```
import numpy as np
import pandas as pd
import optuna
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_auc_score
from lightgbm import LGBMClassifier

# Настройки pandas
pd.set_option('display.max_rows', 100)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.max_columns', None)

# Отображение в Jupyter
from IPython.display import display
%matplotlib inline
```

Большая часть идей для оформления графиков (если график оформляется нетривиальным образом) взята из моих же ноутбуков (в частности, https://github.com/vladimirrsmirnovv/MOEX_research_paper/tree/main)

Загрузим данные

- Данные упорядочены по времени
- Задача - предсказать `target`
- Фичи - `feature_i`
- Трейн, валидация и тест уже определены (см. колонку `sample_part`)

Вопрос

Указано, что признаки - feature_i. А почему нельзя использовать даты в качестве признаков сразу? Попробую данный подход уже в "творческой" части задания.

```
df = pd.read_parquet('df.parquet')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500000 entries, 22620 to 310653
Columns: 235 entries, date to sample_part
dtypes: category(2), datetime64[ms](1), datetime64[ns](2),
float64(220), int64(9), object(1)
memory usage: 893.6+ MB
```

```
df.head()
```

```
             date      month    quarter  feature_217  feature_66
feature_9  \
```

```
22620  2021-01-01 2021-01-01 2021-01-01        1.181       0.483
13.977
478621 2021-01-01 2021-01-01 2021-01-01        3.428       0.887
27.158
372254 2021-01-01 2021-01-01 2021-01-01      -38.555       1.227
19.894
2596   2021-01-01 2021-01-01 2021-01-01      -14.667       0.753
18.203
216892 2021-01-01 2021-01-01 2021-01-01        6.735       1.129
11.246

        feature_193  feature_15  feature_199  feature_25  feature_195
\
22620         1.135      43.272      -46.889      97.558     -173.966

478621       -0.379      80.088      -67.243     105.803     -100.820

372254        0.499      98.464      -67.403      81.858      -83.455

2596         -1.200     117.331      -25.558      90.277      -99.937

216892       -0.109      61.206      -78.397      82.637      -98.028


        feature_154  feature_126  feature_218  feature_19  feature_96
\
22620         0.355      -61.496        0.000      95.060     116.886

478621       11.771      -31.562        0.000     167.121     103.470

372254      -17.036      -69.425        0.000     157.788     122.015

2596         12.579      -12.452        0.000     189.285      57.268

216892       14.798      -96.745        0.000     163.964     127.391


        feature_177  feature_223  feature_137  feature_146  feature_30
\
22620        23.965       38.981      -77.860        1.523       0.037

478621       19.821      190.191     -102.354       12.880       0.060

372254       43.998          NaN     -110.378        6.746       0.084

2596         31.321      100.130     -112.599       -1.449       0.021

216892       50.016       53.473      -60.523       18.961       0.090


        feature_225  feature_184  feature_108  feature_162
```

```
feature_204  \
22620        12.506      -68.489      -15.652       42.140      -
29.446
478621     -107.082      -74.598      -38.710       29.498      -
27.851
372254       -2.640      -35.450      -18.685       40.838       -
5.667
2596         28.147      -21.772      -28.991       25.363     -
103.964
216892      -55.495      -15.294      -11.310       29.490      -
68.760

         feature_180  feature_125  feature_18  feature_147  feature_101
\
22620        33.369      -49.226      15.865            1       18.338

478621        9.449        4.178      14.934            1       32.957

372254        8.319      -11.541      28.788            0        0.987

2596         18.185       24.168      12.479            0        9.529

216892       25.419       56.772       7.208            1       -6.332


         feature_104  feature_84  feature_197  feature_60  feature_118
\
22620        16.955      91.781       34.180    -116.365      130.684

478621        5.784     137.566       28.170    -158.773      151.633

372254       -0.081     145.124       18.497     -93.969       53.913

2596         10.354     187.479       12.969    -169.816      111.911

216892        0.157     182.656        0.818    -148.135      169.084


         feature_45  feature_16  feature_127  feature_159  feature_119
\
22620        4.627     187.238       48.935        2.829      -83.383

478621       8.406      76.368       89.374      -24.589      -52.683

372254       6.785     115.597       66.472      -12.866      -68.470

2596         9.832     177.790       93.701       -8.853      -54.264

216892       6.027      76.512       87.059      -22.558      -67.290
```

```
        feature_82  feature_144  feature_93  feature_168  feature_143
\
22620      -17.192            1     -77.919            1     -134.605
478621     -24.921            1    -126.716            0      -88.918
372254     -20.369            1     -53.191            1      -76.857
2596        -7.155            1     -79.669            0     -148.433
216892      14.523            1     -88.054            1      -89.784

        feature_73  feature_113  feature_112  feature_150  feature_55
\
22620      187.476        0.189        0.143      -75.169     -13.793
478621     168.039       -4.244        0.073      -48.910      -2.601
372254     101.012       -1.508        0.179      -36.070      -5.636
2596       182.624       -0.776        0.156     -116.707      17.692
216892      75.010       -1.300        0.239      -44.794      18.671

        feature_165  feature_69  feature_167  feature_109  feature_124
\
22620             1     -87.365      141.101      -20.604      -61.495
478621            1     -70.292      159.616      -25.524      -28.477
372254            1     -88.989      186.244      -30.064          NaN
2596              1    -100.694      137.007      -37.357      -42.118
216892            0     -87.919      157.030      -38.641      -48.599

        feature_79  feature_86  feature_95  feature_213
feature_117  \
22620      -99.060       3.628    -132.258        7.458       -17.227
478621    -131.036      15.079    -187.867       10.885        -8.564
372254     -77.346       6.276    -211.018       17.206       -10.249
2596      -133.031     -12.070    -102.541       18.618       -13.115
216892    -224.086      14.581    -133.384        8.274        -7.378
```

```
        feature_38  feature_24  feature_67  feature_185
feature_198  \
22620       81.614      -7.180      83.035      121.590       106.620

478621      64.992     -23.798      44.081       42.190       103.292

372254      45.299     -22.977      35.251       63.276        53.121

2596        41.408     -14.088      24.073       77.043       101.308

216892     104.813     -17.196      44.252       71.059        93.234


        feature_10  feature_120  feature_29  feature_153
feature_68  \
22620          NaN       65.210      -0.317      -55.940     -67.237

478621       0.000       59.559       1.325      -49.355    -103.629

372254       1.000       95.916       0.998     -115.400          NaN

2596         1.000       49.346      -1.149     -110.864    -108.854

216892         NaN       22.010      -1.553     -110.317     -73.858


        feature_75  feature_5  feature_56  feature_114  feature_139  \
22620      -37.566   -215.946    -258.725      -37.069      -17.930
478621     -51.686   -170.714     -75.131      -19.271      -37.153
372254     -32.604   -154.503    -270.127      -22.637        1.675
2596       -45.033   -131.553    -463.492       -4.883      -24.975
216892     -58.049   -162.093     327.178       14.696        8.972

        feature_186  feature_65  feature_142  feature_194  feature_136
\
22620        -2.311     163.888       43.686       15.095       66.272

478621     -62.370     -24.941      154.849       -8.353       11.989

372254       80.467     -55.034      305.262        7.529          NaN

2596         59.383     -88.956       64.488       12.097     -106.539

216892    -148.090     -30.291       88.845        8.704      221.844


        feature_31  feature_12  feature_35  feature_28  feature_42  \
22620            2     327.804     -10.316    -147.398     442.757
478621           0     145.239      -2.953     -10.632    -319.719
372254           0      -0.711      18.446    -155.017    -144.521
```

```
2596              0      99.609      22.967    -181.545     -45.809
216892            0     -29.085       1.828     -71.107     112.515

        feature_227  feature_115  feature_155  feature_51  feature_132
\
22620      -173.334     -209.629      -24.298     -50.931      -51.036

478621     -479.616     -113.581       10.729      50.032      -12.897

372254     -222.994     -282.365      -11.249       4.908      -42.238

2596        312.396     -143.675      -58.606     -13.816      -21.098

216892     -201.997      297.919       68.831      -7.798      -21.927


        feature_182  feature_76  feature_41  feature_97
feature_140  \
22620       -67.803      24.512     -26.470      -0.420        15.000

478621       -1.614    -131.929     -12.022     124.632        28.000

372254       -61.911    -130.156     -48.438     -69.458        10.000

2596         -87.202      20.597     -45.001     105.278        22.000

216892     -193.628     178.125     -24.746      36.812        50.000


        feature_78  feature_135  feature_26  feature_211  feature_171
\
22620       -59.890     -209.048      45.124        5.757      -81.663

478621      -19.073     -110.640      33.406        3.106      -15.415

372254          NaN       88.093    -186.826       -7.539      -96.512

2596          3.804       92.171     -74.749      -14.222      -69.923

216892       27.339      394.331     186.121       46.305       67.471


        feature_158 feature_0  feature_2  feature_77  feature_46
feature_100  \
22620        61.183         c     25.846     216.181     115.320     -
248.700
478621       57.095         d     69.958     198.211     172.981
167.405
372254      -15.850         c     77.523    -295.798    -119.580
44.884
2596         51.884         b      8.587    -107.226      85.009
```

```
58.589
216892       34.363         c    -55.348      -6.044      43.109     -
125.506

        feature_111  feature_138  feature_164  feature_90  feature_85
\
22620      -294.392     -135.357       83.424      76.595       -8.157

478621     -482.424      350.920     -192.861      87.053       -6.965

372254     -247.903      197.881      111.868       4.347        3.490

2596         19.096      203.119      -87.258      90.792      -18.724

216892     -238.630      179.954      -55.945      65.864       12.504


        feature_161  feature_152  feature_98  feature_36  feature_181
\
22620          0.000       -5.454      15.142      64.097       -23.541

478621         0.000       -0.418      10.568    -158.395        19.243

372254           NaN        3.094      15.698      46.128        24.112

2596           0.000       -6.581      -5.818      55.446       -34.486

216892         1.000        3.840      -7.056     -31.857       -76.527


        feature_61  feature_22 feature_209  feature_183  feature_33  \
22620       -8.659     126.982        PRIV     -309.840      61.284
478621     -25.389      29.449         GVM      234.698     -51.806
372254     -22.496     -48.221       UNEMP      -58.677      -9.734
2596       -65.080      -6.673        PRIV      336.292     -50.371
216892     -59.141      40.590       UNEMP      230.065     -87.018

        feature_11  feature_224  feature_228  feature_6  feature_27  \
22620      104.977        0.635      246.898    -22.187     109.827
478621      19.588        1.162     -148.415     -2.435      93.826
372254      44.855        0.863          NaN      2.779      70.812
2596       115.083        1.218      101.575    -28.165     113.366
216892      27.006        1.132       78.013     -2.809      96.061

        feature_141  feature_220  feature_221  feature_156  feature_4
\
22620        14.415      -13.085      -97.530       67.237   -179.752

478621       83.803       30.582     -132.501      -23.582   -158.853

372254       63.379      -18.682     -125.532        5.000   -128.796
```

```
2596         109.776      125.930      -79.832        4.022   -184.373
216892       152.139     -138.231      -71.237        8.348   -161.772

         feature_122  feature_32  feature_163  feature_62  feature_128
\
22620         19.595       83.026      -14.136       0.000       -10.854
478621        -4.892       58.832      -56.234       1.000        -7.450
372254        20.959        9.973          NaN         NaN        -8.373
2596          25.708        8.970      -17.724       0.000        -1.277
216892         5.935      -65.317      -56.179       0.000        -0.979

         feature_206  feature_172  feature_70  feature_175  feature_64
\
22620       -339.389       21.311    -242.062       -28.774       17.826
478621       12.301        75.191      71.929        90.201       65.810
372254       293.458           NaN    -216.702      -209.663       -6.491
2596         100.201        41.056    -229.853      -181.364       19.546
216892       348.117       -54.095    -106.647       -14.576       27.792

         feature_44  feature_148  feature_40  feature_123
feature_23  \
22620       588.375      -167.855      419.900      -151.478      -26.248
478621      832.826      -104.884      134.214       298.246        6.461
372254      786.598           NaN      483.933        97.247      -14.063
2596       1030.431      -165.598      369.273        13.847      -58.286
216892     1225.000      -154.776     -223.218        40.652       60.718

         feature_170  feature_178  feature_81  feature_39  feature_216
\
22620        363.840        62.724      -27.197    -306.568        63.074
478621       271.068        79.506       27.552     210.116      -199.566
```

```
372254       83.719       58.278       20.461       57.980     -241.424

2596         79.873       90.220       44.391       74.997     -288.605

216892     -204.979       42.962       25.153     -153.595     -207.008


        feature_47  feature_94  feature_173  feature_43
feature_145  \
22620        9.093      81.439      -63.107     -19.456       -15.254

478621      43.592    -178.524     -101.009     -35.033        39.965

372254      24.957      60.443      -66.481     -42.808        -2.487

2596         0.066      71.331      -51.998     -50.790       -90.951

216892      62.062     -30.675        8.945     -27.044        37.564


        feature_205  feature_3  feature_105  feature_53
feature_133  \
22620        28.052    -50.013      -23.103     -76.552       -47.857

478621      -74.677    -56.263      -28.258     -60.416       -50.860

372254      169.643    -57.440          NaN         NaN           NaN

2596        133.578    -65.826      -31.454     -78.346       -46.876

216892     -221.299    -56.410      -26.947     -78.481       -53.546


        feature_207  feature_176  feature_212  feature_49  feature_80
\
22620        51.926      -44.408       -8.734     -19.969        7.598

478621       59.565      -48.011       -9.867     -16.854        3.359

372254       53.574      -45.130       -8.652     -14.878          NaN

2596         52.241      -45.016       -9.543     -10.603       10.750

216892       57.733      -54.199       -8.221     -14.964       14.869


        feature_34  feature_7  feature_110  feature_91  feature_83  \
22620       53.702    135.175      -59.770      15.974     -98.783
478621      55.723    109.011      -50.481      18.203     -93.351
372254      46.277     98.945      -36.951      18.336         NaN
2596        45.664    128.199      -57.712      16.575     -86.957
```

```
216892      51.618    128.961      -55.065      18.350     -88.030

        feature_203  feature_208  feature_89  feature_8  feature_13  \
22620         21.408       -52.267    -161.735     -0.016     -73.571
478621        20.625       -64.838    -153.978     -0.047     -76.884
372254           NaN       -55.948    -158.620        NaN         NaN
2596          15.324       -43.970    -163.810      0.027     -68.620
216892        19.797       -53.166    -142.453     -0.088     -74.988

        feature_59  feature_196  feature_131  feature_17  feature_166
\
22620       -17.578       -96.206       -13.907       38.684       142.940

478621      -25.264      -104.415       -16.271       37.820       154.076

372254          NaN           NaN           NaN          NaN       137.008

2596        -17.801       -93.988       -11.650       48.020       132.534

216892      -21.171      -110.029       -17.532       44.776       150.958


        feature_72  feature_200  feature_134  feature_192  feature_219
\
22620       127.193        19.727     -143.082       156.746        18.878

478621      109.519        15.559     -128.834       156.707        24.013

372254          NaN        22.581           NaN          NaN          NaN

2596        114.688        23.578     -138.962       160.952        22.971

216892      126.752        17.575     -138.706       148.154        20.580


        feature_63  feature_54  feature_107  feature_50
feature_174  \
22620       -31.626       -7.896       -46.545       -9.221         6.183

478621      -25.711       -6.857       -55.906      -12.882         0.584

372254      -28.989       -5.588           NaN       -0.046         5.824

2596        -28.717       -7.872       -57.512       -6.850        -3.200

216892      -25.807       -5.604       -43.131       -7.338         3.995


        feature_190  feature_189  feature_226  feature_201
feature_169  \
22620       -60.625       -96.469        89.161     -120.011        -
```

```
0.605
478621      -49.047      -90.039       91.580     -149.926       -
0.575
372254          NaN          NaN       95.783     -135.141
NaN
2596        -48.636      -95.354       92.526     -128.508       -
0.637
216892      -52.624      -91.185       98.265     -133.948       -
0.495

        feature_58  feature_48  feature_88  feature_21  feature_57  \
22620       68.169      -7.047      -9.152     120.164     -47.648
478621      75.578      -5.808     -10.503     136.696     -42.210
372254      58.896      -6.215     -12.104     123.149     -48.025
2596        64.576      -8.033     -11.553     135.820     -44.388
216892      38.680      -7.054      -8.497     134.329     -45.555

        feature_160  feature_222  feature_187  feature_191
feature_129  \
22620       -99.201            1      -87.544      -33.754     -
128.628
478621      -84.985            0      -94.522      -30.665     -
119.555
372254      -84.697            0      -84.765      -32.813     -
124.360
2596        -88.773            0      -79.484      -31.255     -
128.595
216892      -74.112            1      -96.451      -35.608     -
118.576

        feature_37  feature_157  feature_215  feature_1  feature_52  \
22620            5            4      142.659     37.598     -83.477
478621          11            6      139.632     43.354     -81.877
372254           3            9      121.806     41.456     -81.889
2596             6            1      134.765     37.093     -86.371
216892           6            2      130.902     35.436     -81.023

        feature_149  feature_130  feature_151  feature_103  feature_99
\
22620       118.495      -16.819       88.368      -19.000      29.620

478621      103.376      -36.987       81.816      -13.428      26.844

372254      123.764      -10.272       77.428          NaN         NaN

2596        115.287        1.922       67.009      -17.681      29.720

216892      106.623      -16.137       86.344      -14.453      26.981


        feature_116  feature_87  feature_202  feature_74  feature_214
```

```
\
22620        52.631     -94.253      -67.188      50.895      -10.686

478621       51.690    -106.230      -53.675      52.458      -10.771

372254       53.604     -91.873      -53.926      56.016      -15.786

2596         48.536    -107.784      -71.514      44.896      -15.021

216892       45.313     -94.542      -68.335      47.732      -13.757


        feature_210  feature_121  feature_229  feature_20  feature_188
\
22620      -111.006       29.340        1.833      23.745      -32.001

478621     -121.484       24.561        3.015      22.902      -37.436

372254     -123.005       35.267        2.246      21.848      -51.873

2596       -119.885       33.572        2.381      26.135      -49.703

216892     -117.775       30.967        4.879      24.201      -41.070


        feature_71  feature_106  feature_14  feature_92
feature_179  \
22620       85.490        1.454    -195.726       3.719            1

478621      71.736        1.657    -176.250       3.278            1

372254      65.363        1.627    -178.117       2.029            0

2596        65.014        1.170    -192.232       2.218            1

216892      69.986        1.631    -200.156       3.055            1


        feature_102  target sample_part
22620       -28.814   0.000       train
478621      -45.988   1.000       train
372254      -31.074   1.000       train
2596        -36.704   0.000       train
216892      -44.672   1.000       train

TARGET = 'target'
N_FEATURES = 230
features = [f'feature_{i}' for i in range(N_FEATURES)]
```

Первичный отбор признаков... (1 балл)

Задание: Сначала отсеем совсем уж мусорные признаки.

Воспользовавшись вашим кодом для вычисления `IV` из прошлой домашки, для всех **числовых** фичей вычислите `IV` на 20 бакетах. Отсейте признаки с `IV < 0.005`.

Hint: паркет не всегда сохраняет `dtype` колонки. Чтобы проверить признаки на "реальный" тип данных, лучше на всякий случай посмотреть на `nunique`

```
# your code here
```

Вопрос: Почему некорректно сравнивать `IV` у категориальных и числовых фичей?

Вычислите IV для категориальных фичей на `n` бакетах, где `n = min(число категорий фичи, 20)`. Возможно, придётся перекодировать некоторые фичи (*только не OneHot-ом!*)

Опционально: примените также и к категориальным фичам предварительный отбор по `IV` с менее строгим порогом

Ответ: сравнивать IV у категориальных и числовых фичей некорректно, потому что IV критически зависит от того, как признак разбит на бины. Для числовых признаков биннинг выбирается заранее и задан и может сильно менять IV, а для категориальных "бины" заданы самими категориями, и при множестве редких категорий IV легко становится нестабильным или завышенным. В итоге IV отражает не только важность признака / его потенциальный высокий/низкий вклад в предикты, но и особенности дискретизации, поэтому значения между типами признаков не сопоставимы напрямую.

```
# your code here

data_types = pd.DataFrame(df.dtypes, columns=['type'])
data_types['feature_name'] = data_types.index
data_types['type'] = data_types['type'].astype(str)

float_int_features = data_types.query("type == 'float64' or type ==
'int64'")

non_float_int_features = data_types.query("type != 'float64' and
type != 'int64'")

float_int_features
```

```
                type feature_name
feature_217  float64  feature_217
feature_66   float64   feature_66
feature_9    float64    feature_9
feature_193  float64  feature_193
feature_15   float64   feature_15
...              ...          ...
feature_14   float64   feature_14
```

```
feature_92   float64   feature_92
feature_179    int64  feature_179
feature_102  float64  feature_102
target       float64       target

[229 rows x 2 columns]
```

```
non_float_int_features
```

```
                      type feature_name
date         datetime64[ns]         date
month        datetime64[ms]        month
quarter      datetime64[ns]      quarter
feature_0          category    feature_0
feature_209        category  feature_209
sample_part          object  sample_part
```

```
data_types.query("type != 'float64' and type != 'int64'")
```

```
                      type feature_name
date         datetime64[ns]         date
month        datetime64[ms]        month
quarter      datetime64[ns]      quarter
feature_0          category    feature_0
feature_209        category  feature_209
sample_part          object  sample_part
```

..."вычислите `IV` на 20 бакетах... **Hint:** паркет не всегда сохраняет `dtype` колонки. Чтобы проверить признаки на "реальный" тип данных, лучше на всякий случай посмотреть на `nunique`".

Делаем вывод, что если нам и нужны категориальные признаки, то с количеством категорий >=20 (и именно категориальные в уже закодированном виде!)

```
data_types_nunique = pd.DataFrame(df.nunique(), columns=['nunique'])
data_types_nunique.query("nunique < 20")
```

```
             nunique
quarter            8
feature_147        2
feature_144        2
feature_168        2
feature_165        2
feature_10         2
feature_31         3
feature_0          4
feature_161        2
feature_209        3
feature_62         2
feature_222        2
```

```
feature_37         12
feature_157        13
feature_179         2
target              2
sample_part         3
```

Проверим подозрительные признаки

```
features_to_check = [
    # 'quarter',
    'feature_147',
    'feature_144',
    'feature_168',
    'feature_165',
    'feature_10',
    'feature_31',
    'feature_0',
    'feature_161',
    'feature_209',
    'feature_62',
    'feature_222',
    'feature_179',
    'feature_37',
    'feature_157',
    # 'target',
    # 'sample_part'
]

for feature in features_to_check:
    print(f"Уникальные значения для {feature}:
{df[feature].unique()}")
```

```
Уникальные значения для feature_147: [1 0]
Уникальные значения для feature_144: [1 0]
Уникальные значения для feature_168: [1 0]
Уникальные значения для feature_165: [1 0]
Уникальные значения для feature_10: [nan  0.  1.]
Уникальные значения для feature_31: [2 0 1]
Уникальные значения для feature_0: ['c', 'd', 'b', 'a']
Categories (4, object): ['a' < 'b' < 'c' < 'd']
Уникальные значения для feature_161: [ 0. nan  1.]
Уникальные значения для feature_209: ['PRIV', 'GVM', 'UNEMP']
Categories (3, object): ['UNEMP' < 'GVM' < 'PRIV']
Уникальные значения для feature_62: [ 0.  1. nan]
Уникальные значения для feature_222: [1 0]
Уникальные значения для feature_179: [1 0]
Уникальные значения для feature_37: [ 5 11  3  6  8  1  9  7  0  4  2
10]
```

```
Уникальные значения для feature_157: [ 4  6  9  1  2 11 12  5  8 10  7
0  3]
```

Задание: Сначала отсеем совсем уж мусорные признаки.

Воспользовавшись вашим кодом для вычисления `IV` из прошлой домашки, для всех **числовых** фичей вычислите `IV` на 20 бакетах. Отсейте признаки с `IV < 0.005`.

Hint: паркет не всегда сохраняет `dtype` колонки. Чтобы проверить признаки на "реальный" тип данных, лучше на всякий случай посмотреть на `nunique`

Итак, признаки были проверены, логика описана выше. Нужны непрерывные признаки или признаки, с количеством категорий от 20 включительно и выше. Все, что строго меньше 20 по числу категорий будет рассмотрено в следующем пункте задания.

Сразу разделю данные на train, test и val. Важно это сделать сразу, так как в дальнейшем я обрабатываю пропуски, заменяя их на медианное значение, поэтому крайне важно не "ликануть" данные. В деревьях на пропуски можно не обращать внимания, но это не отменяет потенциальной необходимости в замене, особенно учитывая дальнейшие задействованные методы отбора признаков.

В блоке по расчету IV наличие пропусков не играет особой роли. Но вот сплит желателен - можно посмотреть в разрезе выборок различия в IV.

```
df['sample_part'].unique()

train = df.query("sample_part == 'train'")
val = df.query("sample_part == 'val'")
test = df.query("sample_part == 'test'")

train.head()
```

```
             date      month    quarter  feature_217  feature_66
feature_9  \
22620  2021-01-01 2021-01-01 2021-01-01        1.181       0.483
13.977
478621 2021-01-01 2021-01-01 2021-01-01        3.428       0.887
27.158
372254 2021-01-01 2021-01-01 2021-01-01      -38.555       1.227
19.894
2596   2021-01-01 2021-01-01 2021-01-01      -14.667       0.753
18.203
216892 2021-01-01 2021-01-01 2021-01-01        6.735       1.129
11.246

        feature_193  feature_15  feature_199  feature_25  feature_195
\
22620         1.135      43.272      -46.889      97.558     -173.966
```

```
478621       -0.379      80.088      -67.243     105.803     -100.820

372254        0.499      98.464      -67.403      81.858      -83.455

2596         -1.200     117.331      -25.558      90.277      -99.937

216892       -0.109      61.206      -78.397      82.637      -98.028


        feature_154  feature_126  feature_218  feature_19  feature_96
\
22620         0.355      -61.496        0.000      95.060     116.886

478621       11.771      -31.562        0.000     167.121     103.470

372254      -17.036      -69.425        0.000     157.788     122.015

2596         12.579      -12.452        0.000     189.285      57.268

216892       14.798      -96.745        0.000     163.964     127.391


        feature_177  feature_223  feature_137  feature_146  feature_30
\
22620        23.965       38.981      -77.860        1.523       0.037

478621       19.821      190.191     -102.354       12.880       0.060

372254       43.998          NaN     -110.378        6.746       0.084

2596         31.321      100.130     -112.599       -1.449       0.021

216892       50.016       53.473      -60.523       18.961       0.090


        feature_225  feature_184  feature_108  feature_162
feature_204  \
22620        12.506      -68.489      -15.652       42.140      -
29.446
478621     -107.082      -74.598      -38.710       29.498      -
27.851
372254       -2.640      -35.450      -18.685       40.838       -
5.667
2596         28.147      -21.772      -28.991       25.363     -
103.964
216892      -55.495      -15.294      -11.310       29.490      -
68.760

        feature_180  feature_125  feature_18  feature_147  feature_101
\
22620        33.369      -49.226      15.865            1       18.338
```

```
478621        9.449        4.178      14.934            1       32.957
372254        8.319      -11.541      28.788            0        0.987
2596         18.185       24.168      12.479            0        9.529
216892       25.419       56.772       7.208            1       -6.332

        feature_104  feature_84  feature_197  feature_60  feature_118
\
22620         16.955       91.781       34.180    -116.365       130.684
478621         5.784      137.566       28.170    -158.773       151.633
372254        -0.081      145.124       18.497     -93.969        53.913
2596          10.354      187.479       12.969    -169.816       111.911
216892         0.157      182.656        0.818    -148.135       169.084

        feature_45  feature_16  feature_127  feature_159  feature_119
\
22620         4.627      187.238       48.935        2.829       -83.383
478621        8.406       76.368       89.374      -24.589       -52.683
372254        6.785      115.597       66.472      -12.866       -68.470
2596          9.832      177.790       93.701       -8.853       -54.264
216892        6.027       76.512       87.059      -22.558       -67.290

        feature_82  feature_144  feature_93  feature_168  feature_143
\
22620      -17.192            1     -77.919            1     -134.605
478621     -24.921            1    -126.716            0       -88.918
372254     -20.369            1     -53.191            1       -76.857
2596        -7.155            1     -79.669            0     -148.433
216892      14.523            1     -88.054            1       -89.784

        feature_73  feature_113  feature_112  feature_150  feature_55
\
```

```
22620      187.476        0.189        0.143      -75.169      -13.793
478621     168.039       -4.244        0.073      -48.910       -2.601
372254     101.012       -1.508        0.179      -36.070       -5.636
2596       182.624       -0.776        0.156     -116.707       17.692
216892      75.010       -1.300        0.239      -44.794       18.671

        feature_165  feature_69  feature_167  feature_109  feature_124
\
22620             1     -87.365      141.101      -20.604      -61.495
478621            1     -70.292      159.616      -25.524      -28.477
372254            1     -88.989      186.244      -30.064          NaN
2596              1    -100.694      137.007      -37.357      -42.118
216892            0     -87.919      157.030      -38.641      -48.599

        feature_79  feature_86  feature_95  feature_213
feature_117  \
22620      -99.060       3.628    -132.258        7.458      -17.227
478621    -131.036      15.079    -187.867       10.885       -8.564
372254     -77.346       6.276    -211.018       17.206      -10.249
2596      -133.031     -12.070    -102.541       18.618      -13.115
216892    -224.086      14.581    -133.384        8.274       -7.378

        feature_38  feature_24  feature_67  feature_185
feature_198  \
22620       81.614      -7.180      83.035      121.590      106.620
478621      64.992     -23.798      44.081       42.190      103.292
372254      45.299     -22.977      35.251       63.276       53.121
2596        41.408     -14.088      24.073       77.043      101.308
216892     104.813     -17.196      44.252       71.059       93.234

        feature_10  feature_120  feature_29  feature_153
```

```
feature_68  \
22620          NaN       65.210      -0.317      -55.940     -67.237

478621       0.000       59.559       1.325      -49.355    -103.629

372254       1.000       95.916       0.998     -115.400         NaN

2596         1.000       49.346      -1.149     -110.864    -108.854

216892         NaN       22.010      -1.553     -110.317     -73.858


        feature_75  feature_5  feature_56  feature_114  feature_139  \
22620      -37.566   -215.946    -258.725       -37.069       -17.930
478621     -51.686   -170.714     -75.131       -19.271       -37.153
372254     -32.604   -154.503    -270.127       -22.637         1.675
2596       -45.033   -131.553    -463.492        -4.883       -24.975
216892     -58.049   -162.093     327.178        14.696         8.972

        feature_186  feature_65  feature_142  feature_194  feature_136
\
22620        -2.311     163.888       43.686       15.095       66.272

478621     -62.370     -24.941      154.849       -8.353       11.989

372254       80.467     -55.034      305.262        7.529          NaN

2596         59.383     -88.956       64.488       12.097     -106.539

216892    -148.090     -30.291       88.845        8.704      221.844


        feature_31  feature_12  feature_35  feature_28  feature_42  \
22620            2     327.804     -10.316    -147.398     442.757
478621           0     145.239      -2.953     -10.632    -319.719
372254           0      -0.711      18.446    -155.017    -144.521
2596             0      99.609      22.967    -181.545     -45.809
216892           0     -29.085       1.828     -71.107     112.515

        feature_227  feature_115  feature_155  feature_51  feature_132
\
22620      -173.334     -209.629      -24.298     -50.931      -51.036

478621     -479.616     -113.581       10.729      50.032      -12.897

372254     -222.994     -282.365      -11.249       4.908      -42.238

2596        312.396     -143.675      -58.606     -13.816      -21.098

216892     -201.997      297.919       68.831      -7.798      -21.927
```

```
        feature_182  feature_76  feature_41  feature_97
feature_140  \
22620       -67.803      24.512     -26.470      -0.420       15.000

478621       -1.614    -131.929     -12.022     124.632       28.000

372254      -61.911    -130.156     -48.438     -69.458       10.000

2596        -87.202      20.597     -45.001     105.278       22.000

216892     -193.628     178.125     -24.746      36.812       50.000


        feature_78  feature_135  feature_26  feature_211  feature_171
\
22620      -59.890     -209.048      45.124        5.757      -81.663

478621     -19.073     -110.640      33.406        3.106      -15.415

372254         NaN       88.093    -186.826       -7.539      -96.512

2596         3.804       92.171     -74.749      -14.222      -69.923

216892      27.339      394.331     186.121       46.305       67.471


        feature_158 feature_0  feature_2  feature_77  feature_46
feature_100  \
22620        61.183         c     25.846     216.181     115.320     -
248.700
478621       57.095         d     69.958     198.211     172.981
167.405
372254      -15.850         c     77.523    -295.798    -119.580
44.884
2596         51.884         b      8.587    -107.226      85.009
58.589
216892       34.363         c    -55.348      -6.044      43.109     -
125.506

        feature_111  feature_138  feature_164  feature_90  feature_85
\
22620      -294.392     -135.357       83.424      76.595      -8.157

478621     -482.424      350.920     -192.861      87.053      -6.965

372254     -247.903      197.881      111.868       4.347       3.490

2596         19.096      203.119      -87.258      90.792     -18.724

216892     -238.630      179.954      -55.945      65.864      12.504
```

```
        feature_161  feature_152  feature_98  feature_36  feature_181
\
22620         0.000       -5.454      15.142      64.097      -23.541

478621        0.000       -0.418      10.568    -158.395       19.243

372254          NaN        3.094      15.698      46.128       24.112

2596          0.000       -6.581      -5.818      55.446      -34.486

216892        1.000        3.840      -7.056     -31.857      -76.527


        feature_61  feature_22 feature_209  feature_183  feature_33  \
22620       -8.659     126.982        PRIV     -309.840      61.284
478621     -25.389      29.449         GVM      234.698     -51.806
372254     -22.496     -48.221       UNEMP      -58.677      -9.734
2596       -65.080      -6.673        PRIV      336.292     -50.371
216892     -59.141      40.590       UNEMP      230.065     -87.018

        feature_11  feature_224  feature_228  feature_6  feature_27  \
22620      104.977        0.635      246.898    -22.187     109.827
478621      19.588        1.162     -148.415     -2.435      93.826
372254      44.855        0.863          NaN      2.779      70.812
2596       115.083        1.218      101.575    -28.165     113.366
216892      27.006        1.132       78.013     -2.809      96.061

        feature_141  feature_220  feature_221  feature_156  feature_4
\
22620         14.415      -13.085      -97.530       67.237   -179.752

478621        83.803       30.582     -132.501      -23.582   -158.853

372254        63.379      -18.682     -125.532        5.000   -128.796

2596         109.776      125.930      -79.832        4.022   -184.373

216892       152.139     -138.231      -71.237        8.348   -161.772


        feature_122  feature_32  feature_163  feature_62  feature_128
\
22620         19.595      83.026      -14.136       0.000      -10.854

478621        -4.892      58.832      -56.234       1.000       -7.450

372254        20.959       9.973          NaN         NaN       -8.373

2596          25.708       8.970      -17.724       0.000       -1.277
```

```
216892        5.935     -65.317      -56.179       0.000       -0.979


        feature_206  feature_172  feature_70  feature_175  feature_64
\
22620      -339.389       21.311    -242.062      -28.774      17.826

478621       12.301       75.191      71.929       90.201      65.810

372254      293.458          NaN    -216.702     -209.663      -6.491

2596        100.201       41.056    -229.853     -181.364      19.546

216892      348.117      -54.095    -106.647      -14.576      27.792


        feature_44  feature_148  feature_40  feature_123
feature_23  \
22620      588.375     -167.855     419.900     -151.478     -26.248

478621     832.826     -104.884     134.214      298.246       6.461

372254     786.598          NaN     483.933       97.247     -14.063

2596      1030.431     -165.598     369.273       13.847     -58.286

216892    1225.000     -154.776    -223.218       40.652      60.718


        feature_170  feature_178  feature_81  feature_39  feature_216
\
22620       363.840       62.724     -27.197    -306.568       63.074

478621      271.068       79.506      27.552     210.116     -199.566

372254       83.719       58.278      20.461      57.980     -241.424

2596         79.873       90.220      44.391      74.997     -288.605

216892     -204.979       42.962      25.153    -153.595     -207.008


        feature_47  feature_94  feature_173  feature_43
feature_145  \
22620        9.093      81.439      -63.107     -19.456      -15.254

478621      43.592    -178.524     -101.009     -35.033       39.965

372254      24.957      60.443      -66.481     -42.808       -2.487
```

```
2596          0.066      71.331      -51.998     -50.790      -90.951

216892       62.062     -30.675        8.945     -27.044       37.564


        feature_205  feature_3  feature_105  feature_53
feature_133  \
22620         28.052    -50.013      -23.103     -76.552      -47.857

478621       -74.677    -56.263      -28.258     -60.416      -50.860

372254       169.643    -57.440          NaN         NaN          NaN

2596         133.578    -65.826      -31.454     -78.346      -46.876

216892      -221.299    -56.410      -26.947     -78.481      -53.546


        feature_207  feature_176  feature_212  feature_49  feature_80
\
22620         51.926      -44.408       -8.734     -19.969       7.598

478621        59.565      -48.011       -9.867     -16.854       3.359

372254        53.574      -45.130       -8.652     -14.878         NaN

2596          52.241      -45.016       -9.543     -10.603      10.750

216892        57.733      -54.199       -8.221     -14.964      14.869


        feature_34  feature_7  feature_110  feature_91  feature_83  \
22620        53.702    135.175      -59.770      15.974     -98.783
478621       55.723    109.011      -50.481      18.203     -93.351
372254       46.277     98.945      -36.951      18.336         NaN
2596         45.664    128.199      -57.712      16.575     -86.957
216892       51.618    128.961      -55.065      18.350     -88.030

        feature_203  feature_208  feature_89  feature_8  feature_13  \
22620         21.408      -52.267    -161.735     -0.016     -73.571
478621        20.625      -64.838    -153.978     -0.047     -76.884
372254           NaN      -55.948    -158.620        NaN         NaN
2596          15.324      -43.970    -163.810      0.027     -68.620
216892        19.797      -53.166    -142.453     -0.088     -74.988

        feature_59  feature_196  feature_131  feature_17  feature_166
\
22620       -17.578      -96.206      -13.907      38.684      142.940

478621      -25.264     -104.415      -16.271      37.820      154.076
```

```
372254         NaN          NaN          NaN         NaN      137.008

2596       -17.801      -93.988      -11.650      48.020      132.534

216892     -21.171     -110.029      -17.532      44.776      150.958


        feature_72  feature_200  feature_134  feature_192  feature_219
\
22620      127.193       19.727     -143.082      156.746       18.878

478621     109.519       15.559     -128.834      156.707       24.013

372254         NaN       22.581          NaN          NaN          NaN

2596       114.688       23.578     -138.962      160.952       22.971

216892     126.752       17.575     -138.706      148.154       20.580


        feature_63  feature_54  feature_107  feature_50
feature_174  \
22620      -31.626      -7.896      -46.545      -9.221         6.183

478621     -25.711      -6.857      -55.906     -12.882         0.584

372254     -28.989      -5.588          NaN      -0.046         5.824

2596       -28.717      -7.872      -57.512      -6.850        -3.200

216892     -25.807      -5.604      -43.131      -7.338         3.995


        feature_190  feature_189  feature_226  feature_201
feature_169  \
22620       -60.625      -96.469       89.161     -120.011        -
0.605
478621      -49.047      -90.039       91.580     -149.926        -
0.575
372254          NaN          NaN       95.783     -135.141
NaN
2596        -48.636      -95.354       92.526     -128.508        -
0.637
216892      -52.624      -91.185       98.265     -133.948        -
0.495

        feature_58  feature_48  feature_88  feature_21  feature_57  \
22620       68.169      -7.047      -9.152     120.164     -47.648
478621      75.578      -5.808     -10.503     136.696     -42.210
372254      58.896      -6.215     -12.104     123.149     -48.025
2596        64.576      -8.033     -11.553     135.820     -44.388
```

```
216892      38.680      -7.054      -8.497     134.329     -45.555

        feature_160  feature_222  feature_187  feature_191
feature_129  \
22620       -99.201            1      -87.544      -33.754     -
128.628
478621      -84.985            0      -94.522      -30.665     -
119.555
372254      -84.697            0      -84.765      -32.813     -
124.360
2596        -88.773            0      -79.484      -31.255     -
128.595
216892      -74.112            1      -96.451      -35.608     -
118.576

        feature_37  feature_157  feature_215  feature_1  feature_52  \
22620            5            4      142.659     37.598     -83.477
478621          11            6      139.632     43.354     -81.877
372254           3            9      121.806     41.456     -81.889
2596             6            1      134.765     37.093     -86.371
216892           6            2      130.902     35.436     -81.023

        feature_149  feature_130  feature_151  feature_103  feature_99
\
22620       118.495      -16.819       88.368      -19.000      29.620

478621      103.376      -36.987       81.816      -13.428      26.844

372254      123.764      -10.272       77.428          NaN         NaN

2596        115.287        1.922       67.009      -17.681      29.720

216892      106.623      -16.137       86.344      -14.453      26.981


        feature_116  feature_87  feature_202  feature_74  feature_214
\
22620        52.631     -94.253      -67.188      50.895      -10.686

478621       51.690    -106.230      -53.675      52.458      -10.771

372254       53.604     -91.873      -53.926      56.016      -15.786

2596         48.536    -107.784      -71.514      44.896      -15.021

216892       45.313     -94.542      -68.335      47.732      -13.757


        feature_210  feature_121  feature_229  feature_20  feature_188
\
22620      -111.006       29.340        1.833      23.745      -32.001
```

```
478621     -121.484       24.561        3.015      22.902       -37.436

372254     -123.005       35.267        2.246      21.848       -51.873

2596       -119.885       33.572        2.381      26.135       -49.703

216892     -117.775       30.967        4.879      24.201       -41.070


        feature_71  feature_106  feature_14  feature_92
feature_179  \
22620       85.490        1.454    -195.726       3.719             1

478621      71.736        1.657    -176.250       3.278             1

372254      65.363        1.627    -178.117       2.029             0

2596        65.014        1.170    -192.232       2.218             1

216892      69.986        1.631    -200.156       3.055             1


        feature_102  target sample_part
22620       -28.814   0.000       train
478621      -45.988   1.000       train
372254      -31.074   1.000       train
2596        -36.704   0.000       train
216892      -44.672   1.000       train
```

```python
def calc_iv(values: np.ndarray, target: np.ndarray, buckets:
np.ndarray) -> float:
    """Считает для признака IV"""
    B = (target == 1).sum()
    G = (target == 0).sum()

    IV = 0.0

    for j in np.unique(buckets):
        mask = buckets == j
        y_j = target[mask]

        B_j = (y_j == 1).sum()
        G_j = (y_j == 0).sum()

        # пропускаем бакеты, где в одном классе нет значений
        if B_j == 0 or G_j == 0:
            continue

        woe_j = np.log((B_j / G_j) / (B / G))
```

```
        IV += (B_j / B - G_j / G) * woe_j

    return IV
```

Сделаем проход по всем признакам, кроме вышеуказанных

```
features_to_check = [
    'feature_147',
    'feature_144',
    'feature_168',
    'feature_165',
    'feature_10',
    'feature_31',
    'feature_0',
    'feature_161',
    'feature_209',
    'feature_62',
    'feature_222',
    'feature_179',
    'feature_37',
    'feature_157'
]

to_drop = set(features_to_check)
features_iv = [f for f in features if f not in to_drop]

# N_FEATURES = 230
```

Считаю пропуски как отдельный бакет

```
# Инициализируем словарь для результатов
iv_results = []
y = train[TARGET].astype(int).to_numpy()

for feature_i in features_iv:
    x = train[feature_i]

    # отдельный бакет для NaN
    buckets = pd.Series(index=train.index, dtype="float64")

    mask = x.notna()
    x_non_na = x[mask]

    # квантильные бакеты
    if x_non_na.nunique(dropna=True) <= 1:
        buckets.loc[mask] = 0
    else:
        try:
            buckets.loc[mask] = pd.qcut(x_non_na, q=20, labels=False,
duplicates="drop")
```

```
        except Exception:
            buckets.loc[mask] = pd.cut(x_non_na, bins=20,
labels=False)

    buckets.loc[~mask] = -1

    iv_value = calc_iv(values=x.to_numpy(), target=y,
buckets=buckets.to_numpy())
    iv_results.append([feature_i, iv_value])

iv_df_train = pd.DataFrame(iv_results, columns=["Feature",
"IV"]).sort_values("IV", ascending=False).reset_index(drop=True)
iv_df_train.head()
```

```
       Feature    IV
0  feature_114 0.035
1   feature_44 0.035
2  feature_124 0.031
3  feature_117 0.028
4   feature_30 0.028
```

Отсекаем признаки по предложенному в задании порогу

```
iv_thr = 0.005
selected_features_train = iv_df_train.loc[iv_df_train["IV"] >= iv_thr,
"Feature"].tolist()
dropped_features_train  = iv_df_train.loc[iv_df_train["IV"] <  iv_thr,
"Feature"].tolist()

print(f"Признаки, которые были оставлены (в соответствии с порогом >=
{iv_thr}): {len(selected_features_train)}")
print(f"Признаки, которые были отсеяны (в соответствии с порогом <
{iv_thr}): {len(dropped_features_train)}")
```

```
Признаки, которые были оставлены (в соответствии с порогом >= 0.005):
57
Признаки, которые были отсеяны (в соответствии с порогом < 0.005): 159
```

```
dropped_features_train
```

```
['feature_26',
 'feature_146',
 'feature_47',
 'feature_40',
 'feature_171',
 'feature_118',
 'feature_152',
 'feature_45',
 'feature_67',
 'feature_141',
```

```
 'feature_125',
 'feature_123',
 'feature_84',
 'feature_61',
 'feature_180',
 'feature_51',
 'feature_128',
 'feature_55',
 'feature_197',
 'feature_228',
 'feature_127',
 'feature_224',
 'feature_25',
 'feature_186',
 'feature_120',
 'feature_11',
 'feature_86',
 'feature_145',
 'feature_193',
 'feature_205',
 'feature_68',
 'feature_195',
 'feature_93',
 'feature_64',
 'feature_104',
 'feature_217',
 'feature_184',
 'feature_181',
 'feature_90',
 'feature_101',
 'feature_204',
 'feature_82',
 'feature_148',
 'feature_18',
 'feature_153',
 'feature_60',
 'feature_185',
 'feature_112',
 'feature_85',
 'feature_41',
 'feature_73',
 'feature_175',
 'feature_77',
 'feature_22',
 'feature_108',
 'feature_28',
 'feature_198',
 'feature_122',
 'feature_182',
```

```
 'feature_173',
 'feature_158',
 'feature_42',
 'feature_16',
 'feature_163',
 'feature_211',
 'feature_199',
 'feature_66',
 'feature_113',
 'feature_194',
 'feature_33',
 'feature_95',
 'feature_220',
 'feature_136',
 'feature_58',
 'feature_119',
 'feature_19',
 'feature_81',
 'feature_63',
 'feature_150',
 'feature_83',
 'feature_72',
 'feature_177',
 'feature_178',
 'feature_126',
 'feature_166',
 'feature_29',
 'feature_221',
 'feature_105',
 'feature_121',
 'feature_130',
 'feature_103',
 'feature_74',
 'feature_99',
 'feature_219',
 'feature_212',
 'feature_78',
 'feature_187',
 'feature_188',
 'feature_200',
 'feature_7',
 'feature_80',
 'feature_53',
 'feature_34',
 'feature_189',
 'feature_129',
 'feature_54',
 'feature_69',
 'feature_110',
```

```
 'feature_3',
 'feature_50',
 'feature_215',
 'feature_214',
 'feature_192',
 'feature_202',
 'feature_92',
 'feature_207',
 'feature_71',
 'feature_91',
 'feature_133',
 'feature_8',
 'feature_21',
 'feature_59',
 'feature_20',
 'feature_176',
 'feature_196',
 'feature_169',
 'feature_226',
 'feature_57',
 'feature_13',
 'feature_49',
 'feature_1',
 'feature_134',
 'feature_102',
 'feature_137',
 'feature_17',
 'feature_88',
 'feature_149',
 'feature_106',
 'feature_131',
 'feature_116',
 'feature_210',
 'feature_159',
 'feature_14',
 'feature_89',
 'feature_229',
 'feature_191',
 'feature_203',
 'feature_190',
 'feature_201',
 'feature_52',
 'feature_107',
 'feature_208',
 'feature_174',
 'feature_87',
 'feature_48',
 'feature_151',
 'feature_160',
```

```
 'feature_143',
 'feature_6']
```

Теперь посморим по валидационной выборке

```
# Инициализируем словарь для результатов
iv_results = []
y = val[TARGET].astype(int).to_numpy()

for feature_i in features_iv:
    x = val[feature_i]

    # отдельный бакет для NaN
    buckets = pd.Series(index=val.index, dtype="float64")

    mask = x.notna()
    x_non_na = x[mask]

    # квантильные бакеты
    if x_non_na.nunique(dropna=True) <= 1:
        buckets.loc[mask] = 0
    else:
        try:
            buckets.loc[mask] = pd.qcut(x_non_na, q=20, labels=False,
duplicates="drop")
        except Exception:
            buckets.loc[mask] = pd.cut(x_non_na, bins=20,
labels=False)

    buckets.loc[~mask] = -1

    iv_value = calc_iv(values=x.to_numpy(), target=y,
buckets=buckets.to_numpy())
    iv_results.append([feature_i, iv_value])

iv_df_val = pd.DataFrame(iv_results, columns=["Feature",
"IV"]).sort_values("IV", ascending=False).reset_index(drop=True)
iv_df_val.head()
```

```
       Feature    IV
0   feature_44 0.039
1  feature_114 0.035
2  feature_124 0.034
3  feature_117 0.026
4   feature_30 0.025
```

```
iv_thr = 0.005
selected_features_val = iv_df_val.loc[iv_df_val["IV"] >= iv_thr,
"Feature"].tolist()
dropped_features_val  = iv_df_val.loc[iv_df_val["IV"] <  iv_thr,
```

```
"Feature"].tolist()

print(f"Признаки, которые были оставлены (в соответствии с порогом >=
{iv_thr}): {len(selected_features_val)}")
print(f"Признаки, которые были отсеяны (в соответствии с порогом <
{iv_thr}): {len(dropped_features_val)}")
```

```
Признаки, которые были оставлены (в соответствии с порогом >= 0.005):
69
Признаки, которые были отсеяны (в соответствии с порогом < 0.005): 147
```

Теперь посмотрим по тестовой выборке

```
# Инициализируем словарь для результатов
iv_results = []
y = test[TARGET].astype(int).to_numpy()

for feature_i in features_iv:
    x = test[feature_i]

    # отдельный бакет для NaN
    buckets = pd.Series(index=test.index, dtype="float64")

    mask = x.notna()
    x_non_na = x[mask]

    # квантильные бакеты
    if x_non_na.nunique(dropna=True) <= 1:
        buckets.loc[mask] = 0
    else:
        try:
            buckets.loc[mask] = pd.qcut(x_non_na, q=20, labels=False,
duplicates="drop")
        except Exception:
            buckets.loc[mask] = pd.cut(x_non_na, bins=20,
labels=False)

    buckets.loc[~mask] = -1

    iv_value = calc_iv(values=x.to_numpy(), target=y,
buckets=buckets.to_numpy())
    iv_results.append([feature_i, iv_value])

iv_df_test = pd.DataFrame(iv_results, columns=["Feature",
"IV"]).sort_values("IV", ascending=False).reset_index(drop=True)
iv_df_test.head()
```

```
       Feature    IV
0   feature_44 0.040
1  feature_124 0.034
```

```
2  feature_117 0.028
3   feature_30 0.024
4    feature_2 0.019
```

```
iv_thr = 0.005
selected_features_test = iv_df_test.loc[iv_df_test["IV"] >= iv_thr,
"Feature"].tolist()
dropped_features_test  = iv_df_test.loc[iv_df_test["IV"] <  iv_thr,
"Feature"].tolist()

print(f"Признаки, которые были оставлены (в соответствии с порогом >=
{iv_thr}): {len(selected_features_test)}")
print(f"Признаки, которые были отсеяны (в соответствии с порогом <
{iv_thr}): {len(dropped_features_test)}")
```

```
Признаки, которые были оставлены (в соответствии с порогом >= 0.005):
59
Признаки, которые были отсеяны (в соответствии с порогом < 0.005): 157
```

Теперь посмотрим на различия в выборках

```
train_map = iv_df_train.set_index("Feature")["IV"]
val_map   = iv_df_val.set_index("Feature")["IV"]
test_map  = iv_df_test.set_index("Feature")["IV"]

iv_diff = pd.DataFrame({
    "IV_train": train_map,
    "IV_val": val_map,
    "IV_test": test_map,
})

iv_diff["abs_diff_train_val"]  = (iv_diff["IV_train"] -
iv_diff["IV_val"]).abs()
iv_diff["abs_diff_train_test"] = (iv_diff["IV_train"] -
iv_diff["IV_test"]).abs()

top10_train_val = iv_diff.sort_values("abs_diff_train_val",
ascending=False).head(10)
top10_train_test = iv_diff.sort_values("abs_diff_train_test",
ascending=False).head(10)

top10_train_val
```

Feature	IV_train	IV_val	IV_test	abs_diff_train_val
feature_2	0.017	0.022	0.019	0.005
feature_44	0.035	0.039	0.040	0.004
feature_75	0.011	0.015	0.011	0.003
feature_30	0.028	0.025	0.024	0.003
feature_223	0.010	0.013	0.012	0.003
feature_124	0.031	0.034	0.034	0.003

```
feature_123     0.004   0.007    0.006                0.003
feature_111     0.012   0.009    0.009                0.003
feature_140     0.011   0.014    0.013                0.003
feature_154     0.011   0.009    0.002                0.003
```

```
             abs_diff_train_test
Feature
feature_2                  0.002
feature_44                 0.005
feature_75                 0.000
feature_30                 0.004
feature_223                0.003
feature_124                0.003
feature_123                0.001
feature_111                0.002
feature_140                0.002
feature_154                0.010
```

top10_train_test

```
             IV_train  IV_val  IV_test  abs_diff_train_val  \
Feature
feature_114     0.035   0.035    0.007                0.000
feature_139     0.018   0.016    0.004                0.002
feature_154     0.011   0.009    0.002                0.003
feature_44      0.035   0.039    0.040                0.004
feature_15      0.012   0.013    0.017                0.002
feature_43      0.012   0.013    0.017                0.002
feature_38      0.016   0.015    0.012                0.001
feature_30      0.028   0.025    0.024                0.003
feature_124     0.031   0.034    0.034                0.003
feature_70      0.008   0.009    0.011                0.001
```

```
             abs_diff_train_test
Feature
feature_114                0.028
feature_139                0.014
feature_154                0.010
feature_44                 0.005
feature_15                 0.005
feature_43                 0.005
feature_38                 0.004
feature_30                 0.004
feature_124                0.003
feature_70                 0.003
```

Выводы: можно заметить, что между тренировочным набором данных и валидаций нет заметных отличий по IV признаков, а вот между train/val и тестовым набором - есть. Так как тестовая выборка у нас, фактически, OOT, эту информацию использовать как-либо не совсем корректно в будущем, но имеем в виду, что итоговые скоры так или иначе отличаться между выборками, а предикты быть менее стабильными. Какие признаки в итоге выбрать? Предлагаю те, которые мы видим на train. Глобальной разницы с val не имеется (только 2 фичи взаимено выпадают/добавляются). Тест же для нас недоступен :)

```
iv_thr = 0.005
selected_features_train = iv_df_train.loc[iv_df_train["IV"] >= iv_thr,
"Feature"].tolist()
dropped_features_train  = iv_df_train.loc[iv_df_train["IV"] <  iv_thr,
"Feature"].tolist()

print(f"Признаки, которые были оставлены (в соответствии с порогом >=
{iv_thr}): {len(selected_features_train)}")
print(f"Признаки, которые были отсеяны (в соответствии с порогом <
{iv_thr}): {len(dropped_features_train)}")
```

```
Признаки, которые были оставлены (в соответствии с порогом >= 0.005):
57
Признаки, которые были отсеяны (в соответствии с порогом < 0.005): 159
```

Вопрос: Почему некорректно сравнивать `IV` у категориальных и числовых фичей?

Вычислите IV для категориальных фичей на `n` бакетах, где `n = min(число категорий фичи, 20)`. Возможно, придётся перекодировать некоторые фичи (*только не OneHot-ом!*)

Опционально: примените также и к категориальным фичам предварительный отбор по `IV` с менее строгим порогом

Кодировка

Важно подсветить, что кодировать признаки на основе методов вида Mean Target не имеет смысла, если мы хотим в дальнейшем измерить IV закодированного признака, - мы получим крайне завышенный показатель IV.

Сам по себе пайплайн бы выглядел так (оставлю комментарий, так как далее по ходу работы могу захотеть перекодировать таким образом оставшиеся признаки):

Catboost encoding. Хороший вопрос - стоит ли кодировать численные категории с большим числом категорий? Полагаю, что числа тут могут что-то значить примечательное. В целом, это можно как сделать, так и не сделать. Я сделаю так:

- 0/1, 0/1/2 и подобное останутся как есть;
- 0/1 с наличием NaN - оформим NaN как отдельную категорию, если там достаточно много наблюдений для выделения в категорию (спойлер: далее увидим, что много). Далее переводим все в формат строки и кодируем через catboost encoding (индекс _cb);

- Численные категории с количеством категорий > 3 оставляем как есть;
- Категориальные качественные переменные кодируются через catboost (индекс _cb).

Но итоговый пайплайн будет выглядеть так:

Предлагаю для качественных признаков использовать catboost encoding. Хороший вопрос - стоит ли кодировать численные категории с большим числом категорий? Полагаю, что числа тут могут что-то значить примечательное. В целом, это можно как сделать, так и не сделать. Я сделаю так:

- 0/1, 0/1/2 и подобное останутся как есть;
- 0/1 с наличием NaN - оформим NaN как отдельную категорию, если там достаточно много наблюдений для выделения в категорию (спойлер: далее увидим, что много). Далее переводим все в формат строки и кодируем через label encoding (индекс _le; фактически, добавляем просто категорию, например, "2");
- Численные категории с количеством категорий > 3 оставляем как есть;
- Категориальные качественные переменные кодируются через label encoding (индекс _le).

```
features_to_check = [
    # 'quarter',
    'feature_147',
    'feature_144',
    'feature_168',
    'feature_165',
    'feature_10',
    'feature_31',
    'feature_0',
    'feature_161',
    'feature_209',
    'feature_62',
    'feature_222',
    'feature_179',
    'feature_37',
    'feature_157',
    # 'target',
    # 'sample_part'
]

for feature in features_to_check:
    print(f"Уникальные значения для {feature}:
{df[feature].unique()}")
```

```
Уникальные значения для feature_147: [1 0]
Уникальные значения для feature_144: [1 0]
Уникальные значения для feature_168: [1 0]
Уникальные значения для feature_165: [1 0]
Уникальные значения для feature_10: [nan  0.  1.]
Уникальные значения для feature_31: [2 0 1]
Уникальные значения для feature_0: ['c', 'd', 'b', 'a']
```

```
Categories (4, object): ['a' < 'b' < 'c' < 'd']
Уникальные значения для feature_161: [ 0. nan  1.]
Уникальные значения для feature_209: ['PRIV', 'GVM', 'UNEMP']
Categories (3, object): ['UNEMP' < 'GVM' < 'PRIV']
Уникальные значения для feature_62: [ 0.  1. nan]
Уникальные значения для feature_222: [1 0]
Уникальные значения для feature_179: [1 0]
Уникальные значения для feature_37: [ 5 11  3  6  8  1  9  7  0  4  2
10]
Уникальные значения для feature_157: [ 4  6  9  1  2 11 12  5  8 10  7
0  3]
```

Те самые доли

```
df['feature_10'].isna().sum() / (df['feature_10'].notna().sum() +
df['feature_10'].isna().sum())
```

```
0.333334
```

```
df['feature_161'].isna().sum() / (df['feature_161'].notna().sum() +
df['feature_161'].isna().sum())
```

```
0.10278
```

```
df['feature_62'].isna().sum() / (df['feature_62'].notna().sum() +
df['feature_62'].isna().sum())
```

```
0.125294
```

```
le_cols = ["feature_0", "feature_209", "feature_10", "feature_161",
"feature_62"]

for c in le_cols:
    train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
    val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
    test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)

for c in le_cols:
    codes, uniques = pd.factorize(train[c], sort=True)
    mapping = {cat: i for i, cat in enumerate(uniques)}

    train[c + "_le"] = train[c].map(mapping).astype("int32")
    val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
    test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c + "_le"] = train[c].map(mapping).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c + "_le"] = train[c].map(mapping).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
```

```
  test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c + "_le"] = train[c].map(mapping).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c + "_le"] = train[c].map(mapping).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:12:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train[c + "_le"] = train[c].map(mapping).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2687042426.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")
```

```
features_keep_as_is = [
    "feature_147", "feature_144", "feature_168", "feature_165",
    "feature_222", "feature_179",
    "feature_31",
    "feature_37", "feature_157",
]

features_le = [
    "feature_10_le", "feature_161_le", "feature_62_le",
    "feature_0_le", "feature_209_le",
]
features_to_check_coded = features_keep_as_is + features_le

iv_results = []
y = train[TARGET].astype(int).to_numpy()
```

```
for feature_i in features_to_check_coded:
    x =
train[feature_i].astype("object").where(train[feature_i].notna(),
"missing").astype(str)

    k = x.nunique(dropna=False)

    if k <= 20:
        buckets = pd.factorize(x, sort=True)[0]
        n_used = k
    else:
        top = x.value_counts().head(19).index
        x2 = x.where(x.isin(top), "other")
        buckets = pd.factorize(x2, sort=True)[0]
        n_used = 20

    iv_value = calc_iv(values=x.to_numpy(), target=y, buckets=buckets)
    iv_results.append([feature_i, k, n_used, iv_value])

iv_df_cat_train = (
    pd.DataFrame(iv_results, columns=["Feature", "n_categories",
"n_buckets_used", "IV"])
      .sort_values("IV", ascending=False)
      .reset_index(drop=True)
)
```

val

```
iv_results = []
y = val[TARGET].astype(int).to_numpy()


for feature_i in features_to_check_coded:
    x = val[feature_i].astype("object").where(val[feature_i].notna(),
"missing").astype(str)

    k = x.nunique(dropna=False)

    if k <= 20:
        buckets = pd.factorize(x, sort=True)[0]
        n_used = k
    else:
        top = x.value_counts().head(19).index
        x2 = x.where(x.isin(top), "other")
        buckets = pd.factorize(x2, sort=True)[0]
        n_used = 20

    iv_value = calc_iv(values=x.to_numpy(), target=y, buckets=buckets)
    iv_results.append([feature_i, k, n_used, iv_value])
```

```
iv_df_cat_val = (
    pd.DataFrame(iv_results, columns=["Feature", "n_categories",
"n_buckets_used", "IV"])
      .sort_values("IV", ascending=False)
      .reset_index(drop=True)
)

iv_results = []
y = test[TARGET].astype(int).to_numpy()


for feature_i in features_to_check_coded:
    x =
test[feature_i].astype("object").where(test[feature_i].notna(),
"missing").astype(str)

    k = x.nunique(dropna=False)

    if k <= 20:
        buckets = pd.factorize(x, sort=True)[0]
        n_used = k
    else:
        top = x.value_counts().head(19).index
        x2 = x.where(x.isin(top), "other")
        buckets = pd.factorize(x2, sort=True)[0]
        n_used = 20

    iv_value = calc_iv(values=x.to_numpy(), target=y, buckets=buckets)
    iv_results.append([feature_i, k, n_used, iv_value])

iv_df_cat_test = (
    pd.DataFrame(iv_results, columns=["Feature", "n_categories",
"n_buckets_used", "IV"])
      .sort_values("IV", ascending=False)
      .reset_index(drop=True)
)

iv_df_cat_train
```

	Feature	n_categories	n_buckets_used	IV
0	feature_209_le	3	3	0.025
1	feature_144	2	2	0.013
2	feature_31	3	3	0.012
3	feature_10_le	3	3	0.011
4	feature_147	2	2	0.009
5	feature_165	2	2	0.009
6	feature_168	2	2	0.007
7	feature_62_le	3	3	0.006
8	feature_161_le	3	3	0.002
9	feature_0_le	4	4	0.001
10	feature_157	13	13	0.000

```
11      feature_37            12               12 0.000
12     feature_179             2                2 0.000
13     feature_222             2                2 0.000
```

```
iv_df_cat_val
```

```
           Feature  n_categories  n_buckets_used    IV
0   feature_209_le             3               3 0.025
1       feature_31             3               3 0.015
2    feature_10_le             3               3 0.014
3      feature_144             2               2 0.013
4      feature_165             2               2 0.009
5    feature_62_le             3               3 0.009
6      feature_168             2               2 0.008
7      feature_147             2               2 0.007
8   feature_161_le             3               3 0.003
9     feature_0_le             4               4 0.001
10      feature_37            12              12 0.001
11     feature_157            13              13 0.001
12     feature_179             2               2 0.000
13     feature_222             2               2 0.000
```

```
iv_df_cat_test
```

```
           Feature  n_categories  n_buckets_used    IV
0   feature_209_le             3               3 0.026
1      feature_144             2               2 0.014
2       feature_31             3               3 0.012
3    feature_10_le             3               3 0.012
4      feature_165             2               2 0.010
5    feature_62_le             3               3 0.008
6      feature_147             2               2 0.007
7      feature_168             2               2 0.005
8   feature_161_le             3               3 0.004
9     feature_0_le             4               4 0.001
10      feature_37            12              12 0.000
11     feature_157            13              13 0.000
12     feature_222             2               2 0.000
13     feature_179             2               2 0.000
```

Возьмем более мягкий порог, чем для непрерывных признаков. Рассмотрим просто отобранные признаки по порогу из train, т. к. показатели IV между выборками относительно схожие (снова)

```
iv_thr = 0.001
selected_features_cat_train =
iv_df_cat_train.loc[iv_df_cat_train["IV"] >= iv_thr,
"Feature"].tolist()
dropped_features_cat_train  =
iv_df_cat_train.loc[iv_df_cat_train["IV"] <  iv_thr,
```

```
"Feature"].tolist()

print(f"Признаки, которые были оставлены (в соответствии с порогом >=
{iv_thr}): {len(selected_features_cat_train)}")
print(f"Признаки, которые были отсеяны (в соответствии с порогом <
{iv_thr}): {len(dropped_features_cat_train)}")
```

```
Признаки, которые были оставлены (в соответствии с порогом >= 0.001):
9
Признаки, которые были отсеяны (в соответствии с порогом < 0.001): 5
```

Итоговый признаковый набор

```
features_final = selected_features_cat_train + selected_features_train

train[features_final].dtypes
```

```
feature_209_le      int32
feature_144         int64
feature_31          int64
feature_10_le       int32
feature_147         int64
feature_165         int64
feature_168         int64
feature_62_le       int32
feature_161_le      int32
feature_114       float64
feature_44        float64
feature_124       float64
feature_117       float64
feature_30        float64
feature_227       float64
feature_139       float64
feature_2         float64
feature_98        float64
feature_172       float64
feature_35        float64
feature_109       float64
feature_38        float64
feature_46        float64
feature_76        float64
feature_218       float64
feature_79        float64
feature_167       float64
feature_4         float64
feature_27        float64
feature_225       float64
feature_15        float64
feature_43        float64
feature_24        float64
```

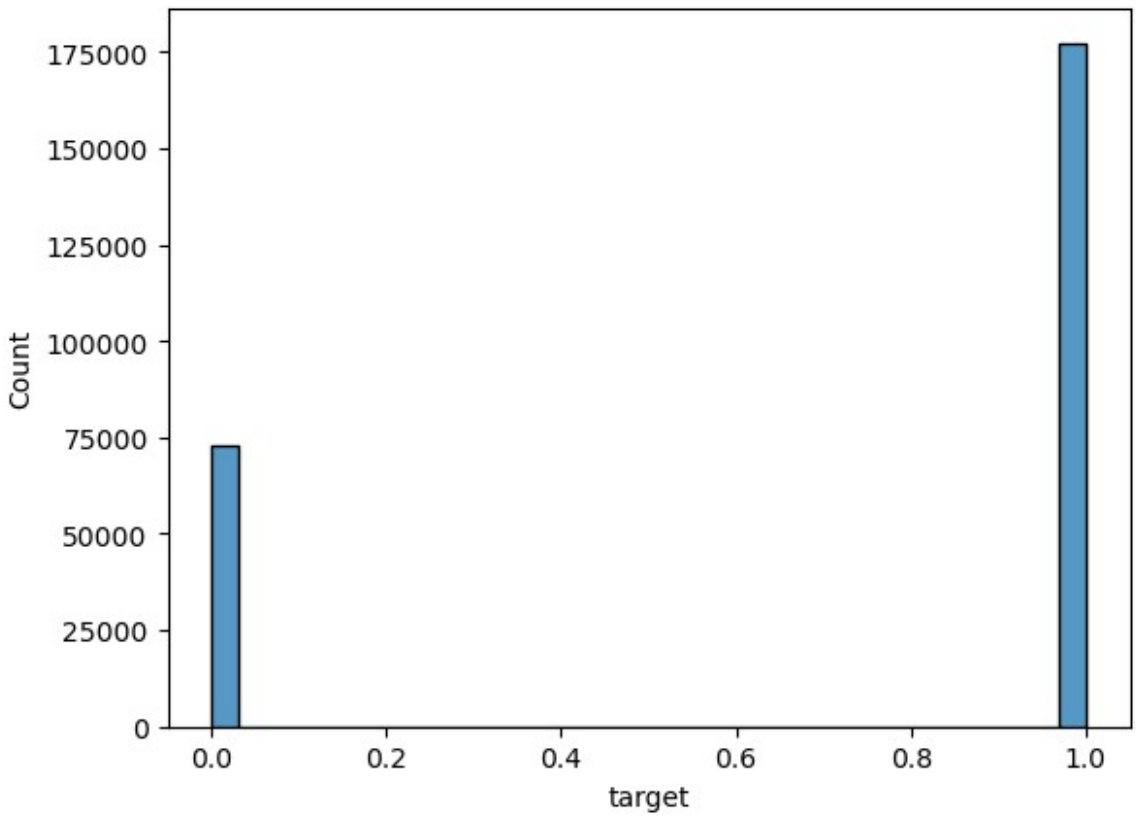
```
feature_111       float64
feature_96        float64
feature_75        float64
feature_154       float64
feature_162       float64
feature_140       float64
feature_56        float64
feature_164       float64
feature_223       float64
feature_132       float64
feature_115       float64
feature_183       float64
feature_70        float64
feature_36        float64
feature_94        float64
feature_100       float64
feature_39        float64
feature_206       float64
feature_12        float64
feature_97        float64
feature_135       float64
feature_170       float64
feature_32        float64
feature_9         float64
feature_213       float64
feature_142       float64
feature_138       float64
feature_156       float64
feature_23        float64
feature_155       float64
feature_65        float64
feature_216       float64
feature_5         float64
dtype: object
```

Предварительно небольшой EDA по таргету

Посмотрим на таргет

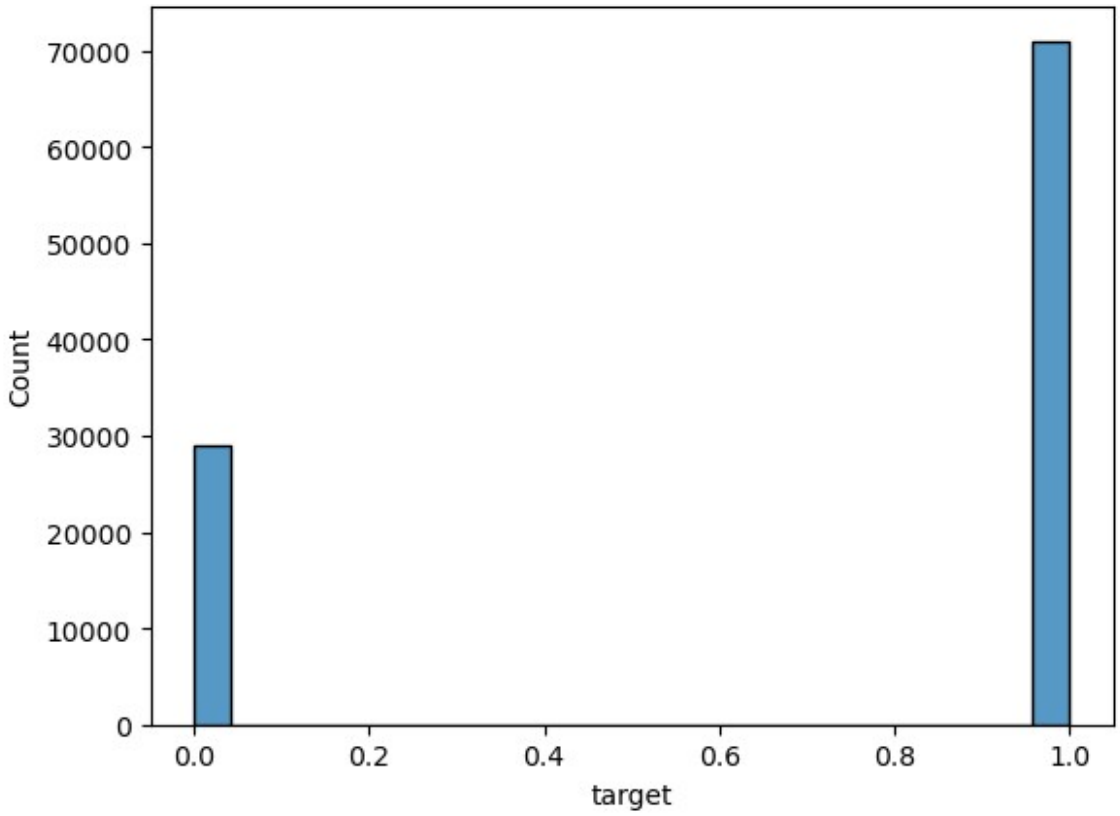
```
sns.histplot(train[TARGET])
```

```
<Axes: xlabel='target', ylabel='Count'>
```




```
sns.histplot(val[TARGET])
```

```
<Axes: xlabel='target', ylabel='Count'>
```



Выводы: дисбаланс классов. Желательно использовать стратифицированное сэмплирование

Бустинг

Подбор гиперпараметров бустинга (3 балла)

Теперь подберем оптимальные гиперпараметры бустинга.

```
# В переменную features_optuna положите список признаков с предыдущего шага
# features_optuna = <your features>

features_optuna = features_final
```

Задание: Заполните пропуски в коде ниже и подберите оптимальные гиперпараметры.

Для успешного решения необходимо преодолеть порог ROC-AUC `0.725` на тестовой выборке.

Я немного изменю код, так как сплит сделал самостоятельно (тут он выбирается через фильтр)

```
def objective(trial):
    params = {
        'learning_rate': trial.suggest_float('learning_rate', 0.01,
0.2, log=True),
        'num_leaves': trial.suggest_int('num_leaves', 10, 200),
        #####
        # your code here
        # Тут вы можете добавить любые гиперпараметры LGBMClassifier
(например, что-то из того, что разбирали в лекции)
        #####

        # что-то базовое
        'objective': 'binary',
        'metric': 'auc',
        'verbosity': -1,
        'boosting_type': 'gbdt',


        "n_estimators": trial.suggest_int("n_estimators", 200, 3000),

        "max_depth": trial.suggest_int("max_depth", 3, 12),

        'feature_fraction': trial.suggest_float('feature_fraction',
0.5, 1.0),
        'bagging_fraction': trial.suggest_float('bagging_fraction',
0.5, 1.0),
        'bagging_freq': trial.suggest_int('bagging_freq', 1, 10),
        # 'min_child_samples': trial.suggest_int('min_child_samples',
5, 100),
        'lambda_l1': trial.suggest_float('lambda_l1', 1e-8, 10.0,
log=True),
        'lambda_l2': trial.suggest_float('lambda_l2', 1e-8, 10.0,
log=True),

        # интересно попробовать, нашел в документации
        'data_sample_strategy': 'bagging', # оставлю так, не очень
понял преимущества GOSS в нашей задаче
        # early_stopping_round

        # Note: while enabling this should increase the overall
performance metric of your model,
        # it will also result in poor estimates of the individual
class probabilities
        # Note: this parameter cannot be used at the same time with
scale_pos_weight, choose only one of them
        'is_unbalance': True,

        'n_jobs': -1,

        'random_state': 42
```

```
    }

    # clf = LGBMClassifier(**params)
    # clf.fit(
    #     X=df.loc[df['sample_part'] == 'train', features_optuna],
    #     y=df.loc[df['sample_part'] == 'train', TARGET]
    # )
    # preds = clf.predict_proba(df.loc[df['sample_part'] == 'val',
features_optuna])[:, 1]
    # auc_valid = roc_auc_score(
    #     y_true=df.loc[df['sample_part'] == 'val', TARGET],
    #     y_score=preds
    # )
    clf = LGBMClassifier(**params)

    clf.fit(
        X=train[features_optuna],
        y=train[TARGET]
    )

    preds = clf.predict_proba(val[features_optuna])[:, 1]
    auc_valid = roc_auc_score(
        y_true=val[TARGET],
        y_score=preds
    )

    return auc_valid

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=10)
```

```
[I 2026-02-08 13:22:17,306] A new study created in memory with name:
no-name-78fd8dd0-542f-4952-850f-c7176dc4fa44
[I 2026-02-08 13:23:40,581] Trial 0 finished with value:
0.7474958606119476 and parameters: {'learning_rate':
0.015510297061334858, 'num_leaves': 116, 'n_estimators': 951,
'max_depth': 11, 'feature_fraction': 0.8962408162908644,
'bagging_fraction': 0.5721189932841437, 'bagging_freq': 8,
'lambda_l1': 0.25724419435600104, 'lambda_l2': 3.8168401580328e-08}.
Best is trial 0 with value: 0.7474958606119476.
[I 2026-02-08 13:26:53,329] Trial 1 finished with value:
0.7541313127731909 and parameters: {'learning_rate':
0.016213044456709194, 'num_leaves': 187, 'n_estimators': 2017,
'max_depth': 8, 'feature_fraction': 0.59584044357557,
'bagging_fraction': 0.9181322319435823, 'bagging_freq': 7,
'lambda_l1': 5.456326538112465e-07, 'lambda_l2': 0.11702251446504433}.
Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:27:38,240] Trial 2 finished with value:
0.7418894657600777 and parameters: {'learning_rate':
0.02978820147412473, 'num_leaves': 26, 'n_estimators': 1261,
```

```
'max_depth': 6, 'feature_fraction': 0.8981166537549339,
'bagging_fraction': 0.7520560730589712, 'bagging_freq': 2,
'lambda_l1': 0.00015125913505380706, 'lambda_l2': 1.4494336199872998e-
06}. Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:28:18,062] Trial 3 finished with value:
0.7137549111219037 and parameters: {'learning_rate':
0.014366240070782871, 'num_leaves': 72, 'n_estimators': 2084,
'max_depth': 3, 'feature_fraction': 0.9251480966505856,
'bagging_fraction': 0.6147247435222888, 'bagging_freq': 2,
'lambda_l1': 1.395291854930904e-08, 'lambda_l2': 9.735737497333495e-
08}. Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:29:14,371] Trial 4 finished with value:
0.7484255041282175 and parameters: {'learning_rate':
0.04010186151792067, 'num_leaves': 102, 'n_estimators': 594,
'max_depth': 9, 'feature_fraction': 0.9210198646265442,
'bagging_fraction': 0.8608450792682913, 'bagging_freq': 2,
'lambda_l1': 1.5815514540096063e-06, 'lambda_l2': 2.102694802385865e-
05}. Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:29:31,038] Trial 5 finished with value:
0.7412752778047595 and parameters: {'learning_rate':
0.12882084095468346, 'num_leaves': 161, 'n_estimators': 642,
'max_depth': 4, 'feature_fraction': 0.6247215086005853,
'bagging_fraction': 0.6145676185556561, 'bagging_freq': 2,
'lambda_l1': 0.00022131431647660492, 'lambda_l2': 9.062829376002267e-
05}. Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:30:04,956] Trial 6 finished with value:
0.7428469329771734 and parameters: {'learning_rate':
0.19104544605289248, 'num_leaves': 42, 'n_estimators': 908,
'max_depth': 5, 'feature_fraction': 0.9675860305005817,
'bagging_fraction': 0.7895415656984563, 'bagging_freq': 3,
'lambda_l1': 3.354476430964262e-05, 'lambda_l2':
0.0037162072613724333}. Best is trial 1 with value:
0.7541313127731909.
[I 2026-02-08 13:31:12,134] Trial 7 finished with value:
0.7404893904808159 and parameters: {'learning_rate':
0.16193258237950078, 'num_leaves': 145, 'n_estimators': 2214,
'max_depth': 5, 'feature_fraction': 0.5441976982995569,
'bagging_fraction': 0.5966130832861942, 'bagging_freq': 7,
'lambda_l1': 0.00029144881441069424, 'lambda_l2': 6.149250438715074e-
08}. Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:32:30,040] Trial 8 finished with value:
0.7467011491015056 and parameters: {'learning_rate':
0.07714953999377144, 'num_leaves': 130, 'n_estimators': 790,
'max_depth': 12, 'feature_fraction': 0.5788501597762103,
'bagging_fraction': 0.5217481755810247, 'bagging_freq': 7,
'lambda_l1': 0.9619113047531965, 'lambda_l2': 1.457639991397227e-07}.
Best is trial 1 with value: 0.7541313127731909.
[I 2026-02-08 13:35:52,589] Trial 9 finished with value:
0.7541726265177271 and parameters: {'learning_rate':
```

```
0.022511004517451885, 'num_leaves': 96, 'n_estimators': 2485,
'max_depth': 10, 'feature_fraction': 0.6349898354943739,
'bagging_fraction': 0.6072608207402718, 'bagging_freq': 10,
'lambda_l1': 0.0002798699047466125, 'lambda_l2': 7.151765760948411e-
08}. Best is trial 9 with value: 0.7541726265177271.
```

Не вижу улучшений, остановлюсь, так как обучаюсь очень долго с ноутбука

```
print(study.best_params)

{'learning_rate': 0.022511004517451885, 'num_leaves': 96,
'n_estimators': 2485, 'max_depth': 10, 'feature_fraction':
0.6349898354943739, 'bagging_fraction': 0.6072608207402718,
'bagging_freq': 10, 'lambda_l1': 0.0002798699047466125, 'lambda_l2':
7.151765760948411e-08}
```

Зафиксирую для train/val

```
params = study.best_params

clf = LGBMClassifier(**params)

clf.fit(
    X=train[features_optuna],
    y=train[TARGET]
)

preds = clf.predict_proba(val[features_optuna])[:, 1]
auc_valid = roc_auc_score(
    y_true=val[TARGET],
    y_score=preds
)

# clf = LGBMClassifier(**study.best_params)
# clf.fit(
#     X=df.loc[df['sample_part'] == 'train', features_optuna],
#     y=df.loc[df['sample_part'] == 'train', TARGET]
# )

# preds_test = clf.predict_proba(df.loc[df['sample_part'] == 'test',
features_optuna])[:, 1]
# auc_test = roc_auc_score(
#     y_true=df.loc[df['sample_part'] == 'test', TARGET],
#     y_score=preds_test
# )
# assert auc_test > 0.725, f'Необходимое значение ROC-AUC 0.725 и
выше, ваше значение: {auc_test}!'

clf = LGBMClassifier(**study.best_params)
```

```
clf.fit(
    X=train[features_optuna],
    y=train[TARGET]
)

preds_test = clf.predict_proba(test[features_optuna])[:, 1]
auc_test = roc_auc_score(
    y_true=test[TARGET],
    y_score=preds_test
)

assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725 и выше,
ваше значение: {auc_test}!"

auc_test
```

```
0.7395093065500715
```

Кривые накопления (2 балла)

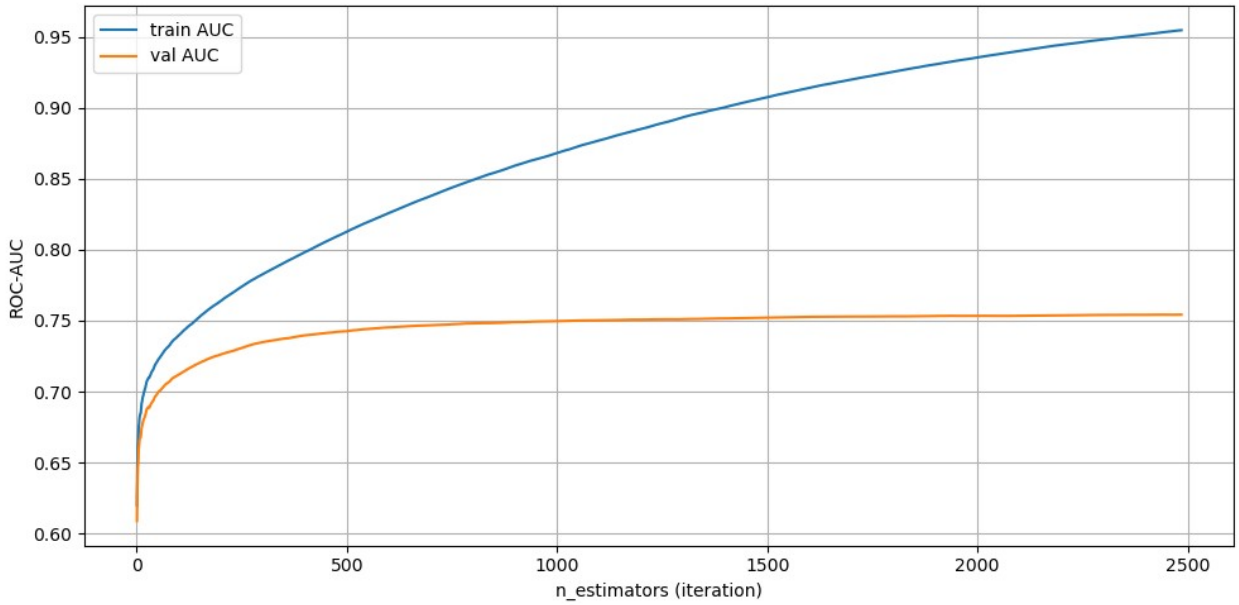
Задание: Отрисуйте кривые накопления `ROC AUC` от `n_estimators` для бустинга с гиперпараметрами из предыдущего пункта.

Hint: для получения auc на каждой итерации обучения `LGBMClassifier`, можно воспользоваться параметрами `eval_set`, `eval_metric` и в методе `fit`.

Имеется в виду именно число последовательно добавляемых деревьев же, не итерации точно? n_estimators (int, optional (default=100)) – Number of boosted trees to fit.

```
#####
# <your code here>
#####

params = study.best_params.copy()
params.update({"random_state": 42, "n_jobs": -1, "verbosity": -1})

clf = LGBMClassifier(**params)
clf.fit(
    X=train[features_optuna],
    y=train[TARGET],
    eval_set=[(train[features_optuna], train[TARGET]),
              (val[features_optuna],   val[TARGET])],
    eval_metric="auc",
)

res = clf.evals_result_
k_train, k_val = list(res.keys())[:2]

train_auc = res[k_train]["auc"]
val_auc   = res[k_val]["auc"]
```

```
iters = np.arange(1, len(val_auc) + 1)

plt.figure(figsize=(10, 5))
plt.plot(iters, train_auc, label="train AUC")
plt.plot(iters, val_auc,   label="val AUC")
plt.xlabel("n_estimators (iteration)")
plt.ylabel("ROC-AUC")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```




```
preds = clf.predict_proba(val[features_optuna])[:, 1]
auc_valid = roc_auc_score(val[TARGET], preds)
print("val ROC-AUC:", auc_valid, "| best val AUC on curve:",
float(np.max(val_auc)), "at iter", int(np.argmax(val_auc)+1))
```

```
val ROC-AUC: 0.7542469407479359 | best val AUC on curve:
0.7542554264205925 at iter 2483
```

Вопрос: Что вы видите на полученных графиках?

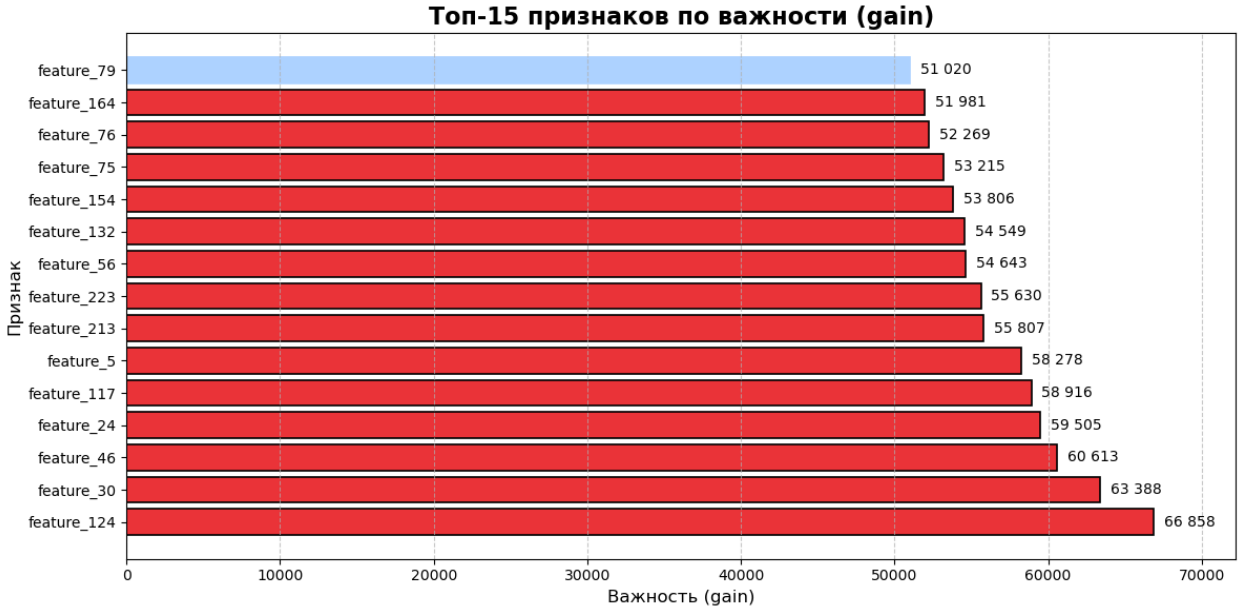
Нужно ли "обрезать" количество деревьев? Ответ обоснуйте.

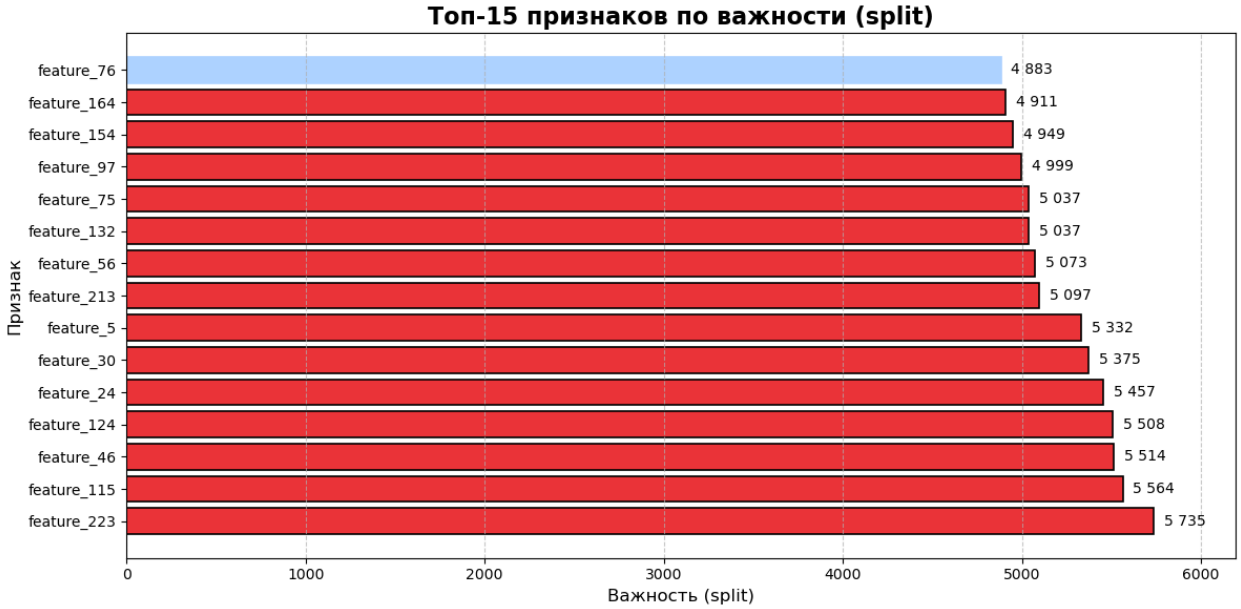
Да, считаю, что нужно; можно ограничить параметр `n_estimators=+-1000`, - уже даст в целом достаточное количество деревьев для хорошего фита, но мы не будем тратить лишний раз время тяжелые итерации при переборе в Optuna. Сам по себе прирост есть послее значения `1000`, но он уже достаточно незначительный и нельзя с уверенностью сказать, что этот прирост заметен на test (если в принципе есть и мы не переобучаемся).

Важность признаков (1 балл)

Задание: Постойте гистограмму важности признаков по `split` и `gain` для бустинга из предыдущего пункта.

```
import lightgbm as lgb

model_lgb = clf
booster = model_lgb.booster_

for imp_type, title in [("gain", "Топ-15 признаков по важности
(gain)"),
                        ("split", "Топ-15 признаков по важности
(split)")]:
    imp = pd.Series(
        booster.feature_importance(importance_type=imp_type),
        index=booster.feature_name()
    ).sort_values(ascending=False).head(15)

    plt.figure(figsize=(12, 6))
    bars = plt.barh(
        imp.index[::-1],
        imp.values[::-1],
        color="#EA3338",
        edgecolor="black",
        linewidth=1.2
    )
    if len(bars) > 0:
        bars[0].set_color("#ADD2FF")

    xmax = max(imp.values) if len(imp) else 0
    pad = xmax * 0.01
    for b in bars:
        w = b.get_width()
        plt.text(w + pad, b.get_y() + b.get_height()/2,
                 f"{w:,.0f}".replace(",", " "),
                 va="center", ha="left", fontsize=10)

    plt.title(title, fontsize=16, fontweight="bold")
    plt.xlabel(f"Важность ({imp_type})", fontsize=12)
    plt.ylabel("Признак", fontsize=12)
    plt.gca().invert_yaxis()
    plt.grid(True, linestyle="--", alpha=0.7, axis="x")
    plt.xlim(0, xmax * 1.08)
    plt.tight_layout()
    plt.show()
```





Вопрос: Какие выводы можно сделать из полученных графиков?

В обоих топах повторяются многие фичи (например feature_164, feature_76, feature_154, feature_75, feature_132, feature_56, feature_213, feature_5, feature_30, feature_24, feature_46, feature_124, feature_223).

Глобально, важно обращать внимание по большей части именно на gain. Но, для окончательных выборов, я бы попробовал разные техники отбора признаков (попробую в творческой части задания!)

Логрег

plot_metric_time (1 балл)

Задание: Реализуйте функцию plot_metric_time, которая будет принимать на вход четыре аргумента, а именно:

- массив значений фичи,
- массив значений таргета,
- массив времени (аггрегированного по месяцам или кварталам, то есть `month` или `quarter` в вашей задаче, по дням рисовать не надо),
- метрику, которую необходимо отрисовать (нужно реализовать функцию для двух метрик: `IV` и `roc_auc`)
- число бакетов для вычисления `IV`, если выбрана эта метрика

Можете добваить какие-то ещё аргументы, если вам нужно

Если в фиче есть пропуски, функция должна убирать строки с пропусками из рассмотрения

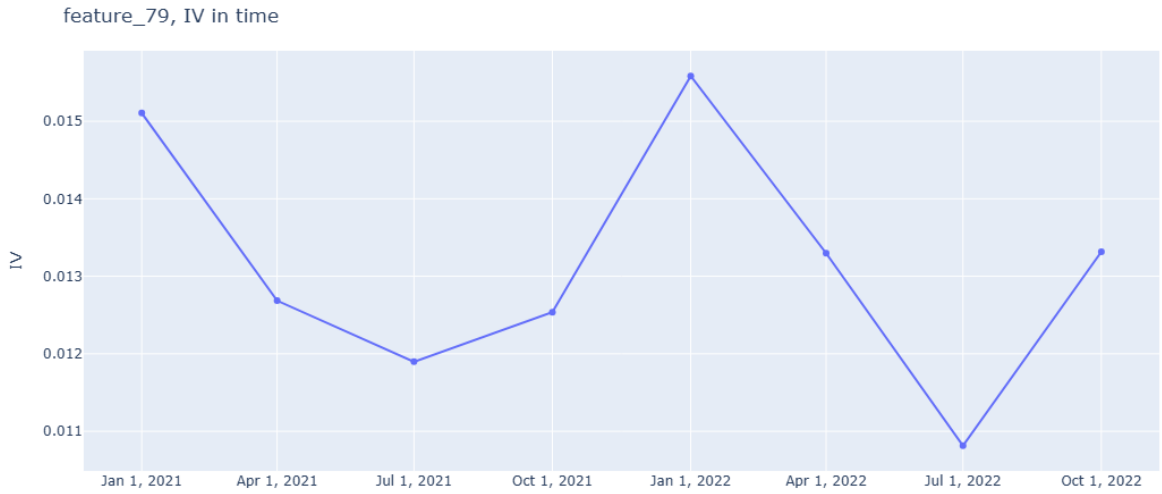
Hint: можно, конечно, реализовать эту функцию через цикл, а можно попробовать разобраться и реализовать её через пандасовские `groupby` -> `apply`, это изящнее и быстрее

```
import plotly.graph_objects as go

def plot_metric_time(
        values: pd.Series,
        target: pd.Series,
        time: pd.Series,
        metric: str = 'IV',
        n_buckets_for_IV: int = 15
) -> go.Figure:

    # код тут
    dfm = pd.DataFrame({"x": values, "y": target, "t":
time}).dropna(subset=["x", "y", "t"])
    dfm["y"] = dfm["y"].astype(int)

    if metric == "IV":
        def _iv_group(g):
            x = g["x"]
            y = g["y"].to_numpy()

            k = min(n_buckets_for_IV, x.nunique(dropna=True))
            if k <= 1:
                return 0.0

            try:
                buckets = pd.qcut(x, q=k, labels=False,
```

```
duplicates="drop").to_numpy()
            except Exception:
                buckets = pd.cut(x, bins=k, labels=False).to_numpy()

            return calc_iv(values=x.to_numpy(), target=y,
buckets=buckets)

        series_metric = dfm.groupby("t", sort=True).apply(_iv_group)

    elif metric == "roc_auc":
        def _auc_group(g):
            y = g["y"]
            x = g["x"]
            if y.nunique() < 2:
                return np.nan
            return roc_auc_score(y_true=y, y_score=x)

        series_metric = dfm.groupby("t", sort=True).apply(_auc_group)

    else:
        raise NotImplementedError("only 'IV' and 'roc_auc' metrics are
implemented")

    series_metric = series_metric.sort_index()
    x_axis = series_metric.index
    y_axis = series_metric.values

    plot_title = f"{values.name}, {metric} in time"
    fig = go.Figure()
    fig.add_trace(
        go.Scatter(
            x=x_axis,
            y=y_axis,
            mode="markers+lines",
            name=values.name
        )
    )
    fig.update_layout(
        title_text=plot_title,
        yaxis=dict(title=metric),
        width=1000,
        height=450,
        xaxis=dict(
            domain=[0, 0.95],
            showgrid=True,
            tickvals=list(x_axis),
        ),
        margin=dict(l=30, r=30, b=30, t=50),
    )
```

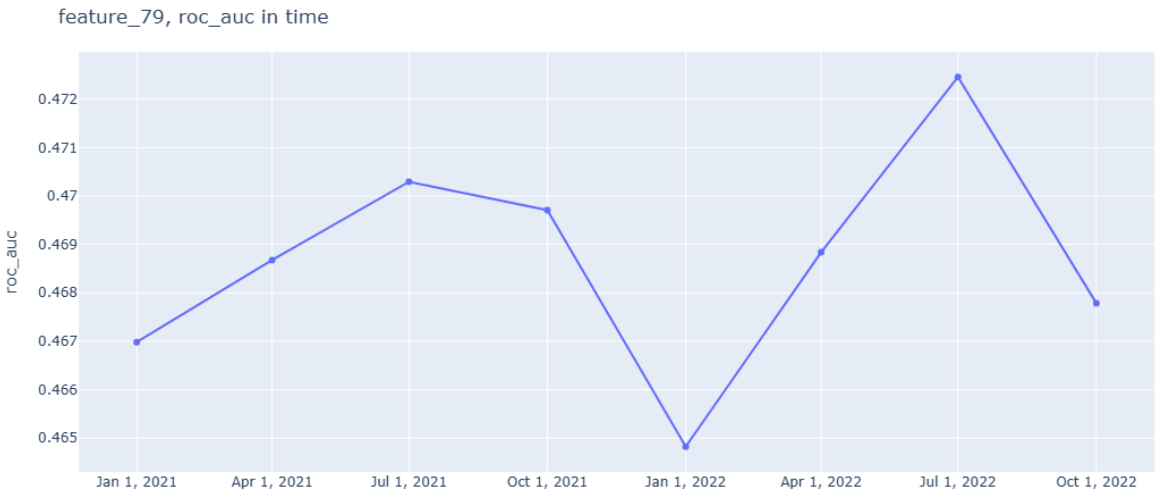
```
    # fig.show()
    return fig

# # примерно так это должно будет выглядеть
# plot_metric_time(df[...], df['target'], df['quarter']) #

# примерно так это должно будет выглядеть
plot_metric_time(df["feature_79"], df["target"], df["quarter"],
metric="IV", n_buckets_for_IV=15)
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```




```
plot_metric_time(df["feature_79"], df["target"], df["quarter"],
metric="roc_auc")
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:39:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```



Задание: Возьмите **топ-15** фичей получившегося бустинга по важности по `gain`. Отрисуйте для них графики стабильности по `IV` во времени и удалите из рассмотрения те признаки, качество которых деградирует

NB! Обращайте внимание на масштаб оси *y!* Иногда признак стабильнее, чем кажется)

Если вам это мешает, можете поменять ось *y*, чтобы она начиналась от нуля в прошлом задании.

```
# your code here
```

Список

```
features = [
    "feature_79",
    "feature_164",
    "feature_76",
    "feature_75",
    "feature_154",
    "feature_132",
    "feature_56",
    "feature_223",
    "feature_213",
    "feature_5",
    "feature_117",
    "feature_24",
    "feature_46",
    "feature_30",
    "feature_124",
]

train[features].nunique()
```

```
feature_79     250000
feature_164    250000
feature_76     250000
feature_75     250000
feature_154    250000
feature_132    250000
feature_56     250000
feature_223    213266
feature_213    250000
feature_5      250000
feature_117    250000
feature_24     250000
feature_46     250000
feature_30     232740
feature_124    220520
dtype: int64
```

```
for f in features:
    fig = plot_metric_time(train[f], train[TARGET], train["month"],
metric="IV", n_buckets_for_IV=20)
    fig.show()
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:
```

```
DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```

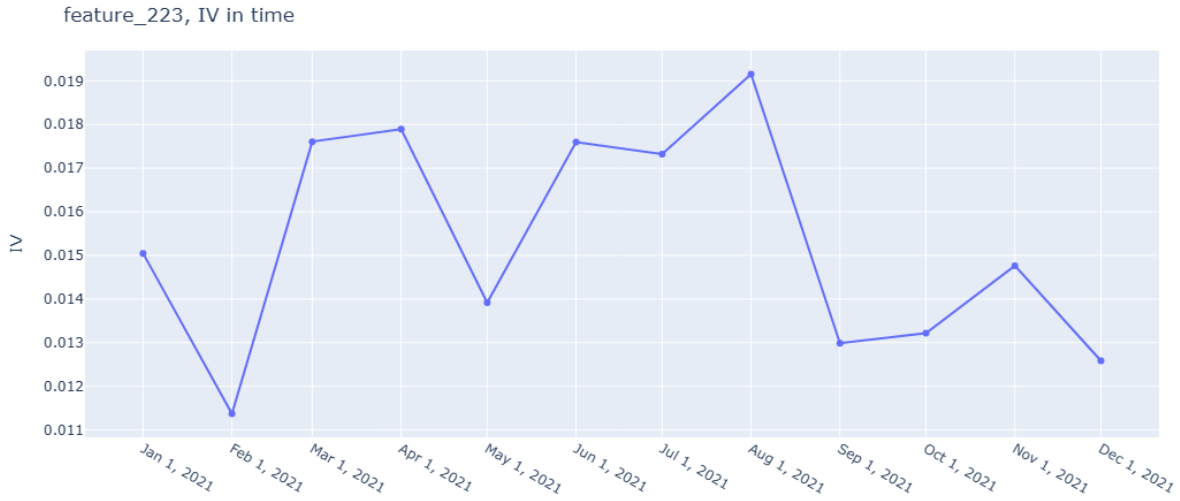

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
```

```
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```




```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
```

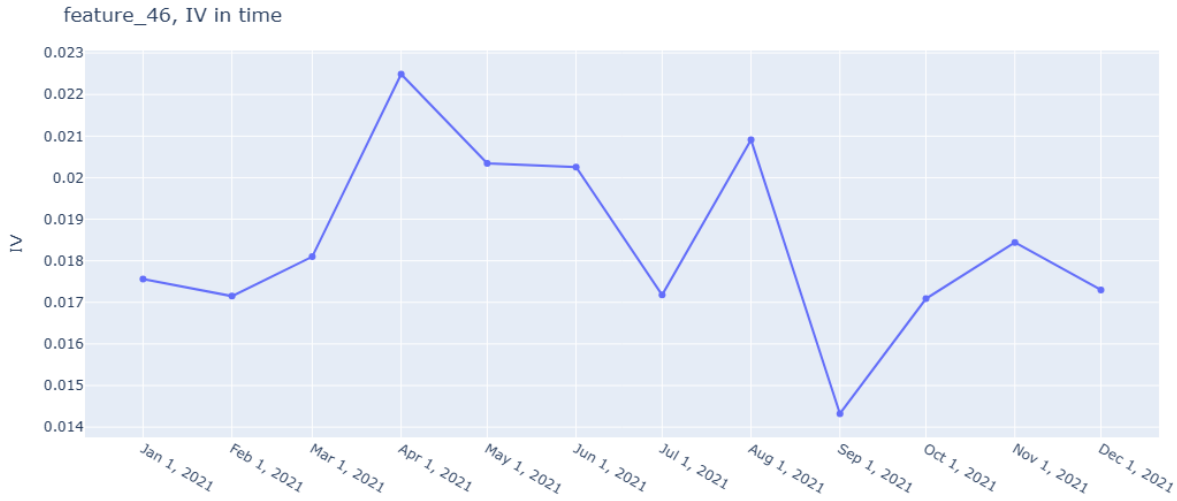
```
is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```




```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
```

```
the grouping columns after groupby to silence this warning.
```


```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\1679618361.py:29:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```

Лично мне не очень нравится признак 132, - слишком сильное падание заметно. Я бы убрал только его, так как с остальными нельзя однозначно сказать, что IV имеет именно негативный тренд, а не просто временно падает.

Предлагаю подытожить признаковый набор:

```
# features_final = selected_features_cat_train +
selected_features_train

features_final = [f for f in features_final if f != "feature_132"]
```

На этом моменте предлагаю сделать новый маленький датафрейм, чтобы не "портить" исходный

И проводить все манипуляции с фичами на нём

```
features_gain = [
    "feature_79",
    "feature_164",
    "feature_76",
    "feature_75",
    "feature_154",
    "feature_132",
    "feature_56",
    "feature_223",
    "feature_213",
    "feature_5",
    "feature_117",
    "feature_24",
    "feature_46",
    "feature_30",
    "feature_124",
]

features_gain = [f for f in features_gain if f != "feature_132"]

top15features_stable = features_gain

df_for_logreg = df[top15features_stable + ['date', 'month', 'quarter',
'target', 'sample_part']]

df_for_logreg[top15features_stable].nunique()
```

```
feature_79     500000
feature_164    500000
feature_76     500000
feature_75     500000
feature_154    500000
feature_56     500000
feature_223    426512
feature_213    500000
feature_5      500000
feature_117    500000
feature_24     500000
feature_46     500000
feature_30     465525
```

```
feature_124    441016
dtype: int64
```

Предобработка признаков (3 балла)

Задание: Используя функцию woe_line из предыдущего ДЗ, проверьте **числовые** фичи из полученного списка фичей на линейность по WoE на трейн-выборке (если в фиче есть пропуски - дропаем их при отрисовке)

Если фичи нелинейные, **линеаризуйте их**.

Преобразования, которые можно/стоит пробовать:

- клипы (`np.clip`) - зачастую их достаточно
- корень
- квадрат
- логарифм

Если нужно, можно прибавлять к фиче константу или менять её знак

При желании можно "распилить фичу на две половины" (если она немонотонна) и линеаризовать их по отдельности

Однако слишком упираться в линеаризацию фичей не нужно. Если фича ну совсем никак не линеаризуется, в крайнем случае можно её дропнуть или оставить как есть.

При отрисовке можно ограничиться 15-20 бактеами

```
# your code here

from typing import Union, Tuple
from scipy.special import logit
from statsmodels.stats.proportion import proportion_confint

from sklearn.metrics import roc_auc_score

def calc_buckets(x : Union[np.ndarray, pd.Series], n_buckets : int) ->
np.ndarray:
    """Разбивает массив значений признака x на
    n_buckets бакетов"""

    x = pd.Series(x).reset_index(drop=True)
    buckets = x.rank(method="dense", pct=True) * n_buckets
    buckets = np.ceil(buckets) - 1   # np.floor дает другой результат
для 5.0, 6.0 и т.д.
    buckets = np.array(buckets, dtype=np.int16)

    return buckets

def woe_ci(target : np.ndarray, buckets : np.ndarray, offset : float)
-> Tuple[pd.Series]:
```

```
    """Для каждого бакета вычисляем WoE и доверительный интервал для
него."""

    woe = {}
    woe_lower = {}
    woe_upper = {}


    for j in np.unique(buckets):
        mask = buckets == j
        y_j = target[mask]

        B_j = (y_j == 1).sum()
        G_j = (y_j == 0).sum()
        n_j = B_j + G_j

        # если WoE не определён
        if B_j == 0 or G_j == 0:
            woe[j] = np.nan
            woe_lower[j] = np.nan
            woe_upper[j] = np.nan
            continue

        # badrate
        badrate_j = B_j / n_j

        # доверительный интервал для badrate по Уилсону
        badrate_low, badrate_up = proportion_confint(
            count=B_j,
            nobs=n_j,
            alpha=0.05,
            method='wilson'
        )

        # переход к WoE
        woe[j] = woe_transform(badrate_j, offset)
        woe_lower[j] = woe_transform(badrate_low, offset)
        woe_upper[j] = woe_transform(badrate_up, offset)


    # # считаем бэдрейт и доверительный интервал для него (любым
способом)
    # badrate, badrate_lower, badrate_upper = None, None, None

    # # переходим от бэдрейта к woe
    # woe, woe_lower, woe_upper = None, None, None

    # return woe, woe_lower, woe_upper

    return (
```

```
            pd.Series(woe).sort_index(),
            pd.Series(woe_lower).sort_index(),
            pd.Series(woe_upper).sort_index()
        )

def calc_buckets_info(values : np.ndarray, target : np.ndarray,
buckets : np.ndarray) -> dict:
    """Для каждого бакета расчитывает
     - среднее значение признака
     - линейную интерполяцию в пространстве woe
     - значение woe и доверительный интервал для него"""

    # buckets_info = {
    #     "mean_feature" : None,
    #     "line"         : None,
    #     "woe"          : None,
    #     "woe_lower"    : None,
    #     "woe_upper"    : None
    # }

    # средние значения признака по бакетам
    mean_feature = (
        pd.Series(values)
        .groupby(buckets)
        .mean()
        .sort_index()
        .values
    )

    # offset
    B = (target == 1).sum()
    G = (target == 0).sum()
    offset = logit(B / (B + G))

    # WoE и доверительные интервалы
    woe, woe_lower, woe_upper = woe_ci(
        target=target,
        buckets=buckets,
        offset=offset
    )

    # линейная интерполяция
    line = calc_line(
        values=values,
        target=target,
        mean_feature=mean_feature,
        offset=offset
    )

    buckets_info = {
```

```
        "mean_feature": mean_feature,
        "line": line,
        "woe": woe,
        "woe_lower": woe_lower,
        "woe_upper": woe_upper
    }

    return buckets_info

def calc_plot_title(
        values : np.ndarray,
        target : np.ndarray,
        buckets : np.ndarray
    ) -> str:
    """Считает для признака roc auc, IV, R^2"""
    # auc = None
    auc = roc_auc_score(target, values)

    # формула выше
    # IV = None

    B = (target == 1).sum()
    G = (target == 0).sum()

    IV = 0.0

    for j in np.unique(buckets):
        mask = buckets == j
        y_j = target[mask]

        B_j = (y_j == 1).sum()
        G_j = (y_j == 0).sum()

        if B_j == 0 or G_j == 0:
            continue

        woe_j = np.log((B_j / G_j) / (B / G))

        IV += (B_j / B - G_j / G) * woe_j

    # используем ранее написанную функцию
    info = calc_buckets_info(
        values=values,
        target=target,
        buckets=buckets
    )

    mean_feature = info["mean_feature"]
    woe = info["woe"].values
    line = info["line"]
```

```
    # веса бакетов - число наблюдений в бакете
    weights = np.array([
        (buckets == j).sum()
        for j in np.unique(buckets)
    ])

    # Взвешенный R^2
    # X - среднее в бакете, Y - woe в бакете, вес - число наблюдений в
бакете
    # R_sqr = None
    w_mean = np.average(woe, weights=weights)
    ss_tot = np.sum(weights * (woe - w_mean) ** 2)

    # линейная инт.
    ss_res = np.sum(weights * (woe - line) ** 2)

    R_sqr = 1 - ss_res / ss_tot

    plot_title = (
        f"AUC = {auc:.3f} "
        f"IV = {IV:.3f} "
        f"R_sqr = {R_sqr:.3f} "
    )

    return plot_title

# в этот код можно не въезжать :)
import plotly.graph_objects as go

def make_figure(buckets_info : dict, plot_title : str) -> go.Figure:
    """Строит график линейности."""
    fig = go.Figure()

    # общие настройки
    title = dict(
        text=plot_title,
        y=0.95,
        x=0.5,
        font=dict(size=12),
        xanchor="center",
        yanchor="top"
    )
    margin = go.layout.Margin(
        l=50,
        r=50,
        b=50,
        t=60
    )
```

```
    fig.add_trace(
        go.Scatter(
            x=buckets_info["mean_feature"],
            y=buckets_info["line"],
            mode='lines',
            name="interpolation_line",
            showlegend=False
        )
    )

    fig.add_trace(
        go.Scatter(
            x=buckets_info["mean_feature"],
            y=buckets_info["woe"],
            line=dict(
                color='firebrick',
                width=1,
                dash='dot'
            ),
            error_y=dict(
                type='data',
                symmetric=False,
                array=buckets_info["woe_upper"],
                arrayminus=buckets_info["woe_lower"]
            ),
            name="WoE",
            showlegend=False)
    )

    fig.update_layout(
        width=1000,
        height=450,
        xaxis_title=dict(
            text='Feature value',
            font=dict(size=12)
        ),
        yaxis_title=dict(
            text="WoE",
            font=dict(size=12)
        ),
        title=title,
        margin=margin
    )

    return fig

def woe_line(
        values : np.ndarray,
        target : np.ndarray,
        n_buckets : int
```

```
    ) -> go.Figure:
    """График линейности переменной по WoE."""
    buckets : np.ndarray = calc_buckets(values, n_buckets)
    buckets_info : pd.DataFrame = calc_buckets_info(values, target,
buckets)
    plot_title : str = calc_plot_title(values, target, buckets)
    fig = make_figure(buckets_info, plot_title)
    return fig


def woe_transform(badrate : float, offset : float) -> float:
    """Считаем WoE для бакета с данным badrate и выборки
    с данным offset."""
    woe = logit(badrate) - offset
    return woe

from sklearn.linear_model import LogisticRegression

def calc_line(values : np.ndarray, target : np.ndarray, mean_feature :
np.ndarray, offset : float) -> np.ndarray:
    """Строим линейную интерполяцию для WoE."""

    # строим логистическую регрессию на одном признаке
    # и считаем ее предсказания в точках - mean_feature

    # логрег на одном признаке
    lr = LogisticRegression(
        # penalty='None', # не регуляризую
        solver='lbfgs' # не принципиально, пробовал разные, результаты
схожие
        # (кроме того, что некоторые солверы игнорируют
мультиколлинеарность)
    )

    # обучаем
    lr.fit(values.reshape(-1, 1), target)

    # предсказываем вероятность bad в точках mean_feature
    proba = lr.predict_proba(mean_feature.reshape(-1, 1))[:, 1]

    # применяем ранее написанную функцию
    line = woe_transform(proba, offset)
    return line
```

Задание: Используя функцию woe_line из предыдущего ДЗ, проверьте **числовые** фичи из полученного списка фичей на линейность по WoE на трейн-выборке (если в фиче есть пропуски - дропаем их при отрисовке)

Если фичи нелинейные, **линеаризуйте их**.

Преобразования, которые можно/стоит пробовать:

- клипы (`np.clip`) - зачастую их достаточно
- корень
- квадрат
- логарифм

Если нужно, можно прибавлять к фиче константу или менять её знак

При желании можно "распилить фичу на две половины" (если она немонотонна) и линеаризовать их по отдельности

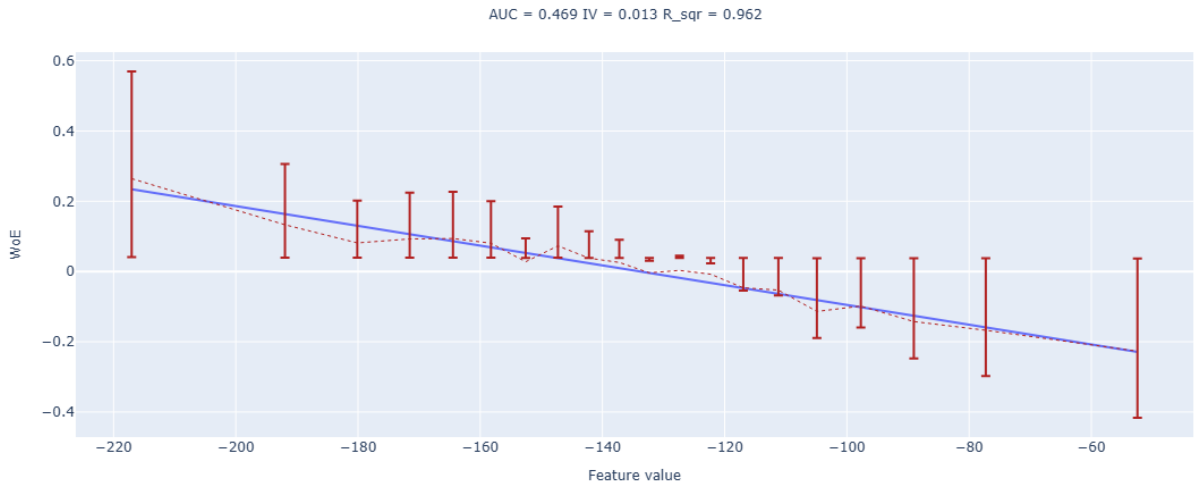
Однако слишком упираться в линеаризацию фичей не нужно. Если фича ну совсем никак не линеаризуется, в крайнем случае можно её дропнуть или оставить как есть.

При отрисовке можно ограничиться 15-20 бактеами

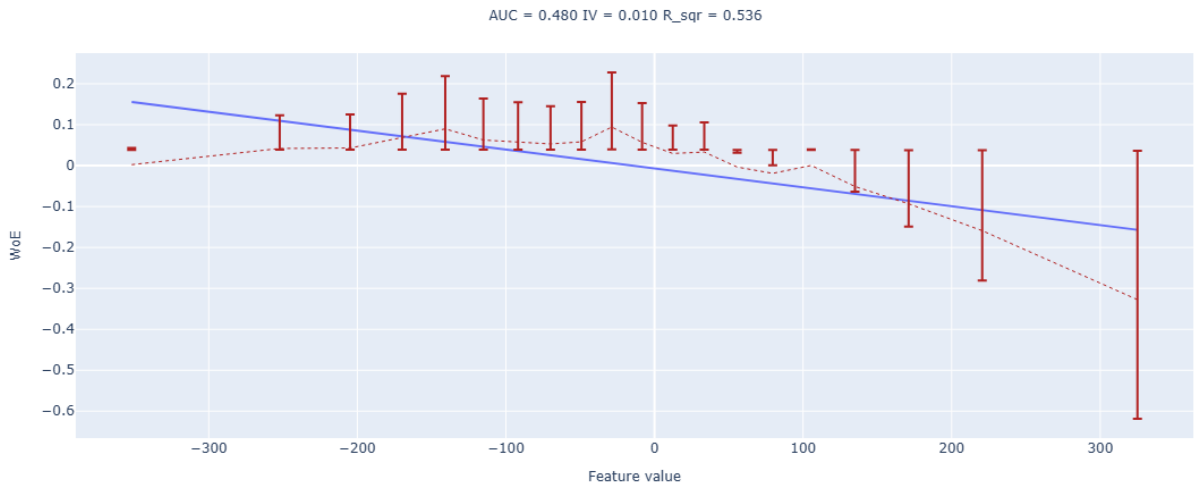
Я буду использовать просто train set (это тот же сет, что и рассматриваем выше как датасет для логистической регрессии! Просто сразу подвыборка train. Причины, почему рассматриваю именно train, описал выше и задания позволяет слегка отклоняться от исходной логики поставленных вопросов).

```
num_feats = [f for f in top15features_stable if
pd.api.types.is_numeric_dtype(train[f])]
num_feats

for f in num_feats:
    print(f'Для признака {f}')
    m = train[f].notna()
    fig = woe_line(train.loc[m, f].to_numpy(), train.loc[m,
TARGET].astype(int).to_numpy(), n_buckets=20)
    fig.show()
```

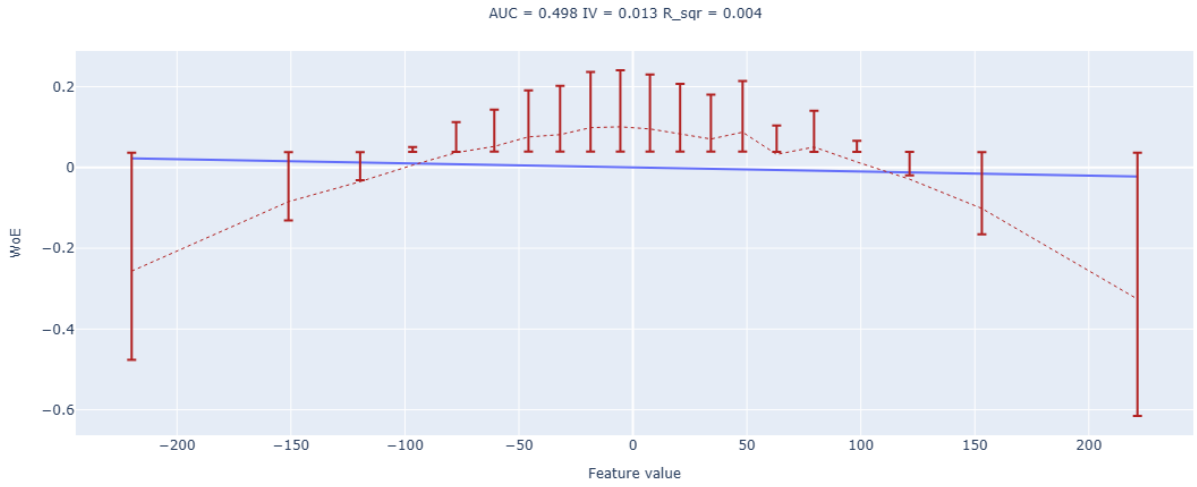
```
Для признака feature_79
```



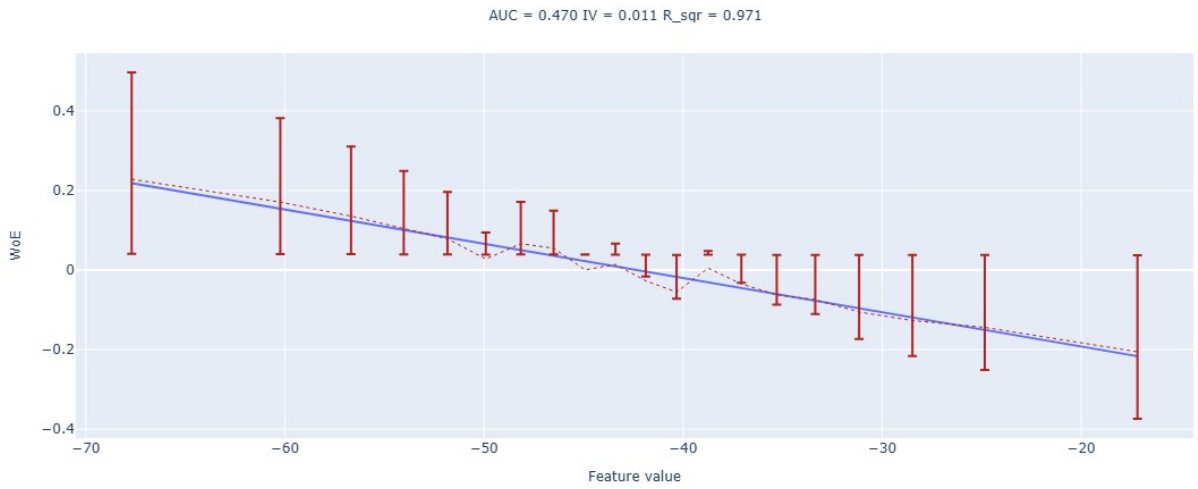
Для признака feature_164



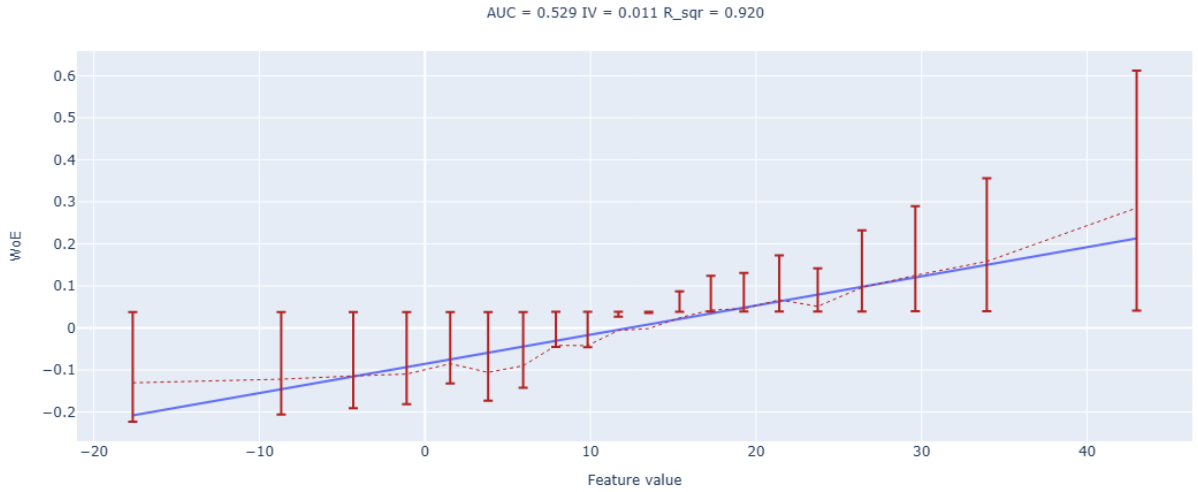
Для признака feature_76



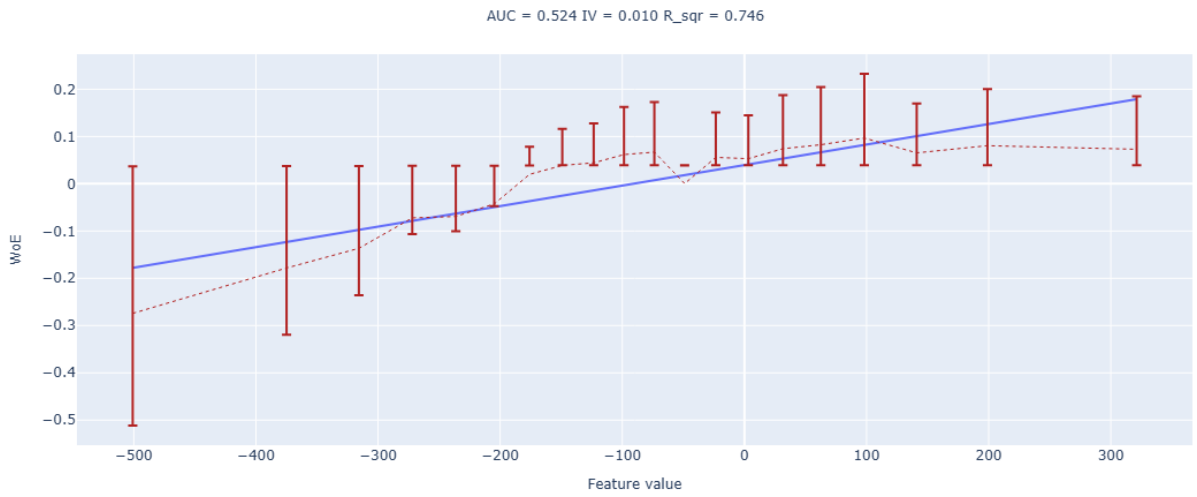
Для признака feature_75



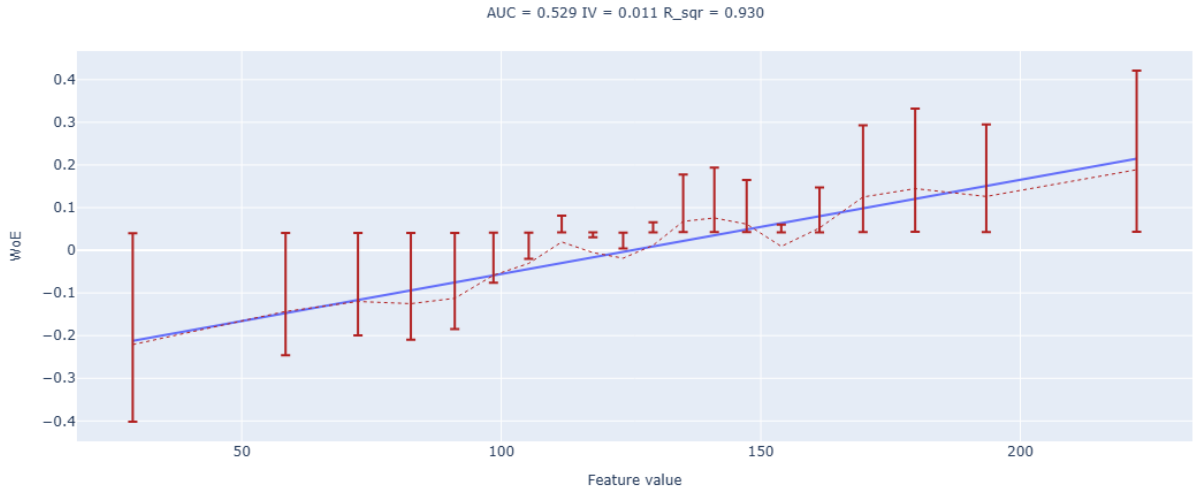

```
Для признака feature_154
```



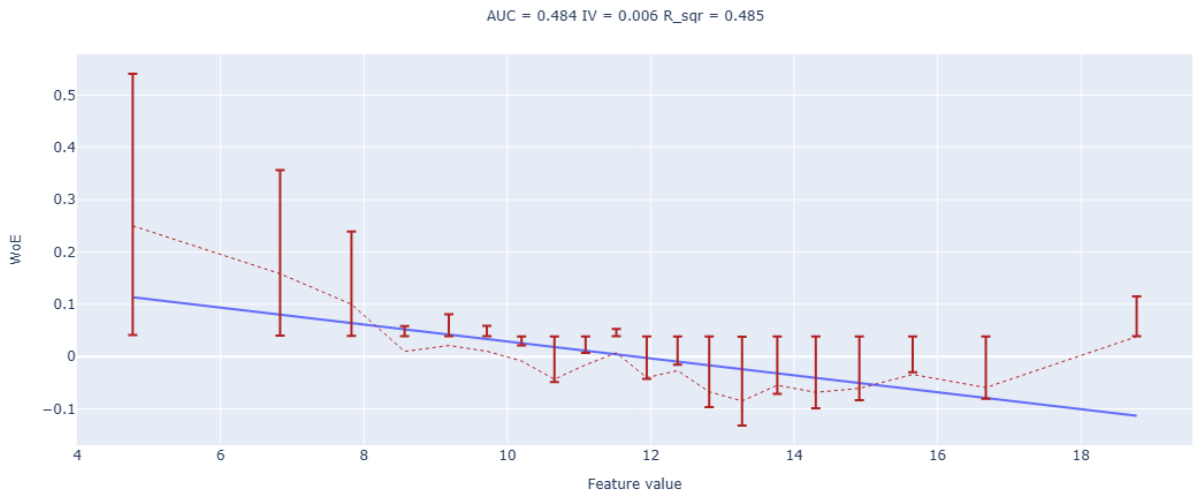

```
Для признака feature_56
```



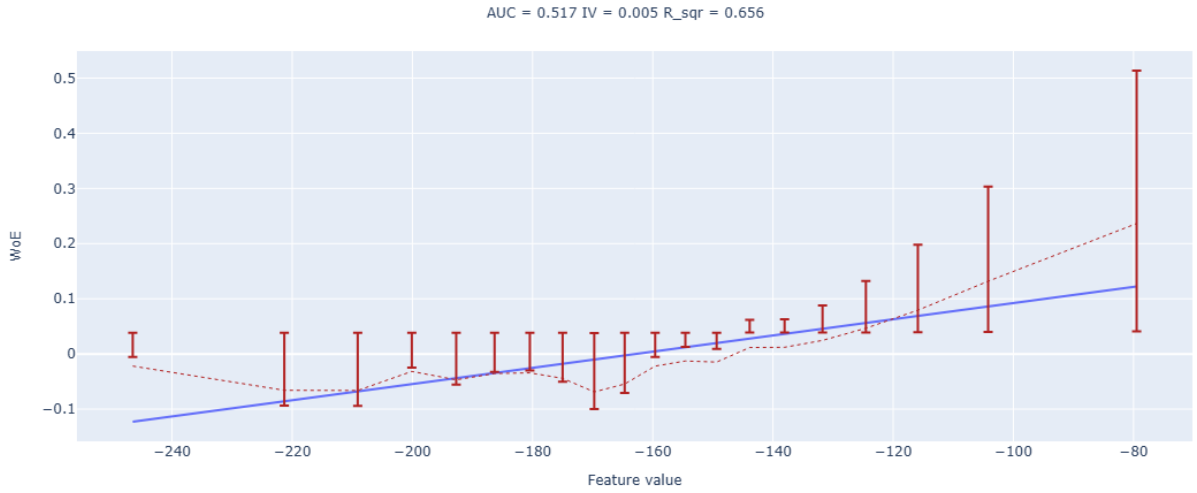

```
Для признака feature_223
```



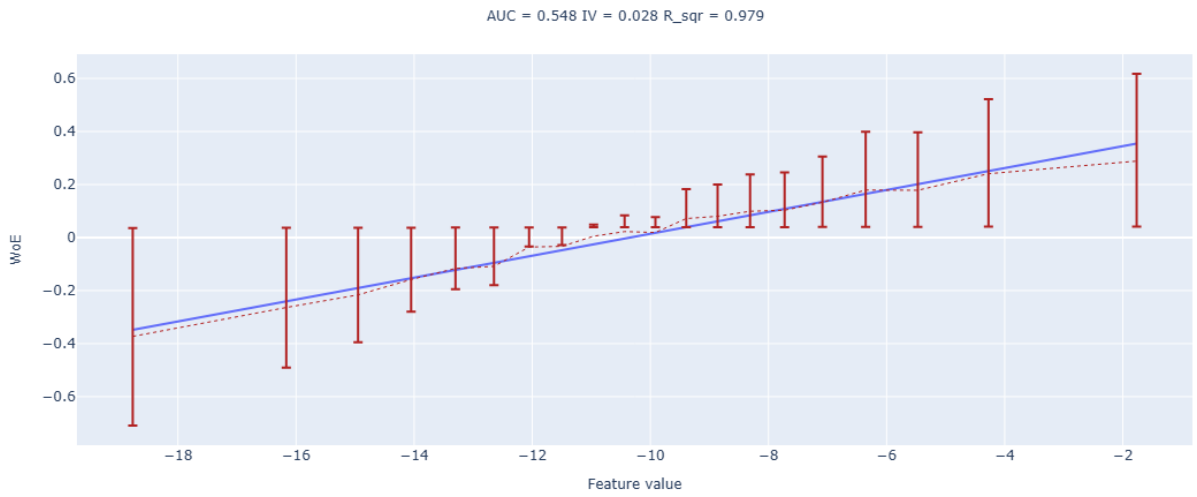

```
Для признака feature_213
```



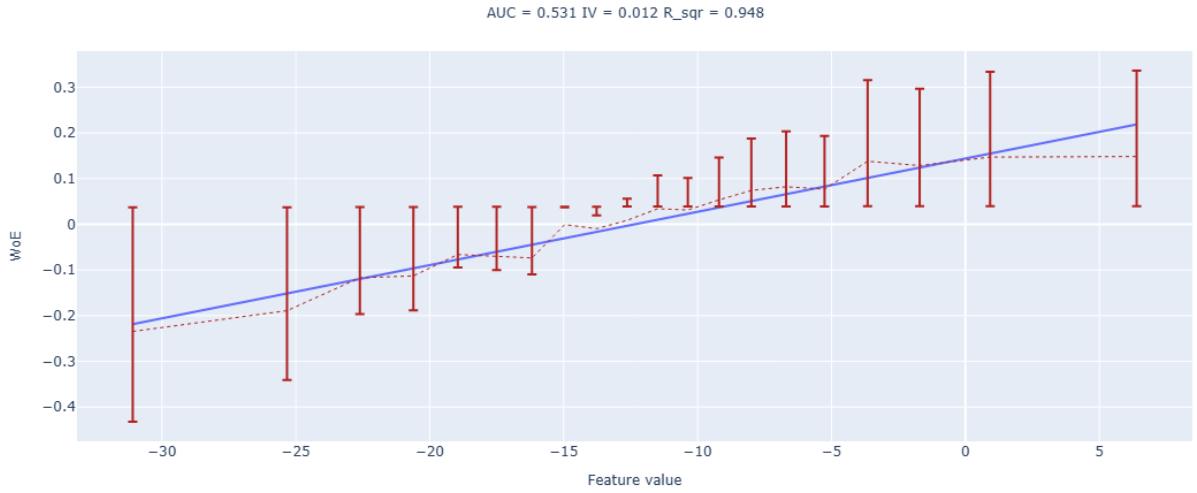

```
Для признака feature_5
```



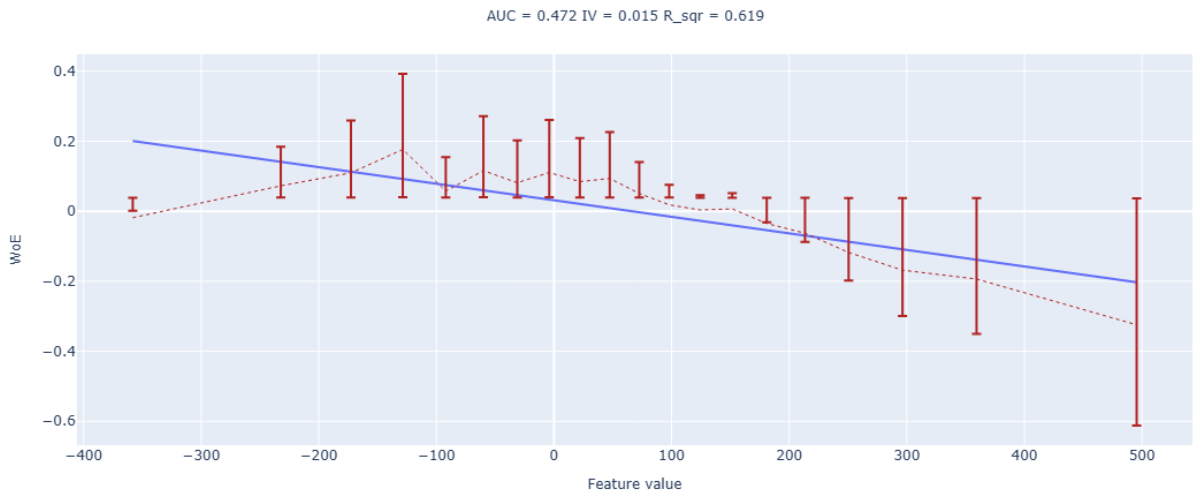

```
Для признака feature_117
```



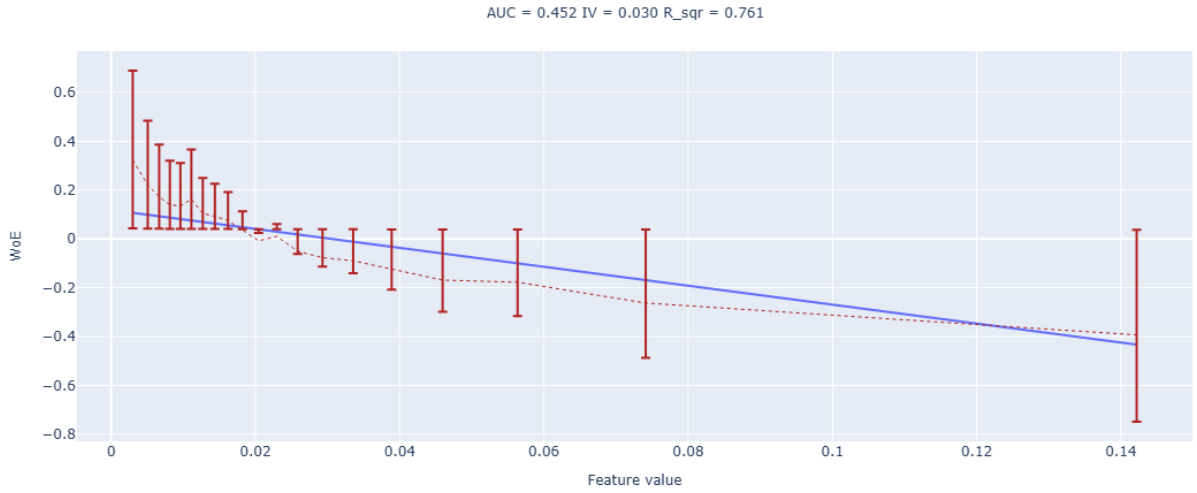
Для признака feature_24



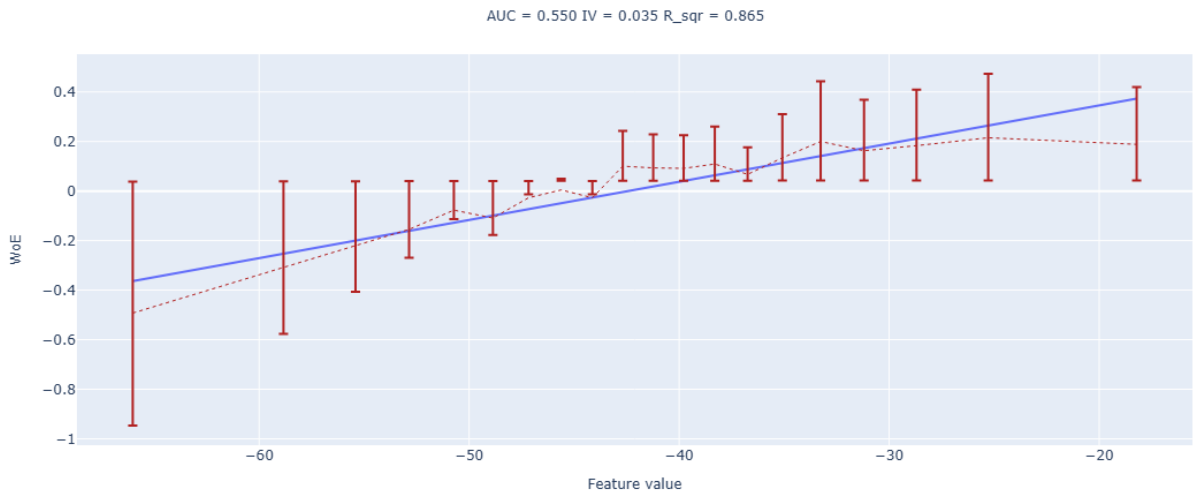
Для признака feature_46



Для признака feature_30



Для признака feature_124



Признаки с индексами:

- 164
- 76
- 56
- 213
- 5
- 46
- 30

Требуют линеаризации. Остальные признаки не пострадают от применения логарифма, так как это просто сгладит распределения и изменит масштабы (спорно, насколько лучше для дерева, хотя, может учиться быстрее, но точно лучше для логистической регрессии). Поэтому применяю логарифмизацию на все признаки из списка.

СРАЗУ ВЕРДИКТ ПО ТЕЗИСУ ВЫШЕ: почти преобразования ухудшают метрики качества, проверил корень, логарифм, клиппинг. Но полиномиальные преобразования вида возведения в квадрат (куб улучшил только у самого проблемного признака и слегка) улучшают метрики У ПРОБЛЕМНЫХ признаков, но не делают этого (скорее, ухудшают) у изначально "хороших" по метрикам. Поэтому применю преобразования только к признакам из списка.

```
df = pd.read_parquet('df.parquet')

train = df.query("sample_part == 'train'")
val = df.query("sample_part == 'val'")
test = df.query("sample_part == 'test'")

sq_cols = [f"{c}_squared" for c in sq_feats]

df_for_logreg = df_for_logreg.drop(columns=sq_cols, errors="ignore")

new_cols = [f"{c}_squared" for c in sq_feats]
print("overlap train:", set(new_cols) & set(train.columns))
```

```
print("overlap val:", set(new_cols) & set(val.columns))
print("overlap test:", set(new_cols) & set(test.columns))
print("overlap df_for_logreg:", set(new_cols) &
set(df_for_logreg.columns))
```

```
overlap train: set()
overlap val: set()
overlap test: set()
overlap df_for_logreg: set()
```

```
sq_feats = [

"feature_164","feature_76","feature_56","feature_213","feature_5","fea
ture_46","feature_30",
]

q_low, q_high = 0.01, 0.99
lo = train[sq_feats].quantile(q_low)
hi = train[sq_feats].quantile(q_high)

def add_squared(df, feats, lo, hi, suffix="_squared"):
    clipped = df[feats].clip(lower=lo, upper=hi, axis=1)
    squared = np.power(clipped, 2)
    squared.columns = [f"{c}{suffix}" for c in feats]
    return df.join(squared)

train = add_squared(train, sq_feats, lo, hi)
val   = add_squared(val,   sq_feats, lo, hi)
test = add_squared(test, sq_feats, lo, hi)

df_for_logreg = add_squared(df_for_logreg, sq_feats, lo, hi)

train['feature_76_squared'].unique()
```

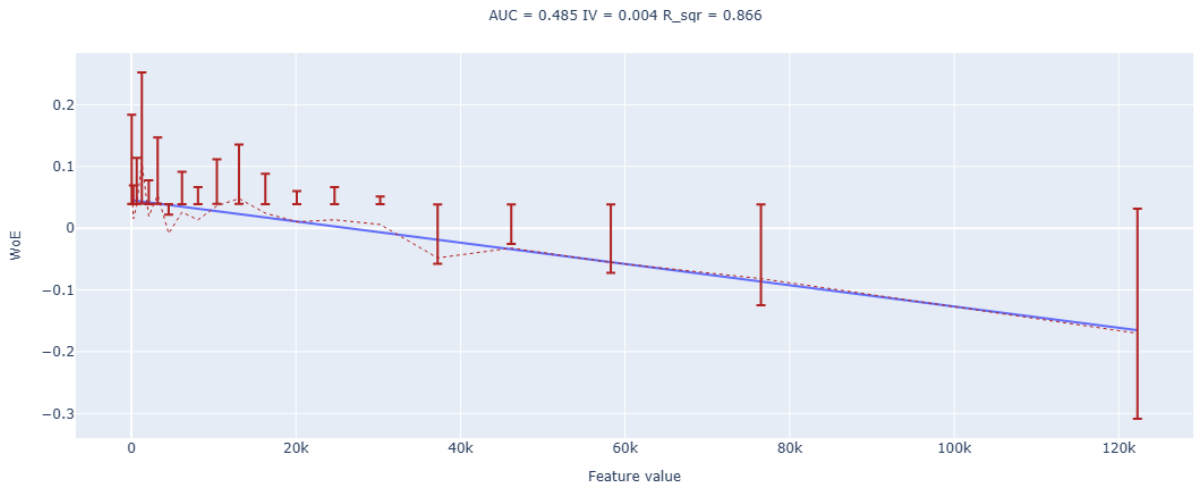
```
array([  600.81985736, 17405.16268576, 16940.64184534, ...,
        5016.75321701,  1343.47584447,  4884.49557618])
```

Проверим!

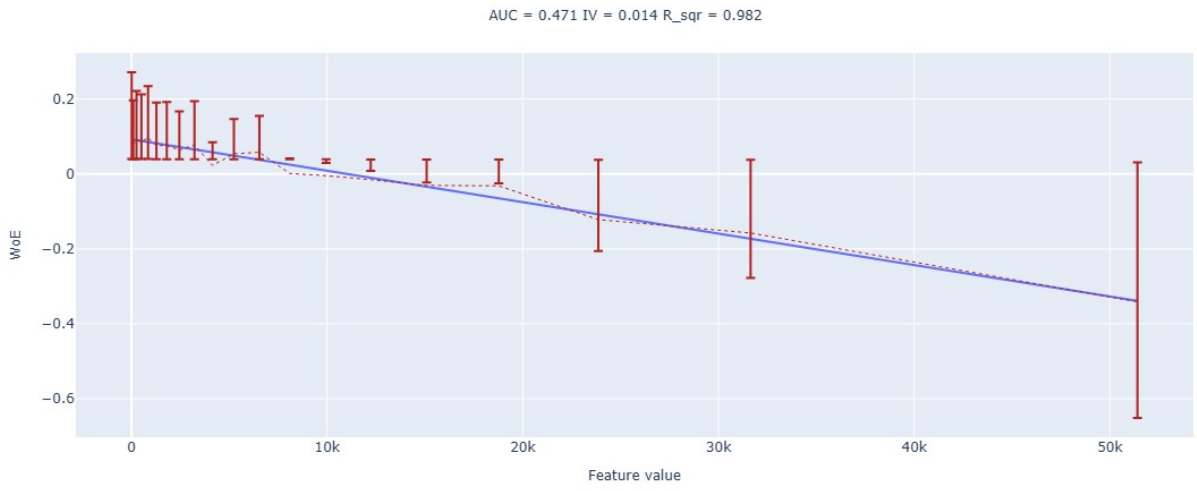
```
sq_feats = [
    "feature_164_squared",
    "feature_76_squared",
    "feature_56_squared",
    "feature_213_squared",
    "feature_5_squared",
    "feature_46_squared",
    "feature_30_squared",
]

sq_feats = [f for f in sq_feats if
```

```
pd.api.types.is_numeric_dtype(train[f])]

for f in sq_feats:
    print(f"Для признака {f}")
    m = train[f].notna()
    fig = woe_line(
        train.loc[m, f].to_numpy(),
        train.loc[m, TARGET].astype(int).to_numpy(),
        n_buckets=20
    )
    fig.show()
```

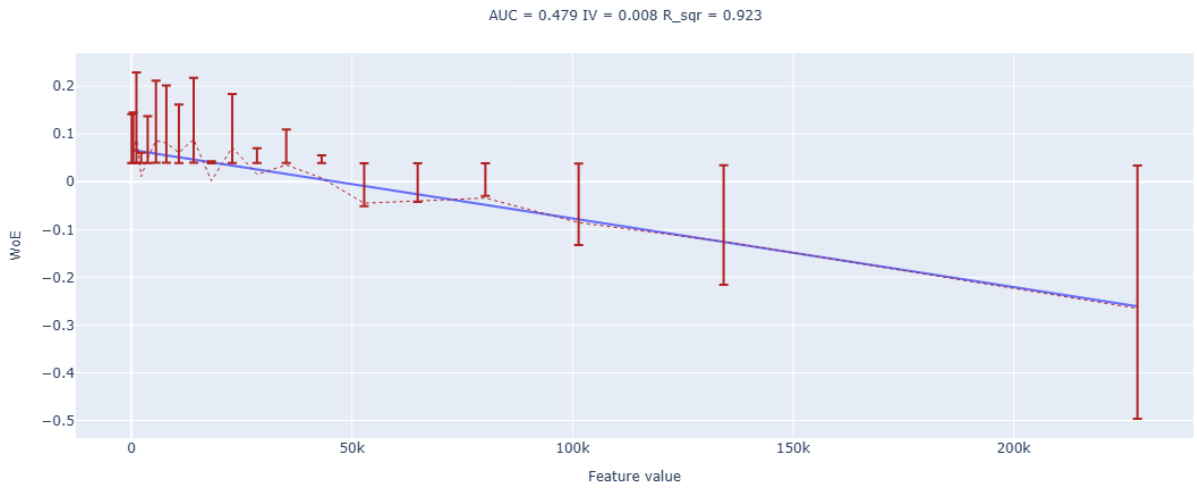
```
Для признака feature_164_squared
```



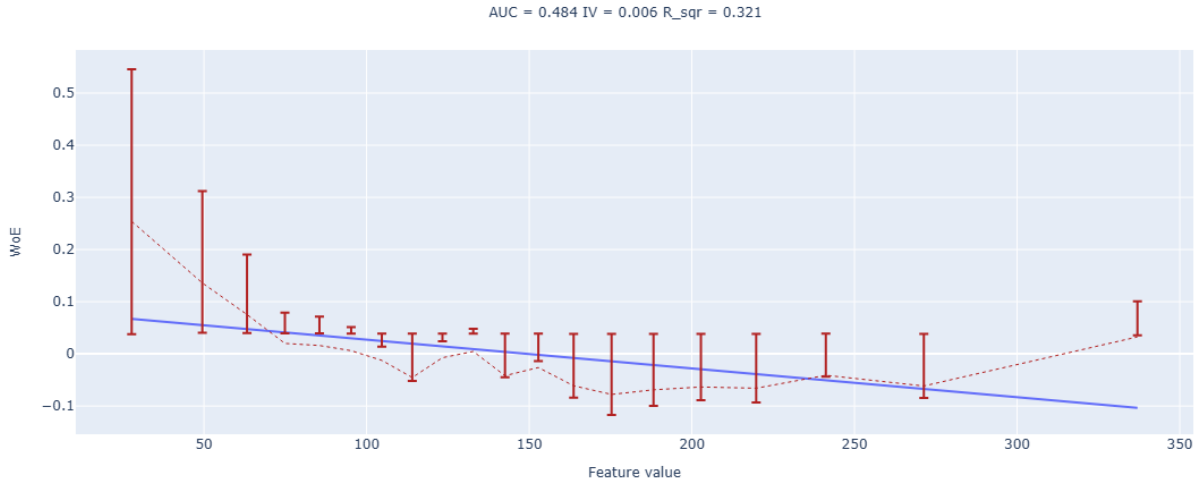

```
Для признака feature_76_squared
```



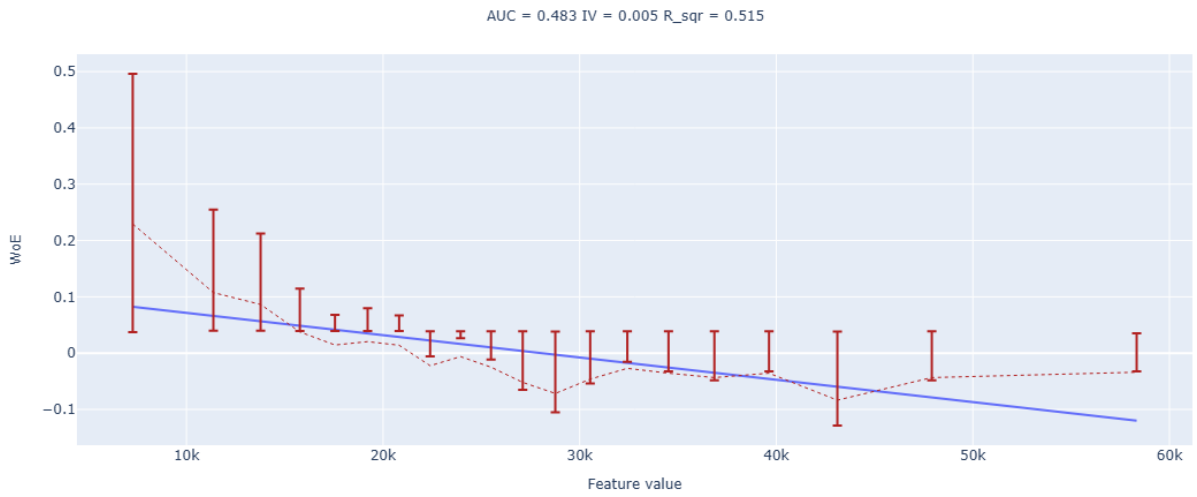

```
Для признака feature_56_squared
```



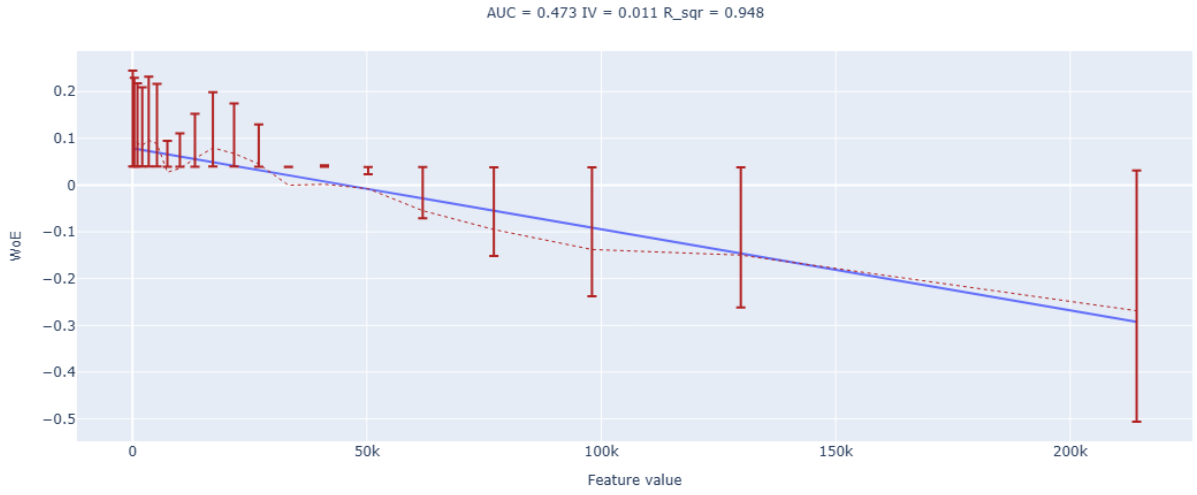
Для признака feature_213_squared



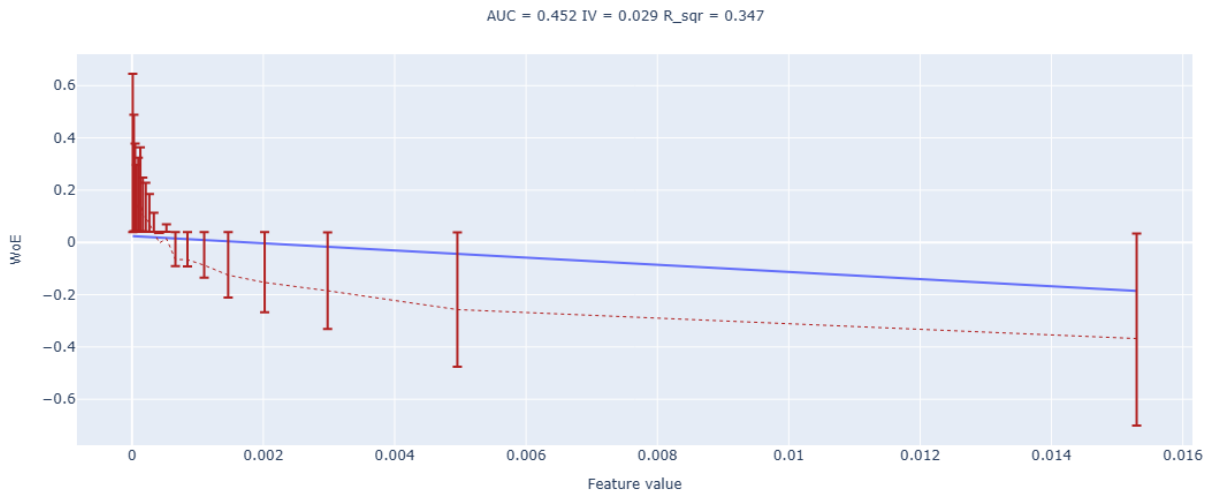
Для признака feature_5_squared



Для признака feature_46_squared



Для признака feature_30_squared



Выводы: улучшения есть! Признаки 213, 30 исключу перед дальнейшим рассмотрением

```
features_all = [
    "feature_79",
    "feature_164_squared",
    "feature_76_squared",
    "feature_75",
    "feature_154",
    "feature_132",
    "feature_56_squared",
    "feature_223",
    "feature_213_squared",
    "feature_5_squared",
    "feature_117",
    "feature_24",
    "feature_46_squared",
    "feature_30_squared",
    "feature_124",
]


to_drop = {"feature_132", "feature_213", "feature_213_squared",
"feature_30", "feature_30_squared"}

final_features_logreg = [f for f in features_all if f not in to_drop]


final_features_logreg = features_all

final_features_logreg = features_all

train[final_features_logreg].nunique()
```

```
feature_79              250000
feature_164_squared     245002
```

```
feature_76_squared     245002
feature_75             250000
feature_154            250000
feature_132            250000
feature_56_squared     245002
feature_223            213266
feature_213_squared    245002
feature_5_squared      245002
feature_117            250000
feature_24             250000
feature_46_squared     245002
feature_30_squared     228086
feature_124            220520
dtype: int64
```

Пропуски

Пришло время заполнить пропуски.

Самый простой вариант для числовых признаков - заполнить их средним значением фичи

Вопрос: какие проблемы могут возникнуть при таком заполнении пропусков?

Задание: Проверьте, что заполнение средним значением адекватно для тех признаков, где есть пропуски (hint: в нашем датасете - почти всегда адекватно). Если нет, придумайте, как ещё можно заполнить пропуски. Ну и заполните их)

Ответ: среднее НЕ ВСЕГДА является репрезентативной оценкой для выборки; а медина - является! Так как уже дан хинт, заполню их сразу медианой!

ПРЕДВАРИТЕЛЬНО ПРОВЕРЮ, ЧТО У ПРИЗНАКОВ НЕТ ВРЕМЕННОЙ СТРУКТУРЫ! ВЕДЬ У НАС ЕСТЬ ДАТЫ, А ЗНАЧИТ, ЕСЛИ СТРУКТУРА ЕСТЬ, ПРОПУЩЕННЫЕ ДАННЫЕ НУЖНО ЗАПОЛНЯТЬ ЭКСТРАПОЛЯЦИЕЙ, А НЕ ВЫБОРОЧНЫМИ СТАТИСТИКАМИ!

```
#### Обработка пропущенных значений

data_is_na = pd.DataFrame(df.isna().mean(),
columns=['Percentage_of_Missed_Values'])
data_is_na['feature_name'] = data_is_na.index

missed_values = data_is_na.query("Percentage_of_Missed_Values > 0.00")

missed_values
```

```
             Percentage_of_Missed_Values feature_name
feature_223                        0.147  feature_223
feature_30                         0.069   feature_30
feature_16                         0.082   feature_16
feature_124                        0.118  feature_124
feature_10                         0.333   feature_10
```

```
feature_68                      0.131   feature_68
feature_136                     0.117  feature_136
feature_28                      0.063   feature_28
feature_115                     0.087  feature_115
feature_51                      0.083   feature_51
feature_140                     0.074  feature_140
feature_78                      0.108   feature_78
feature_161                     0.103  feature_161
feature_224                     0.052  feature_224
feature_228                     0.127  feature_228
feature_220                     0.055  feature_220
feature_163                     0.105  feature_163
feature_62                      0.125   feature_62
feature_172                     0.124  feature_172
feature_175                     0.063  feature_175
feature_148                     0.138  feature_148
feature_205                     0.100  feature_205
feature_3                       0.061    feature_3
feature_105                     0.104  feature_105
feature_53                      0.128   feature_53
feature_133                     0.108  feature_133
feature_207                     0.089  feature_207
feature_176                     0.093  feature_176
feature_212                     0.072  feature_212
feature_49                      0.063   feature_49
feature_80                      0.144   feature_80
feature_34                      0.072   feature_34
feature_7                       0.057    feature_7
feature_110                     0.093  feature_110
feature_91                      0.091   feature_91
feature_83                      0.106   feature_83
feature_203                     0.116  feature_203
feature_208                     0.085  feature_208
feature_89                      0.087   feature_89
feature_8                       0.147    feature_8
feature_13                      0.111   feature_13
feature_59                      0.134   feature_59
feature_196                     0.126  feature_196
feature_131                     0.108  feature_131
feature_17                      0.147   feature_17
feature_166                     0.055  feature_166
feature_72                      0.111   feature_72
feature_200                     0.084  feature_200
feature_134                     0.118  feature_134
feature_192                     0.143  feature_192
feature_219                     0.141  feature_219
feature_63                      0.061   feature_63
feature_54                      0.092   feature_54
feature_107                     0.134  feature_107
```

```
feature_50                          0.067   feature_50
feature_174                         0.097  feature_174
feature_190                         0.118  feature_190
feature_189                         0.115  feature_189
feature_226                         0.091  feature_226
feature_201                         0.093  feature_201
feature_169                         0.138  feature_169
feature_103                         0.104  feature_103
feature_99                          0.114   feature_99
feature_87                          0.063   feature_87
feature_202                         0.092  feature_202
feature_74                          0.058   feature_74
feature_214                         0.087  feature_214
feature_210                         0.065  feature_210
```

```
missed_values_list = missed_values['feature_name'].to_list()
df_missed_values_list = df[missed_values_list]

df_missed_values_list
```

```
        feature_223  feature_30  feature_16  feature_124
feature_10  \
22620        38.981       0.037     187.238      -61.495          NaN

478621      190.191       0.060      76.368      -28.477        0.000

372254          NaN       0.084     115.597          NaN        1.000

2596        100.130       0.021     177.790      -42.118        1.000

216892       53.473       0.090      76.512      -48.599          NaN

...             ...         ...         ...          ...          ...

196513       89.119       0.025     114.723      -34.698          NaN

108604      180.215       0.010     164.591      -57.378        0.000

141359       76.055       0.013     137.813      -46.912          NaN

443018       47.622       0.043     202.592      -28.192          NaN

310653          NaN         NaN         NaN          NaN        0.000


        feature_68  feature_136  feature_28  feature_115
feature_51  \
22620      -67.237       66.272    -147.398     -209.629     -50.931

478621    -103.629       11.989     -10.632     -113.581      50.032
```

```
372254         NaN          NaN    -155.017     -282.365        4.908
2596      -108.854     -106.539    -181.545     -143.675      -13.816
216892     -73.858      221.844     -71.107      297.919       -7.798
...            ...          ...         ...          ...          ...
196513      -3.197       27.590    -188.208      -18.363      -86.980
108604     -30.835     -134.765      57.293      -17.501       70.927
141359     -66.913     -130.157     111.088     -160.477      -48.142
443018     -62.274       -2.009      61.212     -201.116       19.472
310653         NaN          NaN    -233.510          NaN          NaN

        feature_140  feature_78  feature_161  feature_224  feature_228
\
22620        15.000     -59.890        0.000        0.635      246.898
478621       28.000     -19.073        0.000        1.162     -148.415
372254       10.000         NaN          NaN        0.863          NaN
2596         22.000       3.804        0.000        1.218      101.575
216892       50.000      27.339        1.000        1.132       78.013
...             ...         ...          ...          ...          ...
196513       23.000     -36.563        0.000        0.682      388.046
108604       50.000      29.266        0.000        1.127     -230.233
141359       46.000      83.462        0.000        0.930      235.976
443018       26.000     136.784        0.000        0.610      -28.761
310653          NaN         NaN          NaN        0.573          NaN

        feature_220  feature_163  feature_62  feature_172  feature_175
\
22620       -13.085      -14.136       0.000       21.311      -28.774
478621       30.582      -56.234       1.000       75.191       90.201
372254      -18.682          NaN         NaN          NaN     -209.663
```

```
2596         125.930      -17.724       0.000       41.056     -181.364
216892      -138.231      -56.179       0.000      -54.095      -14.576
...              ...          ...         ...          ...          ...
196513        18.032      -41.670       0.000      -14.492      360.183
108604       148.635      -22.736       1.000        6.614      155.552
141359       144.928      -32.903       0.000       39.919      183.698
443018        41.842       -8.307       0.000       11.577      150.294
310653      -133.520          NaN         NaN          NaN      167.083

        feature_148  feature_205  feature_3  feature_105
feature_53  \
22620       -167.855       28.052    -50.013      -23.103      -76.552
478621      -104.884      -74.677    -56.263      -28.258      -60.416
372254           NaN      169.643    -57.440          NaN          NaN
2596        -165.598      133.578    -65.826      -31.454      -78.346
216892      -154.776     -221.299    -56.410      -26.947      -78.481
...              ...          ...         ...          ...          ...
196513      -171.095      -43.066    -51.572      -28.027      -70.575
108604      -101.423       15.954    -57.666      -28.326      -66.988
141359      -149.823      210.137    -64.805      -30.125      -68.800
443018      -195.193      103.768    -62.711      -30.029      -70.232
310653           NaN          NaN    -62.522          NaN          NaN

        feature_133  feature_207  feature_176  feature_212  feature_49
\
22620        -47.857       51.926      -44.408       -8.734      -19.969
478621       -50.860       59.565      -48.011       -9.867      -16.854
372254           NaN       53.574      -45.130       -8.652      -14.878
2596         -46.876       52.241      -45.016       -9.543      -10.603
```

```
216892      -53.546       57.733      -54.199       -8.221     -14.964

...             ...          ...          ...          ...         ...

196513      -52.398       56.156      -66.078       -9.412     -11.948

108604      -51.118       44.451      -55.703       -9.607     -13.806

141359      -46.470       56.320      -57.782       -9.237     -12.333

443018      -53.534       62.785      -57.249       -6.146     -13.963

310653          NaN          NaN          NaN          NaN     -13.873


        feature_80  feature_34  feature_7  feature_110  feature_91  \
22620        7.598      53.702    135.175      -59.770      15.974
478621       3.359      55.723    109.011      -50.481      18.203
372254         NaN      46.277     98.945      -36.951      18.336
2596        10.750      45.664    128.199      -57.712      16.575
216892      14.869      51.618    128.961      -55.065      18.350
...            ...         ...        ...          ...         ...
196513      12.700      52.886    118.148      -45.741      20.238
108604      16.794      54.112    115.309      -41.600      17.481
141359      23.073      49.750    113.509      -51.112      17.729
443018       4.940      49.993    123.622      -49.691      18.557
310653         NaN         NaN    116.849          NaN         NaN

        feature_83  feature_203  feature_208  feature_89  feature_8  \
22620      -98.783       21.408      -52.267    -161.735     -0.016
478621     -93.351       20.625      -64.838    -153.978     -0.047
372254         NaN          NaN      -55.948    -158.620        NaN
2596       -86.957       15.324      -43.970    -163.810      0.027
216892     -88.030       19.797      -53.166    -142.453     -0.088
...            ...          ...          ...         ...         ...
196513     -85.485       24.215      -34.732    -150.759      0.143
108604    -102.023       21.769      -51.027    -154.967      0.038
141359     -94.554       21.148      -54.722    -150.457      0.171
443018     -91.408       19.431      -43.256    -152.291      0.063
310653         NaN          NaN          NaN         NaN         NaN

        feature_13  feature_59  feature_196  feature_131
feature_17  \
22620      -73.571     -17.578      -96.206      -13.907      38.684

478621     -76.884     -25.264     -104.415      -16.271      37.820

372254         NaN         NaN          NaN          NaN         NaN

2596       -68.620     -17.801      -93.988      -11.650      48.020
```

```
216892     -74.988     -21.171     -110.029      -17.532      44.776
...            ...         ...          ...          ...         ...
196513     -74.874     -29.526     -101.223      -16.588      53.879
108604     -65.453     -24.026     -111.567      -17.102      46.034
141359     -71.731     -25.481     -110.698      -17.340      48.347
443018     -66.371     -20.756     -110.203      -14.774      46.706
310653         NaN         NaN          NaN          NaN         NaN

        feature_166  feature_72  feature_200  feature_134  feature_192
\
22620       142.940     127.193       19.727     -143.082      156.746
478621      154.076     109.519       15.559     -128.834      156.707
372254      137.008         NaN       22.581          NaN          NaN
2596        132.534     114.688       23.578     -138.962      160.952
216892      150.958     126.752       17.575     -138.706      148.154
...             ...         ...          ...          ...         ...
196513      143.434     114.728       20.316     -135.992      151.826
108604      151.182     135.165       25.318     -135.304      143.253
141359      142.082     127.237       19.646     -133.723      161.892
443018      136.979     123.306       20.836     -157.733      153.513
310653      143.030         NaN          NaN          NaN          NaN

        feature_219  feature_63  feature_54  feature_107
feature_50  \
22620        18.878     -31.626      -7.896      -46.545      -9.221
478621       24.013     -25.711      -6.857      -55.906     -12.882
372254          NaN     -28.989      -5.588          NaN      -0.046
2596         22.971     -28.717      -7.872      -57.512      -6.850
216892       20.580     -25.807      -5.604      -43.131      -7.338
```

```
...              ...         ...         ...          ...          ...
196513       23.734     -23.142      -5.589      -60.648      -7.894
108604       21.685     -22.066      -6.454      -57.330      -0.571
141359       21.516     -22.048      -6.268      -45.358      -9.795
443018       22.202     -26.920      -6.556      -50.094      -6.835
310653          NaN     -23.527         NaN          NaN         NaN


        feature_174  feature_190  feature_189  feature_226
feature_201  \
22620         6.183      -60.625      -96.469       89.161     -
120.011
478621        0.584      -49.047      -90.039       91.580     -
149.926
372254        5.824          NaN          NaN       95.783     -
135.141
2596         -3.200      -48.636      -95.354       92.526     -
128.508
216892        3.995      -52.624      -91.185       98.265     -
133.948
...              ...          ...          ...          ...          ..
.
196513        8.276      -57.579      -91.092      101.423     -
122.600
108604        5.395      -51.656     -103.124       96.868     -
121.686
141359       11.773      -52.792      -97.155       91.050     -
122.823
443018        0.325      -49.414     -100.487       90.143     -
128.060
310653          NaN          NaN          NaN          NaN
NaN

        feature_169  feature_103  feature_99  feature_87  feature_202
\
22620        -0.605      -19.000      29.620     -94.253      -67.188
478621       -0.575      -13.428      26.844    -106.230      -53.675
372254          NaN          NaN         NaN     -91.873      -53.926
2596         -0.637      -17.681      29.720    -107.784      -71.514
216892       -0.495      -14.453      26.981     -94.542      -68.335
```

```
...              ...          ...          ...          ...          ...

196513       -0.547       -15.621       32.192      -85.827       -54.790

108604       -0.507       -17.405       27.059      -94.680       -62.973

141359       -0.530       -16.853       23.782      -83.221       -58.209

443018       -0.621       -17.750       22.957      -85.346       -51.764

310653          NaN          NaN          NaN      -89.337           NaN
```

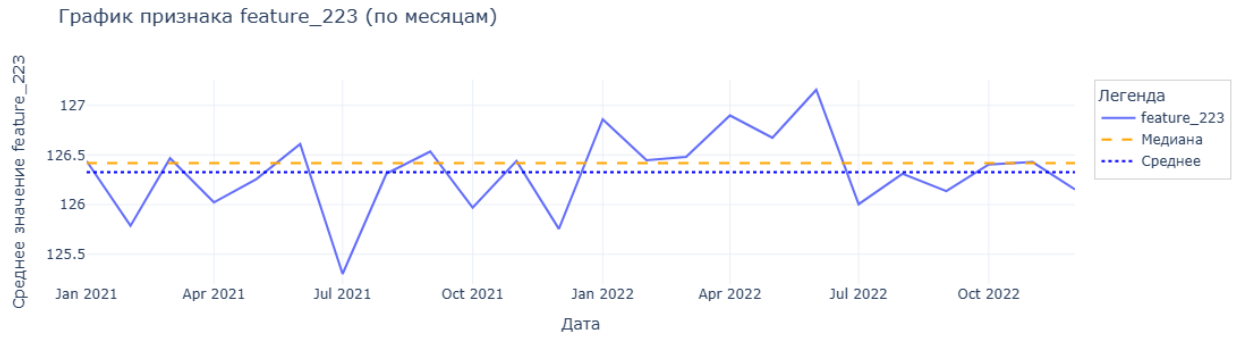
```
        feature_74  feature_214  feature_210
22620       50.895      -10.686     -111.006
478621      52.458      -10.771     -121.484
372254      56.016      -15.786     -123.005
2596        44.896      -15.021     -119.885
216892      47.732      -13.757     -117.775
...            ...          ...          ...
196513      53.891      -17.716     -115.881
108604      50.775      -17.492     -117.535
141359      49.540      -15.612     -130.302
443018      54.464      -17.502     -114.376
310653      51.724          NaN     -122.733

[500000 rows x 68 columns]
```

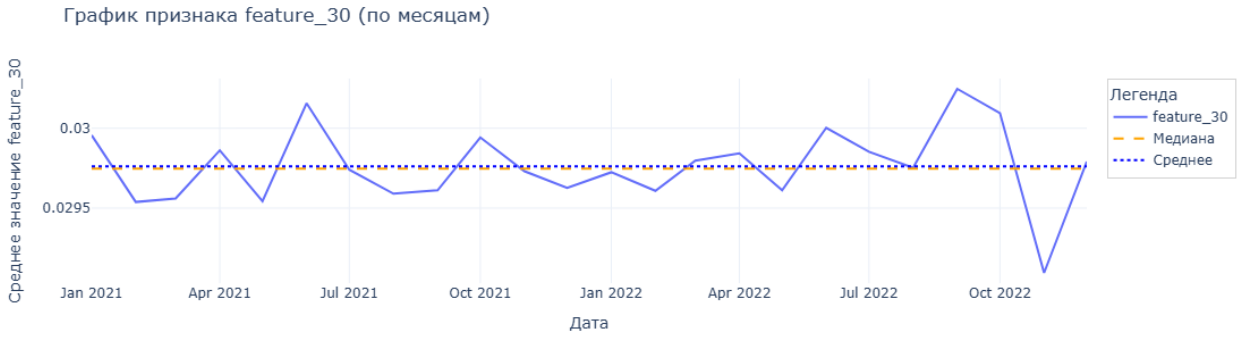
```python
def plotly_single_feature(
    df,
    column,
    index_col='ds',
    title=None,
    yaxis_title='Значение',
    line_name=None,
    color='#636EFA',
    show_mean=True,
    show_median=True,
    save=False,
    save_path='feature_plot.html'
):
    """
    Универсальный график временного ряда с возможностью отобразить среднее и медиану.

    Параметры:
    - df: DataFrame с данными
    - column: имя числового столбца для отображения
    - index_col: колонка с временным индексом (по умолчанию 'ds')
    - title: заголовок графика
```

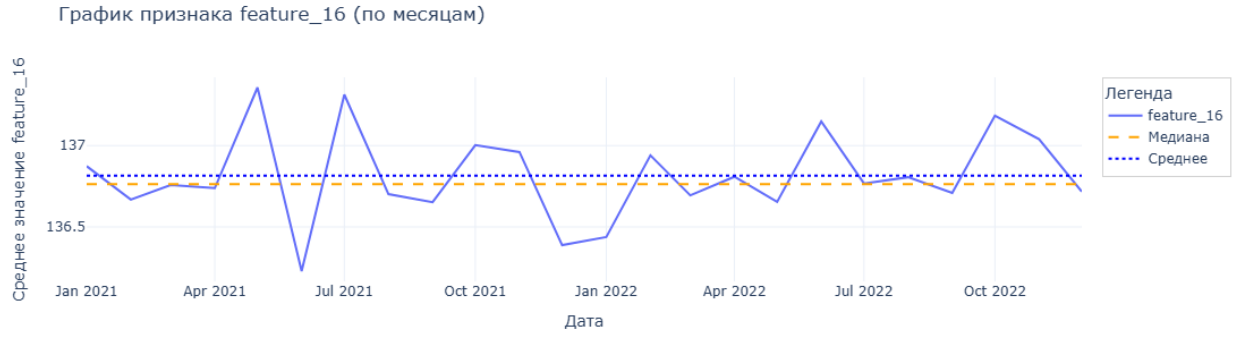
```
    - yaxis_title: подпись оси Y
    - line_name: имя основного ряда (по умолчанию = column)
    - color: цвет линии (по умолчанию Plotly синий)
    - show_mean: показать линию среднего
    - show_median: показать линию медианы
    - save: сохранить график в HTML
    - save_path: путь к файлу (если save=True)
    """
    # Визуализация
    import matplotlib.pyplot as plt
    import seaborn as sns
    import plotly.graph_objs as go
    from plotly.offline import download_plotlyjs, init_notebook_mode,
plot, iplot

    df = df.copy()

    if index_col in df.columns:
        df = df.set_index(index_col)

    line_name = line_name or column
    title = title or f'График: {column}'

    mean_val = df[column].mean()
    median_val = df[column].median()

    traces = []

    # Основная линия
    traces.append(go.Scatter(
        x=df.index,
        y=df[column],
        mode='lines',
        name=line_name,
        line=dict(color=color, width=2)
    ))

    # Медиана
    if show_median:
        traces.append(go.Scatter(
            x=[df.index.min(), df.index.max()],
            y=[median_val, median_val],
            mode='lines',
            name='Медиана',
            line=dict(color='orange', width=2, dash='dash')
        ))

    # Среднее
    if show_mean:
        traces.append(go.Scatter(
```

```
            x=[df.index.min(), df.index.max()],
            y=[mean_val, mean_val],
            mode='lines',
            name='Среднее',
            line=dict(color='blue', width=2, dash='dot')
        ))

    layout = go.Layout(
        title=title,
        xaxis=dict(title='Дата'),
        yaxis=dict(title=yaxis_title),
        template='plotly_white',
        hovermode='x unified',
        legend=dict(
            title='Легенда',
            orientation='v',
            x=1.02,
            y=1,
            xanchor='left',
            yanchor='top',
            bgcolor='rgba(255,255,255,0.8)',
            bordercolor='lightgray',
            borderwidth=1
        )
    )

    fig = go.Figure(data=traces, layout=layout)
    iplot(fig)

    if save:
        dir_to_save = os.path.dirname(save_path)
        if dir_to_save:
            os.makedirs(dir_to_save, exist_ok=True)
        plot(fig, filename=save_path, auto_open=False)
        print(f'Сохранено в файл: {os.path.abspath(save_path)}')

for column_name in df_missed_values_list:
    if column_name in df.columns:
        tmp = (
            df[["month", column_name]]
            .groupby("month", sort=True)[column_name]
            .mean()
            .reset_index()
            .rename(columns={"month": "date"})
        )

        plotly_single_feature(
            df=tmp,
            column=column_name,
            index_col="date",
```

```
        title=f"График признака {column_name} (по месяцам)",
        yaxis_title=f"Среднее значение {column_name}",
    )
```







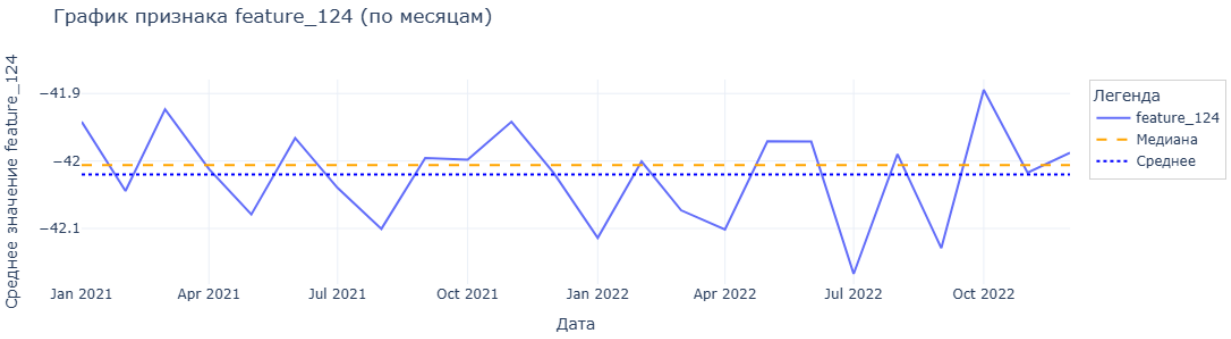


График признака feature_124 (по месяцам)
Среднее значение feature_124
−41.9
−42
−42.1
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_124
Медиана
Среднее

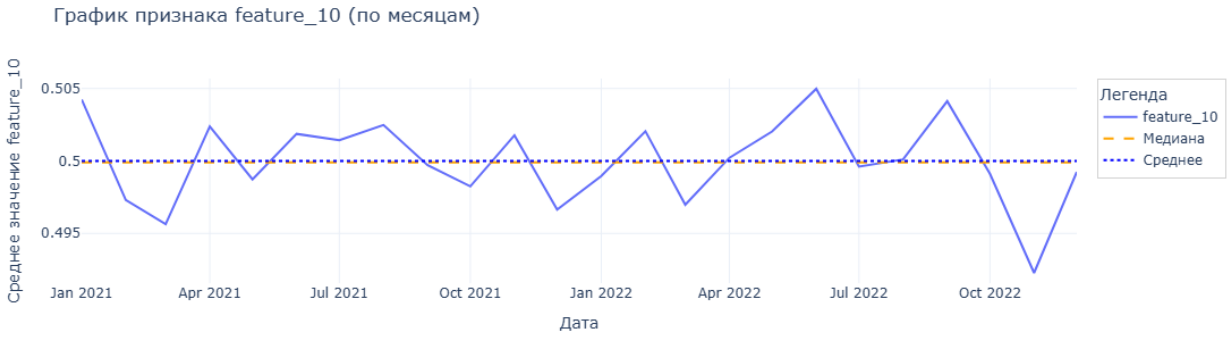


График признака feature_10 (по месяцам)
Среднее значение feature_10
0.505
0.5
0.495
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_10
Медиана
Среднее

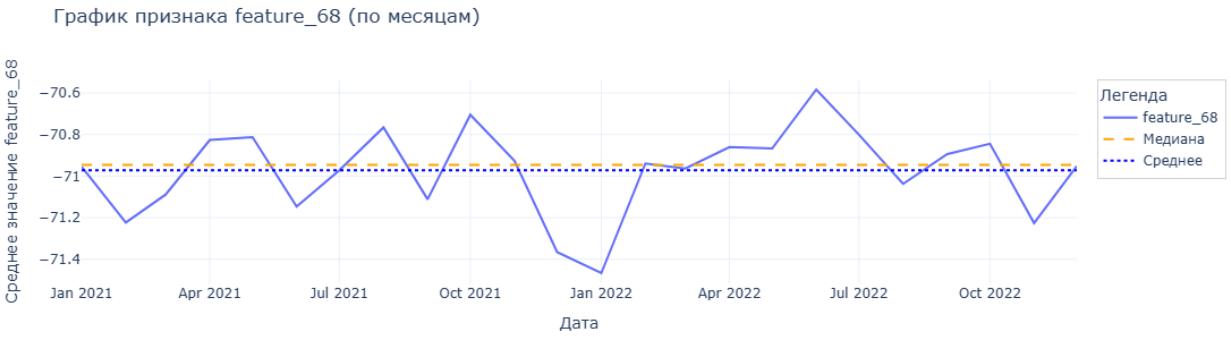


График признака feature_68 (по месяцам)
Среднее значение feature_68
−70.6
−70.8
−71
−71.2
−71.4
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_68
Медиана
Среднее

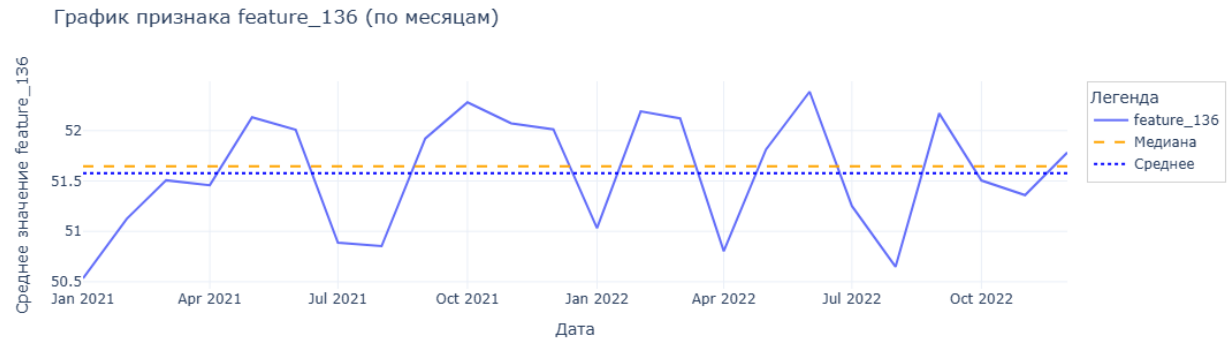


График признака feature_136 (по месяцам)
Среднее значение feature_136
52
51.5
51
50.5
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_136
Медиана
Среднее

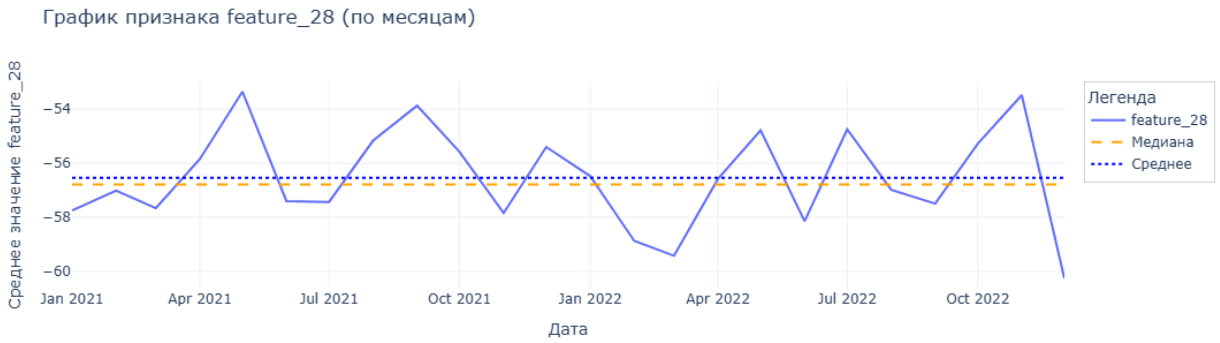


График признака feature_28 (по месяцам)
Среднее значение feature_28
−54
−56
−58
−60
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_28
Медиана
Среднее

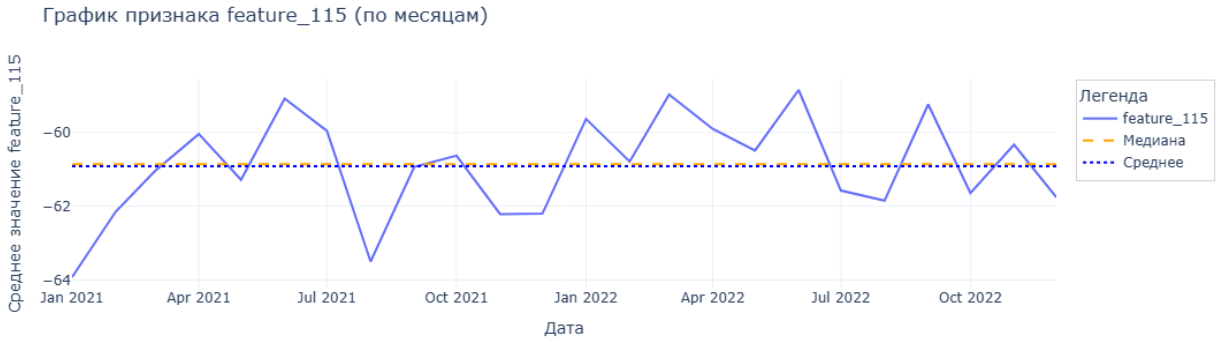


График признака feature_115 (по месяцам)
Среднее значение feature_115
−60
−62
−64
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_115
Медиана
Среднее

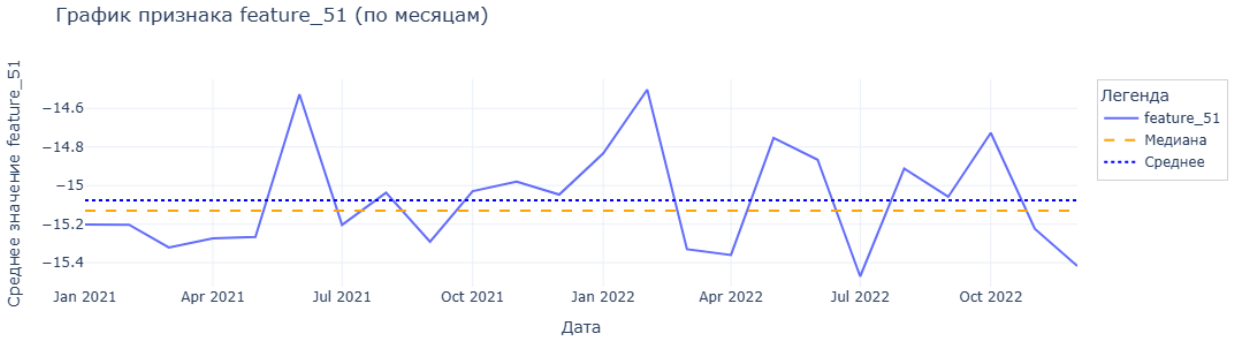


График признака feature_51 (по месяцам)
Среднее значение feature_51
−14.6
−14.8
−15
−15.2
−15.4
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_51
Медиана
Среднее

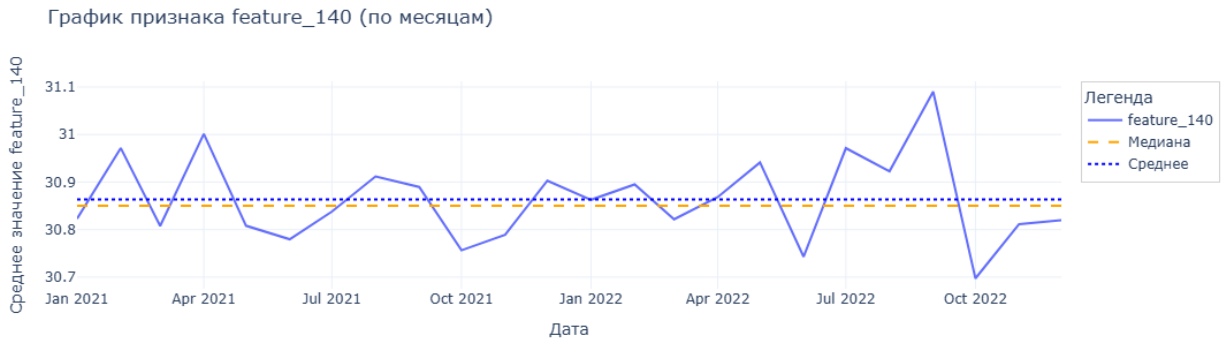


График признака feature_140 (по месяцам)
Среднее значение feature_140
31.1
31
30.9
30.8
30.7
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_140
Медиана
Среднее

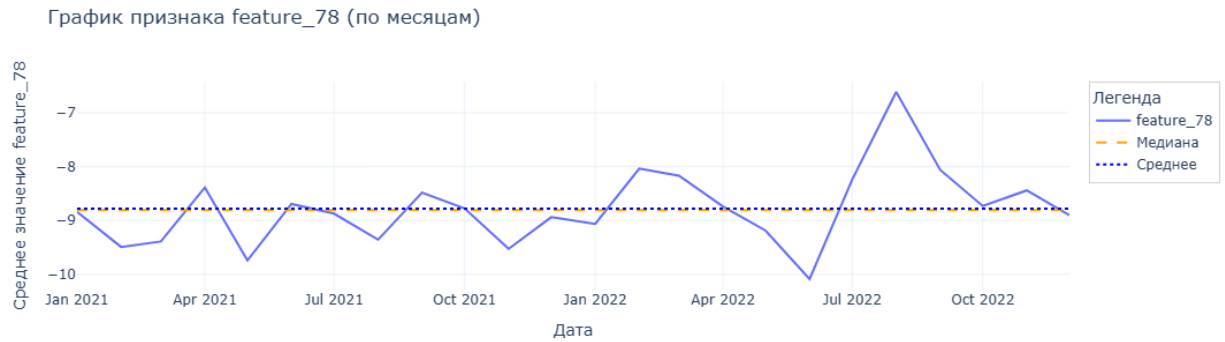


График признака feature_78 (по месяцам)
Среднее значение feature_78
−7
−8
−9
−10
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_78
Медиана
Среднее

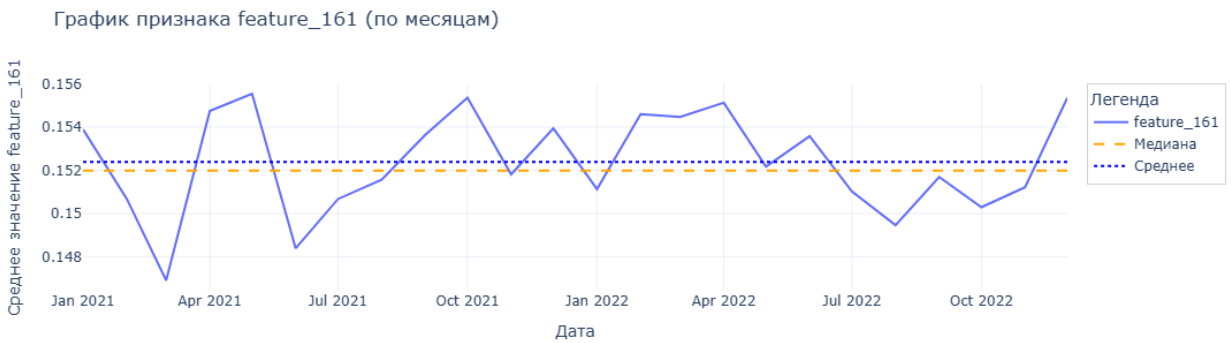


График признака feature_161 (по месяцам)
Среднее значение feature_161
0.156
0.154
0.152
0.15
0.148
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_161
Медиана
Среднее

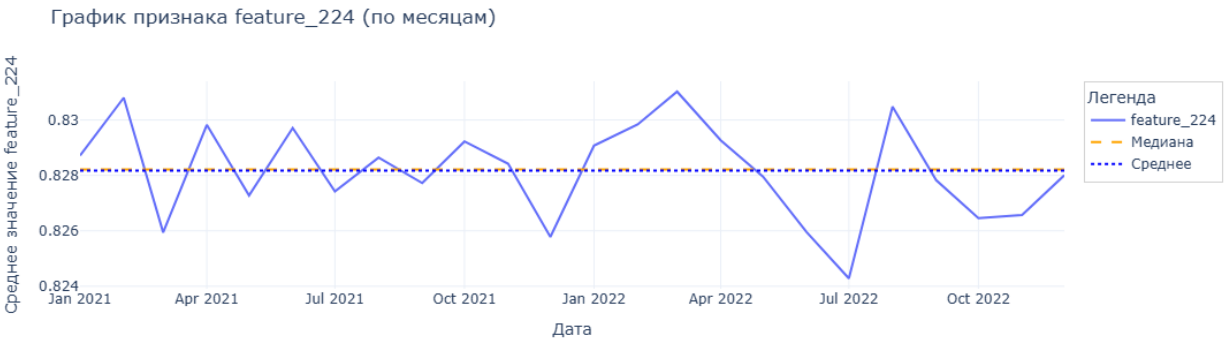


График признака feature_224 (по месяцам)
Среднее значение feature_224
0.83
0.828
0.826
0.824
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_224
Медиана
Среднее

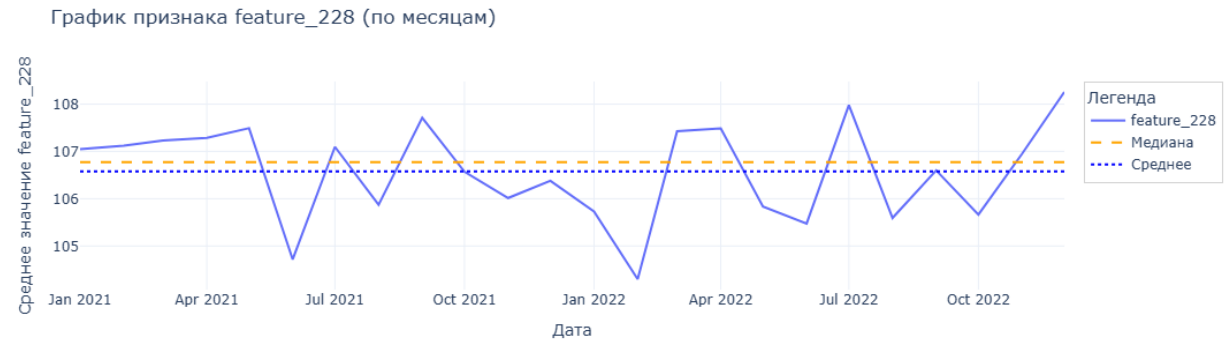


График признака feature_228 (по месяцам)
Среднее значение feature_228
108
107
106
105
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_228
Медиана
Среднее

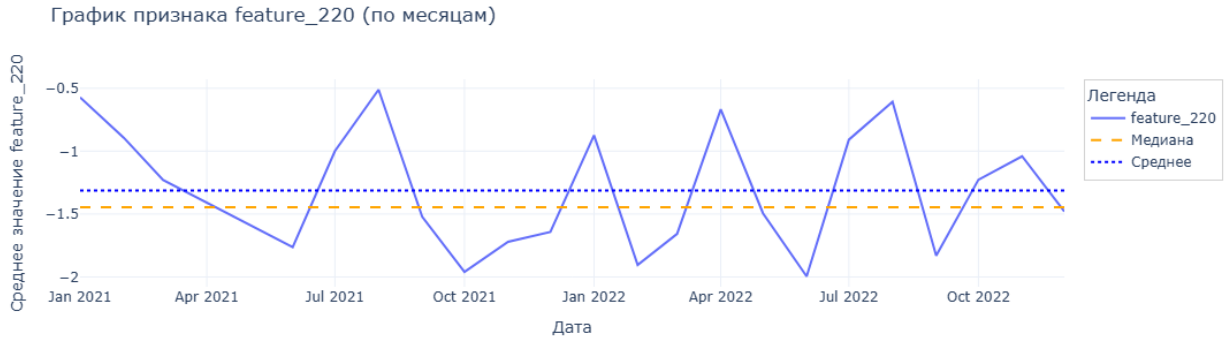


График признака feature_220 (по месяцам)
Среднее значение feature_220
−0.5
−1
−1.5
−2
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_220
Медиана
Среднее

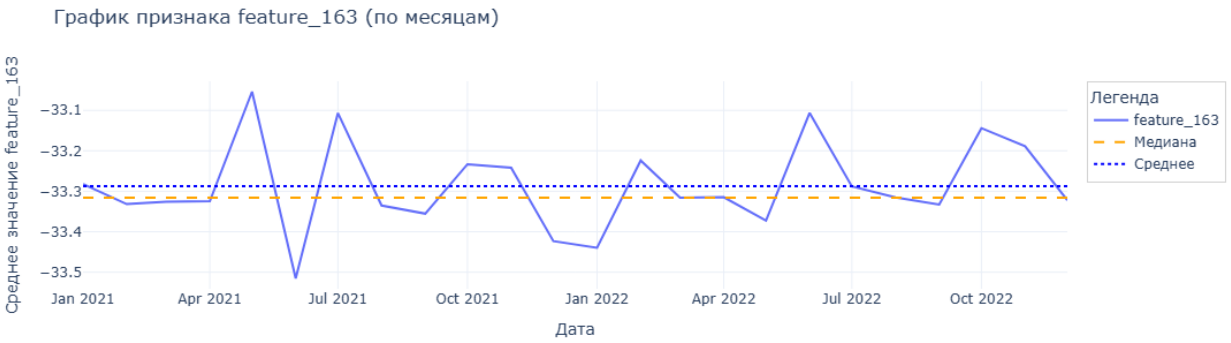


График признака feature_163 (по месяцам)
Среднее значение feature_163
−33.1
−33.2
−33.3
−33.4
−33.5
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_163
Медиана
Среднее

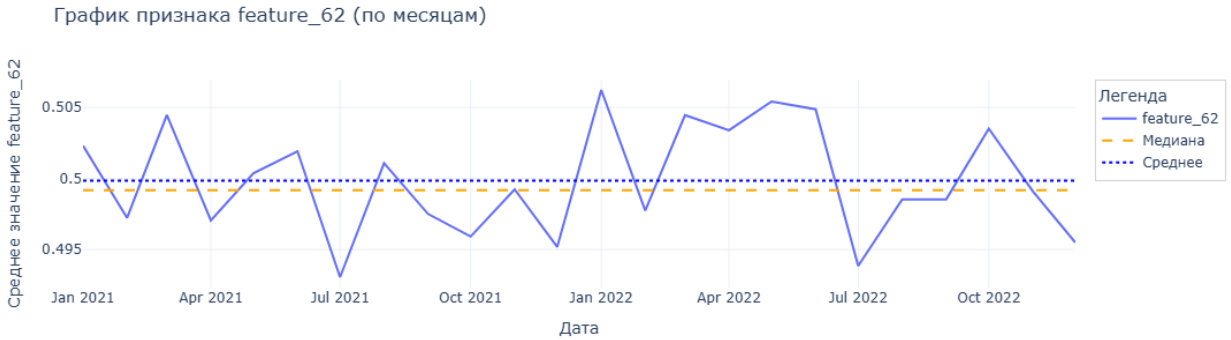


График признака feature_62 (по месяцам)
Среднее значение feature_62
0.505
0.5
0.495
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_62
Медиана
Среднее

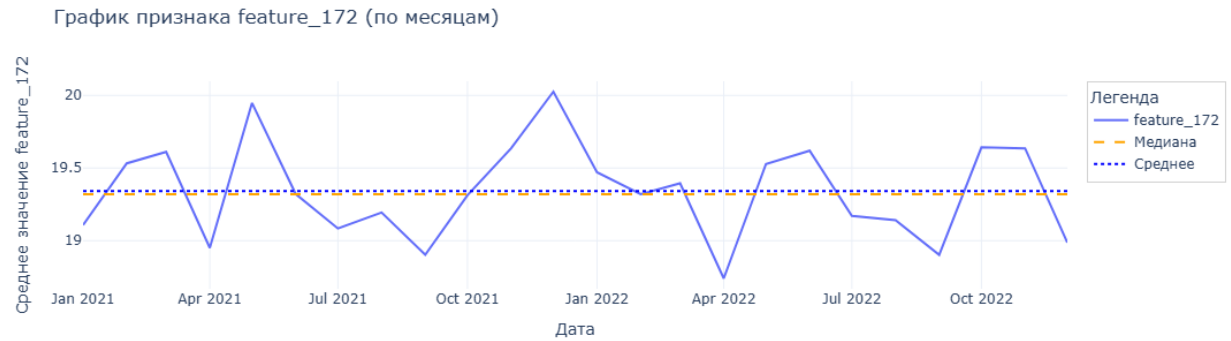


График признака feature_172 (по месяцам)
Среднее значение feature_172
20
19.5
19
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_172
Медиана
Среднее

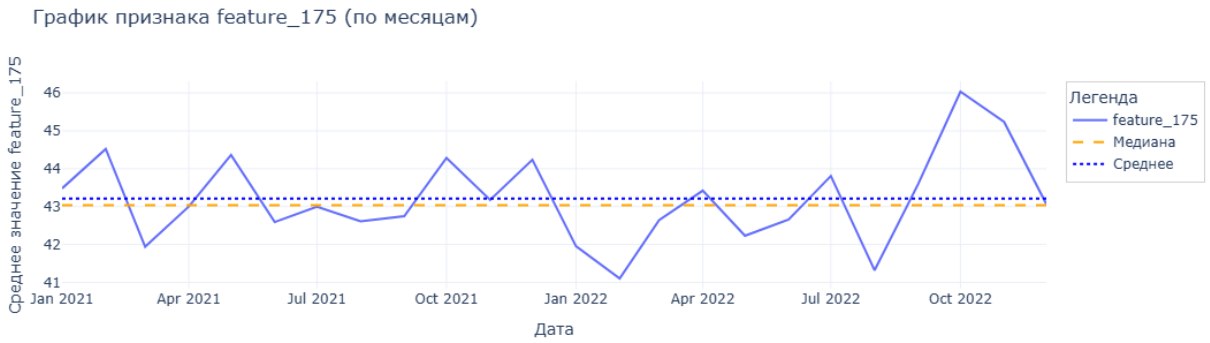


График признака feature_175 (по месяцам)
Среднее значение feature_175
46
45
44
43
42
41
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_175
Медиана
Среднее

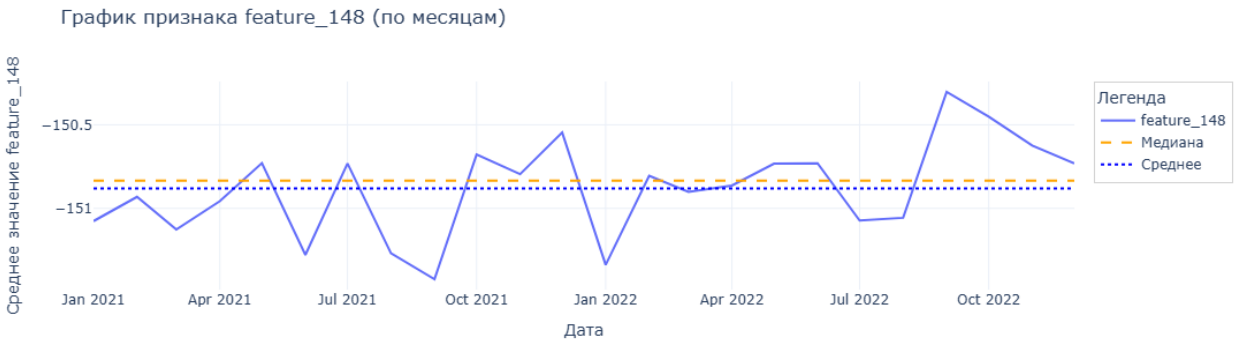


График признака feature_148 (по месяцам)
Среднее значение feature_148
−150.5
−151
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_148
Медиана
Среднее

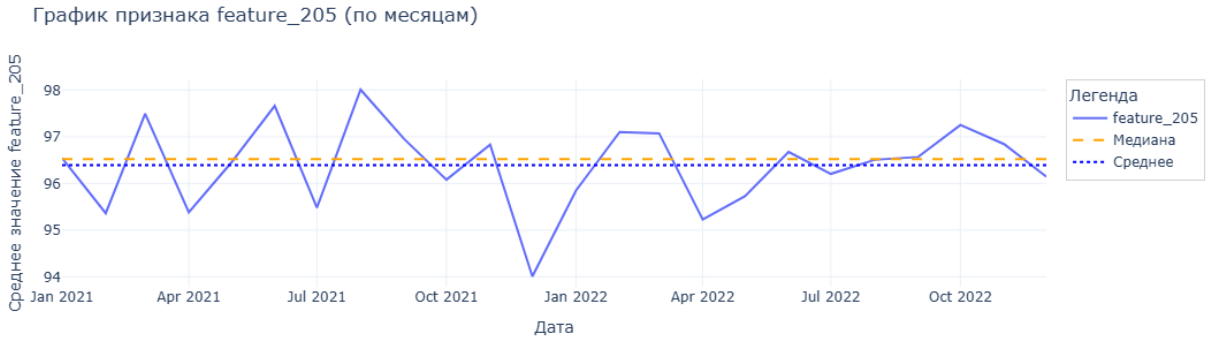


График признака feature_205 (по месяцам)
Среднее значение feature_205
98
97
96
95
94
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_205
Медиана
Среднее

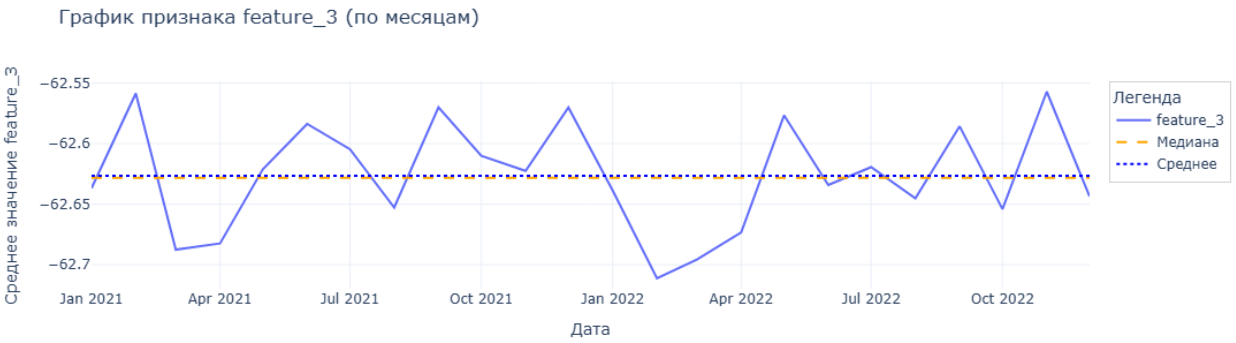


График признака feature_3 (по месяцам)
Среднее значение feature_3
−62.55
−62.6
−62.65
−62.7
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_3
Медиана
Среднее

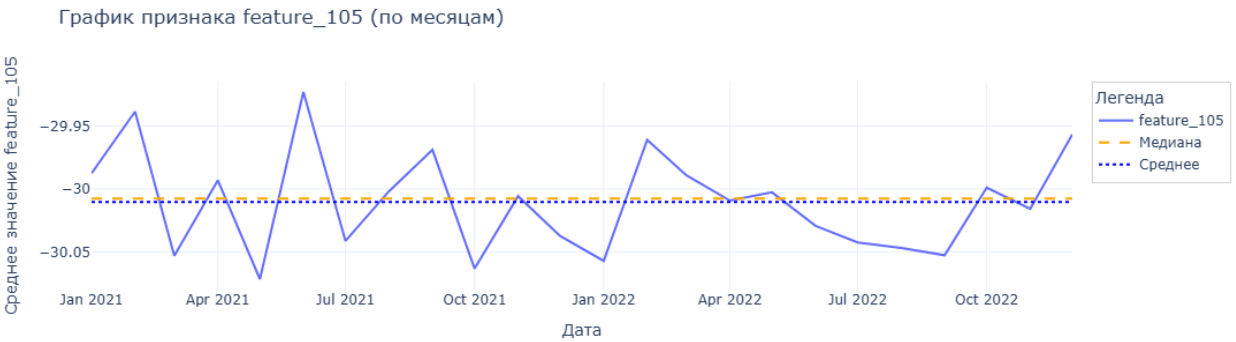


График признака feature_105 (по месяцам)
Среднее значение feature_105
−29.95
−30
−30.05
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_105
Медиана
Среднее

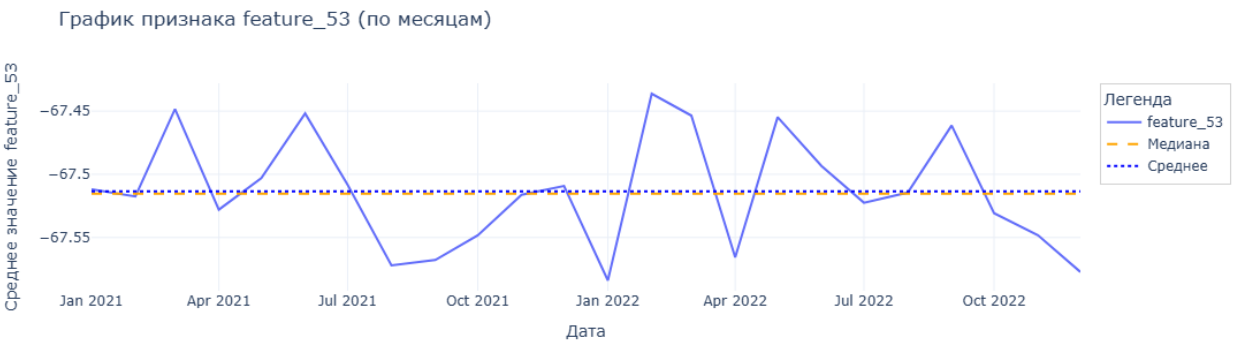


График признака feature_53 (по месяцам)
Среднее значение feature_53
−67.45
−67.5
−67.55
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_53
Медиана
Среднее

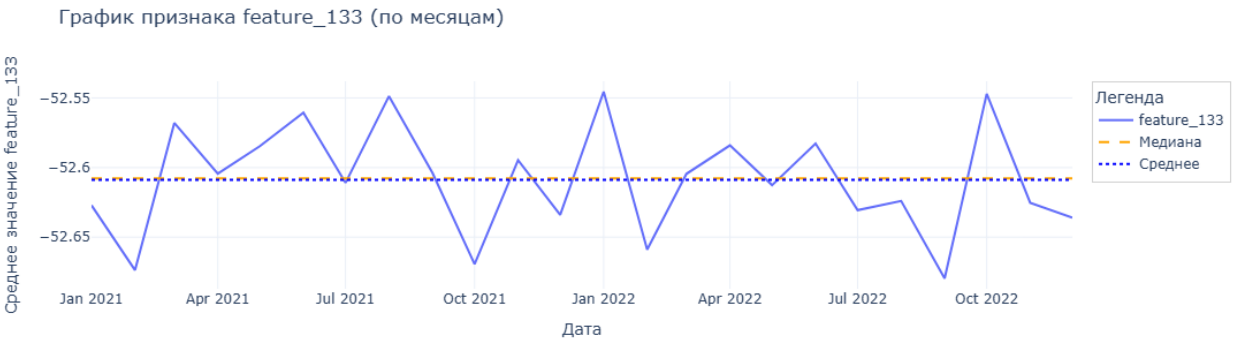


График признака feature_133 (по месяцам)
Среднее значение feature_133
−52.55
−52.6
−52.65
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_133
Медиана
Среднее

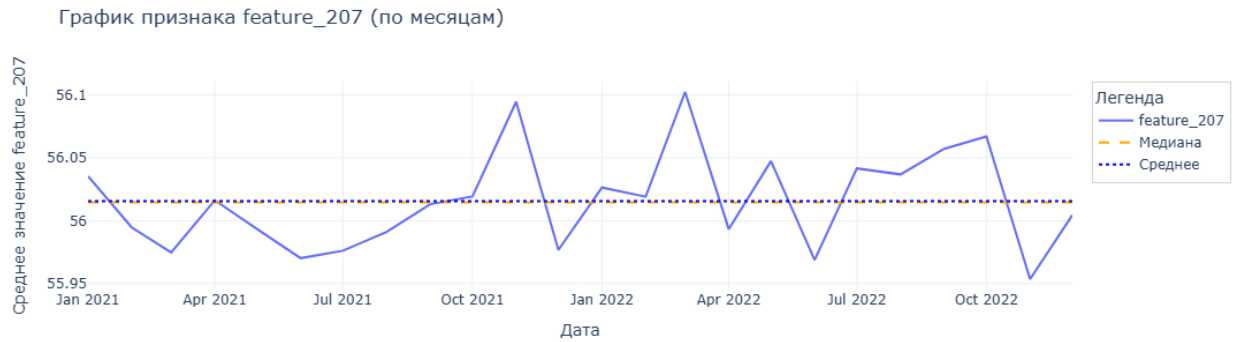


График признака feature_207 (по месяцам)
Среднее значение feature_207
56.1
56.05
56
55.95
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_207
Медиана
Среднее

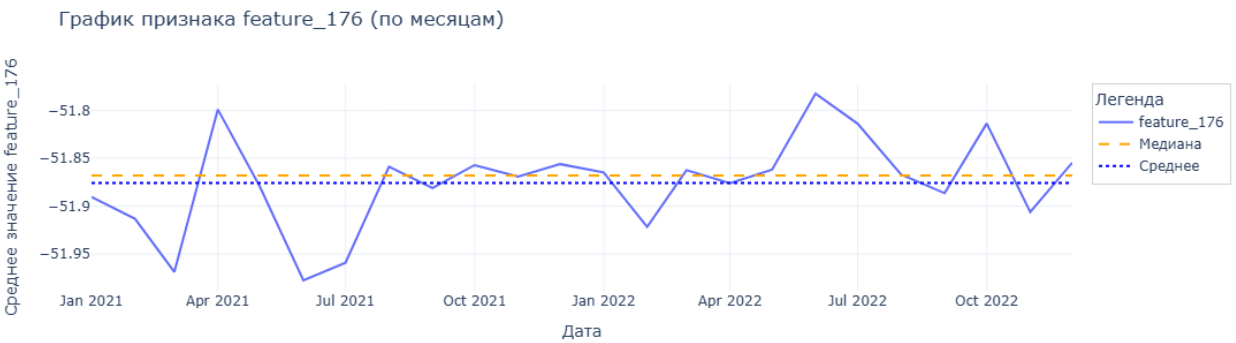


График признака feature_176 (по месяцам)
Среднее значение feature_176
−51.8
−51.85
−51.9
−51.95
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_176
Медиана
Среднее

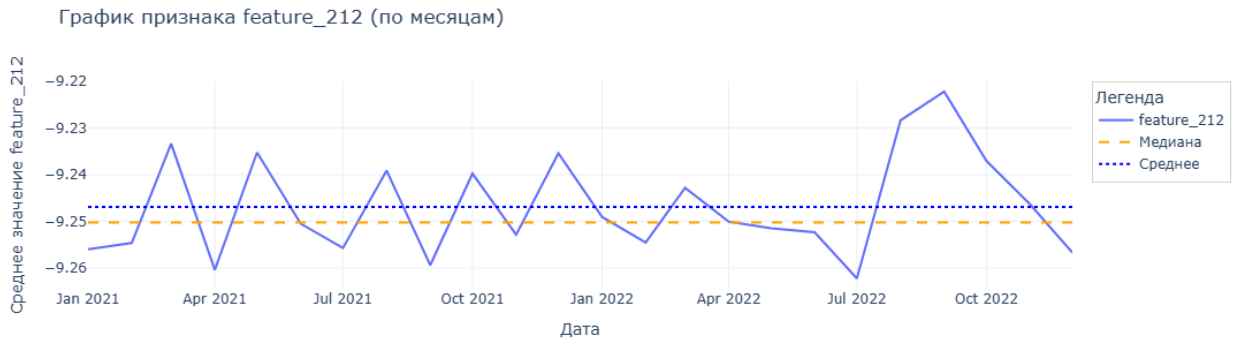


График признака feature_212 (по месяцам)
Среднее значение feature_212
−9.22
−9.23
−9.24
−9.25
−9.26
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_212
Медиана
Среднее

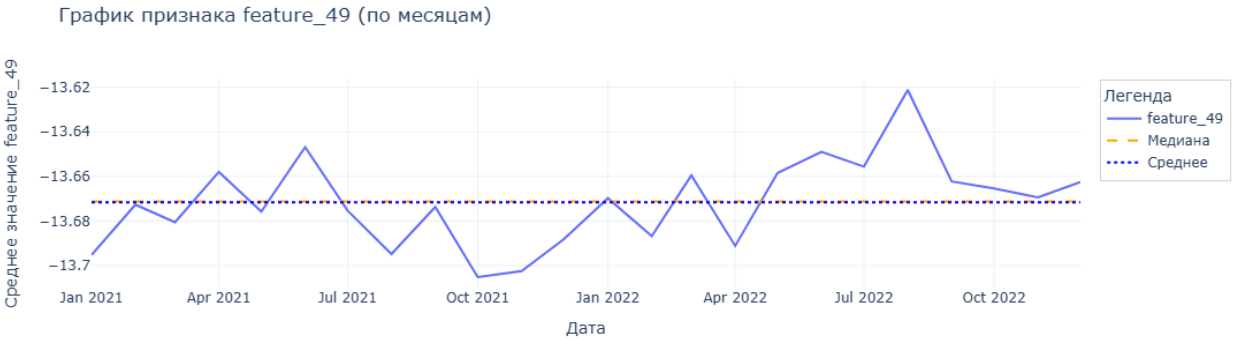


График признака feature_49 (по месяцам)
Среднее значение feature_49
−13.62
−13.64
−13.66
−13.68
−13.7
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_49
Медиана
Среднее

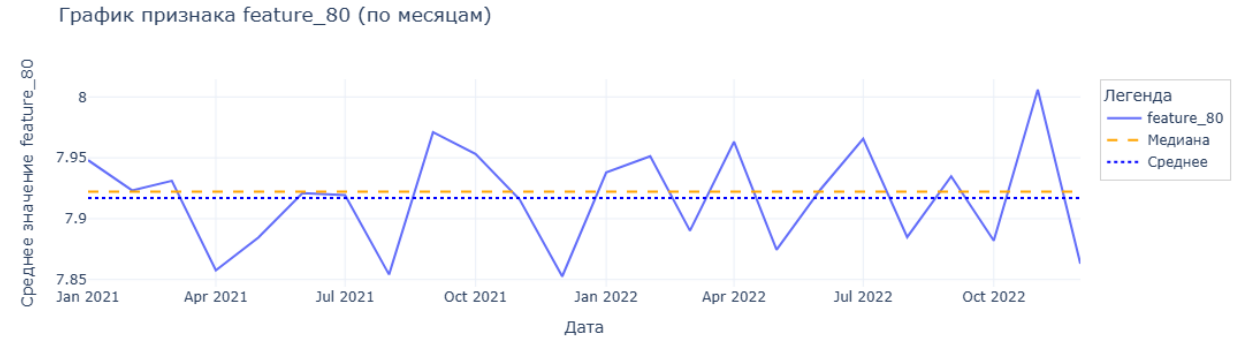


График признака feature_80 (по месяцам)
Среднее значение feature_80
8
7.95
7.9
7.85
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_80
Медиана
Среднее

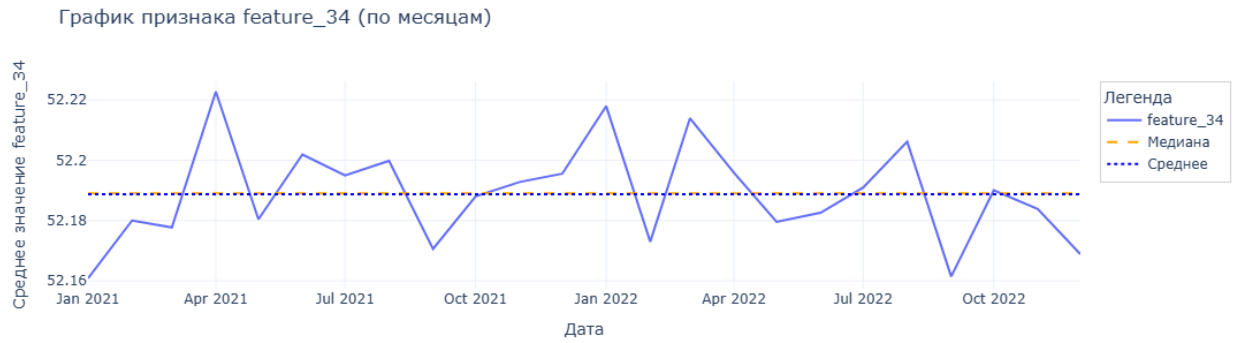


График признака feature_34 (по месяцам)
Среднее значение feature_34
52.22
52.2
52.18
52.16
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_34
Медиана
Среднее

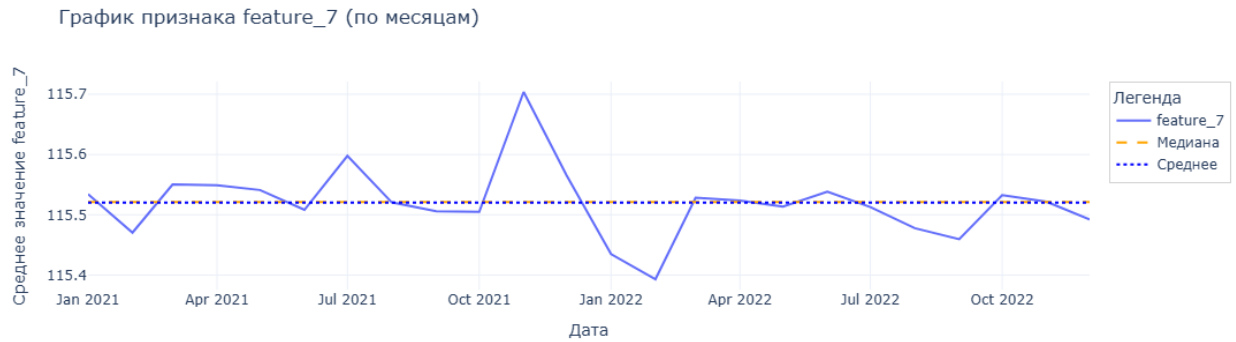


График признака feature_7 (по месяцам)
Среднее значение feature_7
115.7
115.6
115.5
115.4
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_7
Медиана
Среднее

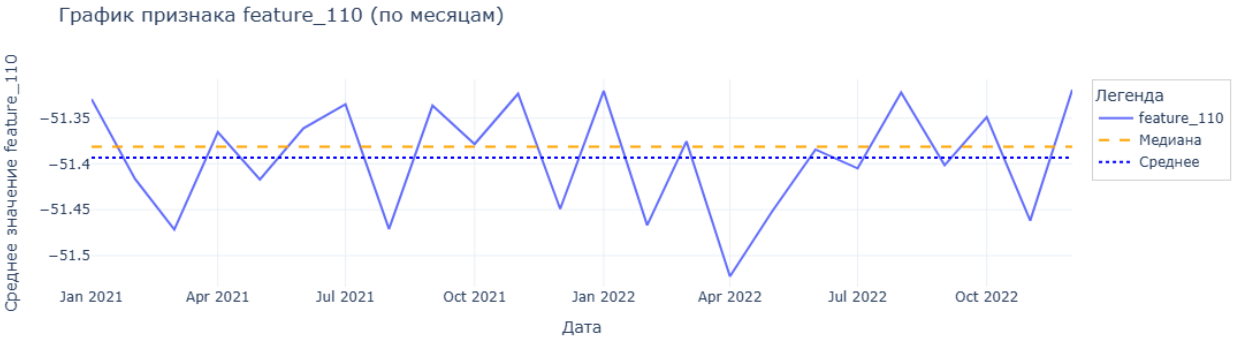


График признака feature_110 (по месяцам)
Среднее значение feature_110
−51.35
−51.4
−51.45
−51.5
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_110
Медиана
Среднее

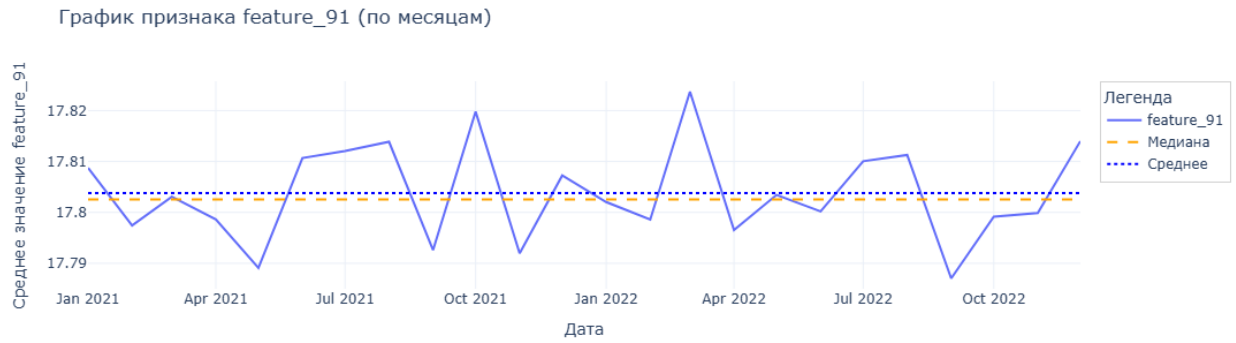


График признака feature_91 (по месяцам)
Среднее значение feature_91
17.82
17.81
17.8
17.79
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_91
Медиана
Среднее

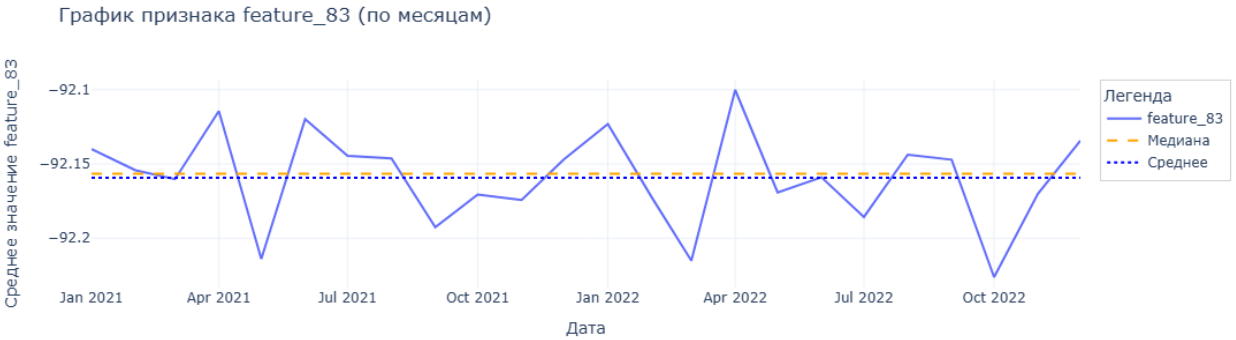


График признака feature_83 (по месяцам)
Среднее значение feature_83
−92.1
−92.15
−92.2
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_83
Медиана
Среднее

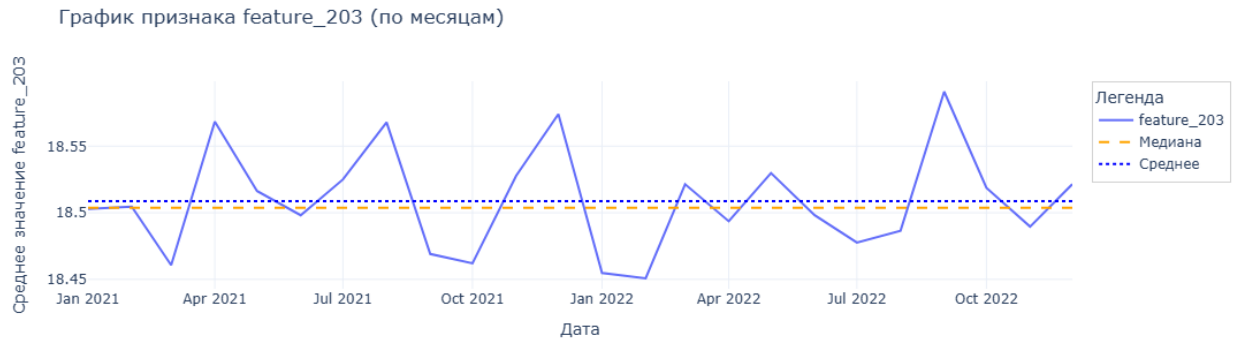


График признака feature_203 (по месяцам)
Среднее значение feature_203
18.55
18.5
18.45
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_203
Медиана
Среднее

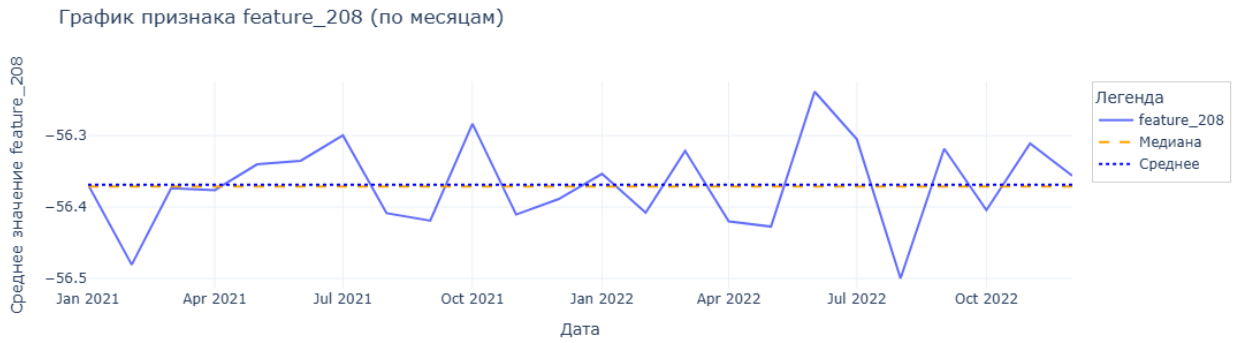


График признака feature_208 (по месяцам)
Среднее значение feature_208
−56.3
−56.4
−56.5
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_208
Медиана
Среднее

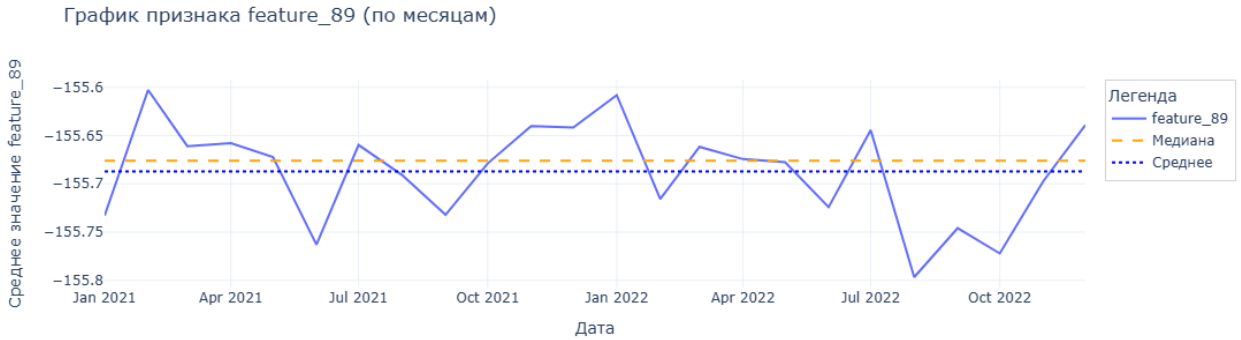


График признака feature_89 (по месяцам)
Среднее значение feature_89
−155.6
−155.65
−155.7
−155.75
−155.8
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_89
Медиана
Среднее

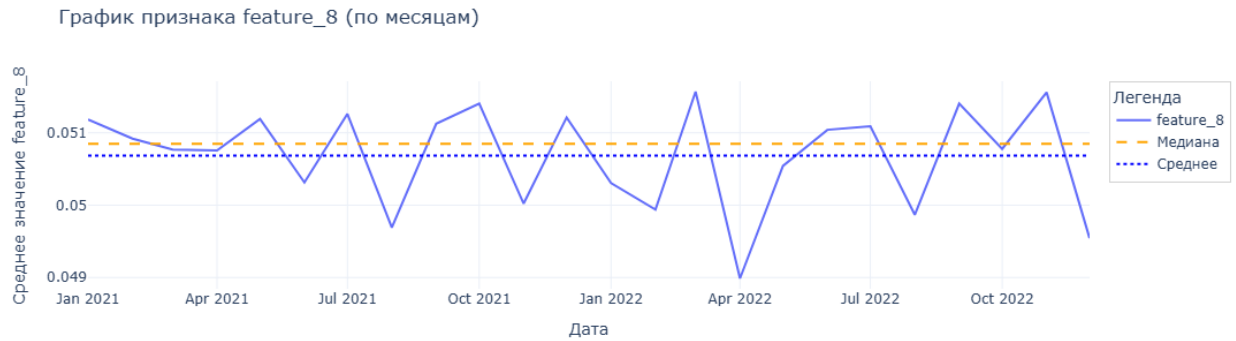


График признака feature_8 (по месяцам)
Среднее значение feature_8
0.051
0.05
0.049
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_8
Медиана
Среднее

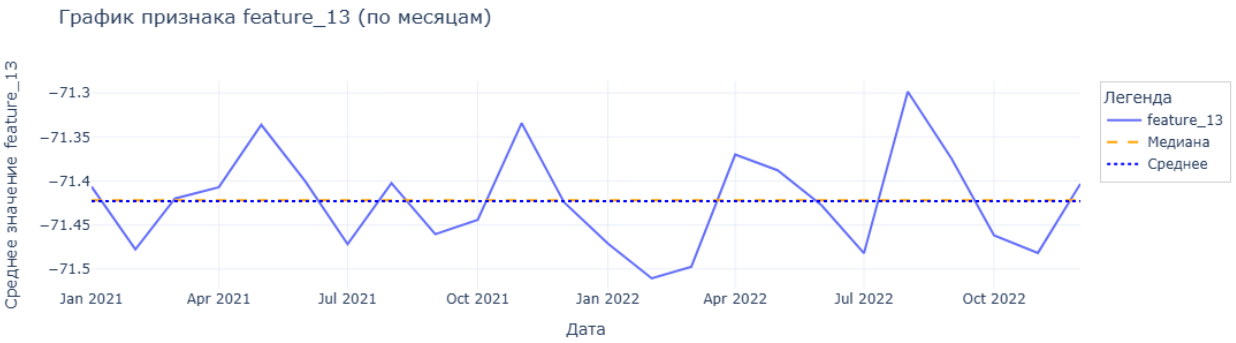


График признака feature_13 (по месяцам)
Среднее значение feature_13
−71.3
−71.35
−71.4
−71.45
−71.5
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_13
Медиана
Среднее

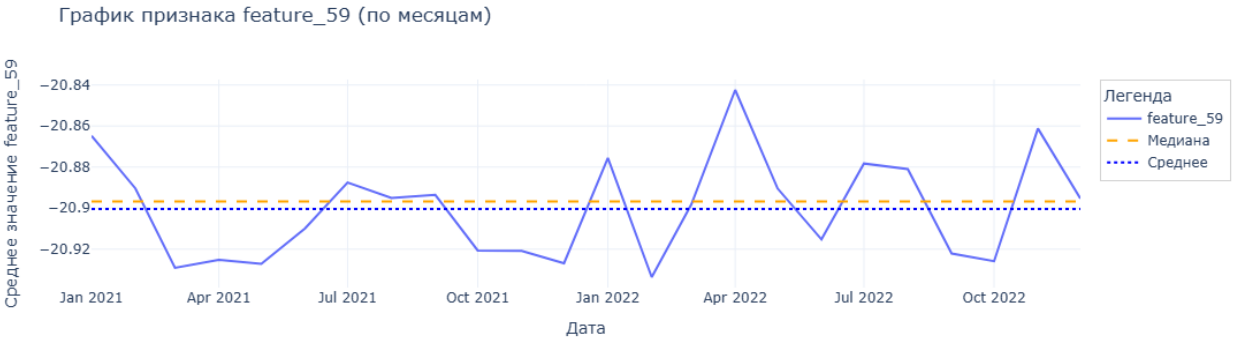


График признака feature_59 (по месяцам)
Среднее значение feature_59
−20.84
−20.86
−20.88
−20.9
−20.92
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_59
Медиана
Среднее

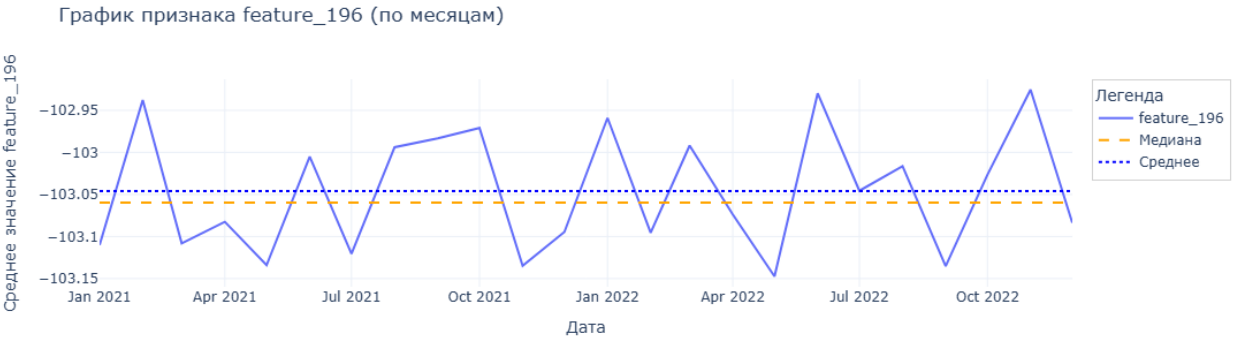


График признака feature_196 (по месяцам)
Среднее значение feature_196
−102.95
−103
−103.05
−103.1
−103.15
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_196
Медиана
Среднее

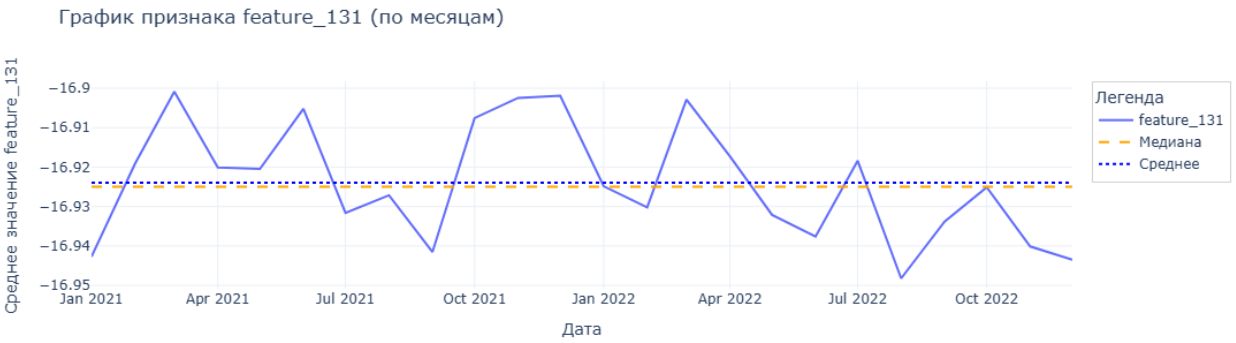


График признака feature_131 (по месяцам)
Среднее значение feature_131
−16.9
−16.91
−16.92
−16.93
−16.94
−16.95
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_131
Медиана
Среднее

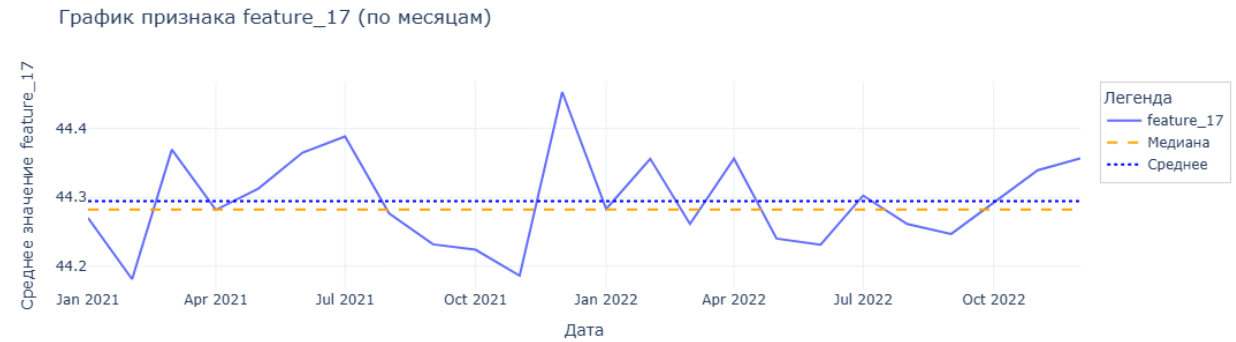


График признака feature_17 (по месяцам)
Среднее значение feature_17
44.4
44.3
44.2
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_17
Медиана
Среднее

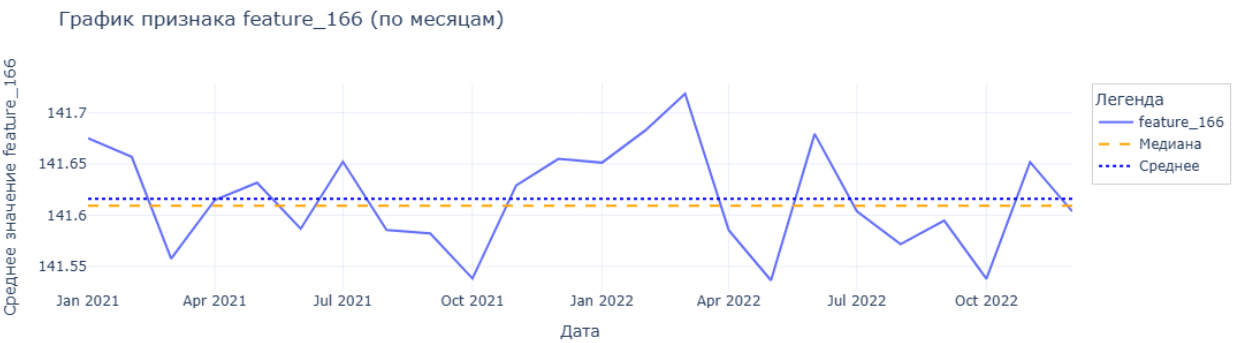


График признака feature_166 (по месяцам)
Среднее значение feature_166
141.7
141.65
141.6
141.55
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_166
Медиана
Среднее

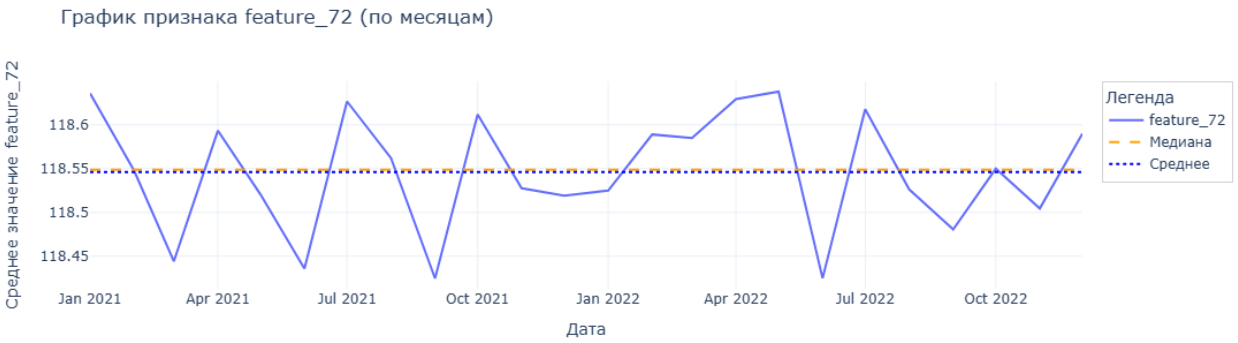


График признака feature_72 (по месяцам)
Среднее значение feature_72
118.6
118.55
118.5
118.45
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_72
Медиана
Среднее

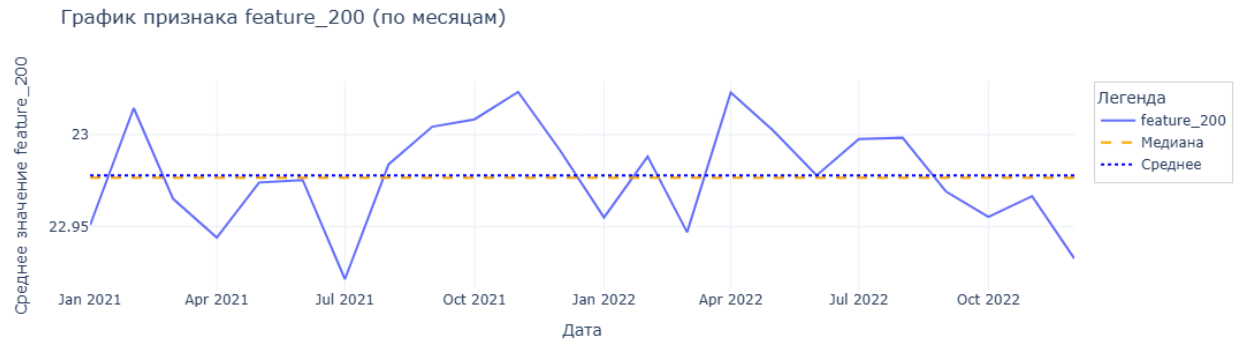


График признака feature_200 (по месяцам)
Среднее значение feature_200
23
22.95
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_200
Медиана
Среднее

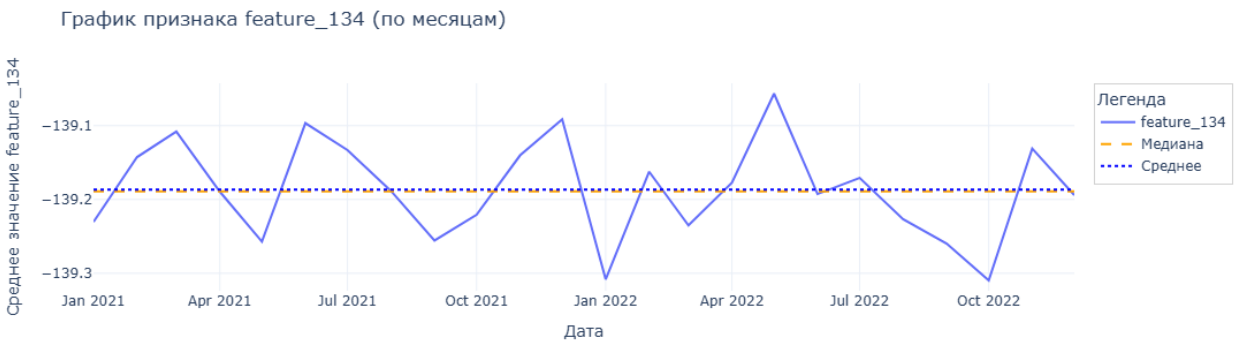
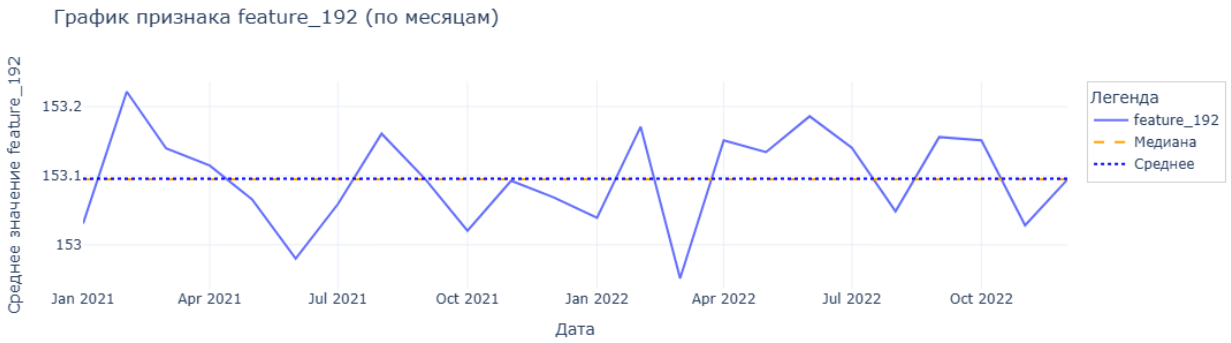


График признака feature_134 (по месяцам)
Среднее значение feature_134
−139.1
−139.2
−139.3
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Дата
Легенда
feature_134
Медиана
Среднее




```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_219","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":
[22.156356161008205,22.168429146373445,22.171521288772375,22.164061510
41365,22.162867727543563,22.15908875891192,22.194392741202915,22.17178
4723738142,22.16458999312691,22.193026319533747,22.180527956181255,22.
178266443059677,22.173133891008764,22.167604605405575,22.1727560157962
1,22.164145858013722,22.17998476872405,22.164689797004858,22.156056201
608806,22.164204596015175,22.168279997315057,22.182787044117706,22.180
281257706874,22.178996489857525]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[22.16997521757291,22.16997521757291]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[22.171576387185002,22.171576387185002]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
```

```
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
```

{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh

```
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_219 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_219"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_63","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-25.53339671899572,-25.564878637324327,-
25.528435865847854,-25.538352616781424,-25.50883205669583,-
25.575265978017296,-25.51682791963312,-25.535276241536465,-
25.554034399220445,-25.520738322790578,-25.50681969903359,-
25.531356325016024,-25.52950756946748,-25.50208312315179,-
25.528671024094972,-25.5444729271361,-25.532041616982557,-
25.49824999102038,-25.55216639481361,-25.53386318962033,-
25.525975031091512,-25.540266821022303,-25.51128235552989,-
25.536059391743102]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
```

```
01T00:00:00"],"y":[-25.53169897099929,-25.53169897099929]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-25.531202259023612,-
25.531202259023612]}],"layout":{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
```

```
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
```

```
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_63 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_63"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_54","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
```

```
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-6.603102618892708,-6.591430751004251,-
6.581288165680997,-6.592843051190018,-6.59833132733634,-
6.60955375792042,-6.585710960503628,-6.608464942436488,-
6.6067852512836005,-6.596096400891548,-6.594851708136471,-
6.611215309301476,-6.594704006653559,-6.6031136695142125,-
6.596671208365983,-6.608156153711325,-6.590018839722515,-
6.584863154410152,-6.598516734752046,-6.59871978936642,-
6.612506859396487,-6.6049724129553296,-6.604899385293373,-
6.602348848288624]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-6.598618262059233,-6.598618262059233]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-6.599131887791998,-6.599131887791998]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
```

```
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
```

```
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
```

```
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_54 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_54"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_107","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-52.45998979900841,-52.48477428038344,-
52.448484609130276,-52.47286732067005,-52.43534789970015,-
52.491626781705236,-52.40512677627266,-52.48394230601839,-
52.50936502121829,-52.46713761030097,-52.47570875261227,-
52.48251027155238,-52.474729064465606,-52.46556056643712,-
52.49530250644859,-52.49204924352729,-52.44437335066088,-
52.490148959797075,-52.505767605922856,-52.47045879089964,-
52.442346143831635,-52.47134394635044,-52.507668173760436,-
52.51621956045033]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-52.475218908538935,-52.475218908538935]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-52.474702055880186,-
52.474702055880186]}],"layout":{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
```

```
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
```

```
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
```

```
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_107 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_107"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_50","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-7.57278099757109,-7.585255597291736,-
7.595993866360754,-7.557925447003025,-7.576247321087353,-
7.5581773984790415,-7.558589213383344,-7.5092928112353725,-
7.522664504973211,-7.560156046868202,-7.5538736823451575,-
7.589165904878475,-7.573517029831413,-7.591378876991792,-
7.553577817928349,-7.567686510367064,-7.603972724229132,-
7.560485512839028,-7.541266864357001,-7.592828884649348,-
7.606338480660792,-7.55921428605909,-7.544451012413968,-
7.558172816379004]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-7.560320779853615,-7.560320779853615]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-7.566375567007615,-7.566375567007615]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
```

{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],

```
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
```

```
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_50 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_50"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_174","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":
[4.093324316359222,4.0508002715548574,4.071593438599799,4.102004898754
505,4.122872242518069,4.071153448548659,4.0792516315319824,4.082436816
566063,4.082362271676385,4.062399486076897,4.111099616862014,4.1369166
```

```
83344209,4.071979771651431,4.0612314386262405,4.0936758718593715,4.106197496472113,4.158745591942739,4.091306209636029,4.061329146575659,4.083276722531439,4.056006016624622,4.109739702187029,4.0812862930335365,4.104755019885668]},{"line":{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Медиана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[4.082856769548751,4.082856769548751]},{"line":{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[4.089406016809106,4.089406016809106]}],"layout":{"hovermode":"x unified","legend":{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwidth":1,"orientation":"v","title":{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"}],"choropleth":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
```

```
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
```

```
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_174 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_174"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
```

{"color":"#636EFA","width":2},"mode":"lines","name":"feature_190","type":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-01T00:00:00"],"y":[-52.14410828905694,-52.11609752940545,-52.14295234689241,-52.16936034971219,-52.069409274468775,-52.13002791350476,-52.10793160072222,-52.14917328410227,-52.18422681965053,-52.12579707581041,-52.08884418597772,-52.11631693103764,-52.137319232034116,-52.14478044702109,-52.09220579125519,-52.146391040293615,-52.07999552243296,-52.15039912660531,-52.09617756147093,-52.12567046325727,-52.10984493234243,-52.08596271184855,-52.08783385163355,-52.16236953450411]},{"line":{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Медиана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[-52.12573376953384,-52.12573376953384]},{"line":{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[-52.123466492293346,-52.123466492293346]}],"layout":{"hovermode":"x unified","legend":{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwidth":1,"orientation":"v","title":{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"}],"choropleth":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":

```
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
```

```
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
```

```
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_190 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_190"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_189","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-94.91184365904121,-94.98571741687208,-
95.03421547411749,-94.982995187588,-94.91864456067599,-
94.92552665891418,-94.91288408359381,-94.93908459174484,-
94.86774275166947,-94.97782773575541,-94.8953062302757,-
94.9487739440462,-94.89918225104223,-94.99985551100237,-
94.89873138685937,-94.9830026389151,-94.8982736143773,-
94.93872154256415,-94.97151679936114,-94.8400182435289,-
94.99517729195574,-94.95163444011666,-94.9072229966552,-
94.90774559650622]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-94.93212410073917,-94.93212410073917]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-94.93715185863243,-94.93715185863243]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
```

```
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
```

```
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
```

{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_189 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_189"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_226","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":
[92.12411860661119,91.97340618769307,92.04791184099777,92.016423954900
16,92.00511344726122,92.00680244854325,92.06326597289653,92.0274934324
6133,91.98403428772384,92.00430896287608,91.99858461383991,92.02414679
502556,92.05163637064214,92.00078584194179,92.0385397640406,92.0482993
6529799,91.97868094355971,91.93102258753426,92.07783478687956,91.99206
520325242,92.05409169829808,92.05092533150741,92.07924969188952,91.985
4484365603]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[92.02028537496287,92.02028537496287]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее

```
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[92.02350794050973,92.02350794050973]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
```

```
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
```

```
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_226 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_226"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_201","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
```

```
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-130.92608314415543,-130.89638207772654,-
130.9890032275903,-130.85733432236273,-131.01915687723505,-
130.93545398209608,-130.95063063333,-130.98865420534761,-
130.95951504686013,-130.92679509947635,-131.03568215817586,-
130.95788996703718,-130.93077481699325,-131.03725876753342,-
130.9236473356663,-130.96010202548126,-130.9166106005037,-
130.90842961302195,-130.87200869585598,-130.9098627335292,-
130.8757199316639,-130.86134344071255,-130.92997296748604,-
130.9965026461425]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-130.93037389223963,-130.93037389223963]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-130.9402005964993,-130.9402005964993]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
```

```
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
```

```
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_201 (по месяцам)"},"xaxis":{"title":
```

{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение feature_201"}}}}

{"config":{"linkText":"Export to plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":[{"line":{"color":"#636EFA","width":2},"mode":"lines","name":"feature_169","type":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-01T00:00:00"],"y":[-0.5954260385672974,-0.5939728989013073,-0.5937887055636659,-0.5956061299434392,-0.5939557295781858,-0.5945767808935701,-0.5960461851927759,-0.5925266226614861,-0.593056418706601,-0.5950341801012407,-0.5945271272247264,-0.5956041305484069,-0.5942478205228481,-0.5944493061959192,-0.5945919094589358,-0.5943591490130096,-0.5935347315518507,-0.5930940251447259,-0.5947950904776027,-0.5942211316084794,-0.5934022853805223,-0.5929976917872324,-0.5934527361409377,-0.5942738126671839]},{"line":{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Медиана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[-0.5942608165950161,-0.5942608165950161]},{"line":{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее","type":"scatter","x":["2021-01-01T00:00:00","2022-12-01T00:00:00"],"y":[-0.5942308599096645,-0.5942308599096645]}],"layout":{"hovermode":"x unified","legend":{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwidth":1,"orientation":"v","title":{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"}],"choropleth":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
```

{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl":[{"marker":{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterternary":[{"marker":{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":{"color":"#C8D4E3"},"line":{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers":"strict","coloraxis":{"colorbar":{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692","#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes":true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":{"align":"left"},"hovermode":"closest","mapbox":{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","polar":{"angularaxis":{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"white","radialaxis":{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":{"xaxis":{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"linecolor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0F8"},"yaxis":{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"linecolor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0F8"},"zaxis":{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco

```
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_169 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_169"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_103","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-15.275153962218313,-15.283675983090237,-
15.30051431870057,-15.264835469204076,-15.269144187288344,-
15.276259322635667,-15.318931835644516,-15.329436732002108,-
15.261167209515138,-15.262516442798859,-15.32805664667078,-
15.274001599664002,-15.272791856570606,-15.252211310232738,-
15.255384065294033,-15.30821591002921,-15.275325295225327,-
15.275268553824446,-15.272253242794829,-15.275534839523113,-
15.296881782848988,-15.277621386709022,-15.311529357067146,-
15.284556376184078]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-15.27543006737422,-15.27543006737422]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-15.283386153572339,-
15.283386153572339]}],"layout":{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
```

```
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
```

```
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
```

```
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_103 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_103"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_99","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":
[28.56496214873198,28.563466059029547,28.51900040747177,28.52494141212
4406,28.523210630980287,28.553690781102567,28.53079286125182,28.529306
823134785,28.56133562159724,28.552582284333287,28.544739225094887,28.5
2780296442839,28.5379176647157,28.549007952659306,28.500257735097,28.4
93787905298383,28.527715240363825,28.551586519406822,28.55955609135363
8,28.546278018209236,28.545495802660813,28.51551440432389,28.524339546
```

```
968594,28.523992351330502]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[28.53435526298376,28.53435526298376]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[28.53630335215286,28.53630335215286]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
```

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
```

```
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_99 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_99"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_87","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
```

```
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-93.99663399401034,-94.01718722873315,-
93.99035283474562,-94.11524412797566,-94.02794841414047,-
94.09657573897776,-93.97853779771066,-94.03358476997755,-
93.97431141304124,-93.94033709174154,-94.06822733381227,-
94.04137809248789,-93.98279098296733,-94.07563596564276,-
94.18951376803756,-94.06438390461821,-94.04058270936798,-
93.95448984062905,-93.99827681203475,-94.10828575276764,-
94.03955128933707,-94.08163887912188,-94.04789393010503,-
94.09170605568804]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-94.04006699935252,-94.04006699935252]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-94.03979453031964,-94.03979453031964]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
```

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
```

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
```

```
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_87 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_87"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_202","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-67.99577378166069,-67.90387854084639,-
67.79743610811622,-67.92186126305631,-67.86860342234334,-
67.90360949968604,-68.07777621268521,-67.76063759135576,-
67.96936716637157,-67.92393132019801,-67.95604814941613,-
67.88521658490681,-67.80284979905986,-67.95606575974509,-
67.91503335256542,-67.90495242318597,-67.92559307884385,-
67.99597839819583,-67.8311447439222,-67.96282465974669,-
67.88989083991703,-67.88891638391085,-67.90732574063276,-
67.90478036463657]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-67.90613908190937,-67.90613908190937]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-67.91039563270853,-67.91039563270853]}],"layout":
{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
```

```
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
```

{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco

```
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_202 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_202"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_74","type
":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-03-
01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":
[49.98380290221249,49.99690091915766,49.97544205275185,49.998510007785
036,49.93060281442449,49.86855136414389,49.924330662300605,49.98141436
992793,49.99098463813388,49.91678869547984,50.021131176278224,49.90203
70383906,50.03682461873394,49.987675048052324,49.93663219562974,49.993
26150957261,49.99621420902012,49.90180158735587,49.94979082042265,49.9
6124663545441,49.98231168095955,49.995908976733766,49.89953366713211,4
9.88666928940224]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[49.97842821133989,49.97842821133989]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[49.95909861997732,49.95909861997732]}],"layout":
{"hovermode":"x unified","legend":
```

{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwidth":1,"orientation":"v","title":{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"template":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpolar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"carpet":[{"aaxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3","minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"}],"choropleth":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],

```
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
```

```
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_74 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_74"}}}}

{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_214","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-14.811079135297117,-14.792894074848082,-
14.827740676228284,-14.823421386162003,-14.811983388495435,-
```

```
14.818937517571143,-14.80092034459141,-14.845884562708232,-
14.82399024865445,-14.793726201426137,-14.807894740471513,-
14.821301060863204,-14.819759974813934,-14.77801353999648,-
14.80171591817744,-14.81744591486925,-14.81049817067033,-
14.817391686888977,-14.796272061958822,-14.792810053829607,-
14.836206324062998,-14.811296523014244,-14.827016952693423,-
14.81174212870701]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-14.811862758601222,-14.811862758601222]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-14.812497607791647,-
14.812497607791647]}],"layout":{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
```

```
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
```

```
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_214 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_214"}}}}
```

```
{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"line":
{"color":"#636EFA","width":2},"mode":"lines","name":"feature_210","typ
e":"scatter","x":["2021-01-01T00:00:00","2021-02-01T00:00:00","2021-
03-01T00:00:00","2021-04-01T00:00:00","2021-05-01T00:00:00","2021-06-
01T00:00:00","2021-07-01T00:00:00","2021-08-01T00:00:00","2021-09-
01T00:00:00","2021-10-01T00:00:00","2021-11-01T00:00:00","2021-12-
01T00:00:00","2022-01-01T00:00:00","2022-02-01T00:00:00","2022-03-
01T00:00:00","2022-04-01T00:00:00","2022-05-01T00:00:00","2022-06-
01T00:00:00","2022-07-01T00:00:00","2022-08-01T00:00:00","2022-09-
01T00:00:00","2022-10-01T00:00:00","2022-11-01T00:00:00","2022-12-
01T00:00:00"],"y":[-116.44078614005386,-116.40384801774802,-
116.35112725022272,-116.35747245453031,-116.37575740337836,-
116.32581996390138,-116.40217675344272,-116.31822424372294,-
116.3984901052004,-116.33769007896066,-116.30625309634574,-
116.44213138271762,-116.38139796420558,-116.39049560507242,-
116.45851282051291,-116.29136792400891,-116.39053107103155,-
116.32404734070192,-116.32829762668354,-116.36959888432322,-
116.47232596452315,-116.28808545447775,-116.35747798948641,-
116.501457349946]},{"line":
{"color":"orange","dash":"dash","width":2},"mode":"lines","name":"Меди
ана","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-116.37267814385079,-116.37267814385079]},{"line":
{"color":"blue","dash":"dot","width":2},"mode":"lines","name":"Среднее
","type":"scatter","x":["2021-01-01T00:00:00","2022-12-
01T00:00:00"],"y":[-116.37555720354992,-
116.37555720354992]}],"layout":{"hovermode":"x unified","legend":
{"bgcolor":"rgba(255,255,255,0.8)","bordercolor":"lightgray","borderwi
dth":1,"orientation":"v","title":
{"text":"Легенда"},"x":1.02,"xanchor":"left","y":1,"yanchor":"top"},"t
emplate":{"data":{"bar":[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"white","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"#C8D4E3","linecolor":"#C8D4E3",
"minorgridcolor":"#C8D4E3","startlinecolor":"#2a3f5f"},"type":"carpet"
}],"choropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
```

```
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
```

```
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"white","showlakes"
:true,"showland":true,"subunitcolor":"#C8D4E3"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"white","pola
r":{"angularaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""},"bgcolor":"wh
ite","radialaxis":
{"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"yaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"},"zaxis":
{"backgroundcolor":"white","gridcolor":"#DFE8F3","gridwidth":2,"lineco
lor":"#EBF0F8","showbackground":true,"ticks":"","zerolinecolor":"#EBF0
F8"}},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"baxis":
{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""},"bgcolor":"wh
ite","caxis":
```

{"gridcolor":"#DFE8F3","linecolor":"#A2B1C6","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"#EBF0F8","linecolor":"#EBF0F8","ticks"
:"","title":
{"standoff":15},"zerolinecolor":"#EBF0F8","zerolinewidth":2}}},"title"
:{"text":"График признака feature_210 (по месяцам)"},"xaxis":{"title":
{"text":"Дата"}},"yaxis":{"title":{"text":"Среднее значение
feature_210"}}}}

Выводы: ощущается временная структура, поэтому я все-таки экстраполирую (пропусков не очень много).

Полиномиальная интерполяция 2-го порядка для заполнения пропусков

```
df_missed_values_list.dtypes
```

```
feature_223    float64
feature_30     float64
feature_16     float64
feature_124    float64
feature_10     float64
feature_68     float64
feature_136    float64
feature_28     float64
feature_115    float64
feature_51     float64
feature_140    float64
feature_78     float64
feature_161    float64
feature_224    float64
feature_228    float64
feature_220    float64
feature_163    float64
feature_62     float64
feature_172    float64
feature_175    float64
feature_148    float64
feature_205    float64
feature_3      float64
feature_105    float64
feature_53     float64
feature_133    float64
feature_207    float64
feature_176    float64
feature_212    float64
feature_49     float64
feature_80     float64
```

```
feature_34     float64
feature_7      float64
feature_110    float64
feature_91     float64
feature_83     float64
feature_203    float64
feature_208    float64
feature_89     float64
feature_8      float64
feature_13     float64
feature_59     float64
feature_196    float64
feature_131    float64
feature_17     float64
feature_166    float64
feature_72     float64
feature_200    float64
feature_134    float64
feature_192    float64
feature_219    float64
feature_63     float64
feature_54     float64
feature_107    float64
feature_50     float64
feature_174    float64
feature_190    float64
feature_189    float64
feature_226    float64
feature_201    float64
feature_169    float64
feature_103    float64
feature_99     float64
feature_87     float64
feature_202    float64
feature_74     float64
feature_214    float64
feature_210    float64
dtype: object
```

ВАЖНО! Лучше использовать полиномиальную аппроксимацию, но она считается заметно дольше, поэтому использую линейную!

```
if isinstance(df_missed_values_list, pd.DataFrame):
    cols = df_missed_values_list.columns.tolist()
else:
    cols = df_missed_values_list.index.tolist()

cols = [c for c in cols if c in train.columns]

train = train.sort_values("date").copy()
```

```
val   = val.sort_values("date").copy()
test  = test.sort_values("date").copy()

train.loc[:, cols] = train[cols].interpolate(method="linear", order=2,
limit_direction="both")
val.loc[:, cols]   = val[cols].interpolate(method="linear", order=2,
limit_direction="both")
test.loc[:, cols]  = test[cols].interpolate(method="linear", order=2,
limit_direction="both")

miss_train = train.isna().sum()
miss_val   = val.isna().sum()
miss_test  = test.isna().sum()

print("TRAIN missing (total):", int(miss_train.sum()))
print("VAL   missing (total):", int(miss_val.sum()))
print("TEST  missing (total):", int(miss_test.sum()))

display(miss_train[miss_train >
0].sort_values(ascending=False).head(30))
display(miss_val[miss_val > 0].sort_values(ascending=False).head(30))
display(miss_test[miss_test >
0].sort_values(ascending=False).head(30))

miss_summary = (
    pd.DataFrame({
        "miss_train": miss_train,
        "miss_val": miss_val,
        "miss_test": miss_test,
    })
    .assign(
        miss_total=lambda d: d["miss_train"] + d["miss_val"] +
d["miss_test"]
    )
    .query("miss_total > 0")
    .sort_values("miss_total", ascending=False)
)

miss_summary.head(50)
```

```
TRAIN missing (total): 17260
VAL   missing (total): 6822
TEST  missing (total): 10393

feature_30_squared    17260
dtype: int64

feature_30_squared    6822
dtype: int64
```

```
feature_30_squared    10393
dtype: int64

                    miss_train  miss_val  miss_test  miss_total
feature_30_squared       17260      6822      10393       34475
```

```
col = "feature_30_squared"

train = train.sort_values("date").copy()
val   = val.sort_values("date").copy()
test  = test.sort_values("date").copy()

train[col] = (
    train.set_index("date")[col]
         .interpolate(method="time", limit_direction="both")
         .to_numpy()
)

val[col] = (
    val.set_index("date")[col]
       .interpolate(method="time", limit_direction="both")
       .to_numpy()
)

test[col] = (
    test.set_index("date")[col]
        .interpolate(method="time", limit_direction="both")
        .to_numpy()
)
```

Вспомним про категориальные признаки, их нужно закодировать.

Задание: Используя `OneHotEncoder` закодируйте категориальные признаки.

Я уже закодировал признаки, используя порядоковое кодирование, но попробуем написать псевдо-код! Почему? Очень много уже признаков, боюсь, можно спутать используемые признаки и пр.

```
from sklearn.preprocessing import OneHotEncoder
```

```
# например! были они прямо category, но вообще логику описывал ранее, надо брать не только их!
cat_cols = ["feature_0", "feature_209"]

enc = OneHotEncoder(handle_unknown="ignore", sparse_output=False)

X_train_cat = enc.fit_transform(train[cat_cols])
X_val_cat   = enc.transform(val[cat_cols])
X_test_cat  = enc.transform(test[cat_cols])
```

```
ohe_cols = enc.get_feature_names_out(cat_cols)

train_ohe = pd.DataFrame(X_train_cat, columns=ohe_cols,
index=train.index)
val_ohe   = pd.DataFrame(X_val_cat,   columns=ohe_cols,
index=val.index)
test_ohe  = pd.DataFrame(X_test_cat,  columns=ohe_cols,
index=test.index)

train = pd.concat([train.drop(columns=cat_cols), train_ohe], axis=1)
val   = pd.concat([val.drop(columns=cat_cols),   val_ohe], axis=1)
test  = pd.concat([test.drop(columns=cat_cols),  test_ohe], axis=1)
```

Stepwise и обучение логрега (4 балла)

Задание (boss): Реализуйте Stepwise-алгоритм.

Ваша функция `stepwise` должна принимать на вход:

- Датафрейм со всеми признаками и таргетом
- список с именами рассматриваемых признаков
- строку-имя таргета
- уровни значимости `alpha_in` и `alpha_out`

И возвращать список отобранных признаков.

Во время работы пусть она также выводит, какой признак был включён или исключён и с каким `p-value`

```
from scipy.stats.distributions import chi2
from sklearn.metrics import log_loss

def likelihood_ratio_test(ll_short, ll_long):

    """
    вспомогательная функция
    рассчитывает значение p-value для теста отношения правдоподобия
    ll_short – логарифм правдоподобия модели на k переменных
    ll_long – логарифм правдоподобия модели на k+1 переменной

    Returns
    -----
    p-value
    """
    # тут ошибка была!!! не тот порядок!
    # lr = 2 * (ll_short - ll_long)
    lr = 2 * (ll_long - ll_short)
    return chi2.sf(lr, 1)
```

```
def stepwise(
        df: pd.DataFrame,
        features: list[str],
        target: str,
        alpha_in: float = 0.01,
        alpha_out: float = 0.05
) -> list[str]:

    features = list(features)
    selected_features = list()

    def fit_ll(cols):
        cols = list(cols)
        use_cols = cols + [target]
        d = df[use_cols].dropna()

        y = d[target].astype(int).to_numpy()
        n = len(y)

        if n == 0:
            return np.nan

        if len(cols) == 0:
            p = np.full(n, y.mean() if n > 0 else 0.5)
            ll = -n * log_loss(y, p, labels=[0, 1])
            return ll

        X = d[cols].to_numpy()
        clf = LogisticRegression(
            penalty="l2",
            C=1e6,
            solver="lbfgs",
            max_iter=2000,
            n_jobs=-1
        )
        clf.fit(X, y)
        p = clf.predict_proba(X)[:, 1]
        ll = -n * log_loss(y, p, labels=[0, 1])
        return ll

    while True:

        # forward:
        best_feature = None
        if (len(selected_features) < len(features)):

            ll_short = fit_ll(selected_features)

            best_p = np.inf
            for f in features:
                if f in selected_features:
```

```
                    continue

                ll_long = fit_ll(selected_features + [f])
                if np.isnan(ll_short) or np.isnan(ll_long):
                    continue

                p = likelihood_ratio_test(ll_short, ll_long)
                if p < best_p:
                    best_p = p
                    best_feature = f

            if best_feature and (best_p < alpha_in):
                selected_features.append(best_feature)
                p_value = best_p
                print(f"В модель была добавлена переменная
{best_feature}, p-value: {round(p_value, 6)}")
            else:
                best_feature = None

        # backward
        worst_feature = None
        if (len(selected_features) > 1):

            ll_long = fit_ll(selected_features)

            worst_p = -np.inf
            for f in list(selected_features):
                cols_short = [c for c in selected_features if c != f]
                ll_short = fit_ll(cols_short)

                if np.isnan(ll_short) or np.isnan(ll_long):
                    continue

                p = likelihood_ratio_test(ll_short, ll_long)
                if p > worst_p:
                    worst_p = p
                    worst_feature = f

            if worst_feature and (worst_p > alpha_out):
                selected_features.remove(worst_feature)
                p_value = worst_p
                print(f"Из модели была удалена переменная
{worst_feature}, p-value: {round(p_value, 6)}")
            else:
                worst_feature = None

        if not (best_feature or worst_feature):
            break

    return selected_features
```

Запустите ваш алгоритм на отобранных фичах со значениями `alpha_in = 0.01`, `alpha_out = 0.02`

Если в степвайз заходят *все переменные*, причём с очень маленькими `p-value` - это неудивительно, ведь вы уже провели серьёзный предварительный отбор фичей.

Чтобы убедиться в коректности работы своего алгоритма, можете попробовать запустить его на каких-нибудь других фичах, откинутых сильно ранее

```
# selected_features = stepwise(..., ..., ..., alpha_in = 0.01,
alpha_out = 0.02)
```

Идея: мы сформировали достаточно маленький датасет после всех обработок (и обработки именно признаков из списка по gain). А давайте как в классическом эконометрическом исследовании и рассмотрим данное признаковое подпространство!

Спойлер: немного не хватило, чтобы побить метрику. Идея была не самая лучшая!

```
alpha_in, alpha_out = 0.01, 0.02

selected = stepwise(
    df=train,
    features=final_features_logreg,
    target=TARGET,
    alpha_in=alpha_in,
    alpha_out=alpha_out
)

print("Итоговые признаки:", len(selected))
print(selected)
```

```
В модель была добавлена переменная feature_117, p-value: 0.0
В модель была добавлена переменная feature_124, p-value: 0.0
В модель была добавлена переменная feature_30_squared, p-value: 0.0
В модель была добавлена переменная feature_76_squared, p-value: 0.0

---------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[728], line 3
      1 alpha_in, alpha_out = 0.01, 0.02
----> 3 selected = stepwise(
      4     df=train,
      5     features=final_features_logreg,
      6     target=TARGET,
      7     alpha_in=alpha_in,
      8     alpha_out=alpha_out
```

```
      9 )
     11 print("Итоговые признаки:", len(selected))
     12 print(selected)

Cell In[726], line 54, in stepwise(df, features, target, alpha_in,
alpha_out)
     51 if f in selected_features:
     52     continue
---> 54 ll_long = fit_ll(selected_features + [f])
     55 if np.isnan(ll_short) or np.isnan(ll_long):
     56     continue

Cell In[726], line 36, in stepwise.<locals>.fit_ll(cols)
     28 X = d[cols].to_numpy()
     29 clf = LogisticRegression(
     30     penalty="l2",
     31     C=1e6,               # почти без регуляризации (ближе к
MLE)
   (...)
     34     n_jobs=-1
     35 )
---> 36 clf.fit(X, y)
     37 p = clf.predict_proba(X)[:, 1]
     38 ll = -n * log_loss(y, p, labels=[0, 1])

File D:\Anaconda\Lib\site-packages\sklearn\base.py:1389, in
_fit_context.<locals>.decorator.<locals>.wrapper(estimator, *args,
**kwargs)
   1382     estimator._validate_params()
   1384 with config_context(
   1385     skip_parameter_validation=(
   1386         prefer_skip_nested_validation or
global_skip_validation
   1387     )
   1388 ):
-> 1389     return fit_method(estimator, *args, **kwargs)

File D:\Anaconda\Lib\site-packages\sklearn\linear_model\
_logistic.py:1350, in LogisticRegression.fit(self, X, y,
sample_weight)
   1347 else:
   1348     n_threads = 1
-> 1350 fold_coefs_ = Parallel(n_jobs=self.n_jobs,
verbose=self.verbose, prefer=prefer)(
   1351     path_func(
   1352         X,
   1353         y,
   1354         pos_class=class_,
   1355         Cs=[C_],
   1356         l1_ratio=self.l1_ratio,
```

```
   1357         fit_intercept=self.fit_intercept,
   1358         tol=self.tol,
   1359         verbose=self.verbose,
   1360         solver=solver,
   1361         multi_class=multi_class,
   1362         max_iter=self.max_iter,
   1363         class_weight=self.class_weight,
   1364         check_input=False,
   1365         random_state=self.random_state,
   1366         coef=warm_start_coef_,
   1367         penalty=penalty,
   1368         max_squared_sum=max_squared_sum,
   1369         sample_weight=sample_weight,
   1370         n_threads=n_threads,
   1371     )
   1372     for class_, warm_start_coef_ in zip(classes_,
warm_start_coef)
   1373 )
   1375 fold_coefs_, _, n_iter_ = zip(*fold_coefs_)
   1376 self.n_iter_ = np.asarray(n_iter_, dtype=np.int32)[:, 0]

File D:\Anaconda\Lib\site-packages\sklearn\utils\parallel.py:77, in
Parallel.__call__(self, iterable)
     72 config = get_config()
     73 iterable_with_config = (
     74     (_with_config(delayed_func, config), args, kwargs)
     75     for delayed_func, args, kwargs in iterable
     76 )
---> 77 return super().__call__(iterable_with_config)

File D:\Anaconda\Lib\site-packages\joblib\parallel.py:2007, in
Parallel.__call__(self, iterable)
   2001 # The first item from the output is blank, but it makes the
interpreter
   2002 # progress until it enters the Try/Except block of the
generator and
   2003 # reaches the first `yield` statement. This starts the
asynchronous
   2004 # dispatch of the tasks to the workers.
   2005 next(output)
-> 2007 return output if self.return_generator else list(output)

File D:\Anaconda\Lib\site-packages\joblib\parallel.py:1711, in
Parallel._get_outputs(self, iterator, pre_dispatch)
   1709     self._running = False
   1710     if not detach_generator_exit:
-> 1711         self._terminate_and_reset()
   1713 while len(_remaining_outputs) > 0:
   1714     batched_results = _remaining_outputs.popleft()
```

```
File D:\Anaconda\Lib\site-packages\joblib\parallel.py:1386, in
Parallel._terminate_and_reset(self)
   1384 self._calling = False
   1385 if not self._managed_backend:
-> 1386     self._backend.terminate()

File D:\Anaconda\Lib\site-packages\joblib\_parallel_backends.py:610,
in LokyBackend.terminate(self)
    605 def terminate(self):
    606     if self._workers is not None:
    607         # Don't terminate the workers as we want to reuse them
in later
    608         # calls, but cleanup the temporary resources that the
Parallel call
    609         # created. This 'hack' requires a private, low-level
operation.
--> 610
self._workers._temp_folder_manager._clean_temporary_resources(
    611             context_id=self.parallel._id, force=False
    612         )
    613         self._workers = None
    615     self.reset_batch_stats()

File D:\Anaconda\Lib\site-packages\joblib\_memmapping_reducer.py:641,
in TemporaryResourcesManager._clean_temporary_resources(self,
context_id, force, allow_non_empty)
    638 # Clean up the folder if possible, either if it is empty or
    639 # if none of the files in it are in used and allow_non_empty.
    640 try:
--> 641     delete_folder(
    642         temp_folder, allow_non_empty=allow_non_empty
    643     )
    644     # Forget the folder once it has been deleted
    645     self._cached_temp_folders.pop(context_id, None)

File D:\Anaconda\Lib\site-packages\joblib\disk.py:136, in
delete_folder(folder_path, onerror, allow_non_empty)
    131     if err_count > RM_SUBDIRS_N_RETRY:
    132         # the folder cannot be deleted right now. It maybe
    133         # because some temporary files have not been deleted
    134         # yet.
    135         raise
--> 136 time.sleep(RM_SUBDIRS_RETRY_TIME)

KeyboardInterrupt:
```

Очень долго идет обработка, конечно! Но работает! Прерву, ноутбук не тянет...

Задание: На получившемся наборе признаков обучим, наконец, логрег!

Для обучения можно использовать трейн + валидацию вместе, либо просто трейн

Не забудьте отскалировать фичи

```
from sklearn.preprocessing import StandardScaler # Или другой скейлер
from sklearn.linear_model import LogisticRegression
from scipy.special import logit
```

Заново прогоним пайплайн

```
le_cols = ["feature_0", "feature_209", "feature_10", "feature_161",
"feature_62"]

for c in le_cols:
    train[c] = train[c].astype("object").where(train[c].notna(),
"missing").astype(str)
    val[c]   = val[c].astype("object").where(val[c].notna(),
"missing").astype(str)
    test[c]  = test[c].astype("object").where(test[c].notna(),
"missing").astype(str)

for c in le_cols:
    codes, uniques = pd.factorize(train[c], sort=True)
    mapping = {cat: i for i, cat in enumerate(uniques)}

    train[c + "_le"] = train[c].map(mapping).astype("int32")
    val[c + "_le"]   = val[c].map(mapping).fillna(-1).astype("int32")
    test[c + "_le"]  = test[c].map(mapping).fillna(-1).astype("int32")

# scaler = StandardScaler()
# scaler.fit(...)
# # your code here
# logreg = LogisticRegression(penalty=None)
# logreg.fit(...)
# df_for_logreg['logreg'] = logreg.predict_proba(...)[:, 1]
# df_for_logreg['logit_logreg'] = logit(df_for_logreg['logreg'])

# список фичей
features_lr = features_final

X_trainval = pd.concat([train[features_lr], val[features_lr]], axis=0)
y_trainval = pd.concat([train[TARGET], val[TARGET]],
axis=0).astype(int)

scaler = StandardScaler()
scaler.fit(X_trainval)

X_trainval_sc = scaler.transform(X_trainval)
logreg = LogisticRegression(penalty=None, max_iter=5000)
logreg.fit(X_trainval_sc, y_trainval)
```

```
# train
X_tr = scaler.transform(train[features_lr])
train = train.copy()
train["logreg"] = logreg.predict_proba(X_tr)[:, 1]
train["logit_logreg"] = logit(np.clip(train["logreg"], 1e-12, 1-1e-
12))

# val
X_va = scaler.transform(val[features_lr])
val = val.copy()
val["logreg"] = logreg.predict_proba(X_va)[:, 1]
val["logit_logreg"] = logit(np.clip(val["logreg"], 1e-12, 1-1e-12))

# test
X_te = scaler.transform(test[features_lr])
test = test.copy()
test["logreg"] = logreg.predict_proba(X_te)[:, 1]
test["logit_logreg"] = logit(np.clip(test["logreg"], 1e-12, 1-1e-12))
```

Задание: Оцените качество получившегося логрега **на тестовой выборке** по метрике `roc_auc`.

Ваша задача - побить `threshold_auc = 0.622`. Если этого сделать ну совсем не получается, попробуйте добавить в модель ещё переменных (но должно получаться :)

Также:

- Постройте графики линейности по WoE получившегося логрега на трейне и тестовой выборке (оценивайте линейность для `logit_logreg`, а не для вероятности!)
- Постройте график стабильности `roc_auc` во времени для получившегося логрега

```
threshold_auc = 0.622

# AUC на test
X_test_sc = scaler.transform(test[features_lr])
preds_test = logreg.predict_proba(X_test_sc)[:, 1]
auc_test = roc_auc_score(test[TARGET].astype(int), preds_test)

print(f"ROC-AUC on test: {auc_test:.4f} (threshold {threshold_auc})")
```

```
ROC-AUC on test: 0.6376 (threshold 0.622)
```

УРА!

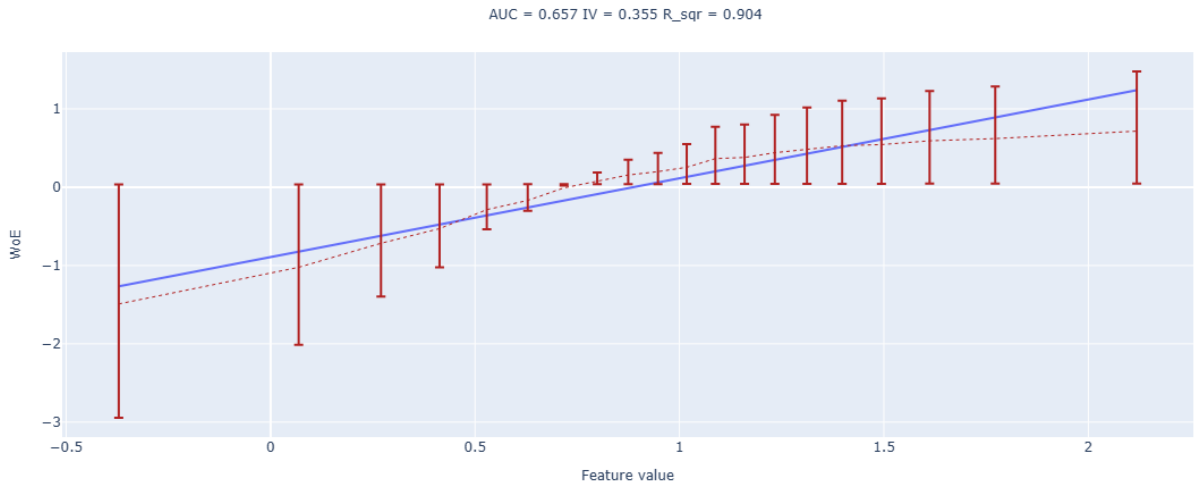
Задание: Визуализируйте важность фичей полученной линейной модели.

Также:

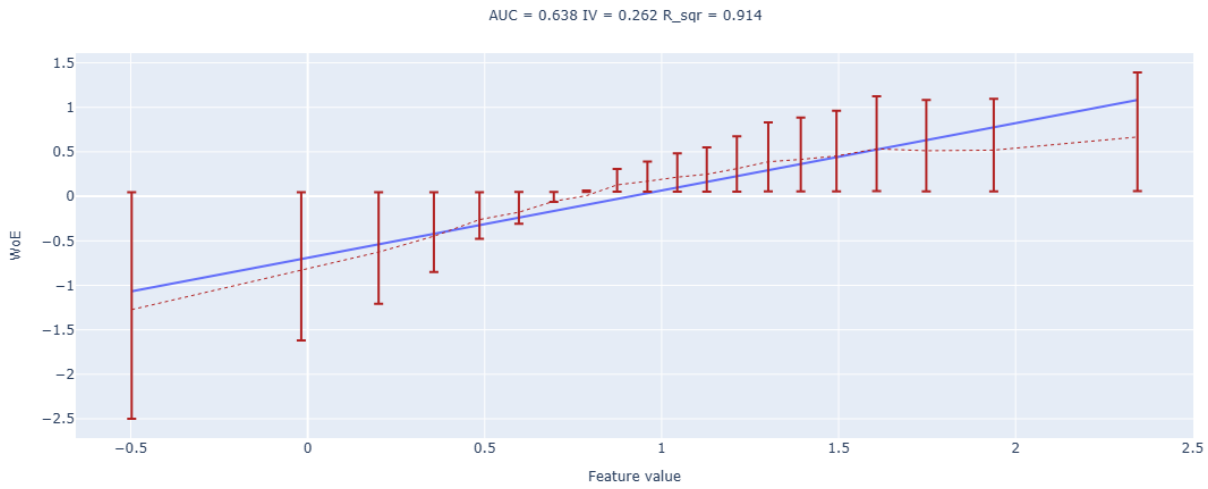
- Постройте графики линейности по WoE получившегося логрега на трейне и тестовой выборке (оценивайте линейность для `logit_logreg`, а не для вероятности!)
- Постройте график стабильности `roc_auc` во времени для получившегося логрега

WOE

```
m = train["logit_logreg"].notna() & train[TARGET].notna()

fig_tr = woe_line(
    values=train.loc[m, "logit_logreg"].to_numpy(),
    target=train.loc[m, TARGET].astype(int).to_numpy(),
    n_buckets=20
)
fig_tr.show()
```




```
m = test["logit_logreg"].notna() & test[TARGET].notna()

fig_te = woe_line(
    values=test.loc[m, "logit_logreg"].to_numpy(),
    target=test.loc[m, TARGET].astype(int).to_numpy(),
    n_buckets=20
)
fig_te.show()
```

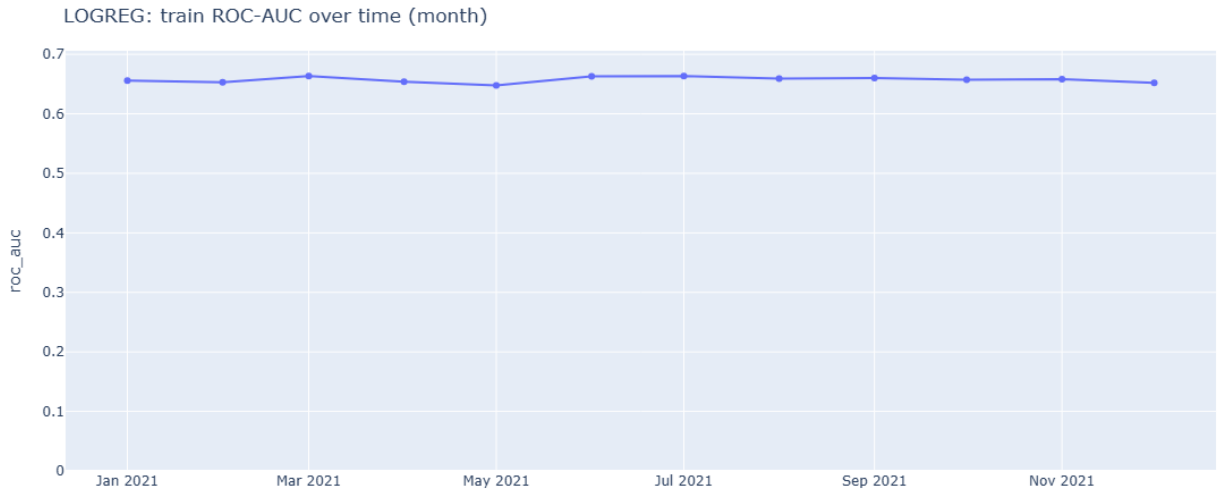


ROC-AUC

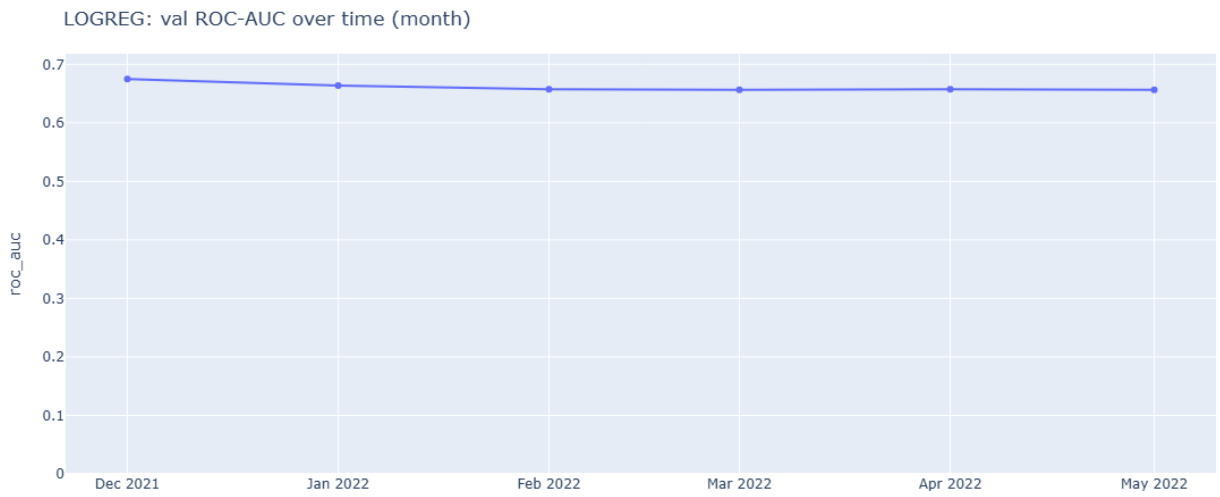
```
features_lr = final_features_logreg

def plot_auc_time(df_part, time_col="month", score_col="logit_logreg",
target_col=TARGET, title=""):
    tmp = df_part[[time_col, score_col, target_col]].dropna()
    auc_by_t = (
        tmp.groupby(time_col, sort=True)
           .apply(lambda g: roc_auc_score(g[target_col].astype(int),
g[score_col]))
    )

    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=auc_by_t.index.astype(str),
        y=auc_by_t.values,
        mode="markers+lines",
        name=f"AUC by {time_col}"
    ))
    fig.update_layout(
        title_text=title or f"ROC-AUC stability over time
({time_col})",
        yaxis=dict(title="roc_auc", rangemode="tozero"),
        width=1000,
        height=450,
        xaxis=dict(showgrid=True),
        margin=dict(l=30, r=30, b=30, t=50),
    )
    fig.show()

plot_auc_time(train, time_col="month", title="LOGREG: train ROC-AUC
over time (month)")
plot_auc_time(val,   time_col="month", title="LOGREG: val ROC-AUC over
time (month)")
```

```
plot_auc_time(test,  time_col="month", title="LOGREG: test ROC-AUC over time (month)")
```

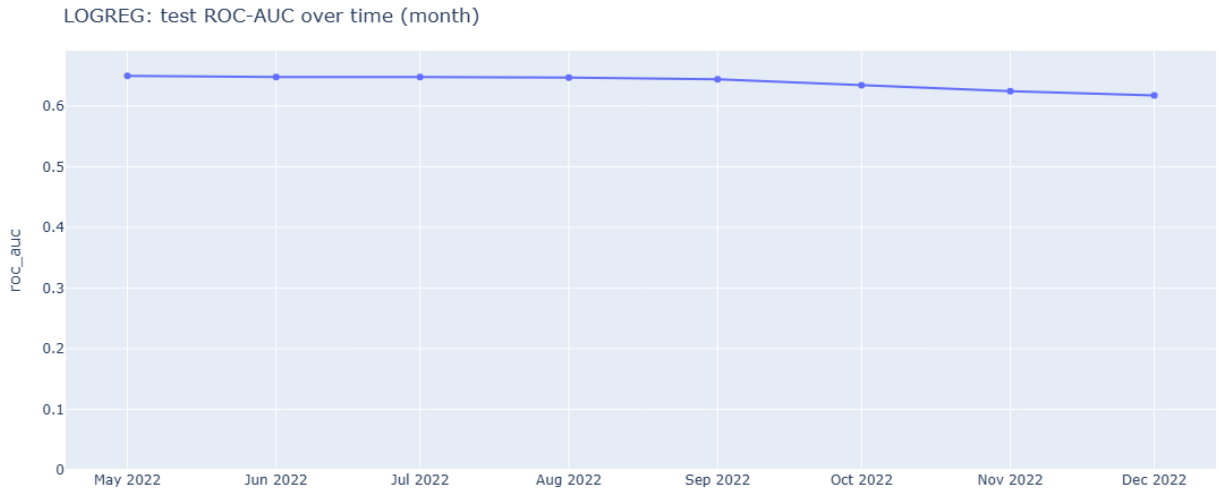
```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2958800306.py:7: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```




```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2958800306.py:7: DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.
```




```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\2958800306.py:7:
DeprecationWarning:

DataFrameGroupBy.apply operated on the grouping columns. This behavior
is deprecated, and in a future version of pandas the grouping columns
will be excluded from the operation. Either pass
`include_groups=False` to exclude the groupings or explicitly select
the grouping columns after groupby to silence this warning.
```



Выводы:

logit_logreg демонстрирует почти линейную и монотонную зависимость по WoE и на train, и на test (R^2 примерно 0.9), а шум в основном в крайних бакетах из-за малого числа наблюдений.

AUC на тесте ниже, чем на трейне (0.638 и 0.657 соответственно), но падение умеренное - модель обобщается нормально.

По времени ROC-AUC на трейне стабилен, а на тесте заметен лёгкий нисходящий тренд к концу периода, что похоже на дрейф данных / их деградацию.

Бонусная часть (до 5 баллов)

В разделе про бустинг мы обучали `LGBMClassifier` на довольно большом наборе фичей. Их количество можно сократить, при этом не теряя в качестве модели.

За начальный набор признаков можно взять все признаки (все 230) или признаки после отбора по IV.

Ваша задача: Отобрать признаки, подобрать оптимальные гиперпараметры и обучить `LGBMClassifier`.

Задание творческое) Можно использовать любые методы отбора признаков / оптимизации гиперпараметров.

Чем меньше признаков, без ухудшения качества модели – тем лучше.

Идеи для отбора признаков:

- Воспользоваться методами из модуля `sklearn.feature_selection` (точно можно попробовать RFE). Документация https://scikit-learn.org/stable/modules/feature_selection.html
- С помощью `BorutaPy` из библиотеки `boruta` (https://towardsdatascience.com/simple-example-using-boruta-feature-selection-in-python-8b96925d5d7a – статья может помочь разобраться с запуском алгоритма (мб не откроется без vpn))

```
# your answere here
```

Попробую использовать Permutation Importance и SHAP вместе!

SHAP

Используем для начала интерпретацию для простой модели; после - кластеризуем (иерархическая кластеризация) по корреляции получившийся признаковый набор!

```
import warnings
from typing import Optional, Union, Tuple

import matplotlib.pyplot as plt

import shap

def calculate_shap_feature_importance_lgbm(
    X_train: pd.DataFrame,
    y_train: Union[pd.Series, np.ndarray],
    X_shap: Optional[pd.DataFrame] = None,
```

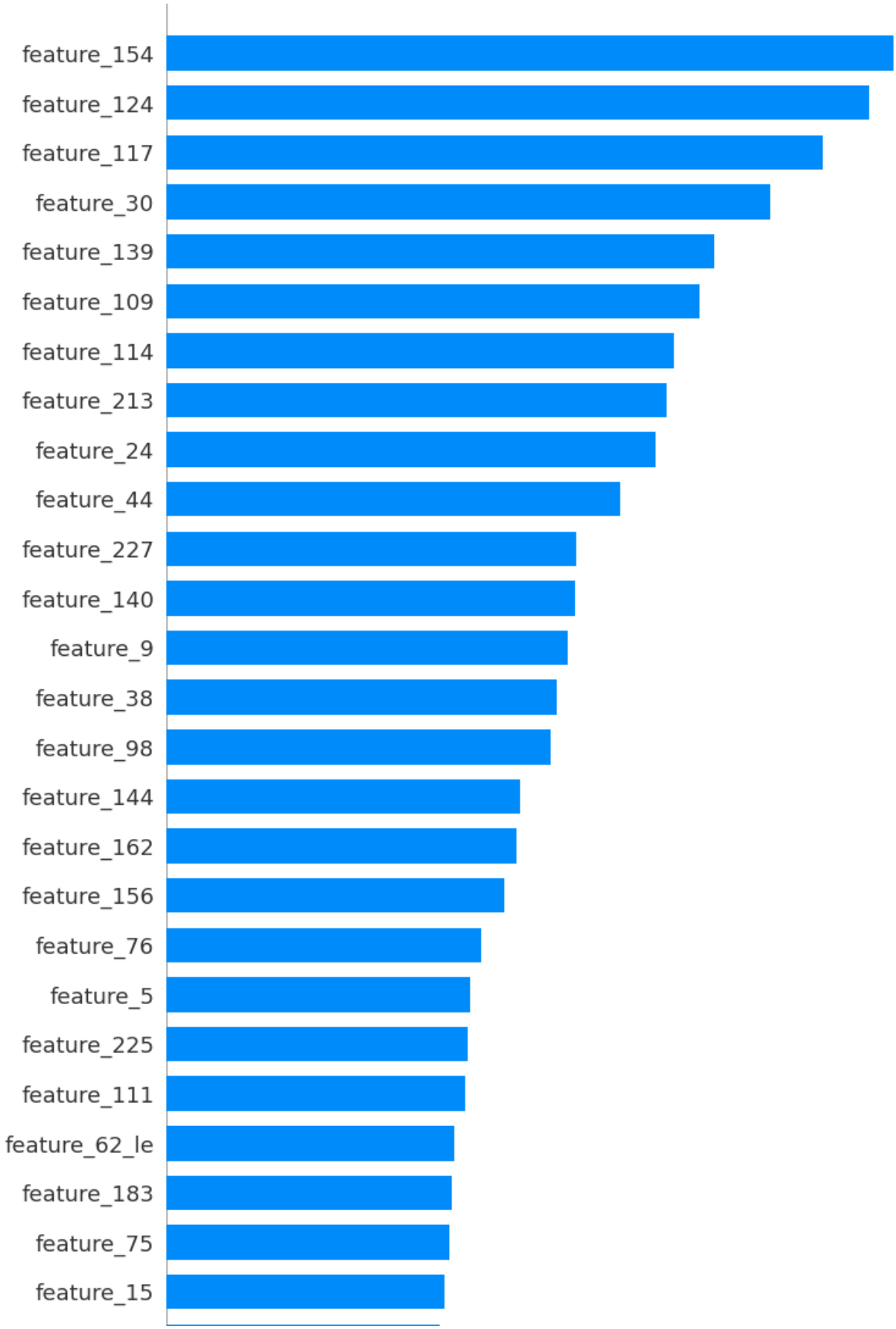
```
    y_shap: Optional[Union[pd.Series, np.ndarray]] = None,
    model_params: Optional[dict] = None,
    max_display: int = 30,
    plot: bool = True,
    sample_size: int = 1000,
    target_name: Optional[str] = None,
    plot_type: str = "bar",
    sampling_strategy: str = "last",
    shap_mode: str = "interventional",  # тут "interventional" or
"tree_path_dependent"
    random_state: int = 42,
) -> Tuple[pd.DataFrame, np.ndarray]:
    """
    Fit LGBMClassifier on X_train/y_train and compute SHAP values on
X_shap.

    Notes for LightGBM + SHAP:
    - shap_mode controls feature_perturbation in TreeExplainer.
      "interventional" is more robust with correlated features but
requires background data.
    - For binary classification, TreeExplainer may return:
        * array of shape (n_samples, n_features)  (often for
model_output="raw")
        * or list of two arrays [class0, class1]; we take class1.
    """

    assert shap_mode in ["tree_path_dependent", "interventional"], \
        "shap_mode must be 'tree_path_dependent' or 'interventional'"

    assert isinstance(X_train, pd.DataFrame), "X_train must be a
pandas DataFrame"
    assert isinstance(y_train, (pd.Series, np.ndarray)), "y_train must
be Series or ndarray"
    assert X_train.shape[0] == len(y_train), "X_train and y_train must
have the same number of samples"

    if model_params is None:
        model_params = {
            "n_estimators": 600,
            "learning_rate": 0.05,
            "num_leaves": 63,
            "max_depth": -1,
            "subsample": 0.8,
            "subsample_freq": 1,
            "colsample_bytree": 0.8,
            "reg_alpha": 0.0,
            "reg_lambda": 0.0,
            "random_state": random_state,
            "n_jobs": -1,
        }
```

```
    if X_shap is None:
        X_shap = X_train.copy()
        y_shap = y_train


    if X_shap.shape[0] > sample_size:
        warnings.warn(
            f"X_shap contains {X_shap.shape[0]} rows. Sampling
{sample_size} rows for SHAP.",
            UserWarning
        )
        if sampling_strategy == "last":
            X_sample = X_shap.iloc[-sample_size:].copy()
            if y_shap is not None:
                if isinstance(y_shap, pd.Series):
                    y_sample = y_shap.iloc[-sample_size:]
                else:
                    y_sample = y_shap[-sample_size:]
            else:
                y_sample = None
        elif sampling_strategy == "random":
            X_sample = X_shap.sample(n=sample_size,
random_state=random_state)
            if y_shap is not None:
                if isinstance(y_shap, pd.Series):
                    y_sample = y_shap.loc[X_sample.index]
                else:
                    y_sample = y_shap[X_sample.index.to_numpy()]
            else:
                y_sample = None
        else:
            raise ValueError(f"Unknown sampling_strategy:
{sampling_strategy}")
    else:
        X_sample = X_shap.copy()
        y_sample = y_shap

    model = LGBMClassifier(**model_params)
    model.fit(X_train, y_train)

    # For interventional SHAP you MUST provide background data.
    # Using X_train is correct, but can be heavy; if needed you can
pass a subsample later.
    if shap_mode == "interventional":
        explainer = shap.TreeExplainer(
            model,
            data=X_train,
            feature_perturbation="interventional",
            model_output="raw"
```

```
        )
    else:
        explainer = shap.TreeExplainer(
            model,
            feature_perturbation="tree_path_dependent",
            model_output="raw"
        )

    shap_values = explainer.shap_values(X_sample)

    if isinstance(shap_values, list):
        if len(shap_values) > 1:
            shap_values = shap_values[1]
        else:
            shap_values = shap_values[0]

    shap_values = np.asarray(shap_values)

    assert shap_values.shape[1] == X_sample.shape[1], \
        "Mismatch between SHAP values and feature dimensions."

    if plot:
        title = f"SHAP summary plot{' for ' + target_name if
target_name else ''}"
        print(title)

        if plot_type == "beeswarm":
            shap.summary_plot(
                shap_values,
                X_sample,
                max_display=max_display,
                plot_type="dot",
                show=False
            )
        else:
            shap.summary_plot(
                shap_values,
                X_sample,
                max_display=max_display,
                plot_type="bar",
                show=False
            )
        plt.show()

    mean_abs_shap = np.abs(shap_values).mean(axis=0)
    shap_df = (
        pd.DataFrame({"feature": X_sample.columns, "mean_abs_shap":
mean_abs_shap})
          .sort_values("mean_abs_shap", ascending=False)
          .reset_index(drop=True)
```

```
    )

    return shap_df, shap_values

shap_df, shap_values = calculate_shap_feature_importance_lgbm(
    X_train=train[features_final],
    y_train=train["target"],
    X_shap=val[features_final],
    y_shap=val["target"],
    max_display=50,
    target_name="target",
    sampling_strategy="last",
    shap_mode="interventional",
    plot_type="bar",
)
```

```
C:\Users\smirn\AppData\Local\Temp\ipykernel_15040\3361923532.py:66:
UserWarning:

X_shap contains 100000 rows. Sampling 1000 rows for SHAP.

100%|====================| 996/1000 [01:06<00:00]

SHAP summary plot for target
```


feature_154
feature_124
feature_117
feature_30
feature_139
feature_109
feature_114
feature_213
feature_24
feature_44
feature_227
feature_140
feature_9
feature_38
feature_98
feature_144
feature_162
feature_156
feature_76
feature_5
feature_225
feature_111
feature_62_le
feature_183
feature_75
feature_15

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

def filter_features_by_shap_and_correlation(
    shap_df: pd.DataFrame,
    X: pd.DataFrame,
    threshold: float = 0.9,
    corr_threshold: float = 0.9,
    min_features: int = 1,
) -> List[str]:

    required_cols = {"feature", "mean_abs_shap"}
    if not required_cols.issubset(shap_df.columns):
        raise ValueError(f"shap_df must contain columns
{required_cols}")

    shap_tbl = (
        shap_df[["feature", "mean_abs_shap"]]
        .dropna(subset=["feature", "mean_abs_shap"])
        .copy()
    )
    shap_tbl = shap_tbl[shap_tbl["feature"].isin(X.columns)]
    if shap_tbl.empty:
        return []

    shap_tbl = shap_tbl.sort_values("mean_abs_shap",
ascending=False).reset_index(drop=True)

    total = float(shap_tbl["mean_abs_shap"].sum())
    if total <= 0:
        return shap_tbl["feature"].head(max(min_features, 1)).tolist()

    shap_tbl["cumulative"] = shap_tbl["mean_abs_shap"].cumsum() /
total

    idx = np.where(shap_tbl["cumulative"].values <= threshold)[0]
    if idx.size == 0:
        k = max(min_features, 1)
    else:
        k = max(int(idx.max() + 2), min_features)
    k = min(k, shap_tbl.shape[0])

    selected_features = shap_tbl["feature"].iloc[:k].tolist()

    X_selected = X[selected_features].copy()
    X_selected = X_selected.apply(pd.to_numeric, errors="coerce")
    X_selected = X_selected.replace([np.inf, -np.inf], np.nan)

    nun = X_selected.nunique(dropna=True)
```

```
    X_selected = X_selected.loc[:, nun[nun > 1].index]
    selected_features = X_selected.columns.tolist()

    if len(selected_features) <= 1:
        return selected_features

    corr = X_selected.corr(method="pearson",
min_periods=2).abs().fillna(0.0)
    corr = corr.clip(lower=0.0, upper=1.0)

    dist = 1.0 - corr
    dist = (dist + dist.T) / 2.0
    np.fill_diagonal(dist.values, 0.0)
    dist = dist.clip(lower=0.0, upper=1.0)

    condensed = squareform(dist.values, checks=False)
    linkage_matrix = linkage(condensed, method="average")
    clusters = fcluster(linkage_matrix, t=1.0 - corr_threshold,
criterion="distance")

    cluster_df = pd.DataFrame({"feature": corr.index.tolist(),
"cluster": clusters})

    shap_map = shap_tbl.set_index("feature")["mean_abs_shap"]

    retained = []
    for cluster_id, grp in cluster_df.groupby("cluster"):
        feats = grp["feature"].tolist()
        best_feat = max(feats, key=lambda f: float(shap_map.get(f, -
np.inf)))
        retained.append(best_feat)

    retained = sorted(retained, key=lambda f: float(shap_map.get(f, -
np.inf)), reverse=True)
    return retained

selected_features_lgbm = filter_features_by_shap_and_correlation(
    shap_df=shap_df,
    X=val[features_final],
    threshold=0.5,
    corr_threshold=0.9999
)
print(len(selected_features_lgbm))
print(selected_features_lgbm[:50])
```

```
19
['feature_154', 'feature_124', 'feature_117', 'feature_30',
'feature_139', 'feature_109', 'feature_114', 'feature_213',
'feature_24', 'feature_44', 'feature_227', 'feature_140', 'feature_9',
```

```
'feature_38', 'feature_98', 'feature_144', 'feature_162',
'feature_156', 'feature_76']
```

Применим, гиперпараметры оставим прежние, так как инкремент малый будет, а перебор с ноутбука, который считает локально, очень долгий...

```
params = study.best_params

clf = LGBMClassifier(**params)

clf.fit(
    X=train[selected_features_lgbm],
    y=train[TARGET]
)

preds = clf.predict_proba(val[selected_features_lgbm])[:, 1]
auc_valid = roc_auc_score(
    y_true=val[TARGET],
    y_score=preds
)

clf = LGBMClassifier(**study.best_params)

clf.fit(
    X=train[selected_features_lgbm],
    y=train[TARGET]
)

LGBMClassifier(bagging_fraction=0.6072608207402718, bagging_freq=10,
               feature_fraction=0.6349898354943739,
               lambda_l1=0.0002798699047466125,
lambda_l2=7.151765760948411e-08,
               learning_rate=0.022511004517451885, max_depth=10,
               n_estimators=2485, num_leaves=96)

preds_test = clf.predict_proba(test[selected_features_lgbm])[:, 1]
auc_test = roc_auc_score(
    y_true=test[TARGET],
    y_score=preds_test
)

assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725 и выше,
ваше значение: {auc_test}!"

# 0.7395093065500715 - старый auc_test
# новый:
print(auc_test)

0.6375542380009138
```

А если не кластеризовать, а просто взять топ?

```
shap_df
```

	feature	mean_abs_shap
0	feature_154	0.124
1	feature_124	0.120
2	feature_117	0.112
3	feature_30	0.103
4	feature_139	0.094
5	feature_109	0.091
6	feature_114	0.087
7	feature_213	0.085
8	feature_24	0.084
9	feature_44	0.078
10	feature_227	0.070
11	feature_140	0.070
12	feature_9	0.069
13	feature_38	0.067
14	feature_98	0.066
15	feature_144	0.060
16	feature_162	0.060
17	feature_156	0.058
18	feature_76	0.054
19	feature_5	0.052
20	feature_225	0.052
21	feature_111	0.051
22	feature_62_le	0.049
23	feature_183	0.049
24	feature_75	0.048
25	feature_15	0.048
26	feature_164	0.047
27	feature_46	0.046
28	feature_206	0.045
29	feature_43	0.045
30	feature_138	0.045
31	feature_12	0.043
32	feature_2	0.041
33	feature_209_le	0.040
34	feature_79	0.039
35	feature_31	0.038
36	feature_35	0.037
37	feature_56	0.037
38	feature_165	0.035
39	feature_97	0.035
40	feature_223	0.031
41	feature_167	0.031
42	feature_135	0.029
43	feature_168	0.029
44	feature_142	0.029

```
45     feature_100           0.028
46   feature_10_le           0.027
47      feature_96           0.027
48     feature_147           0.027
49      feature_27           0.026
50      feature_23           0.026
51     feature_170           0.025
52      feature_36           0.025
53     feature_155           0.025
54      feature_70           0.024
55     feature_115           0.024
56     feature_216           0.024
57      feature_32           0.020
58      feature_65           0.020
59       feature_4           0.018
60     feature_172           0.014
61      feature_94           0.013
62      feature_39           0.010
63  feature_161_le           0.010
64     feature_218           0.004
```

```python
top35_features = (
    shap_df.sort_values("mean_abs_shap", ascending=False)
           .head(35)["feature"]
           .tolist()
)

print("Top-35:", len(top35_features))
print(top35_features)
```

```
Top-35: 35
['feature_154', 'feature_124', 'feature_117', 'feature_30',
'feature_139', 'feature_109', 'feature_114', 'feature_213',
'feature_24', 'feature_44', 'feature_227', 'feature_140', 'feature_9',
'feature_38', 'feature_98', 'feature_144', 'feature_162',
'feature_156', 'feature_76', 'feature_5', 'feature_225',
'feature_111', 'feature_62_le', 'feature_183', 'feature_75',
'feature_15', 'feature_164', 'feature_46', 'feature_206',
'feature_43', 'feature_138', 'feature_12', 'feature_2',
'feature_209_le', 'feature_79']
```

```python
params = study.best_params

clf = LGBMClassifier(**params)

clf.fit(
    X=train[top35_features],
    y=train[TARGET]
)
```

```
preds = clf.predict_proba(val[top35_features])[:, 1]
auc_valid = roc_auc_score(
    y_true=val[TARGET],
    y_score=preds
)

clf = LGBMClassifier(**study.best_params)

clf.fit(
    X=train[top35_features],
    y=train[TARGET]
)
```

```
LGBMClassifier(bagging_fraction=0.6072608207402718, bagging_freq=10,
               feature_fraction=0.6349898354943739,
               lambda_l1=0.0002798699047466125,
lambda_l2=7.151765760948411e-08,
               learning_rate=0.022511004517451885, max_depth=10,
               n_estimators=2485, num_leaves=96)
```

```
preds_test = clf.predict_proba(test[top35_features])[:, 1]
auc_test = roc_auc_score(
    y_true=test[TARGET],
    y_score=preds_test
)

assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725 и выше,
ваше значение: {auc_test}!"
```

```
---------------------------------------------------------------------
-----
AssertionError                            Traceback (most recent call
last)
Cell In[791], line 7
      1 preds_test = clf.predict_proba(test[top35_features])[:, 1]
      2 auc_test = roc_auc_score(
      3     y_true=test[TARGET],
      4     y_score=preds_test
      5 )
----> 7 assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725
и выше, ваше значение: {auc_test}!"

AssertionError: Необходимое значение ROC-AUC 0.725 и выше, ваше
значение: 0.6933898341129493!
```

А если попробуем Permutation Importance?

```
def permutation_importance(
    X_train, y_train, X_test, y_test,
    n_repeats=10, threshold=0.01, metric='accuracy',
    n_top_features=None, plot=True
```

```
):
    """
    Реализация алгоритма важности на основе перестановок (permutation
importance)
    с внешними train/test выборками.
    """

    if metric not in ['accuracy', 'roc_auc']:
        raise ValueError("Метрика должна быть 'accuracy' или
'roc_auc'")


    model = DecisionTreeClassifier(max_depth=5, random_state=42)

    model.fit(X_train, y_train)

    # Предсказания на тестовой выборке
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)[:, 1] if metric ==
'roc_auc' else None

    # Вычисляем базовую метрику
    if metric == 'accuracy':
        baseline_score = accuracy_score(y_test, y_pred)
    else:
        baseline_score = roc_auc_score(y_test, y_pred_proba)

    print(f"Базовая {metric} модели: {baseline_score:.4f}")

    feature_importance = {}
    feature_scores = {}

    for feature in X_test.columns:
        importance_scores = []
        permuted_scores = []

        for _ in range(n_repeats):
            X_test_permuted = X_test.copy()
            X_test_permuted[feature] =
np.random.permutation(X_test_permuted[feature].values)

            y_pred_permuted = model.predict(X_test_permuted)

            if metric == 'accuracy':
                permuted_score = accuracy_score(y_test,
y_pred_permuted)
            else:
                y_pred_proba_permuted =
model.predict_proba(X_test_permuted)[:, 1]
                permuted_score = roc_auc_score(y_test,
```

```
y_pred_proba_permuted)

            permuted_scores.append(permuted_score)
            importance_scores.append(baseline_score - permuted_score)

        feature_importance[feature] = np.mean(importance_scores)
        feature_scores[feature] = np.mean(permuted_scores)

    importance_df = pd.DataFrame({
        'feature': list(feature_importance.keys()),
        'importance': list(feature_importance.values()),
        'permuted_score': list(feature_scores.values())
    }).sort_values('importance', ascending=False)

    importance_df['percent_decrease'] = (importance_df['importance'] /
baseline_score) * 100

    def get_justification(row):
        if row['importance'] <= 0:
            return "Признак не влияет на качество модели или его
перемешивание улучшает модель"
        elif row['percent_decrease'] < 1:
            return "Незначительное влияние на модель (менее 1%
снижения метрики)"
        elif row['percent_decrease'] < 5:
            return "Умеренное влияние на модель (1-5% снижения
метрики)"
        elif row['percent_decrease'] < 10:
            return "Существенное влияние на модель (5-10% снижения
метрики)"
        else:
            return "Критически важный признак (более 10% снижения
метрики)"

    importance_df['justification'] =
importance_df.apply(get_justification, axis=1)

    if n_top_features is not None:
        selected_features = importance_df.head(n_top_features)
['feature'].tolist()
        importance_df['selected'] =
importance_df['feature'].isin(selected_features)
    else:
        selected_features = importance_df[importance_df['importance']
> threshold]['feature'].tolist()
        importance_df['selected'] = importance_df['importance'] >
threshold

    print("\nВажность признаков (permutation importance):")
    print(importance_df)
```

```
    print(f"\nВыбрано {len(selected_features)} признаков:")
    for feature in selected_features:
        row = importance_df[importance_df['feature'] ==
feature].iloc[0]
        print(f"- {feature}: важность = {row['importance']:.4f}, "
              f"снижение метрики = {row['percent_decrease']:.2f}%, "
              f"обоснование: {row['justification']}")

    if plot and not importance_df.empty:
        plt.figure(figsize=(12, 8))
        colors = ['green' if selected else 'gray' for selected in
importance_df['selected']]
        sorted_df = importance_df.sort_values('importance')
        plt.barh(sorted_df['feature'], sorted_df['importance'],
color=colors)
        if n_top_features is None:
            plt.axvline(x=threshold, color='red', linestyle='--',
label=f'Пороговое значение ({threshold})')
            plt.legend()
        plt.xlabel('Важность (снижение метрики)')
        plt.ylabel('Признаки')
        plt.title('Permutation Importance признаков')
        plt.grid(axis='x', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show("permutation_importance.png")
        print("График важности признаков сохранен как
'permutation_importance.png'")

    return importance_df, selected_features

from sklearn.tree import DecisionTreeClassifier

importance_df, selected_features_pi = permutation_importance(
    X_train=train[features_final],
    y_train=train[TARGET],
    X_test=val[features_final],
    y_test=val[TARGET],
    n_repeats=10,
    threshold=0.001,
    metric="roc_auc",
    n_top_features=None,
    plot=True
)

print("Selected by PI:", len(selected_features_pi))
print(selected_features_pi)
```

```
Базовая roc_auc модели: 0.6058
```

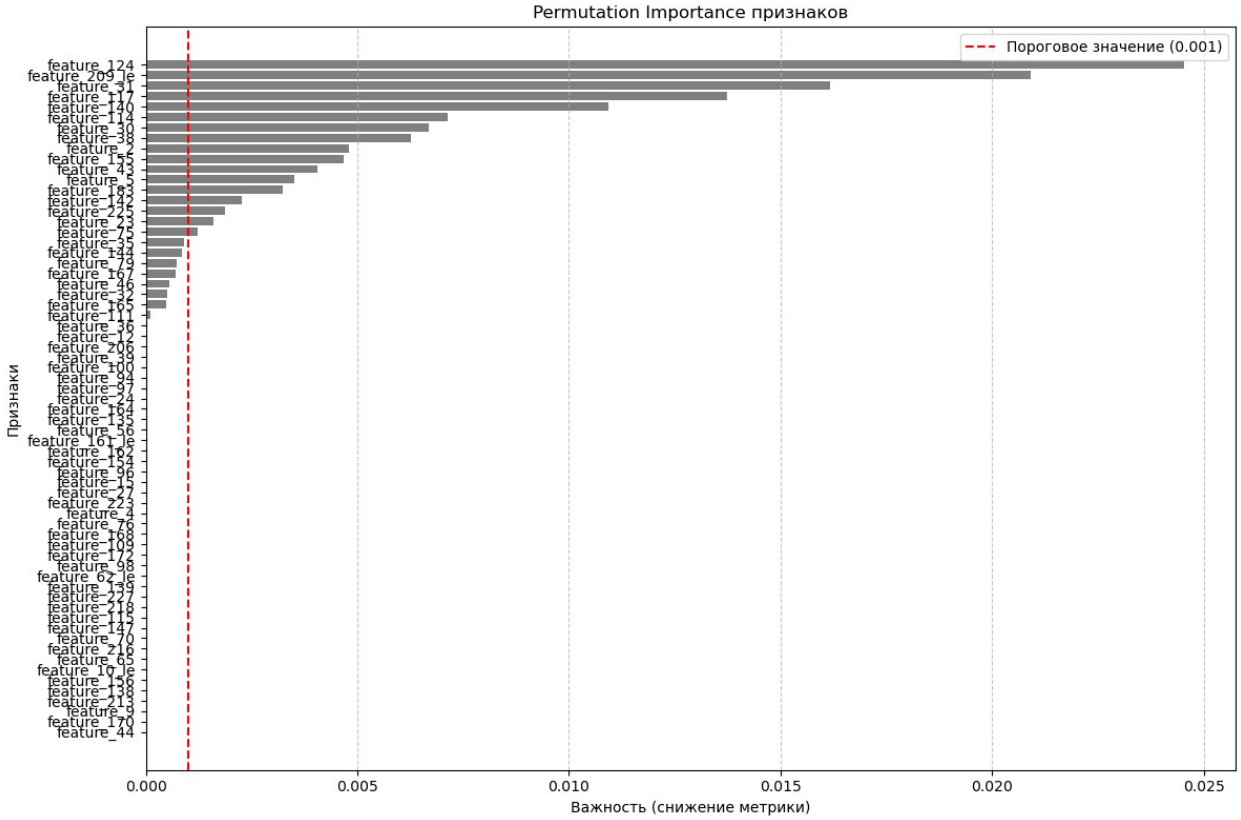
```
Важность признаков (permutation importance):
           feature  importance  permuted_score  percent_decrease  \
11     feature_124       0.025           0.581             4.049
0   feature_209_le       0.021           0.585             3.454
2       feature_31       0.016           0.590             2.669
12     feature_117       0.014           0.592             2.269
38     feature_140       0.011           0.595             1.804
9      feature_114       0.007           0.599             1.178
13      feature_30       0.007           0.599             1.105
21      feature_38       0.006           0.600             1.035
16       feature_2       0.005           0.601             0.792
61     feature_155       0.005           0.601             0.773
31      feature_43       0.004           0.602             0.668
64       feature_5       0.004           0.602             0.580
43     feature_183       0.003           0.603             0.533
57     feature_142       0.002           0.604             0.373
29     feature_225       0.002           0.604             0.309
60      feature_23       0.002           0.604             0.263
35      feature_75       0.001           0.605             0.200
19      feature_35       0.001           0.605             0.149
1      feature_144       0.001           0.605             0.141
25      feature_79       0.001           0.605             0.120
26     feature_167       0.001           0.605             0.114
22      feature_46       0.001           0.605             0.092
54      feature_32       0.000           0.605             0.082
5      feature_165       0.000           0.605             0.078
33     feature_111       0.000           0.606             0.017
50      feature_12       0.000           0.606             0.000
49     feature_206       0.000           0.606             0.000
48      feature_39       0.000           0.606             0.000
47     feature_100       0.000           0.606             0.000
46      feature_94       0.000           0.606             0.000
51      feature_97       0.000           0.606             0.000
45      feature_36       0.000           0.606             0.000
10      feature_44       0.000           0.606             0.000
52     feature_135       0.000           0.606             0.000
53     feature_170       0.000           0.606             0.000
55       feature_9       0.000           0.606             0.000
56     feature_213       0.000           0.606             0.000
58     feature_138       0.000           0.606             0.000
59     feature_156       0.000           0.606             0.000
3    feature_10_le       0.000           0.606             0.000
62       feature_65       0.000           0.606             0.000
63     feature_216       0.000           0.606             0.000
44      feature_70       0.000           0.606             0.000
4      feature_147       0.000           0.606             0.000
42     feature_115       0.000           0.606             0.000
24     feature_218       0.000           0.606             0.000
14     feature_227       0.000           0.606             0.000
```

15	feature_139	0.000	0.606	0.000
7	feature_62_le	0.000	0.606	0.000
17	feature_98	0.000	0.606	0.000
18	feature_172	0.000	0.606	0.000
20	feature_109	0.000	0.606	0.000
6	feature_168	0.000	0.606	0.000
23	feature_76	0.000	0.606	0.000
27	feature_4	0.000	0.606	0.000
41	feature_223	0.000	0.606	0.000
28	feature_27	0.000	0.606	0.000
30	feature_15	0.000	0.606	0.000
34	feature_96	0.000	0.606	0.000
36	feature_154	0.000	0.606	0.000
37	feature_162	0.000	0.606	0.000
8	feature_161_le	0.000	0.606	0.000
39	feature_56	0.000	0.606	0.000
40	feature_164	0.000	0.606	0.000
32	feature_24	0.000	0.606	0.000

	justification	selected
11	Умеренное влияние на модель (1-5% снижения мет...	True
0	Умеренное влияние на модель (1-5% снижения мет...	True
2	Умеренное влияние на модель (1-5% снижения мет...	True
12	Умеренное влияние на модель (1-5% снижения мет...	True
38	Умеренное влияние на модель (1-5% снижения мет...	True
9	Умеренное влияние на модель (1-5% снижения мет...	True
13	Умеренное влияние на модель (1-5% снижения мет...	True
21	Умеренное влияние на модель (1-5% снижения мет...	True
16	Незначительное влияние на модель (менее 1% сни...	True
61	Незначительное влияние на модель (менее 1% сни...	True
31	Незначительное влияние на модель (менее 1% сни...	True
64	Незначительное влияние на модель (менее 1% сни...	True
43	Незначительное влияние на модель (менее 1% сни...	True
57	Незначительное влияние на модель (менее 1% сни...	True
29	Незначительное влияние на модель (менее 1% сни...	True
60	Незначительное влияние на модель (менее 1% сни...	True
35	Незначительное влияние на модель (менее 1% сни...	True
19	Незначительное влияние на модель (менее 1% сни...	False
1	Незначительное влияние на модель (менее 1% сни...	False
25	Незначительное влияние на модель (менее 1% сни...	False
26	Незначительное влияние на модель (менее 1% сни...	False
22	Незначительное влияние на модель (менее 1% сни...	False
54	Незначительное влияние на модель (менее 1% сни...	False
5	Незначительное влияние на модель (менее 1% сни...	False
33	Незначительное влияние на модель (менее 1% сни...	False
50	Признак не влияет на качество модели или его п...	False
49	Признак не влияет на качество модели или его п...	False
48	Признак не влияет на качество модели или его п...	False
47	Признак не влияет на качество модели или его п...	False

```
46  Признак не влияет на качество модели или его п...     False
51  Признак не влияет на качество модели или его п...     False
45  Признак не влияет на качество модели или его п...     False
10  Признак не влияет на качество модели или его п...     False
52  Признак не влияет на качество модели или его п...     False
53  Признак не влияет на качество модели или его п...     False
55  Признак не влияет на качество модели или его п...     False
56  Признак не влияет на качество модели или его п...     False
58  Признак не влияет на качество модели или его п...     False
59  Признак не влияет на качество модели или его п...     False
3   Признак не влияет на качество модели или его п...     False
62  Признак не влияет на качество модели или его п...     False
63  Признак не влияет на качество модели или его п...     False
44  Признак не влияет на качество модели или его п...     False
4   Признак не влияет на качество модели или его п...     False
42  Признак не влияет на качество модели или его п...     False
24  Признак не влияет на качество модели или его п...     False
14  Признак не влияет на качество модели или его п...     False
15  Признак не влияет на качество модели или его п...     False
7   Признак не влияет на качество модели или его п...     False
17  Признак не влияет на качество модели или его п...     False
18  Признак не влияет на качество модели или его п...     False
20  Признак не влияет на качество модели или его п...     False
6   Признак не влияет на качество модели или его п...     False
23  Признак не влияет на качество модели или его п...     False
27  Признак не влияет на качество модели или его п...     False
41  Признак не влияет на качество модели или его п...     False
28  Признак не влияет на качество модели или его п...     False
30  Признак не влияет на качество модели или его п...     False
34  Признак не влияет на качество модели или его п...     False
36  Признак не влияет на качество модели или его п...     False
37  Признак не влияет на качество модели или его п...     False
8   Признак не влияет на качество модели или его п...     False
39  Признак не влияет на качество модели или его п...     False
40  Признак не влияет на качество модели или его п...     False
32  Признак не влияет на качество модели или его п...     False

Выбрано 17 признаков:
- feature_124: важность = 0.0245, снижение метрики = 4.05%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_209_le: важность = 0.0209, снижение метрики = 3.45%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_31: важность = 0.0162, снижение метрики = 2.67%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_117: важность = 0.0137, снижение метрики = 2.27%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_140: важность = 0.0109, снижение метрики = 1.80%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_114: важность = 0.0071, снижение метрики = 1.18%,
```

обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_30: важность = 0.0067, снижение метрики = 1.11%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_38: важность = 0.0063, снижение метрики = 1.04%,
обоснование: Умеренное влияние на модель (1-5% снижения метрики)
- feature_2: важность = 0.0048, снижение метрики = 0.79%, обоснование:
Незначительное влияние на модель (менее 1% снижения метрики)
- feature_155: важность = 0.0047, снижение метрики = 0.77%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_43: важность = 0.0040, снижение метрики = 0.67%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_5: важность = 0.0035, снижение метрики = 0.58%, обоснование:
Незначительное влияние на модель (менее 1% снижения метрики)
- feature_183: важность = 0.0032, снижение метрики = 0.53%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_142: важность = 0.0023, снижение метрики = 0.37%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_225: важность = 0.0019, снижение метрики = 0.31%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_23: важность = 0.0016, снижение метрики = 0.26%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)
- feature_75: важность = 0.0012, снижение метрики = 0.20%,
обоснование: Незначительное влияние на модель (менее 1% снижения
метрики)




```
График важности признаков сохранен как 'permutation_importance.png'
Selected by PI: 17
['feature_124', 'feature_209_le', 'feature_31', 'feature_117',
'feature_140', 'feature_114', 'feature_30', 'feature_38', 'feature_2',
'feature_155', 'feature_43', 'feature_5', 'feature_183',
'feature_142', 'feature_225', 'feature_23', 'feature_75']

params = study.best_params

clf = LGBMClassifier(**params)

clf.fit(
    X=train[selected_features_pi],
    y=train[TARGET]
)

preds = clf.predict_proba(val[selected_features_pi])[:, 1]
auc_valid = roc_auc_score(
    y_true=val[TARGET],
    y_score=preds
)

clf = LGBMClassifier(**study.best_params)

clf.fit(
```

```
    X=train[selected_features_pi],
    y=train[TARGET]
)

preds_test = clf.predict_proba(test[selected_features_pi])[:, 1]
auc_test = roc_auc_score(
    y_true=test[TARGET],
    y_score=preds_test
)

assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725 и выше,
ваше значение: {auc_test}!"
```

```
---------------------------------------------------------------------------
-----
AssertionError                            Traceback (most recent call
last)
Cell In[804], line 29
     23 preds_test = clf.predict_proba(test[selected_features_pi])[:,
1]
     24 auc_test = roc_auc_score(
     25     y_true=test[TARGET],
     26     y_score=preds_test
     27 )
---> 29 assert auc_test > 0.725, f"Необходимое значение ROC-AUC 0.725
и выше, ваше значение: {auc_test}!"

AssertionError: Необходимое значение ROC-AUC 0.725 и выше, ваше
значение: 0.6382026210301834!
```

Выводы

Выбранный признаковый набор уже является достаточно релевантным для задачи. Можно попробовать применить алгоритм на полном наборе данных, но, из-за особенностей обработки, пайплайн необходимо полностью повторить и применить дерево решений на такой большой выборке будет достаточно вычислительно тяжело. Тем не менее, лично для себя я сделал выводы, что отбор по IV является гораздо более понятным для оценки алгоритмов, чем условный SHAP и Permutation. Второй не всегда себя показывает удачно в том или ином контексте и, скорее, является попыткой "прикинуть", чем реально оценить устойчивость признаков во времени. SHAP, в свою очередь, является скорее псевдо-подходом в контексте отбора признаков (хотя, в задачах оценки временных рядов при наличии огромного числа линейно зависимых рядов кластеризация методом иерархической кластеризации показывает себя достаточно интересно и уместно, но, тем не менее, вручную подбирать по порогам признаки крайне тяжело) из-за использования иного базового алгоритма для оценки, нежели по итогу будет в реальной задаче (бустинг с другими параметрами, чем используется по факту), поэтому, подход, скорее, про итоговую интерпретацию важности признаков в модели.

Посмотреть эти же функции (но исходники) можно в ноутбуке, ссылку на который давал выше (имею в виду, что в свое время уже писал реализации и использую свой код или код, созданный для других учебных целей, если это имеет значение при проверке)