	29.19	3991.94	2023-01-13
2023-01-01	27.96	3824.14	2023-01-01
2022-12-01	28.65	3912.38	2022-12-01
2022-11-01	28.74	3917.49	2022-11-01
2022-10-01	27.35	3726.05	2022-10-01
2022-09-01	28.42	3850.52	2022-09-01
2022-08-01	30.80	4158.56	2022-08-01
2022-07-01	29.00	3911.73	2022-07-01
2022-06-01	29.05	3898.95	2022-06-01
2022-05-01	30.67	4040.36	2022-05-01

In [35]:
```
df.tail(10)
```

Out[35]:

Date	PER	Price	Date
1871-11-01	11.60	4.64	1871-11-01
1871-10-01	11.47	4.59	1871-10-01
1871-09-01	12.31	4.84	1871-09-01
1871-08-01	12.55	4.79	1871-08-01
1871-07-01	12.27	4.73	1871-07-01
1871-06-01	12.59	4.82	1871-06-01
1871-05-01	12.59	4.86	1871-05-01
1871-04-01	12.05	4.74	1871-04-01
1871-03-01	11.19	4.61	1871-03-01
1871-02-01	10.92	4.50	1871-02-01

In [36]:
```
for i,e in enumerate(ls_price_p):
    df[e]=df.Price.shift(i)
```

In [37]:
```
df['Price1P'] = df['Price'].shift(-1)
df['Price_Var']=df['Price']/df['Price1P']-1
```

In [38]:
```
df.head(10)
```

Out[38]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917

In [39]:
```
df['PriceFmin']=df[ls_price_p].min(axis=1)
```

In [40]:
```
df.head(10)
```

Out[40]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05

In [41]:
```
col_y = 'Price_Corr_6M'
df[col_y]=1-df['PriceFmin']/df['Price0F']
```

In [42]:
```
df.head(10)
```

Out[42]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	0.000000
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	0.000000
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	0.022554
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	0.023829
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	0.000000
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05	0.032326
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.047467
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.044345
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793

In [43]:
```
df[df.isna().any(axis=1)]
```

Out[43]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	0.000000
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	0.000000
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	0.022554
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	0.023829
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	0.000000
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05	0.032326
1871-02-01	10.92	4.50	1871-02-01	4.50	4.61	4.74	4.86	4.82	4.73	4.79	NaN	NaN	4.50	0.000000

In [44]:
```
df.loc[df.isna().any(axis=1),col_y]=np.nan # set to zero corrections with Nan rows
df[df.isna().any(axis=1)]
```

Out[44]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	NaN
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	NaN
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	NaN
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	NaN
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	NaN
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05	NaN
1871-02-01	10.92	4.50	1871-02-01	4.50	4.61	4.74	4.86	4.82	4.73	4.79	NaN	NaN	4.50	NaN

In [45]:
```
df_initial=df.copy()
```

In [46]:
```
df_initial.head()
```

Out[46]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	NaN
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	NaN
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	NaN
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	NaN
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	NaN

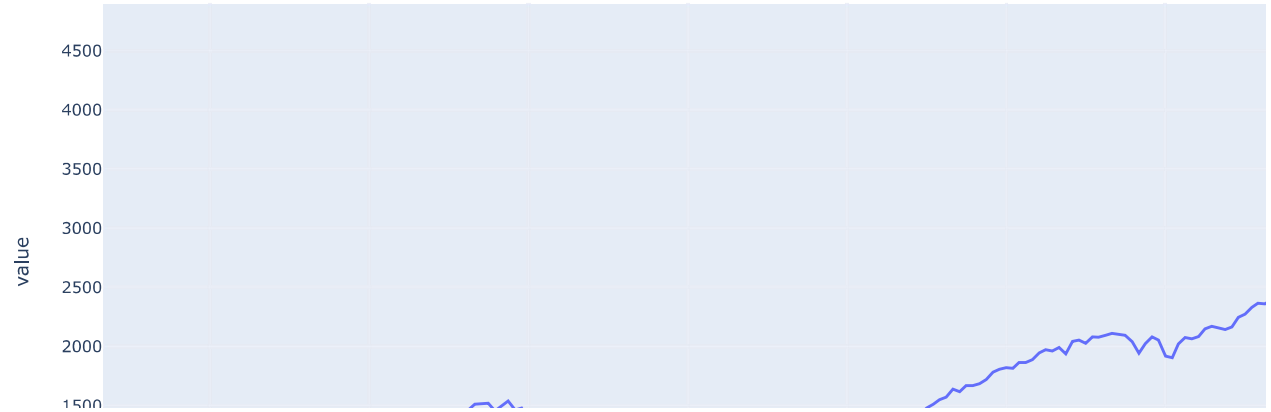
In [47]:
```
df.dropna(axis=0, how='any', inplace=True) # remove NaN rowss
```

In [48]:
```
df[df[col_y]>0].describe()
```

Out[48]:

	PER	Price	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceF
count	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000000	1178.000
mean	16.915085	314.127224	314.127224	310.867496	308.884711	308.165671	307.602453	307.759762	308.359593	313.395806	-0.000352	291.303
std	7.106650	715.704670	715.704670	708.638815	703.509252	701.083985	698.009447	696.476130	696.594704	713.260190	0.041136	660.757
min	4.780000	2.940000	2.940000	2.730000	2.730000	2.730000	2.730000	2.730000	2.730000	2.940000	-0.239709	2.730
25%	11.722500	7.135000	7.135000	7.042500	6.855000	6.812500	6.812500	6.820000	6.850000	7.435000	-0.020523	6.260
50%	15.895000	15.045000	15.045000	14.845000	14.755000	14.720000	14.690000	14.645000	14.355000	15.145000	0.001464	13.970
75%	20.312500	110.875000	110.875000	109.775000	109.700000	109.775000	109.775000	110.175000	109.775000	110.875000	0.023160	106.600
max	44.190000	4674.770000	4674.770000	4674.770000	4674.770000	4674.770000	4674.770000	4573.820000	4435.980000	4674.770000	0.502994	4435.980

In [49]:
```
last_months=240
df_per = pd.melt(df.head(last_months), id_vars='Date', value_vars=['Price'])
px.line(df_per, x='Date', y='value', color='variable')
```



In [50]:
```
# min_correction=0.15
min_correction=0.15 # take a look at big corrections
df[df[col_y]>min_correction].head(10)
```

Out[50]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2022-04-01	33.89	4391.30	2022-04-01	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	4391.27	0.000007	3726.05	0.151493
2021-12-01	38.31	4674.77	2021-12-01	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	4667.39	0.001581	3898.95	0.165959
2020-02-01	30.73	3277.31	2020-02-01	3277.31	2652.39	2761.98	2919.61	3104.66	3207.62	3391.71	3278.20	-0.000271	2652.39	0.190681
2020-01-01	30.99	3278.20	2020-01-01	3278.20	3277.31	2652.39	2761.98	2919.61	3104.66	3207.62	3176.75	0.031935	2652.39	0.190900
2019-12-01	30.33	3176.75	2019-12-01	3176.75	3278.20	3277.31	2652.39	2761.98	2919.61	3104.66	3104.90	0.023141	2652.39	0.165062
2008-10-01	16.39	968.80	2008-10-01	968.80	883.04	877.56	865.58	805.23	757.13	848.15	1216.95	-0.203911	757.13	0.218487

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2008-09-01	20.36	1216.95	2008-09-01	1216.95	968.80	883.04	877.56	865.58	805.23	757.13	1281.47	-0.050348	757.13	0.377846
2008-08-01	21.40	1281.47	2008-08-01	1281.47	1216.95	968.80	883.04	877.56	865.58	805.23	1257.33	0.019199	805.23	0.371636
2008-07-01	20.91	1257.33	2008-07-01	1257.33	1281.47	1216.95	968.80	883.04	877.56	865.58	1341.25	-0.062568	865.58	0.311573
2008-06-01	22.42	1341.25	2008-06-01	1341.25	1257.33	1281.47	1216.95	968.80	883.04	877.56	1403.22	-0.044163	877.56	0.345715

In [51]:
```
df[df[col_y]>min_correction][col_y].describe()
```

Out[51]:
```
count    149.000000
mean       0.228507
std        0.072260
min        0.150127
25%        0.175627
50%        0.205418
75%        0.254902
max        0.469682
Name: Price_Corr_6M, dtype: float64
```

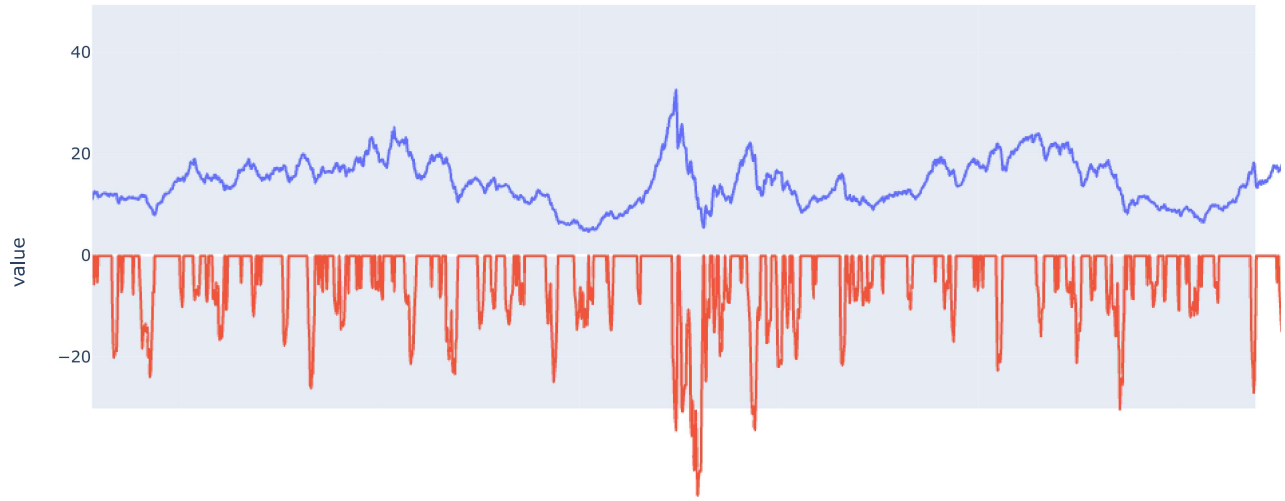
In [52]:
```
# plot PER & Price_correction
```

In [53]:
```
min_correction=0.05 # force learning above ?
# min_correction=0
if min_correction!=0:
    df[col_y] = df[col_y].apply(lambda x: 0 if x<min_correction else x)

# df[col_y] = -df[col_y] # reverse sign

df_per_pc=df[['Date','PER',col_y]].copy()
df_per_pc[[col_y]]=100*df_per_pc[[col_y]]
df_per_pc[col_y]=-df_per_pc[col_y] # reverse sign for plotting
```

In [54]:
```
last_months=2400
df_plot = pd.melt(df_per_pc, id_vars='Date', value_vars=['PER',col_y])
px.line(df_plot, x='Date', y='value', color='variable')
```



In [55]:
```
pd.set_option('display.max_columns', 500)
df.head(10)
```

Out[55]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.000000
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.000000
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793
2022-04-01	33.89	4391.30	2022-04-01	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	4391.27	0.000007	3726.05	0.151493

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2022-03-01	34.27	4391.27	2022-03-01	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	4435.98	-0.010079	3850.52	0.123142
2022-02-01	35.29	4435.98	2022-02-01	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	4573.82	-0.030137	3898.95	0.121062
2022-01-01	36.94	4573.82	2022-01-01	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4674.77	-0.021595	3898.95	0.147551
2021-12-01	38.31	4674.77	2021-12-01	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	4667.39	0.001581	3898.95	0.165959
2021-11-01	38.58	4667.39	2021-11-01	4667.39	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	4460.71	0.046333	4040.36	0.134343

In [56]:
```
df.head(36)
```

Out[56]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.000000
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.000000
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793
2022-04-01	33.89	4391.30	2022-04-01	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	4391.27	0.000007	3726.05	0.151493
2022-03-01	34.27	4391.27	2022-03-01	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	4435.98	-0.010079	3850.52	0.123142
2022-02-01	35.29	4435.98	2022-02-01	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	4573.82	-0.030137	3898.95	0.121062
2022-01-01	36.94	4573.82	2022-01-01	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4674.77	-0.021595	3898.95	0.147551
2021-12-01	38.31	4674.77	2021-12-01	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	4667.39	0.001581	3898.95	0.165959
2021-11-01	38.58	4667.39	2021-11-01	4667.39	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	4460.71	0.046333	4040.36	0.134343
2021-10-01	37.25	4460.71	2021-10-01	4460.71	4667.39	4674.77	4573.82	4435.98	4391.27	4391.30	4445.54	0.003412	4391.27	0.000000
2021-09-01	37.62	4445.54	2021-09-01	4445.54	4460.71	4667.39	4674.77	4573.82	4435.98	4391.27	4454.21	-0.001946	4391.27	0.000000
2021-08-01	37.97	4454.21	2021-08-01	4454.21	4445.54	4460.71	4667.39	4674.77	4573.82	4435.98	4363.71	0.020739	4435.98	0.000000
2021-07-01	37.44	4363.71	2021-07-01	4363.71	4454.21	4445.54	4460.71	4667.39	4674.77	4573.82	4238.49	0.029544	4363.71	0.000000
2021-06-01	36.70	4238.49	2021-06-01	4238.49	4363.71	4454.21	4445.54	4460.71	4667.39	4674.77	4167.85	0.016949	4238.49	0.000000
2021-05-01	36.55	4167.85	2021-05-01	4167.85	4238.49	4363.71	4454.21	4445.54	4460.71	4667.39	4141.18	0.006440	4167.85	0.000000
2021-04-01	36.72	4141.18	2021-04-01	4141.18	4167.85	4238.49	4363.71	4454.21	4445.54	4460.71	3910.51	0.058987	4141.18	0.000000
2021-03-01	35.04	3910.51	2021-03-01	3910.51	4141.18	4167.85	4238.49	4363.71	4454.21	4445.54	3883.43	0.006973	3910.51	0.000000
2021-02-01	35.10	3883.43	2021-02-01	3883.43	3910.51	4141.18	4167.85	4238.49	4363.71	4454.21	3793.75	0.023639	3883.43	0.000000
2021-01-01	34.51	3793.75	2021-01-01	3793.75	3883.43	3910.51	4141.18	4167.85	4238.49	4363.71	3695.31	0.026639	3793.75	0.000000
2020-12-01	33.77	3695.31	2020-12-01	3695.31	3793.75	3883.43	3910.51	4141.18	4167.85	4238.49	3548.99	0.041229	3695.31	0.000000
2020-11-01	32.47	3548.99	2020-11-01	3548.99	3695.31	3793.75	3883.43	3910.51	4141.18	4167.85	3418.70	0.038111	3548.99	0.000000
2020-10-01	31.28	3418.70	2020-10-01	3418.70	3548.99	3695.31	3793.75	3883.43	3910.51	4141.18	3365.52	0.015801	3418.70	0.000000
2020-09-01	30.84	3365.52	2020-09-01	3365.52	3418.70	3548.99	3695.31	3793.75	3883.43	3910.51	3391.71	-0.007722	3365.52	0.000000
2020-08-01	31.16	3391.71	2020-08-01	3391.71	3365.52	3418.70	3548.99	3695.31	3793.75	3883.43	3207.62	0.057391	3365.52	0.000000
2020-07-01	29.60	3207.62	2020-07-01	3207.62	3391.71	3365.52	3418.70	3548.99	3695.31	3793.75	3104.66	0.033163	3207.62	0.000000
2020-06-01	28.84	3104.66	2020-06-01	3104.66	3207.62	3391.71	3365.52	3418.70	3548.99	3695.31	2919.61	0.063382	3104.66	0.000000
2020-05-01	27.33	2919.61	2020-05-01	2919.61	3104.66	3207.62	3391.71	3365.52	3418.70	3548.99	2761.98	0.057071	2919.61	0.000000
2020-04-01	25.93	2761.98	2020-04-01	2761.98	2919.61	3104.66	3207.62	3391.71	3365.52	3418.70	2652.39	0.041317	2761.98	0.000000
2020-03-01	24.82	2652.39	2020-03-01	2652.39	2761.98	2919.61	3104.66	3207.62	3391.71	3365.52	3277.31	-0.190681	2652.39	0.000000
2020-02-01	30.73	3277.31	2020-02-01	3277.31	2652.39	2761.98	2919.61	3104.66	3207.62	3391.71	3278.20	-0.000271	2652.39	0.190681
2020-01-01	30.99	3278.20	2020-01-01	3278.20	3277.31	2652.39	2761.98	2919.61	3104.66	3207.62	3176.75	0.031935	2652.39	0.190900
2019-12-01	30.33	3176.75	2019-12-01	3176.75	3278.20	3277.31	2652.39	2761.98	2919.61	3104.66	3104.90	0.023141	2652.39	0.165062
2019-11-01	29.84	3104.90	2019-11-01	3104.90	3176.75	3278.20	3277.31	2652.39	2761.98	2919.61	2977.68	0.042725	2652.39	0.145741
2019-10-01	28.84	2977.68	2019-10-01	2977.68	3104.90	3176.75	3278.20	3277.31	2652.39	2761.98	2982.16	-0.001502	2652.39	0.109243
2019-09-01	29.23	2982.16	2019-09-01	2982.16	2977.68	3104.90	3176.75	3278.20	3277.31	2652.39	2897.50	0.029218	2652.39	0.110581

In [57]:
```
def gen_features(df,cols, ref_period, band):
    features = []

    features.append(df['Date'].iloc[ref_period])
    for col in cols:
        segment = df[col].iloc[ref_period:ref_period+band]
        features.append(segment.min())
        features.append(segment.max())

        for i in range(2,99,10):
            features.append(np.quantile(segment, i/100))
```

```
        features.append(segment.mean())
        features.append(segment.std())
        features.append(segment.mad())
        features.append(segment.skew())
        features.append(segment.kurtosis())

        features.append(np.sqrt(np.mean(segment**2)))

        features.append(np.abs(segment).mean())
        features.append(np.abs(segment).std())

        features.append(np.abs(np.diff(segment)).mean())
        features.append(np.abs(np.diff(segment)).std())

    return pd.Series(features)
```

In [58]:
```
def generate_dataset(df_f, cols, band=60, start_date='2000-01-01', periods = 0):
#     global cols
    df = df_f.copy()
    X_train = pd.DataFrame()

    y_train = pd.Series(dtype='float64', name=col_y)
#     cols =['PER', 'Price', 'Price_Var']

    if periods>0:
        ref_periods = periods
    else:
        ref_periods = df[df.index>=start_date].shape[0]

    for ref_period in range(ref_periods):
        X = gen_features(df,cols, ref_period, band)
        X_train = X_train.append(X, ignore_index=True)
        y_train = y_train.append(pd.Series(df[col_y][ref_period], index=[df.index[ref_period]], name=col_y))

    X_train = X_train.rename({0:'Date'}, axis='columns')
    X_train = X_train.set_index('Date')
    return X_train, y_train
```

In [59]:
```
start_date = '1950-01-01' # set your own date
band=60 # how many months from past to use to generate statistics
```

In [60]:
```
cols_features = ['PER', 'Price', 'Price_Var']
X_train, y_train = generate_dataset(df, cols_features, band, start_date)
```

In [61]:
```
y_train.shape, X_train.shape
```

Out[61]:
```
((872,), (872, 66))
```

In [62]:
```
y_train.index
```

Out[62]:
```
DatetimeIndex(['2022-08-01', '2022-07-01', '2022-06-01', '2022-05-01',
               '2022-04-01', '2022-03-01', '2022-02-01', '2022-01-01',
               '2021-12-01', '2021-11-01',
               ...
               '1950-10-01', '1950-09-01', '1950-08-01', '1950-07-01',
               '1950-06-01', '1950-05-01', '1950-04-01', '1950-03-01',
               '1950-02-01', '1950-01-01'],
              dtype='datetime64[ns]', length=872, freq=None)
```

In [63]:
```
X_train.index
```

Out[63]:
```
DatetimeIndex(['2022-08-01', '2022-07-01', '2022-06-01', '2022-05-01',
               '2022-04-01', '2022-03-01', '2022-02-01', '2022-01-01',
               '2021-12-01', '2021-11-01',
               ...
               '1950-10-01', '1950-09-01', '1950-08-01', '1950-07-01',
               '1950-06-01', '1950-05-01', '1950-04-01', '1950-03-01',
               '1950-02-01', '1950-01-01'],
              dtype='datetime64[ns]', name='Date', length=872, freq=None)
```

In [64]:
```
dataset = pd.concat([y_train, X_train], axis=1, join="inner") # align to avoid random shifting
dataset.describe()
```

Out[64]:

	Price_Corr_6M	1	2	3	4	5	6	7	8	9	10	11	
count	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.(
mean	0.030044	14.633773	23.853326	14.998240	16.275105	17.326078	18.053220	18.726086	19.427902	20.170131	20.982037	21.854472	22.8
std	0.059539	5.527819	9.166414	5.595779	6.001971	6.369740	6.582409	6.864662	7.245918	7.652873	8.173705	8.529445	8.8
min	0.000000	6.640000	10.010000	6.649000	7.390800	7.839800	8.444000	8.757800	8.843600	8.979000	9.084800	9.260000	9.5
25%	0.000000	9.070000	18.330000	9.624400	10.165600	11.004900	11.738900	12.230600	12.883100	13.852450	14.775300	16.073550	17.5
50%	0.000000	13.670000	22.280000	14.274800	16.076000	17.144000	17.733000	18.301800	19.069200	19.876500	20.421000	20.869500	21.4
75%	0.053825	19.677500	27.550000	19.867100	20.420800	21.258000	22.094000	23.362400	24.736000	25.607550	26.150000	26.523650	27.2
max	0.377846	27.670000	44.190000	28.657400	30.306800	32.317000	32.895200	34.569200	36.970400	38.803200	41.334400	42.320600	43.2

In [65]:
```
Y=dataset.pop(col_y) # target to predict
```

In [66]:
```
Y.describe()
```

Out[66]:
```
count    872.000000
mean       0.030044
std        0.059539
min        0.000000
25%        0.000000
50%        0.000000
75%        0.053825
max        0.377846
Name: Price_Corr_6M, dtype: float64
```

In [67]:
```
X=dataset # features for prediction
X.describe()
```

Out[67]:

	1	2	3	4	5	6	7	8	9	10	11	12	
count	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000	872.000(
mean	14.633773	23.853326	14.998240	16.275105	17.326078	18.053220	18.726086	19.427902	20.170131	20.982037	21.854472	22.844800	19.335
std	5.527819	9.166414	5.595779	6.001971	6.369740	6.582409	6.864662	7.245918	7.652873	8.173705	8.529445	8.868805	7.186
min	6.640000	10.010000	6.649000	7.390800	7.839800	8.444000	8.757800	8.843600	8.979000	9.084800	9.260000	9.535600	8.579
25%	9.070000	18.330000	9.624400	10.165600	11.004900	11.738900	12.230600	12.883100	13.852450	14.775300	16.073550	17.534800	13.298
50%	13.670000	22.280000	14.274800	16.076000	17.144000	17.733000	18.301800	19.069200	19.876500	20.421000	20.869500	21.458600	18.867
75%	19.677500	27.550000	19.867100	20.420800	21.258000	22.094000	23.362400	24.736000	25.607550	26.150000	26.523650	27.291200	23.745
max	27.670000	44.190000	28.657400	30.306800	32.317000	32.895200	34.569200	36.970400	38.803200	41.334400	42.320600	43.212800	36.634

In [68]:
```
# PREDICTION
```

In [69]:
```
def generate_model(df, cols, start_date, verbosity = False):
    best_score = 0
    best_model = None
    best_band = 0
    test_size = 0.3
    bands = 60

    for band in range(12,bands+1,12): # [12,24,36,48,60] months
        X_train, y_train = generate_dataset(df, cols, band, start_date)
        dataset = pd.concat([y_train, X_train], axis=1, join="inner") # align to avoid random shifting
        Y=dataset.pop(col_y)
        X=dataset
        X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size, shuffle = True)
        model = xgb.XGBRegressor()
        model.fit(X_train,Y_train.values)
        score = model.score(X_test, Y_test)
        if best_score < score:
            best_score = score
            best_model = model
            best_band = band
            X_t = X_test
            Y_t = Y_test

#         if verbosity: # verbosit for each band
#             print(score, band)
    if verbosity: # verbosity for best score
        print('retest score:', best_model.score(X_t, Y_t).round(3), best_score.round(3), 'features:', cols, 'X_train.shape:', X_train.

    return best_model, best_band, best_score
```

In [70]:
```
start_date = '1970-01-01'
cols_features =['PER', 'Price', 'Price_Var']
model, band, score = generate_model(df, cols_features, start_date, True)
```

```
retest score: 0.799 0.799 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
```

In [71]:
```
# model.score(X_t, Y_t), best_score # retest best model score
```

In [72]:
```
df_initial['horizon'] = 'past'
df_initial.loc[np.isnan(df_initial['Price6F']),['horizon']]='future'
df_initial.head(10)
```

Out[72]:

	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M	horizon
Date															
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	NaN	future

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M	horizon
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	NaN	future
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	NaN	future
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	NaN	future
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	NaN	future
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05	NaN	future
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005	past
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.047467	past
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.044345	past
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793	past

In [73]:
```
df_initial['Date']=df_initial.index # add required column
df_initial.head(24)
```

Out[73]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M	horizon
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	NaN	future
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	NaN	future
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	NaN	future
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	NaN	future
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	NaN	future
2022-09-01	28.42	3850.52	2022-09-01	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	4158.56	-0.074074	3726.05	NaN	future
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005	past
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.047467	past
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.044345	past
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793	past
2022-04-01	33.89	4391.30	2022-04-01	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	4391.27	0.000007	3726.05	0.151493	past
2022-03-01	34.27	4391.27	2022-03-01	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	4435.98	-0.010079	3850.52	0.123142	past
2022-02-01	35.29	4435.98	2022-02-01	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4158.56	4573.82	-0.030137	3898.95	0.121062	past
2022-01-01	36.94	4573.82	2022-01-01	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	3911.73	4674.77	-0.021595	3898.95	0.147551	past
2021-12-01	38.31	4674.77	2021-12-01	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	3898.95	4667.39	0.001581	3898.95	0.165959	past
2021-11-01	38.58	4667.39	2021-11-01	4667.39	4674.77	4573.82	4435.98	4391.27	4391.30	4040.36	4460.71	0.046333	4040.36	0.134343	past
2021-10-01	37.25	4460.71	2021-10-01	4460.71	4667.39	4674.77	4573.82	4435.98	4391.27	4391.30	4445.54	0.003412	4391.27	0.015567	past
2021-09-01	37.62	4445.54	2021-09-01	4445.54	4460.71	4667.39	4674.77	4573.82	4435.98	4391.27	4454.21	-0.001946	4391.27	0.012208	past
2021-08-01	37.97	4454.21	2021-08-01	4454.21	4445.54	4460.71	4667.39	4674.77	4573.82	4435.98	4363.71	0.020739	4435.98	0.004093	past
2021-07-01	37.44	4363.71	2021-07-01	4363.71	4454.21	4445.54	4460.71	4667.39	4674.77	4573.82	4238.49	0.029544	4363.71	0.000000	past
2021-06-01	36.70	4238.49	2021-06-01	4238.49	4363.71	4454.21	4445.54	4460.71	4667.39	4674.77	4167.85	0.016949	4238.49	0.000000	past
2021-05-01	36.55	4167.85	2021-05-01	4167.85	4238.49	4363.71	4454.21	4445.54	4460.71	4667.39	4141.18	0.006440	4167.85	0.000000	past

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M	horizon
2021-04-01	36.72	4141.18	2021-04-01	4141.18	4167.85	4238.49	4363.71	4454.21	4445.54	4460.71	3910.51	0.058987	4141.18	0.000000	past
2021-03-01	35.04	3910.51	2021-03-01	3910.51	4141.18	4167.85	4238.49	4363.71	4454.21	4445.54	3883.43	0.006973	3910.51	0.000000	past

In [74]:
```
df.head()
```

Out[74]:

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M
2022-08-01	30.80	4158.56	2022-08-01	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3991.94	3911.73	0.063100	3726.05	0.104005
2022-07-01	29.00	3911.73	2022-07-01	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	3824.14	3898.95	0.003278	3726.05	0.000000
2022-06-01	29.05	3898.95	2022-06-01	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	3912.38	4040.36	-0.034999	3726.05	0.000000
2022-05-01	30.67	4040.36	2022-05-01	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	3917.49	4391.30	-0.079917	3726.05	0.077793
2022-04-01	33.89	4391.30	2022-04-01	4391.30	4040.36	3898.95	3911.73	4158.56	3850.52	3726.05	4391.27	0.000007	3726.05	0.151493

In [75]:
```
periods=120 # predicted periods
df_pred = df_initial.copy()
cols_features =['PER', 'Price', 'Price_Var']
X_pred, y_dummy = generate_dataset(df_pred, cols_features, band, periods=periods)
X_pred.head() # generated features
```

Out[75]:

Date	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2023-01-13	24.82	38.58	26.182	28.4384	28.9968	29.5088	30.1846	30.9284	31.4830	33.1720	35.1722	37.3032	31.582333	3.372162	2.745544	0.562205	-0.548512
2023-01-01	24.82	38.58	26.182	28.4384	28.9968	29.5752	30.3014	30.9836	31.7344	33.1720	35.1722	37.3032	31.629833	3.357935	2.722800	0.531863	-0.542356
2022-12-01	24.82	38.58	26.182	28.6548	29.0490	29.5976	30.5952	31.0240	31.8564	33.5308	35.1722	37.3032	31.719000	3.329748	2.707467	0.488084	-0.539814
2022-11-01	24.82	38.58	26.182	28.7124	29.2264	29.8112	30.7168	31.1216	31.9770	33.5308	35.1722	37.3032	31.776333	3.305537	2.672056	0.459111	-0.512261
2022-10-01	24.82	38.58	26.182	28.7204	29.2398	29.9720	30.7846	31.2144	31.9770	33.5308	35.1722	37.3032	31.819000	3.282119	2.636800	0.442491	-0.475400

In [76]:
```
X_pred.shape
```

Out[76]:
```
(120, 66)
```

In [77]:
```
# df_pred = df_pred.iloc[:periods]
# df_pred
```

In [78]:
```
y_pred = model.predict(X_pred).round(2) # readable
# y_pred = model.predict(X_pred)
# y_pred
```

In [79]:
```
df_pred = df_pred.iloc[:periods].copy()
df_pred[col_y+'Pred'] = np.clip(y_pred,0,0.5) # replace outliers
```

In [80]:
```
cols_pred = ['PER', 'Price', 'Price_Corr_6M', 'horizon', 'Price_Corr_6MPred']
df_pred[cols_pred].head(36) # one shoot prediction
```

Out[80]:

Date	PER	Price	Price_Corr_6M	horizon	Price_Corr_6MPred
2023-01-13	29.19	3991.94	NaN	future	0.10
2023-01-01	27.96	3824.14	NaN	future	0.10
2022-12-01	28.65	3912.38	NaN	future	0.10
2022-11-01	28.74	3917.49	NaN	future	0.12
2022-10-01	27.35	3726.05	NaN	future	0.11
2022-09-01	28.42	3850.52	NaN	future	0.08
2022-08-01	30.80	4158.56	0.104005	past	0.10
2022-07-01	29.00	3911.73	0.047467	past	0.00
2022-06-01	29.05	3898.95	0.044345	past	0.00

Date	PER	Price	Price_Corr_6M	horizon	Price_Corr_6MPred
2022-05-01	30.67	4040.36	0.077793	past	0.08
2022-04-01	33.89	4391.30	0.151493	past	0.11
2022-03-01	34.27	4391.27	0.123142	past	0.12
2022-02-01	35.29	4435.98	0.121062	past	0.07
2022-01-01	36.94	4573.82	0.147551	past	0.07
2021-12-01	38.31	4674.77	0.165959	past	0.16
2021-11-01	38.58	4667.39	0.134343	past	0.14
2021-10-01	37.25	4460.71	0.015567	past	0.00
2021-09-01	37.62	4445.54	0.012208	past	0.00
2021-08-01	37.97	4454.21	0.004093	past	0.00
2021-07-01	37.44	4363.71	0.000000	past	0.00
2021-06-01	36.70	4238.49	0.000000	past	0.00
2021-05-01	36.55	4167.85	0.000000	past	0.00
2021-04-01	36.72	4141.18	0.000000	past	0.00
2021-03-01	35.04	3910.51	0.000000	past	0.00
2021-02-01	35.10	3883.43	0.000000	past	0.00
2021-01-01	34.51	3793.75	0.000000	past	0.00
2020-12-01	33.77	3695.31	0.000000	past	0.01
2020-11-01	32.47	3548.99	0.000000	past	0.00
2020-10-01	31.28	3418.70	0.000000	past	0.00
2020-09-01	30.84	3365.52	0.000000	past	0.00
2020-08-01	31.16	3391.71	0.007722	past	0.00
2020-07-01	29.60	3207.62	0.000000	past	0.00
2020-06-01	28.84	3104.66	0.000000	past	0.00
2020-05-01	27.33	2919.61	0.000000	past	0.00
2020-04-01	25.93	2761.98	0.000000	past	0.00
2020-03-01	24.82	2652.39	0.000000	past	0.00

In [81]:
```
ind_y = df_pred['horizon']=='past'
# ind_y
```

In [82]:
```
model.score(X_pred.loc[ind_y], df_pred.loc[ind_y,col_y]) # score for past - could contain seen data (X_train)
```

Out[82]:
```
0.7946193676991046
```

In [83]:
```
# ind_y[ind_y==True] # past months for score
```

In [84]:
```
# RANDOM CROSS FOLD
n_times = 10 # build model n times and predict n times
periods = 6 # predicted last months
df_pred = df_initial.copy()
df_pred_iter = df_pred.iloc[:periods].copy()
ls_pc6m_cols=[]
ls_score = []
cols_features =['PER', 'Price', 'Price_Var']
for i in range(n_times):
    start_date = '1970-01-01'
    model, band, score = generate_model(df, cols_features, start_date, True)
    ls_score.append(score.round(3))
    print('score:',score.round(3), 'band:', band)

    X_pred, y_dummy = generate_dataset(df_pred, cols_features, band, periods=periods)
    y_pred = model.predict(X_pred).round(2) # readable
    pc6m_col = 'PC6M'+str(1+i)
    ls_pc6m_cols.append(pc6m_col)
    df_pred_iter[pc6m_col] = y_pred
```

```
retest score: 0.759 0.759 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.759 band: 60
retest score: 0.799 0.799 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.799 band: 60
retest score: 0.787 0.787 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.787 band: 60
retest score: 0.729 0.729 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.729 band: 48
retest score: 0.706 0.706 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.706 band: 24
retest score: 0.815 0.815 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
```

```
score: 0.815 band: 48
retest score: 0.774 0.774 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.774 band: 60
retest score: 0.758 0.758 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.758 band: 36
retest score: 0.712 0.712 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.712 band: 24
retest score: 0.834 0.834 features: ['PER', 'Price', 'Price_Var'] X_train.shape: (442, 66)
score: 0.834 band: 60
```

In [85]:
```
cols_iter = ['PER', 'Price', 'Price_Var', 'horizon']
df_pred_iter=df_pred_iter[cols_iter+ls_pc6m_cols].copy()
```

In [86]:
```
df_pred_iter['PC6M_AVG']=df_pred_iter[ls_pc6m_cols].mean(axis=1) # average of all predictions
```

In [87]:
```
# RANDOM CROSS FOLD PREDICTION with ['PER', 'Price', 'Price_Var']
df_pred_iter
```

Out[87]:

Date	PER	Price	Price_Var	horizon	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	29.19	3991.94	0.043879	future	0.10	0.11	0.08	0.1	0.09	0.11	0.09	0.01	0.18	0.06	0.093
2023-01-01	27.96	3824.14	-0.022554	future	0.10	0.11	0.07	0.1	0.08	0.10	0.08	-0.00	0.19	0.06	0.089
2022-12-01	28.65	3912.38	-0.001304	future	0.10	0.11	0.08	0.1	0.08	0.10	0.09	0.01	0.11	0.06	0.084
2022-11-01	28.74	3917.49	0.051379	future	0.13	0.12	0.11	0.1	0.10	0.10	0.11	0.02	0.11	0.09	0.099
2022-10-01	27.35	3726.05	-0.032326	future	0.13	0.12	0.11	0.1	0.10	0.07	0.10	0.01	0.11	0.09	0.094
2022-09-01	28.42	3850.52	-0.074074	future	0.13	0.11	0.10	0.1	0.08	0.07	0.09	0.02	0.10	0.09	0.089

In [88]:
```
np.mean(ls_score), ls_score # model score mean and model score list
```

Out[88]:
```
(0.7672999999999999,
 [0.759, 0.799, 0.787, 0.729, 0.706, 0.815, 0.774, 0.758, 0.712, 0.834])
```

In [89]:
```
# Buffet Indicator
```

In [90]:
```
# Get your api_key from FRED
# https://fred.stlouisfed.org/docs/api/api_key.html

fred = Fred(api_key='abcdefghijklmnopqrstuvwxyz123456') # example # get your api_key from FRED
```

In [91]:
```
# wilshire = fred.get_series('WILL5000PRFC')
```

In [92]:
```
gdp_data = fred.get_series_latest_release('GDP')
gdp_data
```

Out[92]:
```
1946-01-01          NaN
1946-04-01          NaN
1946-07-01          NaN
1946-10-01          NaN
1947-01-01      243.164
                 ...
2021-07-01    23550.420
2021-10-01    24349.121
2022-01-01    24740.480
2022-04-01    25248.476
2022-07-01    25723.941
Length: 307, dtype: float64
```

In [93]:
```
# Convert from quarter to day with interpolate
idx = pd.date_range(gdp_data.index.min(), gdp_data.index.max())
gdp_data = gdp_data.reindex(idx, fill_value=np.nan).copy()
gdp_data
```

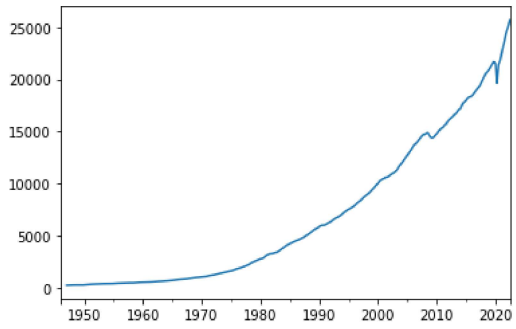
Out[93]:
```
1946-01-01          NaN
1946-01-02          NaN
1946-01-03          NaN
1946-01-04          NaN
1946-01-05          NaN
                 ...
2022-06-27          NaN
2022-06-28          NaN
2022-06-29          NaN
2022-06-30          NaN
2022-07-01    25723.941
Freq: D, Length: 27941, dtype: float64
```

In [94]:
```
gdp_data = pd.to_numeric(gdp_data, errors='coerce').astype('float64').copy()
gdp_data.interpolate(inplace=True)
gdp_data
```

Out[94]:
```
1946-01-01           NaN
1946-01-02           NaN
1946-01-03           NaN
1946-01-04           NaN
1946-01-05           NaN
                  ...
2022-06-27    25703.04144
2022-06-28    25708.26633
2022-06-29    25713.49122
2022-06-30    25718.71611
2022-07-01    25723.94100
Freq: D, Length: 27941, dtype: float64
```

In [95]:
```
# %matplotlib inline
gdp_data.plot.line()
```

Out[95]:
```
<AxesSubplot:>
```



In [96]:
```
wilshire = fred.get_series('WILL5000PRFC')
```

In [97]:
```
idx = pd.date_range(wilshire.index.min(), wilshire.index.max())
wilshire = wilshire.reindex(idx, fill_value=np.nan).copy()
wilshire
```

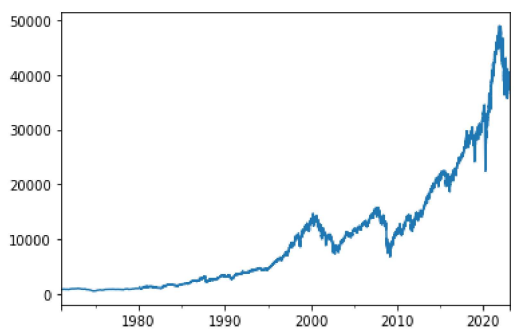
Out[97]:
```
1970-12-31      830.27
1971-01-01         NaN
1971-01-02         NaN
1971-01-03         NaN
1971-01-04         NaN
                 ...
2023-01-08         NaN
2023-01-09    38648.30
2023-01-10    38950.46
2023-01-11    39470.43
2023-01-12    39663.09
Freq: D, Length: 19006, dtype: float64
```

In [98]:
```
wilshire.interpolate(inplace=True)
wilshire
```

Out[98]:
```
1970-12-31      830.270000
1971-01-01      831.754138
1971-01-02      833.238276
1971-01-03      834.722414
1971-01-04      836.206552
                   ...
2023-01-08    38642.433333
2023-01-09    38648.300000
2023-01-10    38950.460000
2023-01-11    39470.430000
2023-01-12    39663.090000
Freq: D, Length: 19006, dtype: float64
```

In [99]:
```
wilshire.plot.line()
```

Out[99]:
```
<AxesSubplot:>
```



In [100…
```
gdp_data.name='GDP'
wilshire.name='Wilshire 5000'
```

```
bf_ind = pd.concat([gdp_data, wilshire], axis=1) # Buffet Indicator
bf_ind
```

Out[100…

	GDP	Wilshire 5000
1946-01-01	NaN	NaN
1946-01-02	NaN	NaN
1946-01-03	NaN	NaN
1946-01-04	NaN	NaN
1946-01-05	NaN	NaN
…	…	…
2023-01-08	NaN	38642.433333
2023-01-09	NaN	38648.300000
2023-01-10	NaN	38950.460000
2023-01-11	NaN	39470.430000
2023-01-12	NaN	39663.090000

28136 rows × 2 columns

In [101…

```
bf_ind.dropna(subset=['Wilshire 5000'], inplace=True) # drop missing GDP
bf_ind
```

Out[101…

	GDP	Wilshire 5000
1970-12-31	1134.650043	830.270000
1971-01-01	1135.156000	831.754138
1971-01-02	1135.390611	833.238276
1971-01-03	1135.625222	834.722414
1971-01-04	1135.859833	836.206552
…	…	…
2023-01-08	NaN	38642.433333
2023-01-09	NaN	38648.300000
2023-01-10	NaN	38950.460000
2023-01-11	NaN	39470.430000
2023-01-12	NaN	39663.090000

19006 rows × 2 columns

In [102…

```
bf_ind.interpolate(method ='spline',order=2, inplace=True)
```

In [103…

```
col_bf_ind='Buffet Indicator'
bf_ind[col_bf_ind] =  bf_ind['Wilshire 5000'] / bf_ind['GDP']
bf_ind
```

Out[103…

	GDP	Wilshire 5000	Buffet Indicator
1970-12-31	1134.650043	830.270000	0.731741
1971-01-01	1135.156000	831.754138	0.732722
1971-01-02	1135.390611	833.238276	0.733878
1971-01-03	1135.625222	834.722414	0.735033
1971-01-04	1135.859833	836.206552	0.736188
…	…	…	…
2023-01-08	26675.791107	38642.433333	1.448596
2023-01-09	26680.609410	38648.300000	1.448554
2023-01-10	26685.425950	38950.460000	1.459615
2023-01-11	26690.240726	39470.430000	1.478834
2023-01-12	26695.053740	39663.090000	1.485784

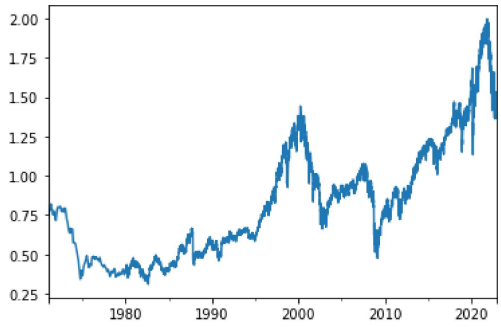
19006 rows × 3 columns

In [104…

```
# https://www.longtermtrends.net/market-cap-to-gdp-the-buffett-indicator/
bf_ind[col_bf_ind].plot.line() # compare with chart from above link
```

Out[104…

```
<AxesSubplot:>
```



In [105…
```
bf_ind.index.min(), bf_ind.index.max()
```

Out[105…
```
(Timestamp('1970-12-31 00:00:00', freq='D'),
 Timestamp('2023-01-12 00:00:00', freq='D'))
```

In [106…
```
df_initial
```

Out[106…

Date	PER	Price	Date	Price0F	Price1F	Price2F	Price3F	Price4F	Price5F	Price6F	Price1P	Price_Var	PriceFmin	Price_Corr_6M	horizon
2023-01-13	29.19	3991.94	2023-01-13	3991.94	NaN	NaN	NaN	NaN	NaN	NaN	3824.14	0.043879	3991.94	NaN	future
2023-01-01	27.96	3824.14	2023-01-01	3824.14	3991.94	NaN	NaN	NaN	NaN	NaN	3912.38	-0.022554	3824.14	NaN	future
2022-12-01	28.65	3912.38	2022-12-01	3912.38	3824.14	3991.94	NaN	NaN	NaN	NaN	3917.49	-0.001304	3824.14	NaN	future
2022-11-01	28.74	3917.49	2022-11-01	3917.49	3912.38	3824.14	3991.94	NaN	NaN	NaN	3726.05	0.051379	3824.14	NaN	future
2022-10-01	27.35	3726.05	2022-10-01	3726.05	3917.49	3912.38	3824.14	3991.94	NaN	NaN	3850.52	-0.032326	3726.05	NaN	future
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
1871-06-01	12.59	4.82	1871-06-01	4.82	4.73	4.79	4.84	4.59	4.64	4.74	4.86	-0.008230	4.59	0.047718	past
1871-05-01	12.59	4.86	1871-05-01	4.86	4.82	4.73	4.79	4.84	4.59	4.64	4.74	0.025316	4.59	0.055556	past
1871-04-01	12.05	4.74	1871-04-01	4.74	4.86	4.82	4.73	4.79	4.84	4.59	4.61	0.028200	4.59	0.031646	past
1871-03-01	11.19	4.61	1871-03-01	4.61	4.74	4.86	4.82	4.73	4.79	4.84	4.50	0.024444	4.61	0.000000	past
1871-02-01	10.92	4.50	1871-02-01	4.50	4.61	4.74	4.86	4.82	4.73	4.79	NaN	NaN	4.50	NaN	past

1825 rows × 15 columns

In [107…
```
df_pred = df_initial.copy()
df_pred = pd.concat([df_pred, bf_ind], axis=1)
df_pred.sort_index(ascending=False, inplace=True)
```

In [108…
```
cols_iter = ['Date','PER', 'Price', 'Price_Var', 'horizon', 'Buffet Indicator', 'Price_Corr_6M']
df_pred[col_bf_ind] = df_pred[col_bf_ind].interpolate(method='linear', limit_direction='backward')
df_pred = df_pred[cols_iter].copy()
df_pred.dropna(subset=['horizon', 'Buffet Indicator'], inplace=True)
df_pred
```

Out[108…

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	Price_Corr_6M
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	NaN
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	NaN
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	NaN
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	NaN
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	NaN
…	…	…	…	…	…	…	…
1971-05-01	1971-05-01	17.56	101.60	-0.013592	past	0.820049	0.086811
1971-04-01	1971-04-01	17.92	103.00	0.034137	past	0.799294	0.055922
1971-03-01	1971-03-01	17.40	99.60	0.025641	past	0.773546	0.023695
1971-02-01	1971-02-01	17.03	97.11	0.038721	past	0.765564	0.000000
1971-01-01	1971-01-01	16.46	93.49	0.038201	past	0.732722	0.000000

626 rows × 7 columns

In [109…

```
cols_train=['Date','PER', 'Price', 'Price_Var', 'Buffet Indicator', 'Price_Corr_6M']
df= df_pred[cols_train].copy()
df.dropna(inplace=True)
df
```

Out[109…

	Date	PER	Price	Price_Var	Buffet Indicator	Price_Corr_6M
2022-08-01	2022-08-01	30.80	4158.56	0.063100	1.585419	0.104005
2022-07-01	2022-07-01	29.00	3911.73	0.003278	1.477178	0.047467
2022-06-01	2022-06-01	29.05	3898.95	-0.034999	1.595022	0.044345
2022-05-01	2022-05-01	30.67	4040.36	-0.079917	1.635094	0.077793
2022-04-01	2022-04-01	33.89	4391.30	0.000007	1.814186	0.151493
…	…	…	…	…	…	…
1971-05-01	1971-05-01	17.56	101.60	-0.013592	0.820049	0.086811
1971-04-01	1971-04-01	17.92	103.00	0.034137	0.799294	0.055922
1971-03-01	1971-03-01	17.40	99.60	0.025641	0.773546	0.023695
1971-02-01	1971-02-01	17.03	97.11	0.038721	0.765564	0.000000
1971-01-01	1971-01-01	16.46	93.49	0.038201	0.732722	0.000000

620 rows × 6 columns

In [110…

```
# RANDOM CROSS FOLD for Bootstrapping
ls_cols_features=['PER', 'Price', 'Price_Var','Buffet Indicator'] # test with one by one feature
ls_cols_features.append(['PER','Buffet Indicator'])  # test with set of features
ls_cols_features.append(['PER', 'Price','Buffet Indicator'])  # test with set of features
start_date = df.index[-periods]

for e in ls_cols_features:
    n_times = 10 # build model n times and predict n times
    periods = 6 # predicted last months
    # df_pred = df_initial.copy()
    df_pred_iter = df_pred.iloc[:periods].copy()
    ls_pc6m_cols=[]
    ls_score = []
    cols_features = [e]
    if isinstance(e, list):
        cols_features = e # set of features
    else:
        cols_features = [e] # one feature

    for i in range(n_times):
    #     start_date = '1974-01-01'
        model, band, score = generate_model(df, cols_features, start_date, True)
        score = score.round(3)
        ls_score.append(score)
        print('score:',score, 'band:', band)

        X_pred, y_dummy = generate_dataset(df_pred, cols_features, band, periods=periods)
        y_pred = model.predict(X_pred).round(2) # recols_features, adable
        y_pred = np.clip(y_pred,0,0.5) # replace outliers
        pc6m_col = 'PC6M'+str(1+i)
        ls_pc6m_cols.append(pc6m_col)
        df_pred_iter[pc6m_col] = y_pred

    # RANDOM CROSS FOLD PREDICTION on PER and BUFFET INDICATOR
    df_pred_iter['PC6M_AVG']=df_pred_iter[ls_pc6m_cols].mean(axis=1) # average of all predictions
    display(df_pred_iter.loc[:,df_pred_iter.columns!=col_y])
    print('scores:', np.mean(ls_score).round(3), ls_score) # model score mean and model score list)
```

```
retest score: 0.754 0.754 features: ['PER'] X_train.shape: (430, 22)
score: 0.754 band: 36
retest score: 0.729 0.729 features: ['PER'] X_train.shape: (430, 22)
score: 0.729 band: 48
retest score: 0.748 0.748 features: ['PER'] X_train.shape: (430, 22)
score: 0.748 band: 60
retest score: 0.746 0.746 features: ['PER'] X_train.shape: (430, 22)
score: 0.746 band: 36
retest score: 0.778 0.778 features: ['PER'] X_train.shape: (430, 22)
score: 0.778 band: 24
retest score: 0.763 0.763 features: ['PER'] X_train.shape: (430, 22)
score: 0.763 band: 36
retest score: 0.788 0.788 features: ['PER'] X_train.shape: (430, 22)
score: 0.788 band: 60
retest score: 0.641 0.641 features: ['PER'] X_train.shape: (430, 22)
score: 0.641 band: 24
retest score: 0.72 0.72 features: ['PER'] X_train.shape: (430, 22)
score: 0.72 band: 60
retest score: 0.64 0.64 features: ['PER'] X_train.shape: (430, 22)
score: 0.64 band: 12
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.03	0.1	0.00	0.10	0.09	0.02	0.02	0.11	0.03	0.11	0.061

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.07	0.1	0.00	0.12	0.13	0.02	0.02	0.12	0.03	0.12	0.073
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.05	0.1	0.00	0.11	0.11	0.02	0.03	0.13	0.03	0.09	0.067
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.06	0.1	0.03	0.10	0.11	0.03	0.04	0.15	0.06	0.03	0.071
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.06	0.1	0.03	0.09	0.12	0.02	0.04	0.19	0.05	0.12	0.082
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.10	0.1	0.04	0.10	0.09	0.04	0.04	0.13	0.05	0.03	0.072

```
scores: 0.731 [0.754, 0.729, 0.748, 0.746, 0.778, 0.763, 0.788, 0.641, 0.72, 0.64]
retest score: 0.799 0.799 features: ['Price'] X_train.shape: (430, 22)
score: 0.799 band: 24
retest score: 0.781 0.781 features: ['Price'] X_train.shape: (430, 22)
score: 0.781 band: 60
retest score: 0.76 0.76 features: ['Price'] X_train.shape: (430, 22)
score: 0.76 band: 60
retest score: 0.702 0.702 features: ['Price'] X_train.shape: (430, 22)
score: 0.702 band: 60
retest score: 0.788 0.788 features: ['Price'] X_train.shape: (430, 22)
score: 0.788 band: 24
retest score: 0.768 0.768 features: ['Price'] X_train.shape: (430, 22)
score: 0.768 band: 48
retest score: 0.813 0.813 features: ['Price'] X_train.shape: (430, 22)
score: 0.813 band: 60
retest score: 0.746 0.746 features: ['Price'] X_train.shape: (430, 22)
score: 0.746 band: 60
retest score: 0.759 0.759 features: ['Price'] X_train.shape: (430, 22)
score: 0.759 band: 48
retest score: 0.837 0.837 features: ['Price'] X_train.shape: (430, 22)
score: 0.837 band: 48
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.09	0.05	0.07	0.1	0.08	0.10	0.13	0.11	0.12	0.10	0.095
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.09	0.05	0.07	0.1	0.07	0.10	0.13	0.10	0.12	0.09	0.092
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.08	0.05	0.06	0.1	0.07	0.10	0.13	0.10	0.12	0.09	0.090
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.09	0.05	0.06	0.1	0.10	0.10	0.10	0.10	0.12	0.10	0.092
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.09	0.05	0.06	0.1	0.11	0.10	0.10	0.10	0.11	0.10	0.092
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.08	0.04	0.06	0.1	0.10	0.11	0.10	0.10	0.10	0.10	0.089

```
scores: 0.775 [0.799, 0.781, 0.76, 0.702, 0.788, 0.768, 0.813, 0.746, 0.759, 0.837]
retest score: 0.626 0.626 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.626 band: 36
retest score: 0.732 0.732 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.732 band: 36
retest score: 0.712 0.712 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.712 band: 36
retest score: 0.797 0.797 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.797 band: 36
retest score: 0.712 0.712 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.712 band: 60
retest score: 0.717 0.717 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.717 band: 60
retest score: 0.676 0.676 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.676 band: 36
retest score: 0.668 0.668 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.668 band: 36
retest score: 0.706 0.706 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.706 band: 24
retest score: 0.751 0.751 features: ['Price_Var'] X_train.shape: (430, 22)
score: 0.751 band: 36
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.06	0.05	0.09	0.08	0.07	0.09	0.07	0.09	0.03	0.04	0.067
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.07	0.05	0.11	0.12	0.07	0.06	0.04	0.06	0.04	0.08	0.070
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.08	0.05	0.05	0.07	0.07	0.08	0.07	0.08	0.04	0.08	0.067
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.08	0.05	0.04	0.07	0.07	0.09	0.07	0.07	0.04	0.08	0.066
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.08	0.05	0.04	0.07	0.07	0.07	0.07	0.07	0.05	0.07	0.064

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.08	0.05	0.04	0.07	0.07	0.09	0.07	0.08	0.04	0.07	0.066

```
scores: 0.71 [0.626, 0.732, 0.712, 0.797, 0.712, 0.717, 0.676, 0.668, 0.706, 0.751]
retest score: 0.668 0.668 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.668 band: 48
retest score: 0.616 0.616 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.616 band: 24
retest score: 0.725 0.725 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.725 band: 60
retest score: 0.783 0.783 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.783 band: 60
retest score: 0.804 0.804 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.804 band: 60
retest score: 0.724 0.724 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.724 band: 36
retest score: 0.77 0.77 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.77 band: 60
retest score: 0.629 0.629 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.629 band: 24
retest score: 0.78 0.78 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.78 band: 24
retest score: 0.758 0.758 features: ['Buffet Indicator'] X_train.shape: (430, 22)
score: 0.758 band: 24
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.09	0.14	0.10	0.11	0.04	0.11	0.10	0.16	0.20	0.17	0.122
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.10	0.15	0.10	0.11	0.04	0.08	0.10	0.15	0.10	0.13	0.106
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.10	0.13	0.10	0.11	0.04	0.08	0.10	0.11	0.10	0.10	0.097
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.10	0.14	0.10	0.10	0.06	0.08	0.11	0.13	0.15	0.15	0.112
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.07	0.13	0.10	0.10	0.06	0.08	0.11	0.08	0.14	0.16	0.103
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.10	0.10	0.11	0.11	0.07	0.08	0.11	0.11	0.09	0.09	0.097

```
scores: 0.726 [0.668, 0.616, 0.725, 0.783, 0.804, 0.724, 0.77, 0.629, 0.78, 0.758]
retest score: 0.775 0.775 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.775 band: 12
retest score: 0.684 0.684 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.684 band: 48
retest score: 0.834 0.834 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.834 band: 60
retest score: 0.738 0.738 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.738 band: 60
retest score: 0.778 0.778 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.778 band: 60
retest score: 0.73 0.73 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.73 band: 60
retest score: 0.791 0.791 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.791 band: 60
retest score: 0.784 0.784 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.784 band: 24
retest score: 0.741 0.741 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.741 band: 60
retest score: 0.747 0.747 features: ['PER', 'Buffet Indicator'] X_train.shape: (430, 44)
score: 0.747 band: 60
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.04	0.1	0.04	0.03	0.04	0.04	0.02	0.08	0.03	0.06	0.048
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.04	0.1	0.04	0.03	0.04	0.03	0.02	0.16	0.03	0.07	0.056
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.05	0.1	0.04	0.03	0.05	0.05	0.04	0.09	0.03	0.06	0.054
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.08	0.1	0.07	0.04	0.05	0.05	0.07	0.09	0.06	0.10	0.071
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.14	0.1	0.07	0.04	0.05	0.06	0.08	0.09	0.07	0.10	0.080
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.07	0.1	0.07	0.04	0.05	0.04	0.07	0.05	0.04	0.08	0.061

```
scores: 0.76 [0.775, 0.684, 0.834, 0.738, 0.778, 0.73, 0.791, 0.784, 0.741, 0.747]
retest score: 0.722 0.722 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.722 band: 48
retest score: 0.779 0.779 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.779 band: 60
retest score: 0.787 0.787 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.787 band: 48
retest score: 0.797 0.797 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.797 band: 24
```

```
retest score: 0.816 0.816 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.816 band: 60
retest score: 0.778 0.778 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.778 band: 48
retest score: 0.739 0.739 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.739 band: 60
retest score: 0.765 0.765 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.765 band: 48
retest score: 0.768 0.768 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.768 band: 60
retest score: 0.821 0.821 features: ['PER', 'Price', 'Buffet Indicator'] X_train.shape: (430, 66)
score: 0.821 band: 24
```

	Date	PER	Price	Price_Var	horizon	Buffet Indicator	PC6M1	PC6M2	PC6M3	PC6M4	PC6M5	PC6M6	PC6M7	PC6M8	PC6M9	PC6M10	PC6M_AVG
2023-01-13	2023-01-13	29.19	3991.94	0.043879	future	1.485784	0.10	0.03	0.15	0.10	0.05	0.11	0.06	0.1	0.05	0.13	0.088
2023-01-01	2023-01-01	27.96	3824.14	-0.022554	future	1.425834	0.07	0.03	0.14	0.12	0.05	0.10	0.03	0.1	0.05	0.15	0.084
2022-12-01	2022-12-01	28.65	3912.38	-0.001304	future	1.530109	0.07	0.03	0.14	0.11	0.05	0.10	0.03	0.1	0.05	0.15	0.083
2022-11-01	2022-11-01	28.74	3917.49	0.051379	future	1.464387	0.07	0.05	0.10	0.06	0.10	0.10	0.06	0.1	0.08	0.15	0.087
2022-10-01	2022-10-01	27.35	3726.05	-0.032326	future	1.379913	0.07	0.05	0.10	0.09	0.10	0.10	0.05	0.1	0.07	0.13	0.086
2022-09-01	2022-09-01	28.42	3850.52	-0.074074	future	1.519690	0.07	0.06	0.10	0.06	0.09	0.10	0.06	0.1	0.07	0.11	0.082

```
scores: 0.777 [0.722, 0.779, 0.787, 0.797, 0.816, 0.778, 0.739, 0.765, 0.768, 0.821]
```

In [111…

```
#
```