Машинне навчання з scikit-learn

Розглянемо набір даних про погоду, який являє собою часовий ряд.

```
df=pd.read_csv(weather.csv',index_col=['CET'],parse_dates=True)
df.index
```

```
DatetimeIndex(['2005-01-01', '2005-01-02', '2005-01-03', '2005-01-04',
               '2005-01-05', '2005-01-06', '2005-01-07', '2005-01-08',
               '2005-01-09', '2005-01-10',
               ...
               '2014-12-22', '2014-12-23', '2014-12-24', '2014-12-25',
               '2014-12-26', '2014-12-27', '2014-12-28', '2014-12-29',
               '2014-12-30', '2014-12-31'],
              dtype='datetime64[ns]', name='CET', length=3652, freq=None)
```

За замовчуванням індекс не матиме частоти

Машинне навчання з scikit-learn

Тому потрібно встановити частоту за допомогою метода asfreq:

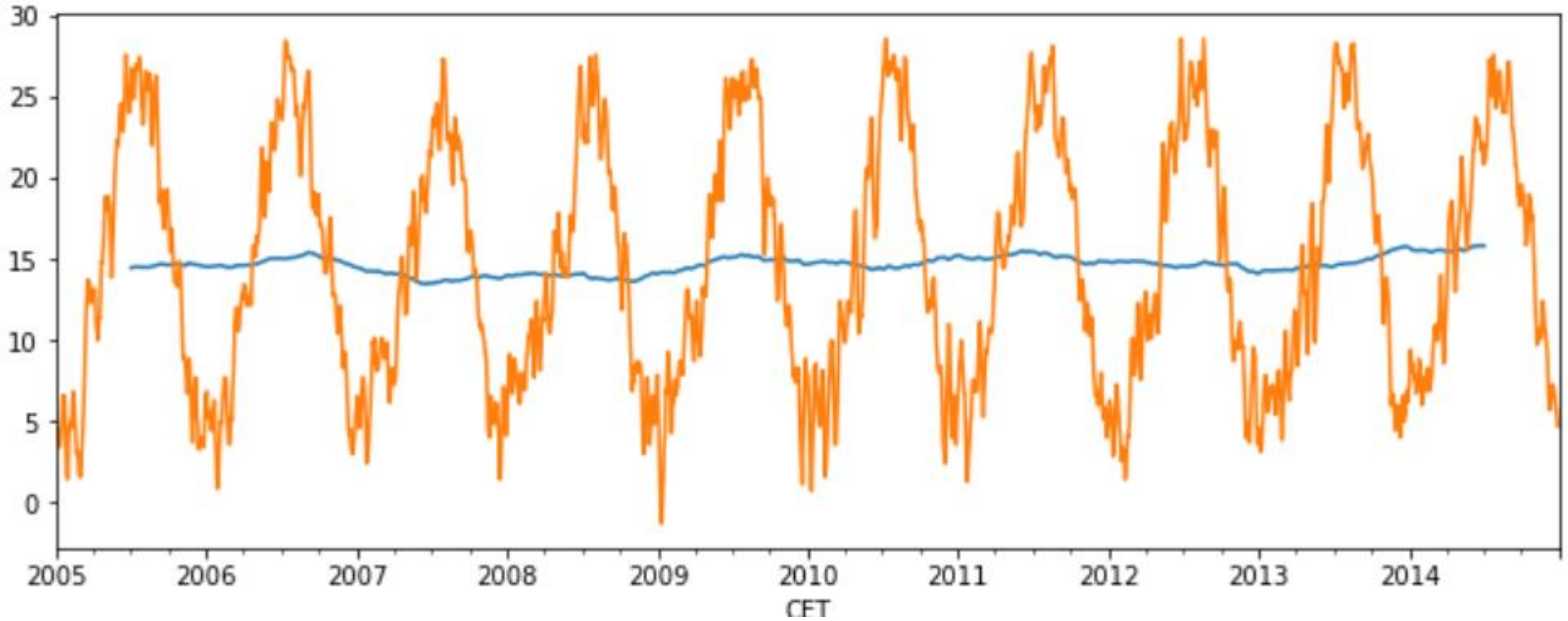
df=df.asfreq('D')

```
DatetimeIndex(['2005-01-01', '2005-01-02', '2005-01-03', '2005-01-04',
               '2005-01-05', '2005-01-06', '2005-01-07', '2005-01-08',
               '2005-01-09', '2005-01-10',
               ...
               '2014-12-22', '2014-12-23', '2014-12-24', '2014-12-25',
               '2014-12-26', '2014-12-27', '2014-12-28', '2014-12-29',
               '2014-12-30', '2014-12-31'],
              dtype='datetime64[ns]', name='CET', length=3652, freq='D')
```

Машинне навчання з scikit-learn

За допомогою ковзних характеристик (наприклад, ковзного середнього) можна виявити наявність тренду в даних.

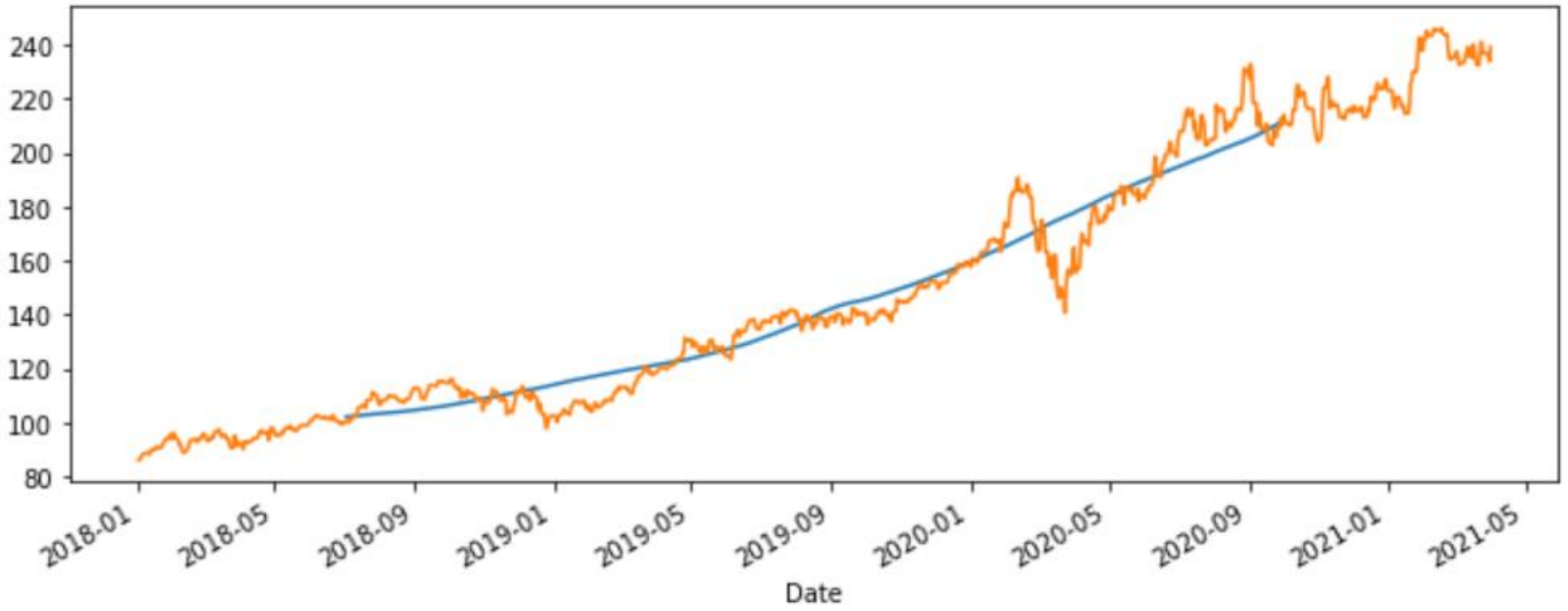
df.rolling(365, center = True).mean().plot()
df.rolling(7, center = True).mean().plot()



Машинне навчання з scikit-learn

В даних про погоду тренд не спостерігається. Розглянемо для порівняння дані про найвищу за день ціну акцій компанії Майкрософт на біржі:

```
mss.rolling(365, center = True).mean().plot()
mss.plot()
```



Машинне навчання з scikit-learn

Іноді було б корисно видалити тренд з даних, особливо якщо він досить виражений, щоб можна було оцінити сезонні коливання або шум у часових рядах. Видалення тренду також може спростити процес моделювання та покращити продуктивність моделі.

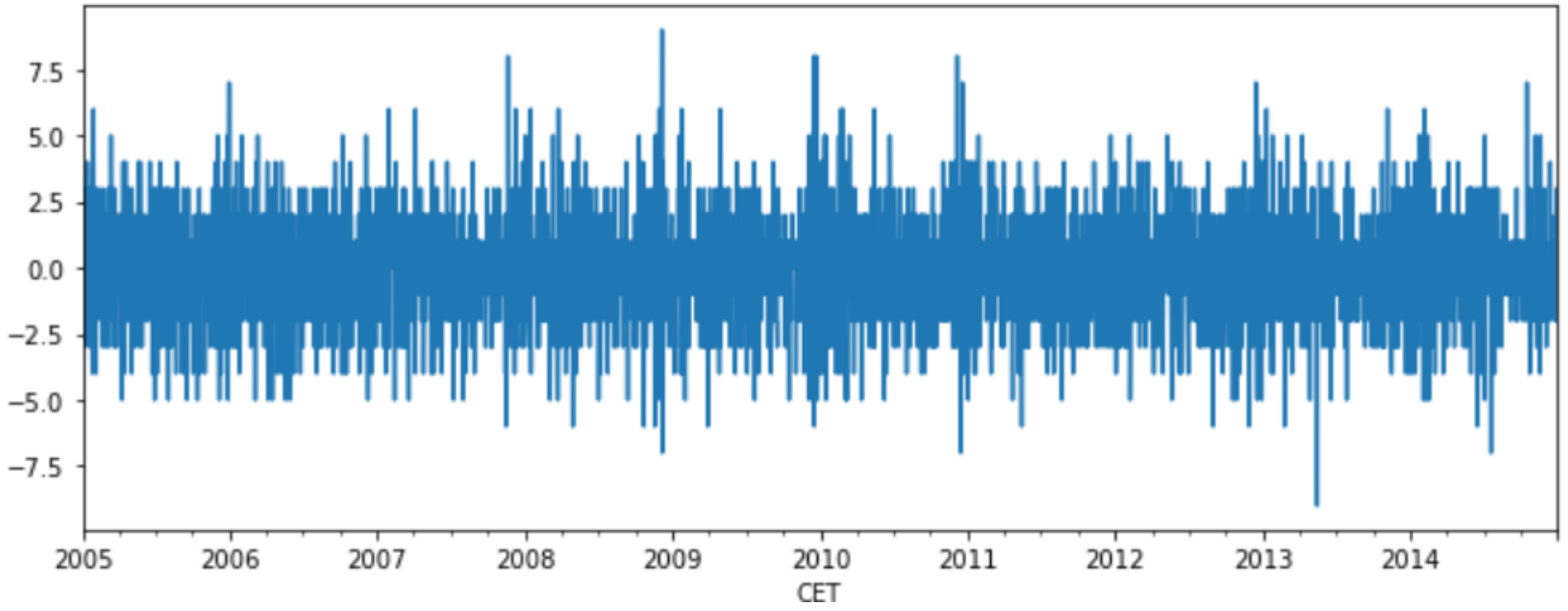
Часовий ряд із трендом називається нестаціонарним.
Часовий ряд, який не має тренду або який було вилучено, називається стаціонарним.

Машинне навчання з scikit-learn

Для вилучення тренду може бути достатнім відняти від кожного значення попереднє.

df_diff= df.diff()

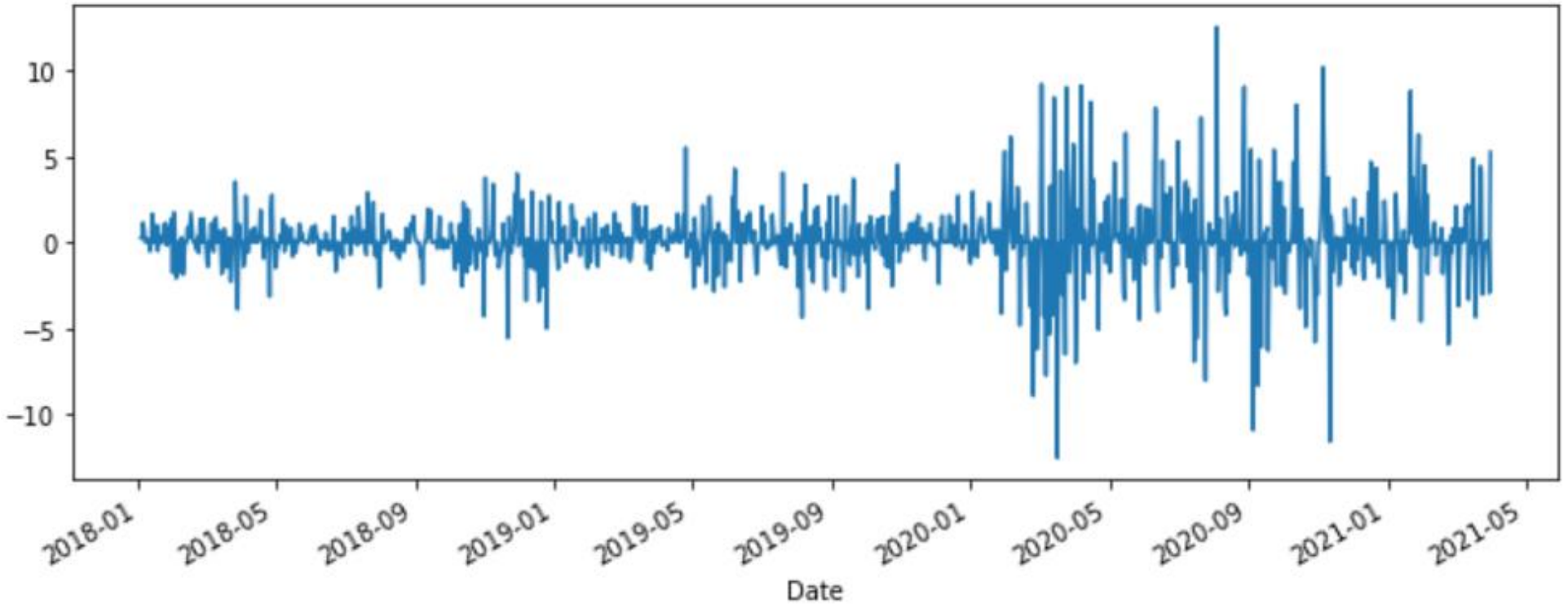
df_diff.plot()



Машинне навчання з scikit-learn

Для даних, в яких спостерігався тренд:

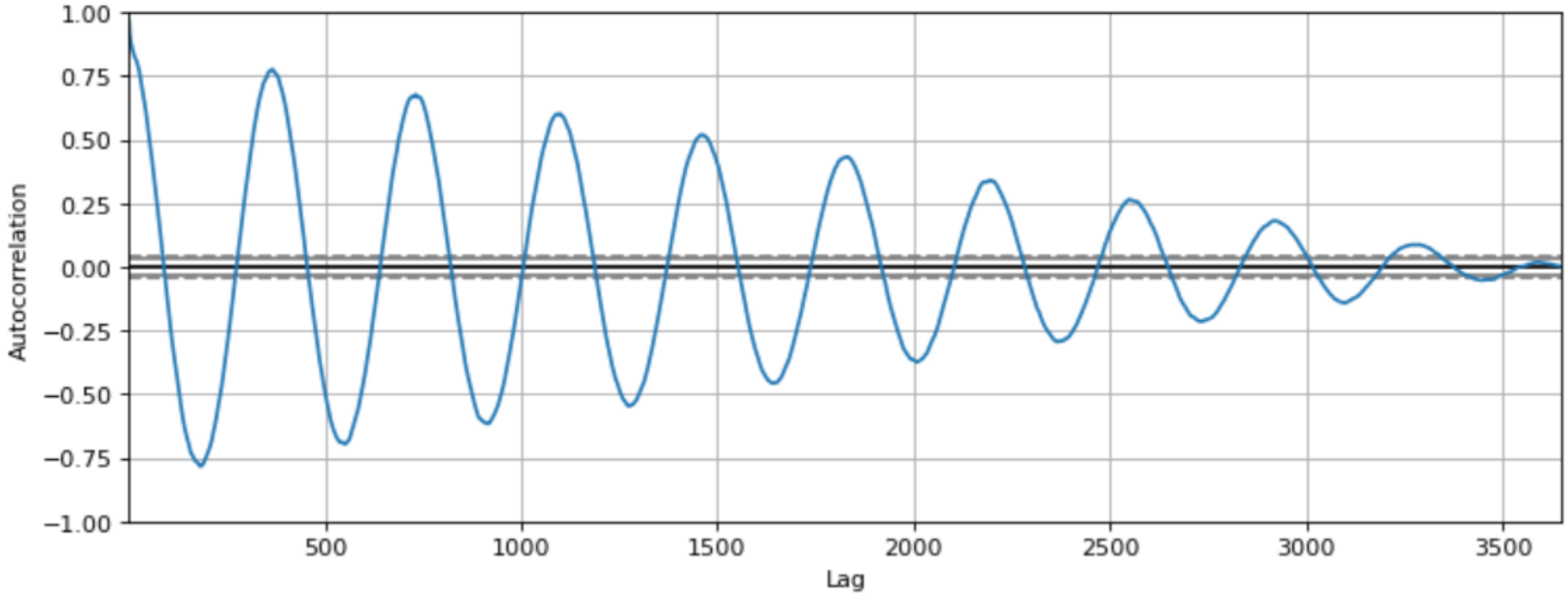
mss_diff = mss.diff()



Машинне навчання з scikit-learn

Автокореляція – це техніка для аналізу сезонності. Вона демонструє кореляцію часового ряду з самим собою, зсунутим в часі.

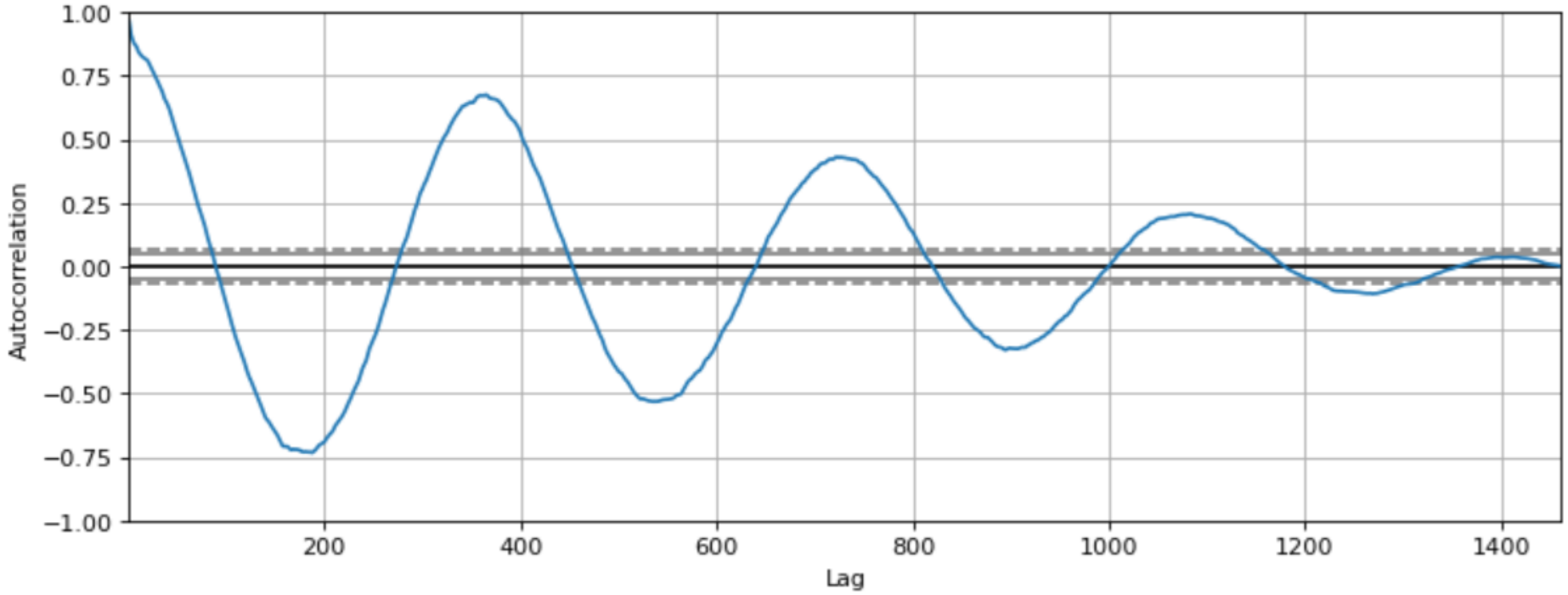
pd.plotting.autocorrelation_plot(df)



Машинне навчання з scikit-learn

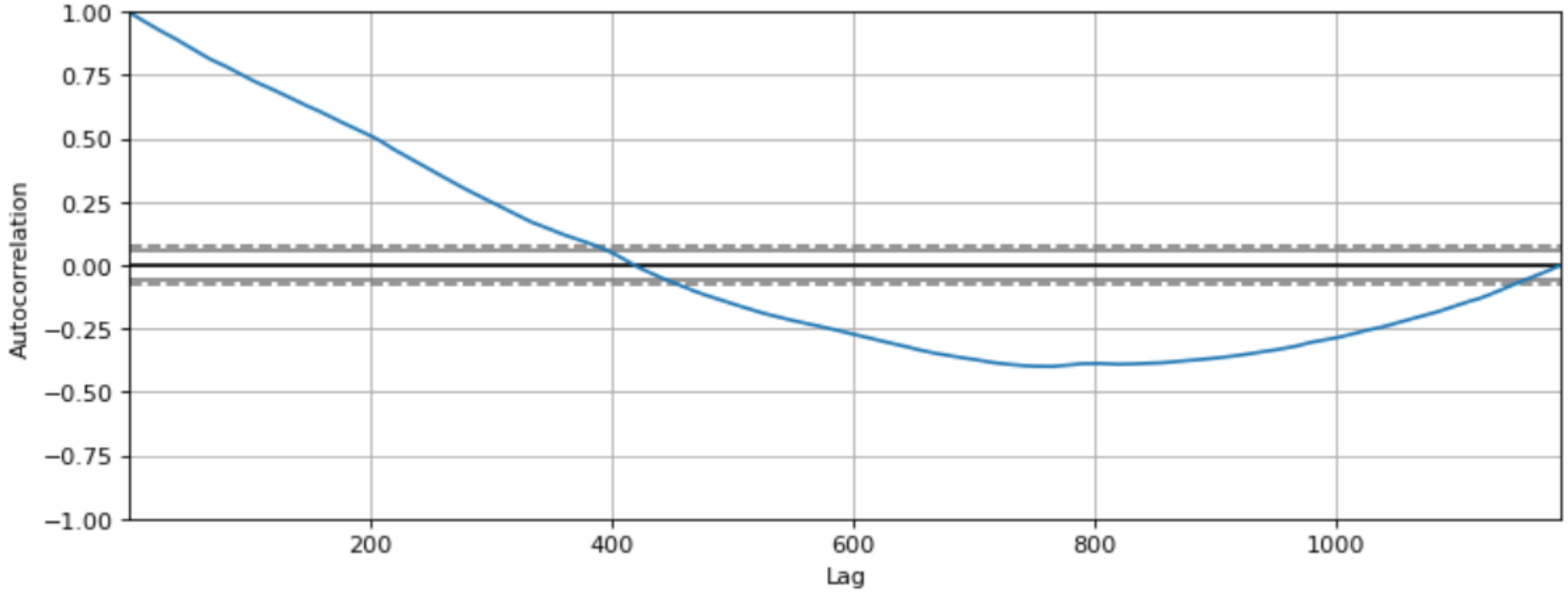
Візьмемо роки 2011-2015:

```
pd.plotting.autocorrelation_plot(df.loc['2011':'2015']);
```



Машинне навчання з scikit-learn

І для біржових даних:
pd.plotting.autocorrelation_plot(mss);



Машинне навчання з scikit-learn

Процес прогнозування складається з передбачення майбутнього значення часового ряду шляхом моделювання ряду виключно на основі його поведінки в минулому (авторегресія) або за допомогою інших зовнішніх змінних.

Щоб застосувати моделі машинного навчання до проблем прогнозування, часові ряди потрібно перетворити на таблицю, у якій кожне значення пов’язане з часовим вікном, що йому передує.

У контексті часових рядів відставання від часового кроку t визначається як значення ряду на попередніх часових кроках.

Машинне навчання з scikit-learn

Створимо два додаткових стовпця: значення середньої температури за попередній день та різниця між днями.

```
df['Yesterday'] = df['Mean TemperatureC'].shift()
df['Yesterday_Diff'] = df['Yesterday'].diff()
df = df.dropna()
```

CET	Mean TemperatureC	Yesterday	Yesterday_Diff
2005-01-03	4.0	6.0	2.0
2005-01-04	3.0	4.0	-2.0
2005-01-05	3.0	3.0	-1.0
2005-01-06	6.0	3.0	0.0
2005-01-07	6.0	6.0	3.0

Машинне навчання з scikit-learn

Розіб’ємо дані на навчальні та тестові. Часові ряди не можна розбивати випадковим чином, тому візьмемо дані за 2005-2013 роки як навчальні і дані за 2014 рік як тестові.
Вхідні ознаки будуть містити температуру за попередній день та різницю температур, вихідна ознака – поточна середня температура.

```
X_train = df['2005':'2013'].drop(['Mean TemperatureC'], axis = 1)
y_train = df.loc['2005':'2013','Mean TemperatureC']
X_test = df.loc['2014'].drop(['Mean TemperatureC'], axis = 1)
y_test = df.loc['2014','Mean TemperatureC’]
```

Машинне навчання з scikit-learn

Оскільки температура – це неперервна змінна, то задача її прогнозування є задачею регресії. Спочатку застосуємо лінійну регресію.

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
lr.score(X_test,y_test)
```

0.9194836530923867

Машинне навчання з scikit-learn

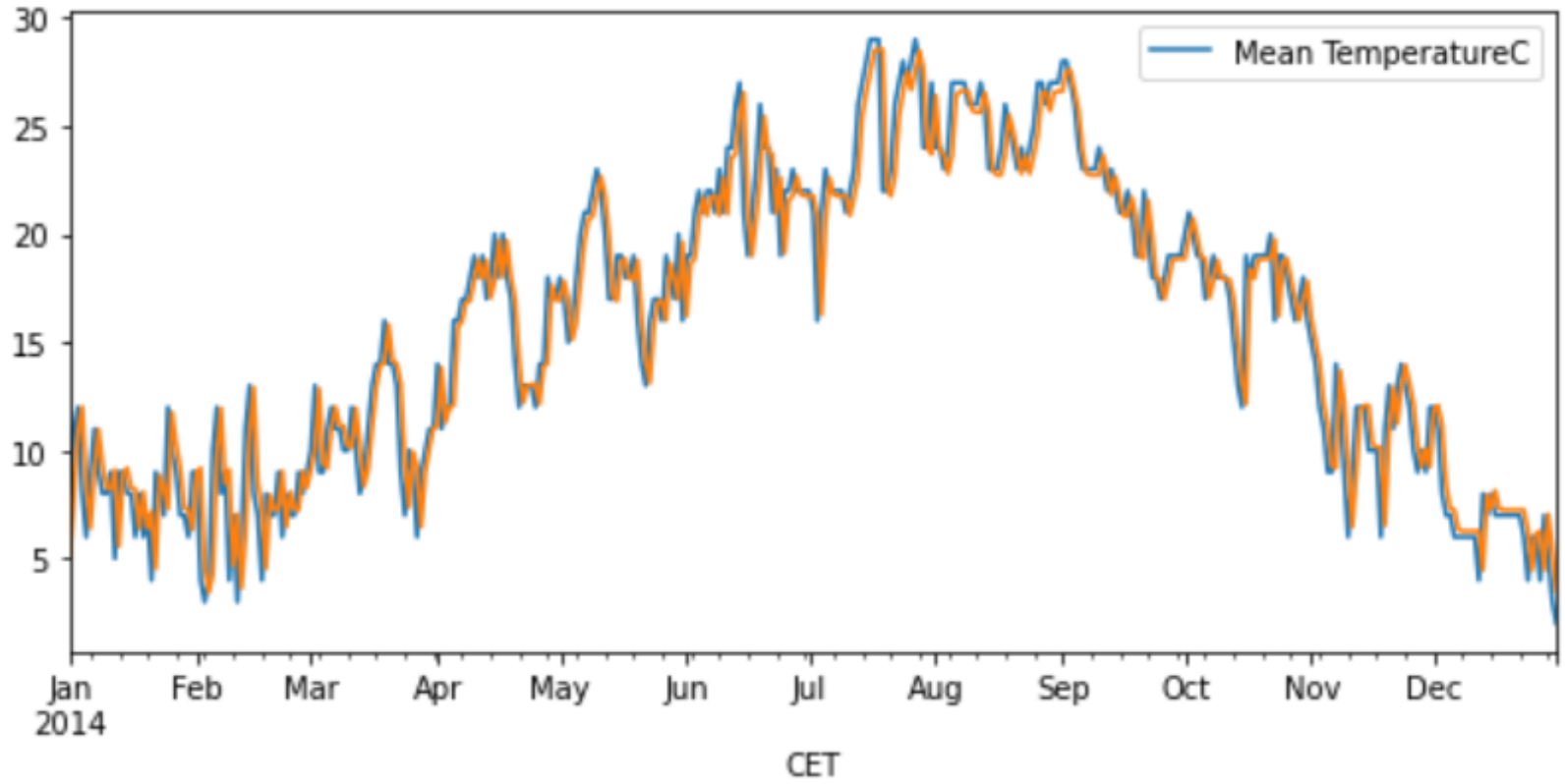
Можна покращити результат за допомогою перехресної перевірки. Але звичайна k-кратна перехресна перевірка не підходить для часових рядів. Scikit-learn має спеціальну функцію для часових рядів – TimeSeriesSplit.

```
from sklearn.model_selection import TimeSeriesSplit
from sklearn.model_selection import cross_val_score
tscv = TimeSeriesSplit(n_splits=8)
cv_results = cross_val_score(lr, X_train, y_train, cv=tscv, scoring='r2')
cv_results.mean()
```

```
0.9308245971010781
```

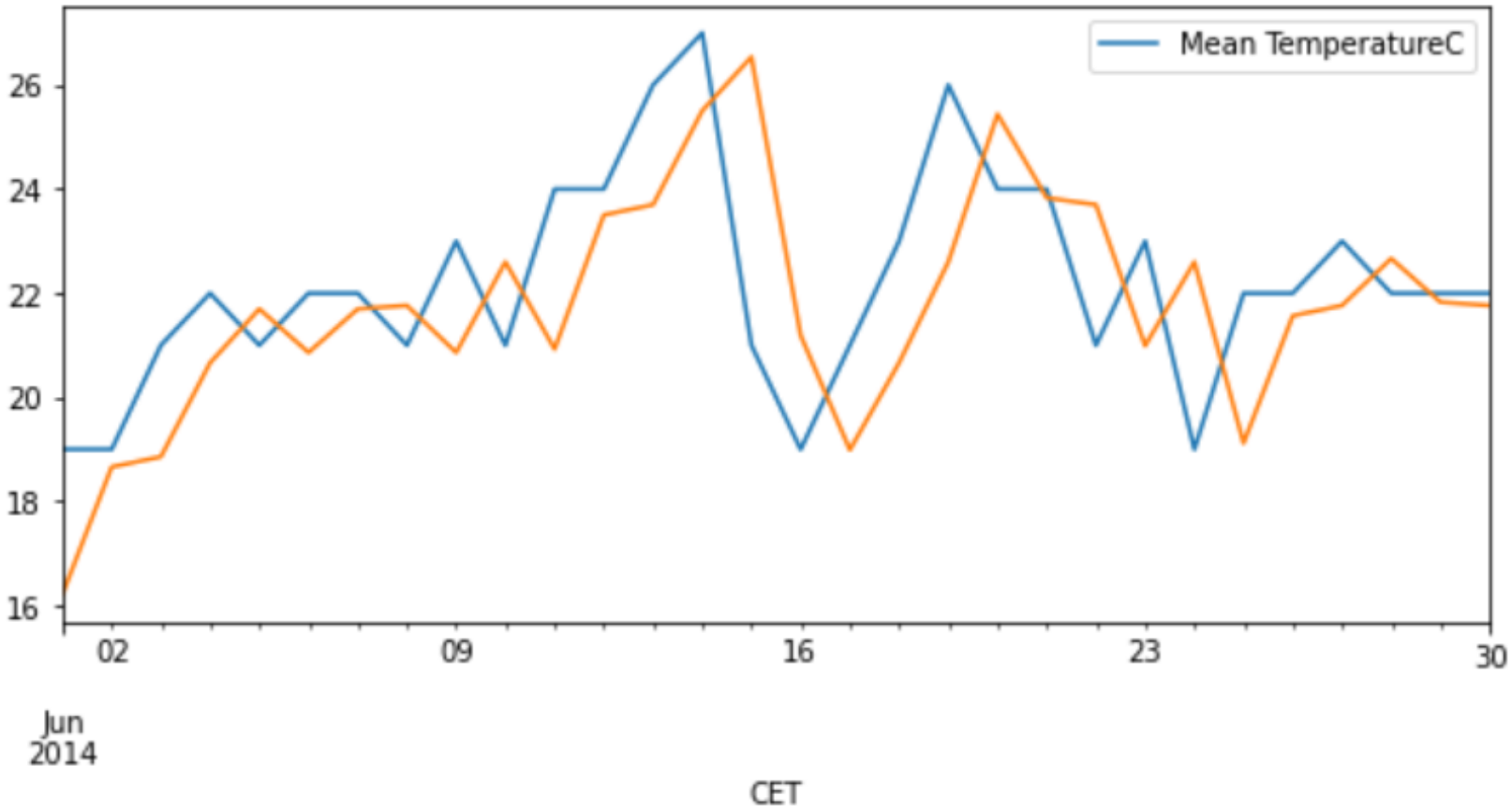
Машинне навчання з scikit-learn

Зобразимо результат графічно:



Машинне навчання з scikit-learn

За певний місяць, наприклад, червень:



Машинне навчання з scikit-learn

Використаємо іншу модель, а саме, випадковий ліс:

```
from sklearn.ensemble import RandomForestRegressor
RF=RandomForestRegressor(max_depth=5,n_estimators=100,random_
state=0)
RF.fit(X_train,y_train)
RF.score(X_test,y_test)
```

0.9186379885178024

Машинне навчання з scikit-learn

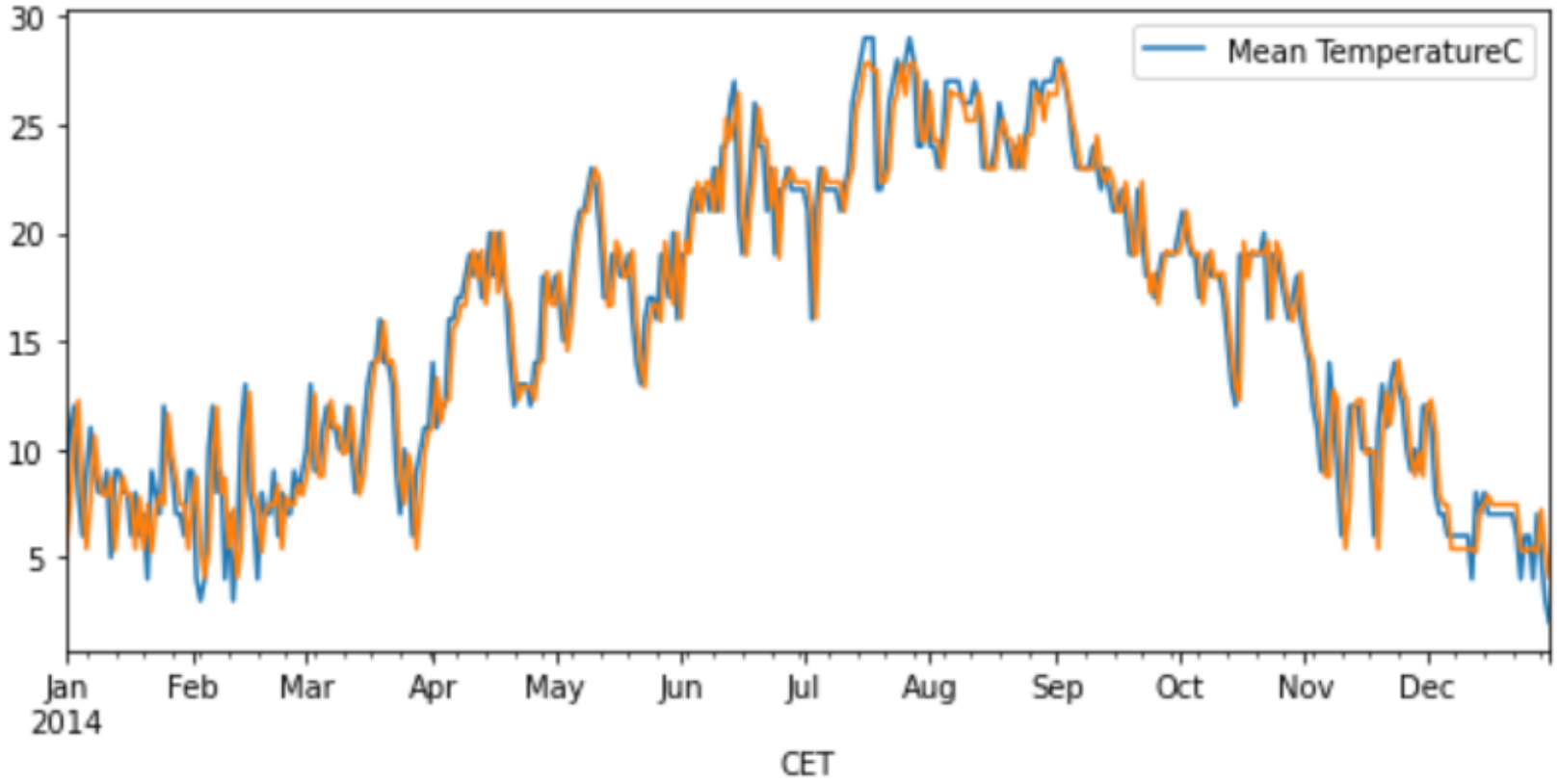
Зробимо підбір гіперпараметрів:

```
search_space = {'max_features': ['auto', 'sqrt', 'log2'],'max_depth' : [i
for i in range(5,15)]}
RF=RandomForestRegressor()
tscv = TimeSeriesSplit(n_splits=8)
gs = GridSearchCV(RF, cv=tscv, param_grid=search_space, scoring='r2')
gs.fit(X_train, y_train)
gs.best_score_
```

```
0.9308689868805313
```

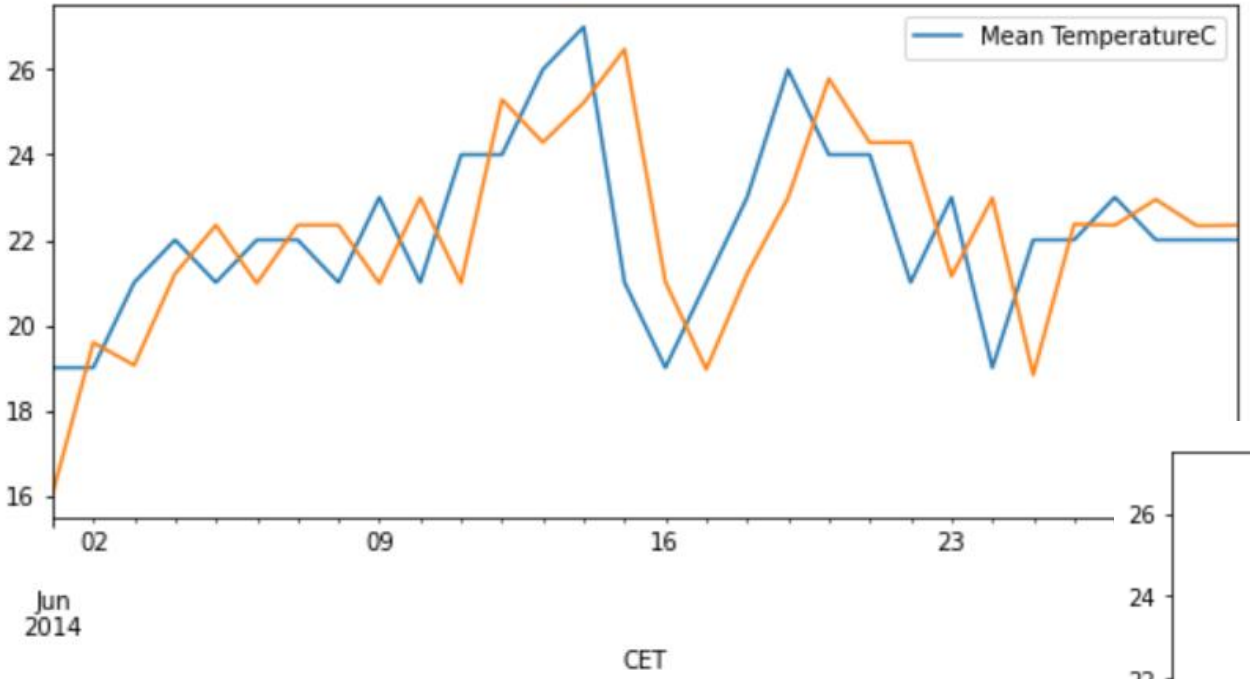
Машинне навчання з scikit-learn

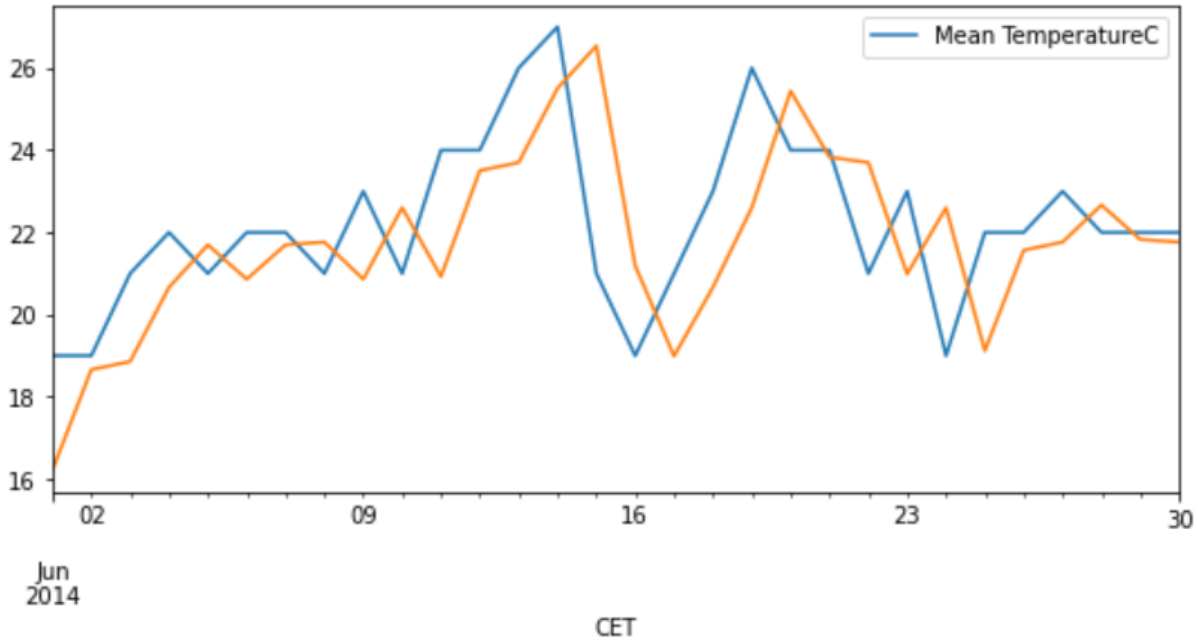
Отриманий в результаті графік:



Машинне навчання з scikit-learn

За червень в порівнянні з лінійною регресією:





Машинне навчання з scikit-learn

Виявлення аномалій в даних включає дві задачі:

Виявлення викидів - навчальні дані містять викиди, які визначаються як спостереження, далекі від інших. Таким чином, моделі для виявлення викидів намагаються підібрати регіони, де навчальні дані є найбільш сконцентрованими, ігноруючи спостереження, що сильно відхиляються.

Виявлення аномалій в нових даних - навчальні дані не забруднені викидами і модель повинна виявити, чи є нове спостереження викидом.

Машинне навчання з scikit-learn

Одним з алгоритмів для виявлення аномалій є ізоляційний ліс. Він повертає масив значень, де -1 означає викид, 1 – звичайне спостереження.

```
from sklearn.ensemble import IsolationForest
isf =  IsolationForest(random_state=0)
isf.fit(weather[['Mean TemperatureC']])
result=isf.predict(weather[['Mean TemperatureC’]])
```

```
array([-1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1, -1, -1, -1, -1,
       -1, -1, -1, -1, -1, -1, -1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1, -1,  1, -1,
        1,  1,  1, -1, -1, -1,  1,  1,  1, -1,  1, -1, -1, -1, -1, -1,  1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, -1, -1,  1,
        1,  1, -1,  1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1,  1,  1, -1,
        1, -1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
```

Машинне навчання з scikit-learn

Виведемо середню температуру разом з максимальною в тих спостереженнях, що були класифіковані як викиди:

```
weather_n[weather_n.outliers==-1]
```

CET	Max TemperatureC	Mean TemperatureC	outliers
1997-01-02	7.0	3.0	-1
1997-06-01	22.0	18.0	-1
1997-07-09	28.0	23.0	-1
1997-07-15	33.0	24.0	-1
1997-07-22	30.0	24.0	-1

Машинне навчання з scikit-learn

Візьмемо 10 найбільших значень:
weather_n[weather_n.outliers==-1].nlargest(10,columns='Mean TemperatureC’)

CET	Max TemperatureC	Mean TemperatureC	outliers
2008-06-23	31.0	45.0	-1
2011-04-30	16.0	30.0	-1
1997-08-23	35.0	29.0	-1
2002-06-26	36.0	29.0	-1
2005-06-20	36.0	29.0	-1
2010-07-08	36.0	29.0	-1
2011-08-21	37.0	29.0	-1
2013-08-12	37.0	29.0	-1
2013-08-16	37.0	29.0	-1
2015-07-22	35.0	29.0	-1

Машинне навчання з scikit-learn

```
isf =  IsolationForest(random_state=0,contamination=0.01)
isf.fit(weather[['Mean TemperatureC']])
result=isf.predict(weather[['Mean TemperatureC']])
weather_n[weather_n.outliers==-1].nlargest(10,columns='Mean TemperatureC’)
```

CET	Max TemperatureC	Mean TemperatureC	outliers
2008-06-23	31.0	45.0	-1
2011-04-30	16.0	30.0	-1
2008-11-28	6.0	0.0	-1
2010-01-09	2.0	0.0	-1
2009-01-13	2.0	-1.0	-1

Машинне навчання з scikit-learn

Розглянемо іншу задачу визначення аномалій.

datetime	source_ip	username	success	failure_reason
2018-01-01 00:05:32.988414	223.178.55.3	djones	True	NaN
2018-01-01 00:08:00.343636	223.178.55.3	djones	False	error_wrong_password
2018-01-01 00:08:01.343636	223.178.55.3	djones	True	NaN
2018-01-01 01:06:59.640823	208.101.11.88	wbrown	True	NaN
2018-01-01 02:40:47.769630	11.76.99.35	tkim	True	NaN
...	...	...	...	...
2018-12-31 21:13:21.846800	150.64.165.159	pkim	True	NaN
2018-12-31 21:29:34.632700	204.182.96.183	pkim	True	NaN
2018-12-31 22:05:56.299770	110.235.65.232	ybrown	True	NaN
2018-12-31 23:29:41.482166	68.150.183.142	rkim	False	error_wrong_password
2018-12-31 23:29:42.482166	68.150.183.142	rkim	True	NaN

Машинне навчання з scikit-learn

log_data.info()

```
DatetimeIndex: 38700 entries, 2018-01-01 00:05:32.988414 to 2018-12-31 23:29:42.482166
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   source_ip       38700 non-null  object
 1   username        38700 non-null  object
 2   success         38700 non-null  bool
 3   failure_reason  11368 non-null  object
```

log_data.describe(include='all')

	source_ip	username	success	failure_reason
count	38700	38700	38700	11368
unique	4956	1797	2	3
top	168.123.156.81	wlopez	True	error_wrong_password
freq	314	387	27332	6646

Машинне навчання з scikit-learn

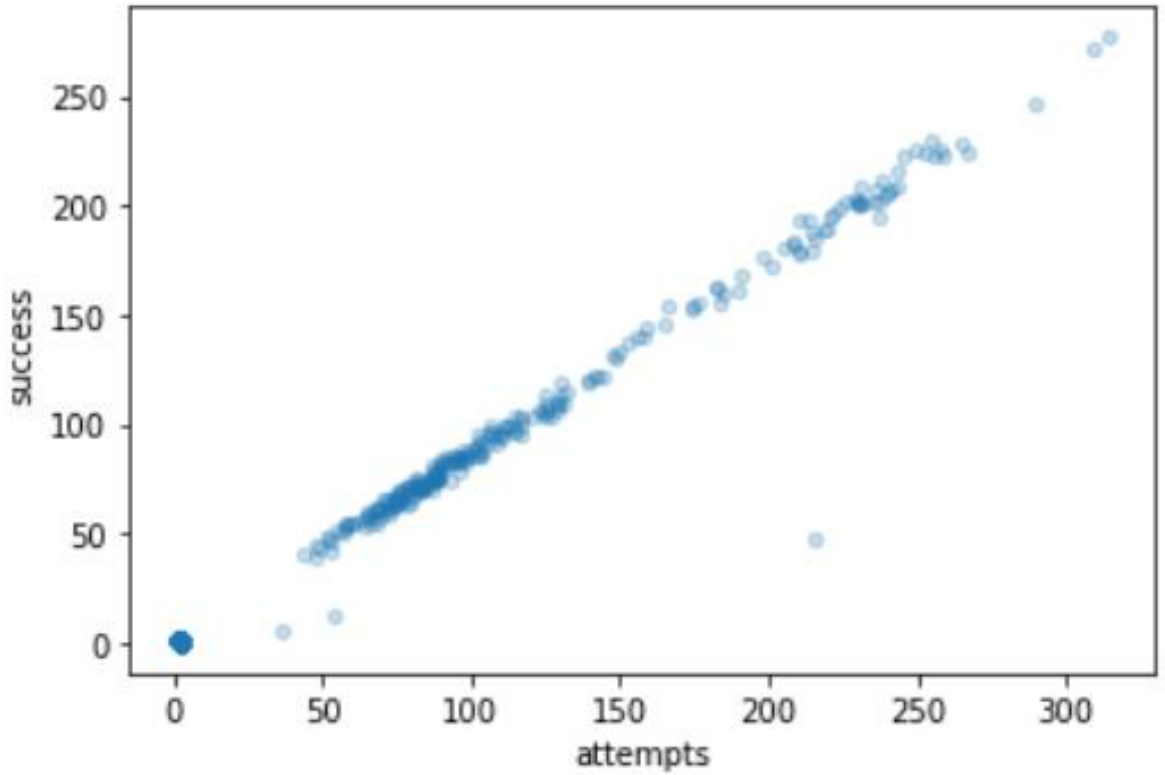
log_data.groupby('source_ip').agg(dict(username='nunique')).username.describe()

```
count    4956.000000
mean        1.146287
std         1.916782
min         1.000000
25%         1.000000
50%         1.000000
75%         1.000000
max       129.000000
```

failure_reason	attempts	error_account_locked	error_wrong_password	error_wrong_username	success	success_rate	error_rate
source_ip							
168.123.156.81	314	0	37	0	277	0.882166	0.117834
24.112.17.125	309	0	37	0	272	0.880259	0.119741
16.118.156.50	289	0	41	1	247	0.854671	0.145329
25.246.225.197	267	0	43	0	224	0.838951	0.161049
30.67.241.95	265	0	37	0	228	0.860377	0.139623

Машинне навчання з scikit-learn

pivot.plot(kind='scatter', x='attempts', y='success',alpha=0.25);



Машинне навчання з scikit-learn

Виділимо наступні дані:

datetime	2018-01-01 00:08:00	2018-01-01 00:09:00	2018-01-01 00:10:00	2018-01-01 00:11:00	2018-01-01 00:12:00	2018-01-01 00:13:00	2018-01-01 00:14:00	2018-01-01 00:15:00	2018-01-01 00:16:00	2018-01-01 00:17:00	...	2018-01-31 22:35:00	2018-01-31 22:36:00	2018-01-31 22:37:00	2018 22:38
usernames_with_failures	1	0	0	0	0	0	0	0	0	0	...	0	0	0	
day_of_week_0	1	1	1	1	1	1	1	1	1	1	...	0	0	0	
day_of_week_1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
day_of_week_2	0	0	0	0	0	0	0	0	0	0	...	1	1	1	
day_of_week_3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
day_of_week_4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
day_of_week_5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
day_of_week_6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	
hour_0	1	1	1	1	1	1	1	1	1	1	...	0	0	0	

Машинне навчання з scikit-learn

Алгоритм ізоляційного лісу використовує методи розщеплення, щоб ізолювати викиди від решти даних; тому його можна використовувати для виявлення аномалій. Він використовує випадковий ліс, де розгалуження відбуваються за випадковими ознаками. Для поділу вибирається випадкове значення ознаки між її максимумом і мінімумом.

```
from sklearn.ensemble import IsolationForest
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
iso_forest_pipeline = Pipeline([('scale', StandardScaler()),('iforest', IsolationForest(random_state=0, contamination=0.05))]).fit(X)
```

Машинне навчання з scikit-learn

Параметр contamination визначає, яку частку даних очікувано займають викиди. Також можливо провести статистичний аналіз даних, щоб задати цей параметр.
Метод predict() можна використовувати, щоб перевірити, чи є точка даних викидом. Алгоритми виявлення аномалій, реалізовані в scikit-learn, зазвичай повертають 1, якщо точка виглядає нормальною або -1, якщо точка є аномалією.

```
isolation_forest_preds = iso_forest_pipeline.predict(X)
pd.Series(np.where(isolation_forest_preds == -1, 'outlier',
'inlier')).value_counts()
```

```
inlier     42556
outlier     2001
```

Машинне навчання з scikit-learn

У той час як нормальні елементи зазвичай розташовуються в більш щільних областях набору даних, викиди, як правило, розташовуються в більш ізольованих областях з кількома сусідніми точками. Алгоритм локального коефіцієнта викидів (LOF) шукає ці малонаселені області, щоб ідентифікувати викиди. Він оцінює всі точки на основі відношення щільності навколо кожної точки до щільності її найближчих сусідів. Точки, які вважаються нормальними, матимуть ту ж щільність, що й їхні сусіди; ті, у кого поблизу небагато інших точок, вважатимуться аномаліями.

Машинне навчання з scikit-learn

```
from sklearn.neighbors import LocalOutlierFactor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
lof_pipeline = Pipeline([('scale', StandardScaler()),('lof',
LocalOutlierFactor())]).fit(X)
```

Клас LocalOutlierFactor не має методу predict(), то ж використовуємо атрибут negative_outlier_factor_.

```
lof_preds = lof_pipeline.named_steps['lof'].negative_outlier_factor_
```

```
array([-1.33898756e+10, -1.00000000e+00, -1.00000000e+00, ...,
       -1.00000000e+00, -1.00000000e+00, -1.11582297e+10])
```

Машинне навчання з scikit-learn

Тут неможливо прямо підрахувати кількість викидів, тому потрібно порівняти атрибут negative_outlier_factor_ з атрибутом offset_ моделі LOF, який повідомляє граничне значення, визначене моделлю LOF під час навчання.

```
pd.Series(np.where(lof_preds < 
lof_pipeline.named_steps['lof'].offset_,'outlier', 'inlier')).value_counts()
```

```
inlier     44248
outlier      309
```

Машинне навчання з scikit-learn

Використаємо метрику cohen_kappa_score(), щоб визначити, наскільки узгоджені дві моделі:

```
from sklearn.metrics import cohen_kappa_score
is_lof_outlier = np.where(lof_preds <
lof_pipeline.named_steps['lof'].offset_,'outlier', 'inlier')
is_iso_outlier = np.where(isolation_forest_preds == -1, 'outlier', 'inlier')
cohen_kappa_score(is_lof_outlier, is_iso_outlier)
```

```
0.2586251797335677
```

Отже, моделі мають низький рівень узгодження, тобто зовсім неочевидно, які саме точки є аномальними.