Задание Hoff

Неделя 1

In [27]:

```
import pandas as pd
import numpy as np
import scipy.stats as sps
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import numba
from tqdm.notebook import tqdm

sns.set(font_scale=1.4)
```

Загрузим данные:

In [3]:

```
data = pd.read_csv('../data/raw/train.csv', sep='\t')
data.head()
```

Out[3]:

	TRANSACTIONID	ITEMID	PRICE	TRANSDATE
0	6	80275632.0	149.0	2016-02-26
1	10	80275632.0	149.0	2016-02-26
2	12	80088007.0	8.0	2016-02-26
3	13	80088007.0	8.0	2016-02-26
4	14	80088007.0	8.0	2016-02-26

Посмотрим, сколько есть значений, не снабженных ITEMID

In [4]:

```
data.count(), len(data)
```

Out[4]:

```
(TRANSACTIONID    6398799
 ITEMID           6388960
 PRICE            6398799
 TRANSDATE        6398799
 dtype: int64,
 6398799)
```

Удалим значения NULL строках и отсортируем данные по дате:

In [5]:

```
data = data.dropna()
data = data.sort_values('TRANSDATE')
```

Выберем случайный товар и посмотрим на временной ряд:

In [6]:

```
random_items = np.random.choice(data['ITEMID'], 10)

def get_random_item(item_id):
    unique_item = data[data['ITEMID'] == item_id]
    return unique_item['TRANSDATE'], unique_item['PRICE']
```

Построим график зависимости от цены от времени:

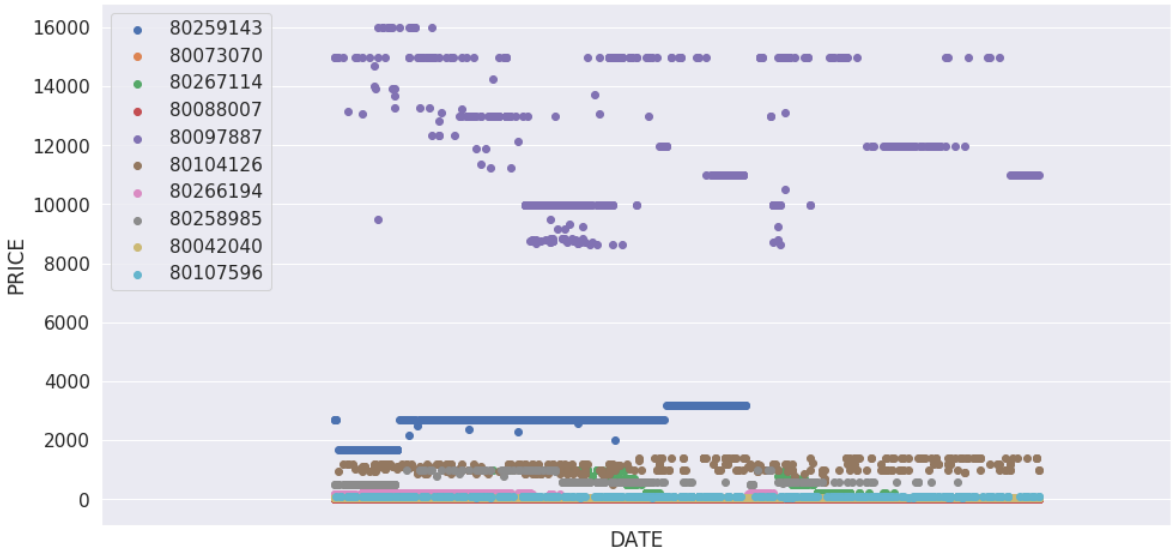
In [7]:

```
plt.figure(figsize=(16, 8))

for item in random_items:
    x, y = get_random_item(item)
    plt.scatter(x, y, label=int(item))

plt.xlabel('DATE')
plt.ylabel('PRICE')
plt.xlim(-100, 360)
plt.xticks([])
plt.legend()

plt.show()
```

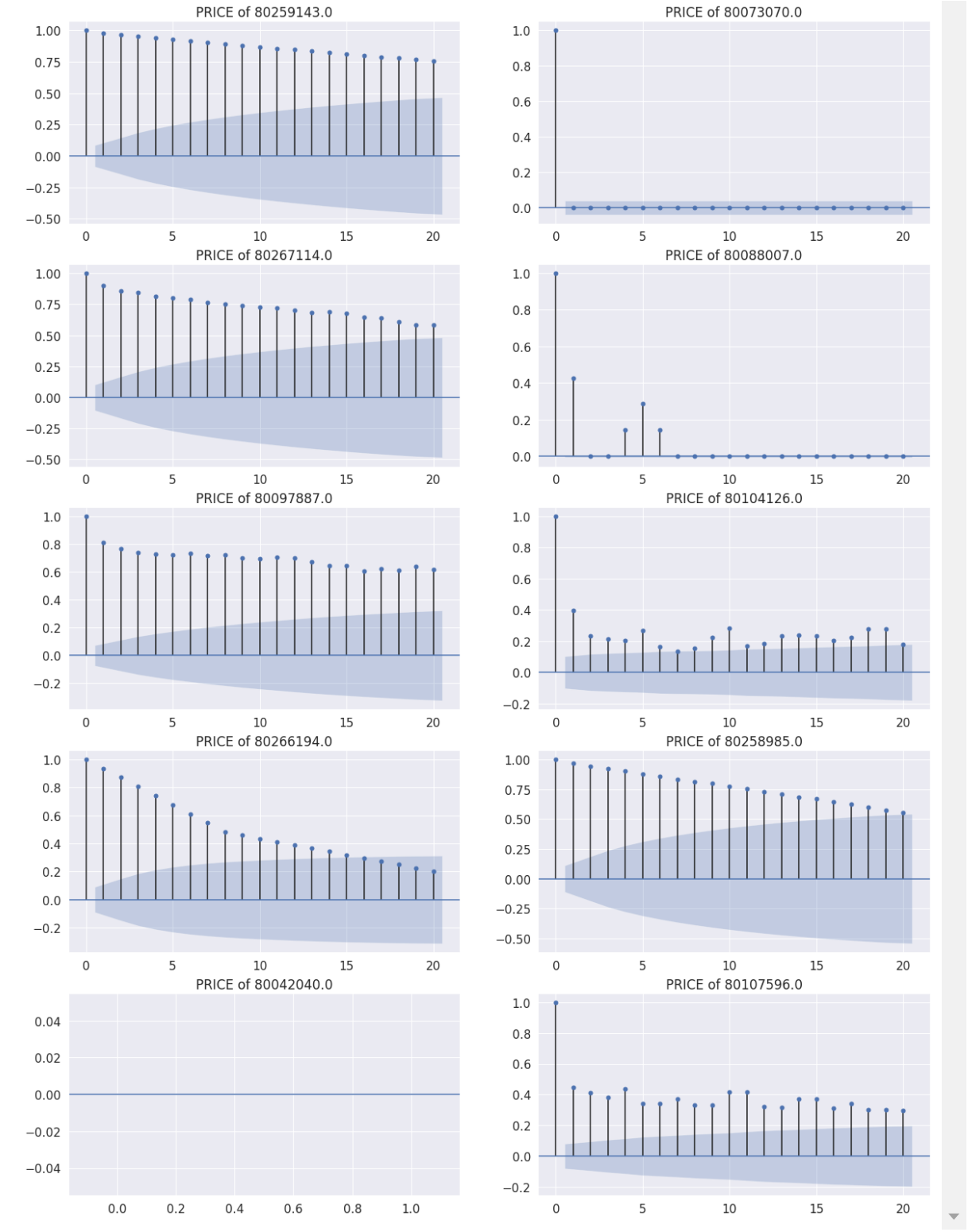


Видно, что временные ряды ведут себя очень по-разному. И, главное, каждый из них плохо прогнозируется моделью SARIMA. Кажется, она мало сюда подходит. Более того, в большинстве случае автокорреляционная функция близка к нулю. Проверим это:

In [8]:

```
fig = plt.figure(figsize=(20, 28))
for i, item in enumerate(random_items):
    ax = fig.add_subplot(5, 2, i+1)
    x, y = get_random_item(item)
    fig = sm.graphics.tsa.plot_acf(y, lags=20, ax=ax, title=f'PRICE of {item}')
plt.show()
```

```
/home/fedor/anaconda3/envs/mipt-stats/lib/python3.7/site-packages/stat
smodels/tsa/stattools.py:578: RuntimeWarning: invalid value encountere
d in true_divide
  acf = avf[:nlags + 1] / avf[0]
```



Видно, что цена может вести себя очень по-разному в зависимости от товара. Посмотрим, есть кластеризация и зависит ли вид цены от нее.

Кластеризация

Попробуем понять, что хранят в себе чеки. Сделаем `crosstab`, случайным образом выберя 500 товаров (для всех не хватает памяти)

In [32]:

```
random_items = np.unique(np.random.choice(data['ITEMID'], 500).astype(int))
random_items.size
```

Out[32]:

423

In [33]:

```
indices = np.zeros(len(data), dtype=np.bool)

for i in range(len(random_items)):
    indices |= (data['ITEMID'] == random_items[i])

chosen_data = data[indices].reset_index(drop=True)
chosen_data.head()
```

Out[33]:

	TRANSACTIONID	ITEMID	PRICE	TRANSDATE
0	2743	80090775.0	5.0	2015-08-01
1	2744	80075990.0	19.0	2015-08-01
2	2744	80075990.0	19.0	2015-08-01
3	2744	80090775.0	5.0	2015-08-01
4	2747	80071144.0	499.0	2015-08-01

Посмотрим на товары в одном чеке:

In [105]:

```
chosen_data[(chosen_data['TRANSACTIONID'] == 2743) &
            (chosen_data['TRANSDATE'] == '2015-08-01')]
```

Out[105]:

	TRANSACTIONID	ITEMID	PRICE	TRANSDATE
0	2743	80090775.0	5.0	2015-08-01
3	2743	80099433.0	149.0	2015-08-01
6	2743	80099433.0	149.0	2015-08-01
7	2743	80101869.0	39.0	2015-08-01

Хотя мы и знаем о существовании `pd.crosstab` , но, не тратя времени на тонскости (самый простой вариант попытаться агрегировать данные с поомщью `pd.crosstab(index=chosen_data['ITEMID'], columns=chosen_data['ITEMID'], values=zip(chosen_data['TRANSACTIONID'], chosen_data['TRANSDATE']), aggfunc='count')` , но он выдает только значения на диагонали, а занчит, где-то ошибка, так что мы посчитаем все в ручную.

In [21]:

```
chosen_data.sort_values(by=['TRANSDATE', 'TRANSACTIONID'], inplace=True)
chosen_data
```

Out[21]:

	TRANSACTIONID	ITEMID	PRICE	TRANSDATE
1428	42	20000189.0	1290.0	2015-08-01
1427	45	20000190.0	1590.0	2015-08-01
1075	116	20000065.0	0.1	2015-08-01
1076	120	20000246.0	400.0	2015-08-01
1077	125	20000246.0	400.0	2015-08-01
...	...	...	...	...
1336667	254017	80007846.0	89.0	2016-05-31
1336668	254017	80007846.0	89.0	2016-05-31
1336671	254019	80263147.0	269.0	2016-05-31
1336670	254025	80260285.0	79.0	2016-05-31
1336688	254035	20000006.0	14495.0	2016-05-31

1337575 rows × 4 columns

In [35]:

```
item2index = dict()
for i in range(len(random_items)):
    item2index[random_items[i]] = i
```

In [72]:

```
sorted_data = np.array(chosen_data)

def make_crosstab(sorted_data, item2index):
    crosstab = np.zeros((len(item2index), len(item2index)), dtype=np.int)
    current_key = (None, None)
    check_list = []

    for row in tqdm(sorted_data):
        if current_key != (row[0], row[-1]):
            check_list = []
            current_key = (row[0], row[-1])
        check_list.append(row[1])
        for item in check_list:
            crosstab[item2index[item], item2index[row[1]]] += 1
            crosstab[item2index[row[1]], item2index[item]] += 1
    return crosstab

cross_tab = make_crosstab(sorted_data, item2index)
```

100% 1326760/1326760 [01:42<00:00, 12998.38it/s]

In [57]:

```
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from scipy.spatial.distance import squareform
```

Воспользуемся методом иерархической кластеризации:

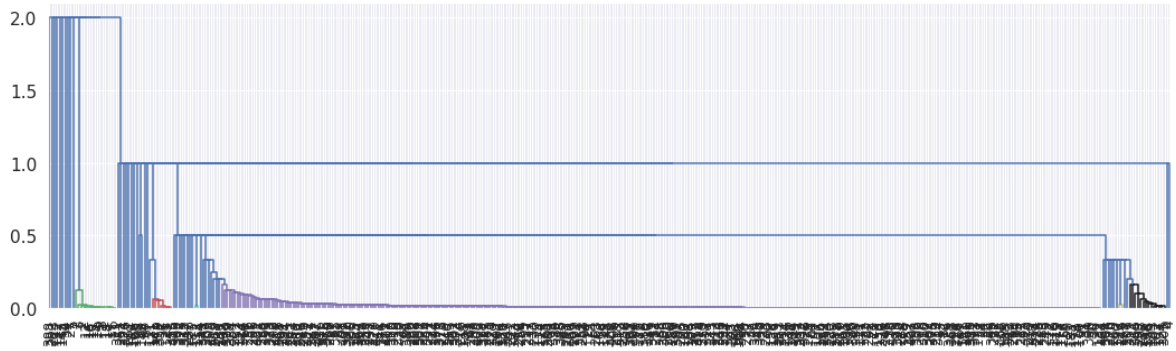
In [63]:

```
dist_matrix = 1 / cross_tab
dist_matrix[np.arange(len(cross_tab)), np.arange(len(cross_tab))] = 0
dist_matrix[dist_matrix == np.inf] = 2

# Определение вспомогательных объектов для кластеризации
dists = squareform(dist_matrix)
linkage_matrix = linkage(dists, "single")

# Отрисовка дентрограммы
plt.figure(figsize=(18, 5))
dendrogram_result = dendrogram(linkage_matrix, orientation='top',
                               color_threshold=0.2, distance_sort='descending',
                               show_leaf_counts=True)
plt.xticks(fontsize=12)
plt.show()
```

```
/home/fedor/anaconda3/envs/mipt-stats/lib/python3.7/site-packages/ipyk
ernel_launcher.py:1: RuntimeWarning: divide by zero encountered in tru
e_divide
  """Entry point for launching an IPython kernel.
```

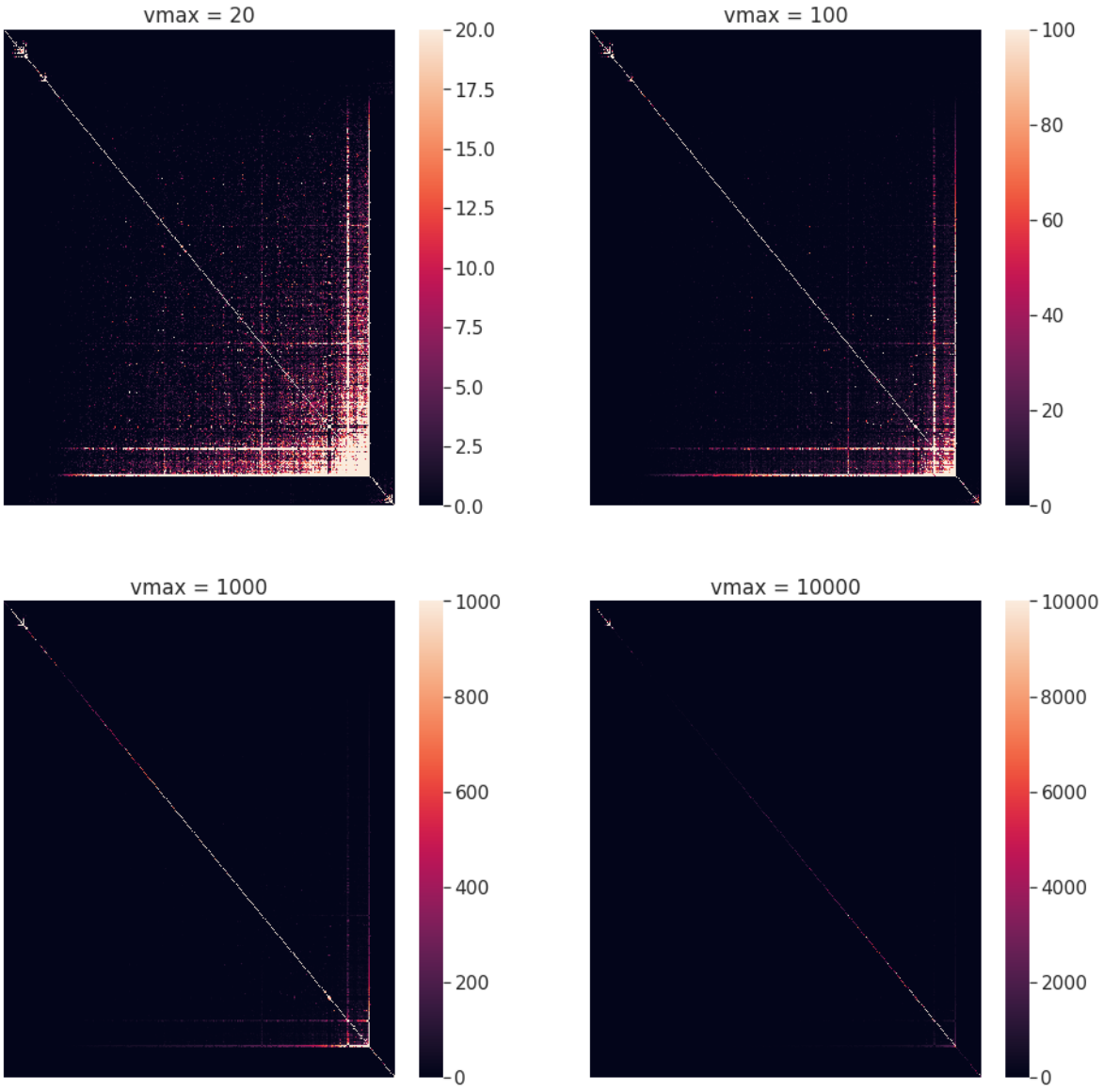


In [73]:

```
index = [int(x) for x in dendrogram_result['ivl']]
cross_tab = cross_tab[:, index][index, :]
```

In [88]:

```
plt.figure(figsize=(16, 16))
for i, vmax in enumerate([20, 100, 1000, 10000]):
    plt.subplot(2, 2, i+1)
    sns.heatmap(cross_tab, vmax=vmax)
    plt.axis('Off')
    plt.title(f'vmax = {vmax}')
plt.show()
```



In [87]:

```
# Печать самих кластеров
clusters = fcluster(linkage_matrix, 0.2, criterion='distance')
elements, repeats = np.unique(clusters, return_counts=True)
for cl in elements[repeats > 1]:
    print((clusters == cl).sum(), 'elements:', list(random_items[clusters == cl
    print()
```

```
16 elements: [20000065, 20000094, 20000096, 20000165, 20000188, 200001
89, 20000190, 20000192, 20000241, 20000247, 20000254, 20000255, 200002
84, 20000288, 20000362, 20000368]

8 elements: [80035102, 80056630, 80087882, 80088221, 80094380, 8026864
2, 80268650, 80271397]

2 elements: [80085677, 80085683]

335 elements: [80001219, 80007817, 80007829, 80007846, 80007860, 80009
015, 80009017, 80009041, 80015461, 80018233, 80018235, 80018236, 80018
247, 80020112, 80020955, 80021667, 80021669, 80024353, 80024372, 80030
105, 80035453, 80036042, 80039090, 80042063, 80042930, 80045771, 80045
853, 80046112, 80047332, 80049067, 80054208, 80057276, 80058746, 80059
963, 80060781, 80064011, 80064287, 80064536, 80064651, 80064657, 80064
661, 80065117, 80066178, 80066640, 80067440, 80067841, 80068012, 80068
340, 80068493, 80071143, 80071144, 80071391, 80071698, 80071765, 80071
852, 80072073, 80072578, 80073068, 80073072, 80073277, 80074136, 80074
362, 80074450, 80075894, 80075990, 80076952, 80077231, 80078520, 80078
920, 80080192, 80081126, 80081369, 80082093, 80082354, 80083905, 80084
305, 80084323, 80084767, 80084772, 80084775, 80084932, 80085231, 80085
782, 80085834, 80085838, 80086049, 80086064, 80086288, 80086473, 80088
007, 80088717, 80088982, 80089525, 80089570, 80089625, 80090775, 80091
036, 80091066, 80092050, 80092063, 80093084, 80093375, 80093922, 80095
406, 80095974, 80096036, 80096602, 80097204, 80097595, 80097611, 80097
748, 80097864, 80098166, 80098375, 80098720, 80098915, 80099507, 80099
589, 80099591, 80099606, 80099611, 80099953, 80100070, 80100350, 80100
672, 80100765, 80100848, 80100859, 80100876, 80100879, 80101363, 80101
583, 80101584, 80101585, 80101586, 80101870, 80102228, 80102335, 80102
357, 80102740, 80102939, 80103078, 80103110, 80103457, 80103703, 80104
208, 80104212, 80104213, 80104559, 80104583, 80104795, 80105190, 80105
191, 80105423, 80105662, 80105970, 80106283, 80106615, 80106918, 80106
932, 80107496, 80107578, 80107603, 80107818, 80108118, 80108164, 80108
212, 80108256, 80108257, 80108354, 80108398, 80108445, 80108447, 80108
553, 80257038, 80257323, 80257327, 80257328, 80257424, 80257625, 80257
861, 80257889, 80258088, 80258098, 80258263, 80258274, 80258300, 80258
805, 80258834, 80258894, 80259120, 80259121, 80259143, 80259345, 80259
355, 80259426, 80259617, 80259669, 80259694, 80259996, 80260015, 80260
399, 80260400, 80260468, 80260662, 80260908, 80260911, 80260940, 80261
020, 80261266, 80261562, 80261618, 80261724, 80262317, 80262319, 80262
357, 80262408, 80262623, 80262701, 80262793, 80262926, 80262999, 80263
014, 80263189, 80263342, 80264570, 80264833, 80264843, 80264996, 80265
096, 80265221, 80265380, 80265558, 80265602, 80265604, 80265727, 80265
991, 80265993, 80266004, 80266025, 80266120, 80266253, 80266254, 80266
281, 80266357, 80266490, 80266647, 80266801, 80266949, 80267070, 80267
238, 80267244, 80267264, 80267292, 80267320, 80267345, 80267588, 80267
744, 80267784, 80268042, 80268045, 80268047, 80268049, 80268063, 80268
319, 80268327, 80268342, 80268366, 80268370, 80268381, 80268391, 80268
396, 80268416, 80268447, 80268786, 80268789, 80268836, 80268842, 80268
924, 80269352, 80269450, 80269613, 80269689, 80269817, 80269856, 80269
960, 80270073, 80270137, 80270490, 80270494, 80270626, 80270627, 80270
630, 80270631, 80270787, 80270896, 80271223, 80271451, 80271514, 80271
```

```
556, 80271565, 80271643, 80272368, 80272484, 80272717, 80272747, 80272
767, 80273018, 80273535, 80273542, 80273546, 80273664, 80273835, 80273
876, 80274021, 80274036, 80274151, 80274540, 80275013, 80275307, 80275
464, 80275477, 80275568, 80275644, 80275716, 80276152, 80276465, 80276
469, 80276822, 80277226, 80277599, 80278030, 80278328, 80279890, 80279
920]

2 elements: [80088042, 80258338]

15 elements: [80082177, 80088044, 80093377, 80104390, 80104406, 801044
07, 80104413, 80105538, 80106164, 80257737, 80263511, 80265467, 802655
16, 80269191, 80269861]
```

Видно наличие кластеризации. Следующий этап -- проверить, меняется ли в зависимости от кластера временной ряд.