Анализ оттока клиентов оператора связи (учебный проект)

Задача - научиться прогнозировать отток клиентов для оператора связи «Ниединогоразрыва.ком»

Оператор предоставляет два основных типа услуг:

1. Стационарную телефонную связь. Возможно подключение телефонного аппарата к нескольким линиям одновременно.
2. Интернет. Подключение может быть двух типов: через телефонную линию (DSL*,* от англ. *digital subscriber line*, «цифровая абонентская линия») или оптоволоконный кабель (*Fiber optic*).

Также доступны такие услуги:

- Интернет-безопасность: антивирус (*DeviceProtection*) и блокировка небезопасных сайтов (*OnlineSecurity*);
- Выделенная линия технической поддержки (*TechSupport*);
- Облачное хранилище файлов для резервного копирования данных (*OnlineBackup*);
- Стриминговое телевидение (*StreamingTV*) и каталог фильмов (*StreamingMovies*).

За услуги клиенты могут платить каждый месяц или заключить договор на 1–2 года. Доступны различные способы расчёта и возможность получения электронного чека.

Данные состоят из файлов, полученных из разных источников:

- `contract.csv` — информация о договоре;
- `personal.csv` — персональные данные клиента;
- `internet.csv` — информация об интернет-услугах;
- `phone.csv` — информация об услугах телефонии.

Во всех файлах столбец `customerID` содержит код клиента.

Информация о договорах актуальна на 1 февраля 2020.

План работы

Загрузка данных
Проверка типов данных
Составить общий датасет
Обработать пропуски
Проверить дубли
Посмотреть на распределение данных, выбросы и аномалии
Проверка гипотез
Подготовить синтетические признаки: таргет, сумма потрачено денег, продолжительность контрактов, классы клиентов по длительности контрактов, по сумме потраченных денег
Разделить выборку на треин и тест
Проверить мультиколлениарность признаков, масштабировать признаки (для логистической регрессии)
Обработать категориальные признаки
Обучить модели
Подбор гиперпараметров

Выбор лучшей модели
Анализ важности признаков
Подготовка отчета

Загрузка данных

In [8]:

```
!pip install phik
import pandas as pd
import numpy as np
import random
import phik
from phik import resources, report
import matplotlib
import matplotlib.pyplot as plt
import seaborn
import datetime as dt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from catboost import CatBoostClassifier

np.random.seed(seed = 261222)
random.seed = 261222

import warnings
warnings.filterwarnings('ignore')
from catboost import CatBoostClassifier, cv
```

```
Collecting phik
  Downloading phik-0.12.3-cp39-cp39-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl (679 kB)
     |████████████████████████████████| 679 kB 2.0 MB/s eta 0:00:01
Requirement already satisfied: joblib>=0.14.1 in /opt/conda/lib/python
3.9/site-packages (from phik) (1.1.0)
Requirement already satisfied: pandas>=0.25.1 in /opt/conda/lib/python
3.9/site-packages (from phik) (1.2.4)
Requirement already satisfied: matplotlib>=2.2.3 in /opt/conda/lib/pyt
hon3.9/site-packages (from phik) (3.3.4)
Requirement already satisfied: scipy>=1.5.2 in /opt/conda/lib/python3.
9/site-packages (from phik) (1.9.1)
Requirement already satisfied: numpy>=1.18.0 in /opt/conda/lib/python
3.9/site-packages (from phik) (1.21.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/pyt
hon3.9/site-packages (from matplotlib>=2.2.3->phik) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.
9/site-packages (from matplotlib>=2.2.3->phik) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.
```

In [9]:

```
try:
    contract = pd.read_csv('/datasets/final_provider/contract.csv')
    internet = pd.read_csv('/datasets/final_provider/internet.csv')
    personal = pd.read_csv('/datasets/final_provider/personal.csv')
    phone = pd.read_csv('/datasets/final_provider/phone.csv')
    print('Данные загружены')
except:
    print('Ошибка. Неверный путь')
```

```
Данные загружены
```

In [10]:

```
contract.EndDate.value_counts()
```

Out[10]:

```
No                    5174
2019-11-01 00:00:00    485
2019-12-01 00:00:00    466
2020-01-01 00:00:00    460
2019-10-01 00:00:00    458
Name: EndDate, dtype: int64
```

In [11]:

```
for one in [contract,personal,phone,internet]:
    print(one.index)
    print(one.describe())
    print(one.info())
    print(one.head)
    print()

```

```
RangeIndex(start=0, stop=7043, step=1)
       MonthlyCharges
count     7043.000000
mean        64.761692
std         30.090047
min         18.250000
25%         35.500000
50%         70.350000
75%         89.850000
max        118.750000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   BeginDate         7043 non-null   object
 2   EndDate           7043 non-null   object
 3   Type              7043 non-null   object
 4   PaperlessBilling  7043 non-null   object
```

В представленных файлах:

Договоры - 7043 строк

Интернет - 7043

Персональные данные - 6361
Телефония - 5571

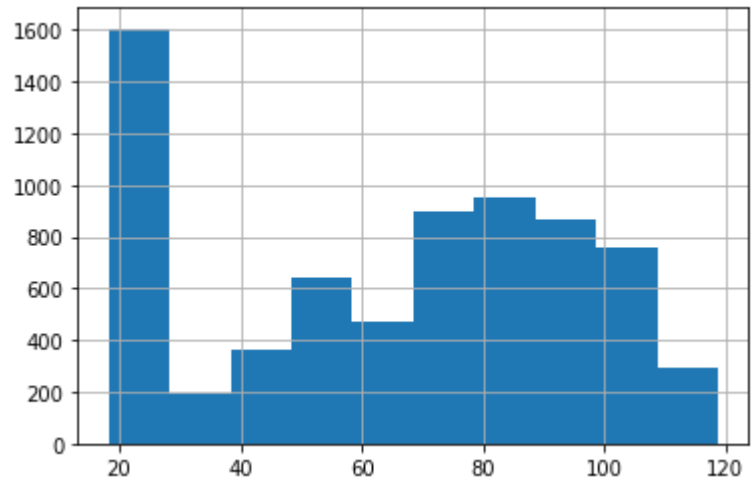
Все customerID встречаются по одному разу.
В данных есть пользоатели, заключившие контракт 2020-02-01, у них TotalCharges ==0

In [12]:

```
1 contract.MonthlyCharges.hist()
```

Out[12]:

```
<AxesSubplot:>
```



In [13]:

```
1 contract.loc[contract.EndDate=='No','EndDate']='2020-02-01 00:00:00' #28 февраля
2 contract[['BeginDate','EndDate']] = contract[['BeginDate','EndDate']].astype('da
3 print(contract.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   customerID        7043 non-null   object        
 1   BeginDate         7043 non-null   datetime64[ns]
 2   EndDate           7043 non-null   datetime64[ns]
 3   Type              7043 non-null   object        
 4   PaperlessBilling  7043 non-null   object        
 5   PaymentMethod     7043 non-null   object        
 6   MonthlyCharges    7043 non-null   float64       
 7   TotalCharges      7043 non-null   object        
dtypes: datetime64[ns](2), float64(1), object(5)
memory usage: 440.3+ KB
None
```

In [14]:

```
contract.loc[contract.TotalCharges==' ','TotalCharges']=0
contract.TotalCharges = contract.TotalCharges.astype('float')
contract.TotalCharges.sort_values().unique()
```

Out[14]:

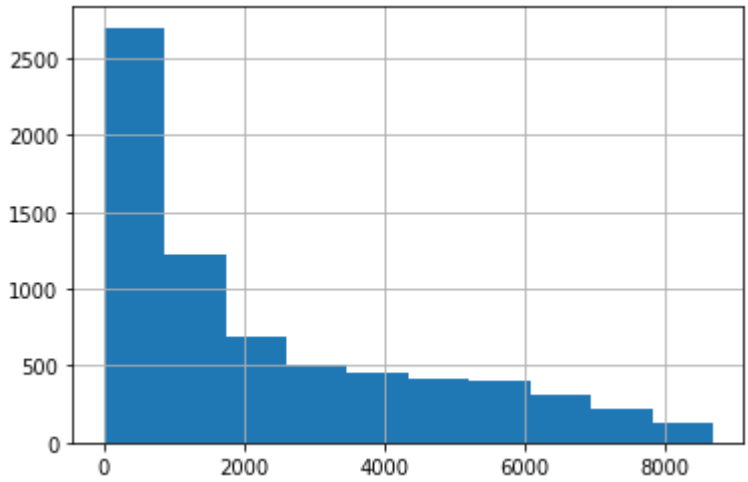
```
array([   0.  ,   18.8 ,   18.85, ..., 8670.1 , 8672.45, 8684.8 ])
```

In [15]:

```
contract.TotalCharges.hist()
```

Out[15]:

```
<AxesSubplot:>
```



In [16]:

```
1 contract.head
```

Out[16]:

```
<bound method NDFrame.head of       customerID  BeginDate    EndDate
Type PaperlessBilling  \
0     7590-VHVEG 2020-01-01 2020-02-01  Month-to-month              Ye
s
1     5575-GNVDE 2017-04-01 2020-02-01        One year               N
o
2     3668-QPYBK 2019-10-01 2019-12-01  Month-to-month              Ye
s
3     7795-CFOCW 2016-05-01 2020-02-01        One year               N
o
4     9237-HQITU 2019-09-01 2019-11-01  Month-to-month              Ye
s
...          ...        ...        ...             ...
...
7038  6840-RESVB 2018-02-01 2020-02-01        One year              Ye
s
7039  2234-XADUH 2014-02-01 2020-02-01        One year              Ye
s
7040  4801-JZAZL 2019-03-01 2020-02-01  Month-to-month              Ye
s
7041  8361-LTMKD 2019-07-01 2019-11-01  Month-to-month              Ye
s
7042  3186-AJIEK 2014-08-01 2020-02-01        Two year              Ye
s

                 PaymentMethod  MonthlyCharges  TotalCharges
0             Electronic check           29.85         29.85
1                 Mailed check           56.95       1889.50
2                 Mailed check           53.85        108.15
3     Bank transfer (automatic)           42.30       1840.75
4             Electronic check           70.70        151.65
...                        ...             ...           ...
7038              Mailed check           84.80       1990.50
7039    Credit card (automatic)          103.20       7362.90
7040          Electronic check           29.60        346.45
7041              Mailed check           74.40        306.60
7042  Bank transfer (automatic)          105.65       6844.50

[7043 rows x 8 columns]>
```

Вопросы

1. Есть ли учет пользователей, которые возвращаются в компанию? Если пользователь возвращается, ему присваивается новый customerID? Может ли пользователь делать перерывы в оплате? (пользоваться услугами не каждый месяц, уехать в отпуск и не платить)
2. Ведется ли учет обращений пользователей в техподдержку? возможно, уход связан с техническими проблемами, это тоже можно исследовать? Можно было бы посмотреть на зависимость кол-ва обращений, дальнейшего ухода и типа подключенных услуг
3. По каким признакам принимается решение, что пользователь перестал пользоваться услугами? Как долго он должен для этого не платить или же должен разорвать контракт?

4. Есть ли данные об изменении MonthlyCharges (повышение тарифов и их связь с уходом пользователей?)
5. Есть ли данные об отключении/подключении пользователем доп услуг в процессе действия договора?

Ответы

1) хороший вопрос. Такой информацией не обладаем, поэтому пока не отвечу. Но по опыту обычно у таких клиентов новый ID.

2) Это расширение данных, идея хорошая, но сначала нужно сделать прототип на том, что есть, а дальше уже работать по сбору новых доп.данных.

3) В данной компании ушел = разорвал контракт.

4) Пока нет. Тарифы выгруженные актуальны на дату среза. Тут такая же ситуация как с п.2 - вопросы и идеи отличные и правильные, но для прототипа нереализуемые.

5) Таких данных тоже пока нет.

Общий датасет

In [17]:

```
def ind(one):
    one.index = one.customerID
    one = one.drop('customerID', axis = 1)
    return one
```

In [18]:

```
contract = ind(contract)
internet = ind(internet)
personal = ind(personal)
phone = ind(phone)
```

In [19]:

```
data = pd.concat([contract,internet,personal,phone], axis = 1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7043 entries, 7590-VHVEG to 3186-AJIEK
Data columns (total 19 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   BeginDate         7043 non-null   datetime64[ns]
 1   EndDate           7043 non-null   datetime64[ns]
 2   Type              7043 non-null   object        
 3   PaperlessBilling  7043 non-null   object        
 4   PaymentMethod     7043 non-null   object        
 5   MonthlyCharges    7043 non-null   float64       
 6   TotalCharges      7043 non-null   float64       
 7   InternetService   5517 non-null   object        
 8   OnlineSecurity    5517 non-null   object        
 9   OnlineBackup      5517 non-null   object        
 10  DeviceProtection  5517 non-null   object        
 11  TechSupport       5517 non-null   object        
 12  StreamingTV       5517 non-null   object        
 13  StreamingMovies   5517 non-null   object        
```

Обработка пропусков и дублей

In [20]:

```
data.isna().sum()
```

Out[20]:

```
BeginDate              0
EndDate                0
Type                   0
PaperlessBilling       0
PaymentMethod          0
MonthlyCharges         0
TotalCharges           0
InternetService     1526
OnlineSecurity      1526
OnlineBackup        1526
DeviceProtection    1526
TechSupport         1526
StreamingTV         1526
StreamingMovies     1526
gender                 0
SeniorCitizen          0
Partner                0
Dependents             0
MultipleLines        682
dtype: int64
```

Пропуски во всех парамах заполню как 'No'. Для InternetService это станет третим категориальным значением

In [21]:

```
data = data.fillna('No')
data.isna().sum()

```

Out[21]:

```
BeginDate           0
EndDate             0
Type                0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
MultipleLines       0
dtype: int64
```

In [22]:

```
1 data[data.duplicated()].sort_values(by='customerID')
```

Out[22]:

customerID	BeginDate	EndDate	Type	PaperlessBilling	PaymentMethod	MonthlyCharges	Tot
0970-QXPXW	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	19.65	
1000-AJSLD	2019-10-01	2019-11-01	Month-to-month	Yes	Mailed check	20.10	
2636-ALXXZ	2019-12-01	2020-01-01	Month-to-month	Yes	Electronic check	69.60	
2668-TZSPS	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.45	
2676-ISHSF	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.30	
3247-MHJKM	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.20	
5996-DAOQL	2020-01-01	2020-02-01	Month-to-month	Yes	Mailed check	20.45	
6457-GIRWB	2019-10-01	2019-11-01	Month-to-month	Yes	Electronic check	69.35	
7096-UCLNH	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.05	
7878-RTCZG	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	19.90	
8048-DSDFQ	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.20	
8605-ITULD	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	19.55	
8749-CLJXC	2020-01-01	2020-02-01	Month-to-month	No	Mailed check	20.05	

Дубликатов нет. Найденные дубли имеют разные customerID, значит это, скорее всего пользователи, заключившие одинаковые договоры в один день

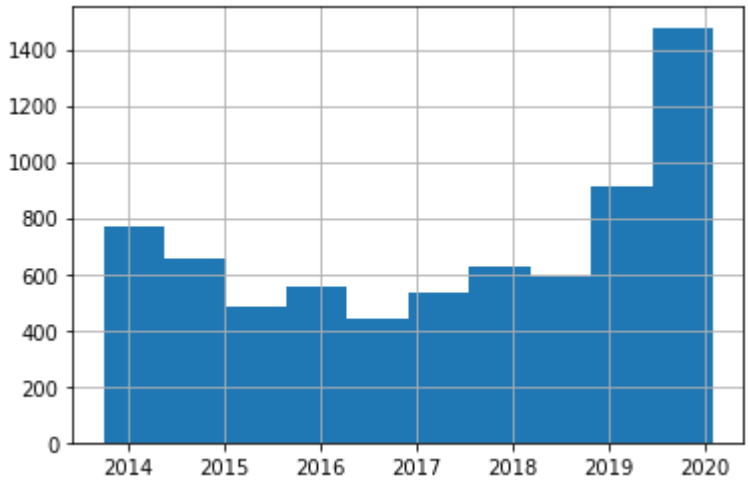
Посмотреть на распределение данных, выбросы и аномалии

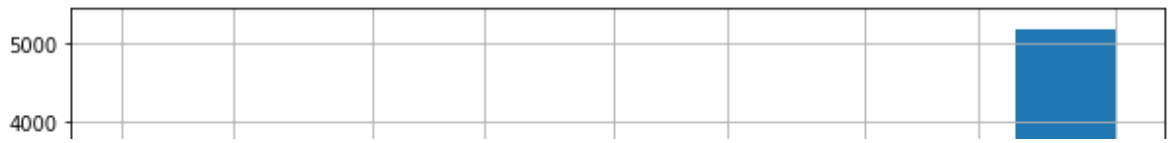
In [23]:

```
data.reset_index()
data.index = data.BeginDate

```

In [24]:

```
data['BeginDate'].hist()
plt.show()
data['EndDate'].hist(figsize=(10,4))
plt.show()
data['MonthlyCharges'].hist()
plt.show()
data['TotalCharges'].hist()
plt.show()

```



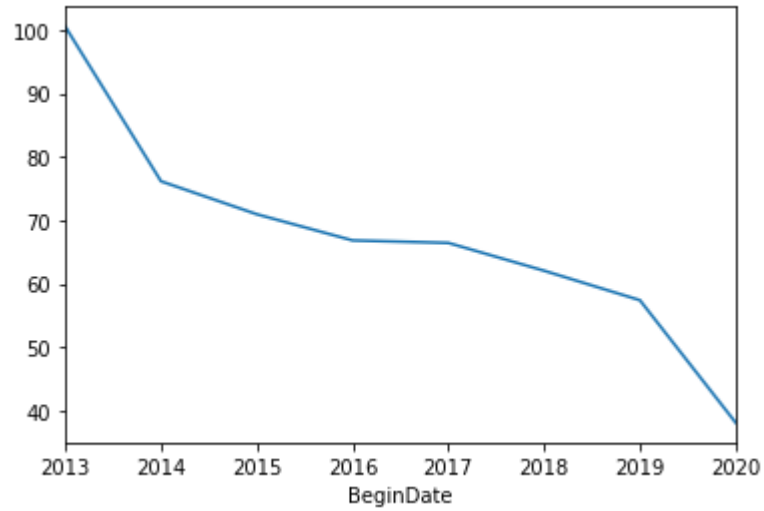


In [25]:

```
data_year = data.resample('1Y').mean()
data_year['MonthlyCharges'].plot()
```

Out[25]:

```
<AxesSubplot:xlabel='BeginDate'>
```



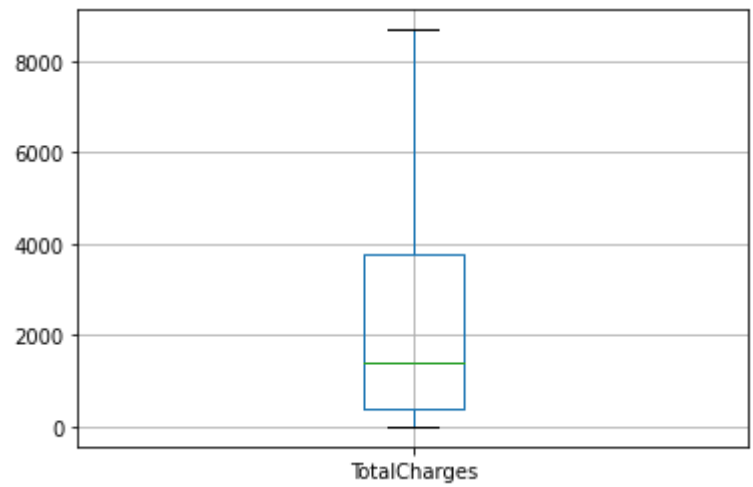
Месячная плата постепенно снижается

In [26]:

```
data[['TotalCharges']].boxplot()
```

Out[26]:

```
<AxesSubplot:>
```

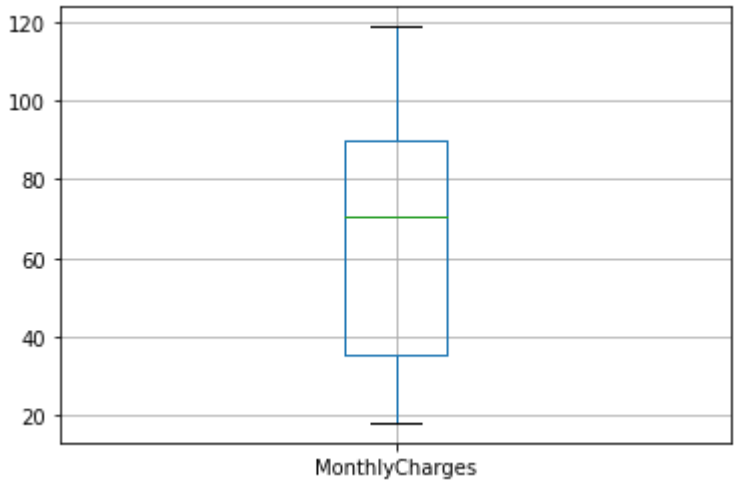


In [27]:

```
1  data[['MonthlyCharges']].boxplot()
```

Out[27]:

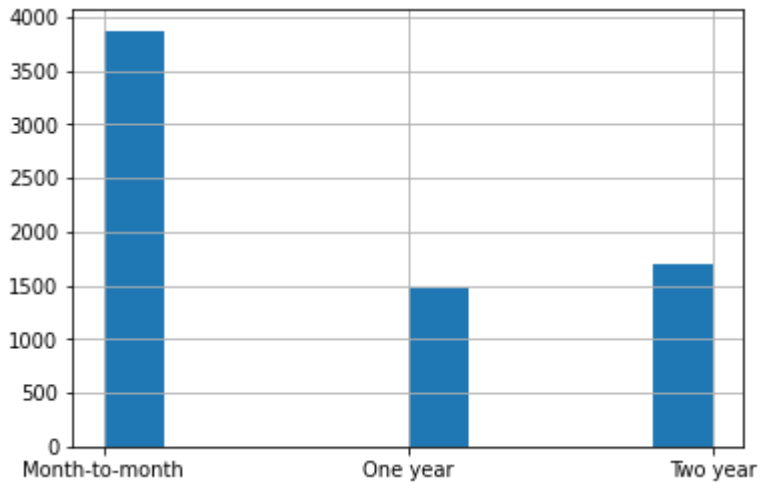
<AxesSubplot:>



Выбросов в месячной и тотал оплате нет

In [28]:

```
1  for one in ['Type','PaperlessBilling','PaymentMethod','InternetService',
2  'OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport',
3  'StreamingTV','StreamingMovies','gender','SeniorCitizen','Partner',
4  'Dependents','MultipleLines']:
5          print(
6              one,
7              data[one].hist(),
8              plt.show()
9          )
```




```
Type AxesSubplot(0.125,0.125;0.775x0.755) None
```

Больше половины пользователей оплачивают услуги помесячно
Практически половина пользатель - мужчины и половина - женщины. Остальные данные распределены не равномеро

Гипотезы и синтетические признаки

Исследовать MonthlyCharges <30 - вероятно самая дешевая месячная оплата больше всего пользователей, мб это отдельная группа пользователей
Type - помесячная, 1 год, 2 года - исследовать взаимосвязь PaperlessBilling и PaymentMethod
Contract.PaperlessBilling - Yes/no (безнал) - исследовать взаимосвязь PaperlessBilling и PaymentMethod
Исследовать зависимость разных категорий клиентов и длительности контрактов - пенсионеры, в браке, с иждевенцами, с мулььтителефонией
Есть ли разница в тотал расходах оптического интернета и dsl
Есть ли зависимость короткого срока контракта и отсутствия OnlineSecurity и других характеристик?
Много ли контрактов после 2 лет, которые не продлевают?
Может ли быть такое, что контратк длится 2 года, а пользователь перестал платить после 1 месяца?
Есть ли ошибки в данных, когда дурация больше 1 месяца, а тотал и месячная оплата одинаковые
Изучить, еслить ли пики ухода и узнать, было ли повышение тарифов в это время? Изучить вероятность банковских ошибок как причины ухода? (зависимость типа платежа от кол-ва ушедших пользователей)
Влияет ли MultipleLines на TotalCharges?

In [29]:

```
data['Monthly'] = data['MonthlyCharges']//10 #признак категории месячной оплаты
```

In [30]:

```
data[['Type','PaymentMethod','BeginDate']].groupby(['Type','PaymentMethod']).cou
```

Out[30]:

		BeginDate
Type	**PaymentMethod**	
Month-to-month	**Bank transfer (automatic)**	589
	Credit card (automatic)	543
	Electronic check	1850
	Mailed check	893
One year	**Bank transfer (automatic)**	391
	Credit card (automatic)	398
	Electronic check	347
	Mailed check	337
Two year	**Bank transfer (automatic)**	564
	Credit card (automatic)	581
	Electronic check	168
	Mailed check	382

Электронных чеков больше в помесячной оплате и почти нет в оплате на 2 года

In [31]:

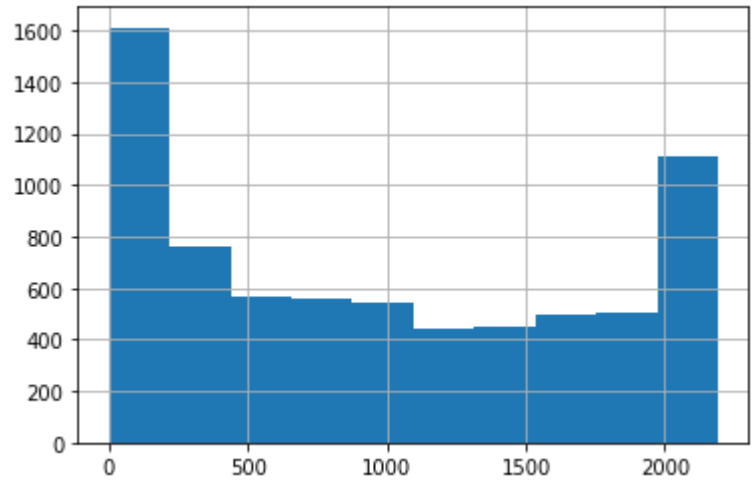
```
data['duration'] = (data['EndDate'] - data['BeginDate']).dt.days #количество дне
```

In [32]:

```
data['duration'].hist()
```

Out[32]:

<AxesSubplot:>



In [33]:

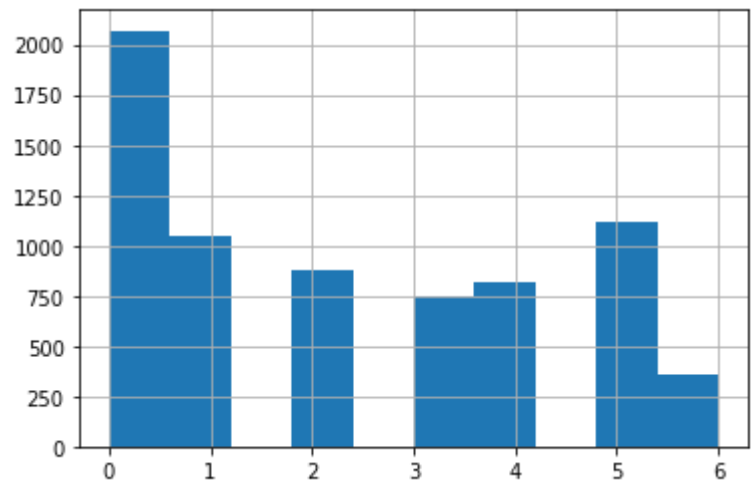
```
data['duration_year'] = (data['EndDate'] - data['BeginDate']).dt.days//365 #коли
```

In [34]:

```
data['duration_year'].hist()
```

Out[34]:

<AxesSubplot:>



почти половина договоров живут меньше года

In [35]:

```
data['duration_month'] = (data['EndDate'].dt.month - data['BeginDate'].dt.month)
12*(data['EndDate'].dt.year - data['BeginDate'].dt.year) #количество месяцев исп
```

In [36]:

```
data[['Type','PaymentMethod','BeginDate']].groupby(['Type','PaymentMethod']).cou
```

Out[36]:

		BeginDate
Type	**PaymentMethod**	
Month-to-month	**Bank transfer (automatic)**	589
	Credit card (automatic)	543
	Electronic check	1850
	Mailed check	893
One year	**Bank transfer (automatic)**	391
	Credit card (automatic)	398
	Electronic check	347
	Mailed check	337
Two year	**Bank transfer (automatic)**	564
	Credit card (automatic)	581
	Electronic check	168
	Mailed check	382

In [37]:

```
for one in ['Type','PaperlessBilling','PaymentMethod','InternetService',
'OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport',
'StreamingTV','StreamingMovies','gender','SeniorCitizen','Partner',
'Dependents','MultipleLines']:

    print(data[[one, 'duration_month','BeginDate']].groupby(one).mean())
```

```
                duration_month
Type
Month-to-month       18.036645
One year             42.044807
Two year             56.735103
                   duration_month
PaperlessBilling
No                    32.189067
Yes                   32.496524
                          duration_month
PaymentMethod
Bank transfer (automatic)       43.656736
Credit card (automatic)         43.269382
Electronic check                25.174630
Mailed check                    21.830025
                  duration_month
InternetService
DSL                   32.821561
Fiber optic           32.917959
No                    30.547182
                 duration_month
OnlineSecurity
No                   27.277269
Yes                  45.046558
              duration_month
OnlineBackup
No                 25.951669
Yes                44.565253
                  duration_month
DeviceProtection
No                    25.959100
Yes                   44.604872
             duration_month
TechSupport
No                27.279856
Yes               44.822896
             duration_month
StreamingTV
No                26.942804
Yes               41.066125
                 duration_month
StreamingMovies
No                    26.777778
Yes                   41.197291
        duration_month
gender
Female       32.244553
Male         32.495359
               duration_month
SeniorCitizen
0                  32.192171
1                  33.295972
         duration_month
Partner
No            23.357869
Yes           42.017637
            duration_month
Dependents
No               29.806000
Yes              38.368246
               duration_month
```

```
MultipleLines
No                  25.408153
Yes                 41.914507
```

Длительность контрактов с помесячной оплатой меньше.
Длительность контрактов с электронным чеком или чеком по почте меньше.
Длительность контрактов без бэкапов, онлайн-защиты, защиты устройств, техподдержки, стриминга видео, телефонной линии меньше.
Длительность контакров не зависит от гендера и пенсионного статуса статуса.
Длительность контарктов без партнеров и иждевенцев ниже.

In [38]:

```
for one in ['Type','PaperlessBilling','PaymentMethod','InternetService',
'OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport',
'StreamingTV','StreamingMovies','gender','SeniorCitizen','Partner',
'Dependents','MultipleLines']:

    print(data[[one, 'TotalCharges','BeginDate']].groupby(one).mean())
```

```
                TotalCharges
Type
Month-to-month   1369.254581
One year         3032.622878
Two year         3706.934336
                  TotalCharges
PaperlessBilling
No                1846.580449
Yes               2577.988408
                           TotalCharges
PaymentMethod
Bank transfer (automatic)   3075.310816
Credit card (automatic)     3069.378022
Electronic check            2090.868182
Mailed check                1049.250744
                  TotalCharges
InternetService
DSL               2115.411338
Fiber optic       3205.304570
No                 662.604784
                 TotalCharges
OnlineSecurity
No                1688.236236
Yes               3751.594775
               TotalCharges
OnlineBackup
No              1442.268942
Yes             3870.539234
                   TotalCharges
DeviceProtection
No                 1423.176542
Yes                3913.984269
              TotalCharges
TechSupport
No             1653.775045
Yes            3810.639555
              TotalCharges
StreamingTV
No             1357.448201
Yes            3757.027447
                  TotalCharges
StreamingMovies
No                 1341.226931
Yes                3760.665959
        TotalCharges
gender
Female   2279.918062
Male     2279.554008
               TotalCharges
SeniorCitizen
0               2177.023801
1               2810.465193
         TotalCharges
Partner
No        1584.089810
Yes       3024.249765
            TotalCharges
Dependents
No            2187.709254
Yes           2494.881019
              TotalCharges
```

```
MultipleLines
No              1372.662046
Yes             3522.951481
```

TotalCharges ниже при:
Type Month-to-month
PaperlessBilling: No
Mailed check
InternetService: No
OnlineSecurity: No
Onlinebackup: No
DeviceProtection: No
TechSupport: No
StreamingTV: No
SeniorCitizen: 0
Partner:No
Dependents: Nov
MultipleLines: No

Пол не влияет на TotalCharges

In [39]:

```
for one in ['OnlineSecurity','OnlineBackup','DeviceProtection','TechSupport','St
    data.loc[data[one]=='No', one]=0
    data.loc[data[one]=='Yes', one]=1
data['services'] =  data[['OnlineSecurity','OnlineBackup','DeviceProtection','Te
```

In [40]:

```
data['services'] #количество подключенных услуг
```

Out[40]:

```
BeginDate
2020-01-01    1.0
2017-04-01    2.0
2019-10-01    2.0
2016-05-01    3.0
2019-09-01    0.0
             ...
2018-02-01    5.0
2014-02-01    4.0
2019-03-01    1.0
2019-07-01    1.0
2014-08-01    4.0
Name: services, Length: 7043, dtype: float64
```

In [41]:

```
#количество продлений контракта +1
data = data.drop('BeginDate', axis=1)
data = data.reset_index()
```

In [42]:

```
data.loc[data['Type']=='Month-to-month','continiue']=data['duration_month']
data.loc[data['Type']=='One year','continiue']=data['duration_month']/12
data.loc[data['Type']=='Two year','continiue']=data['duration_month']/24
data['continiue'] = data['continiue'].apply(np.ceil)
```

In [43]:

```
data[['Type','continiue']].groupby('Type').mean()

```

Out[43]:

Type	continiue
Month-to-month	18.036645
One year	3.952478
Two year	2.646018

In [44]:

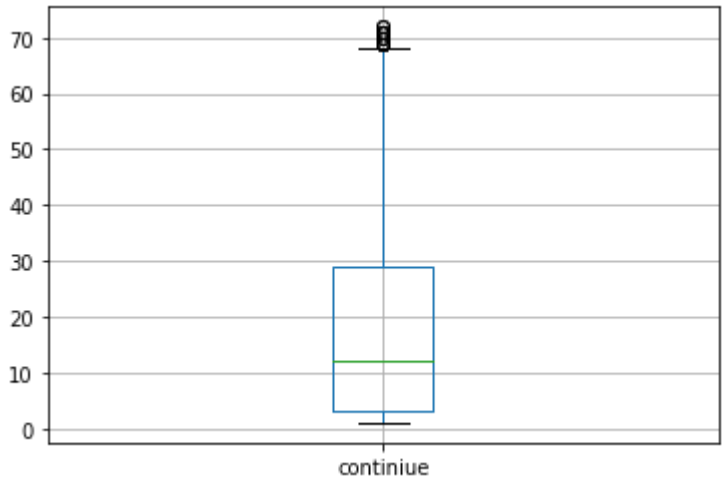
```
data[['Type','continiue']].groupby('Type').median()
```

Out[44]:

Type	continiue
Month-to-month	12.0
One year	4.0
Two year	3.0

In [45]:

```
data[data['Type']=='Month-to-month'][['continiue']].boxplot()
plt.show()
data[data['Type']=='One year'][['continiue']].boxplot()
plt.show()
data[data['Type']=='Two year'][['continiue']].boxplot()
plt.show()
```



In [46]:

```
data[data['Type']=='Two year']['continiue'].value_counts()
```

Out[46]:

```
3.0    1263
2.0     274
1.0     148
0.0      10
Name: continiue, dtype: int64
```

Контракты, заключенные на 2 года, чаще всего продлевают 1 раз

In [47]:

```
data[(data.MonthlyCharges == data.TotalCharges)&(data.continiue>1)] #проверяем к
```

Out[47]:

	BeginDate	EndDate	Type	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	In

0 rows × 25 columns

In [48]:

```
1 data.EndDate.value_counts()
```

Out[48]:

```
2020-02-01    5174
2019-11-01     485
2019-12-01     466
2020-01-01     460
2019-10-01     458
Name: EndDate, dtype: int64
```

Почему-то всего 4 даты завершения контрактов (их начали собирать с ноября 2019)

Подготовить синтетические признаки: таргет, сумма потраченных денег

In [49]:

```
1 data['target'] = 1 #ушедшие
2 data.loc[data.EndDate=='2020-02-01', 'target'] = 0
```

In [50]:

```
1 data.target.value_counts()
```

Out[50]:

```
0    5174
1    1869
Name: target, dtype: int64
```

In [51]:

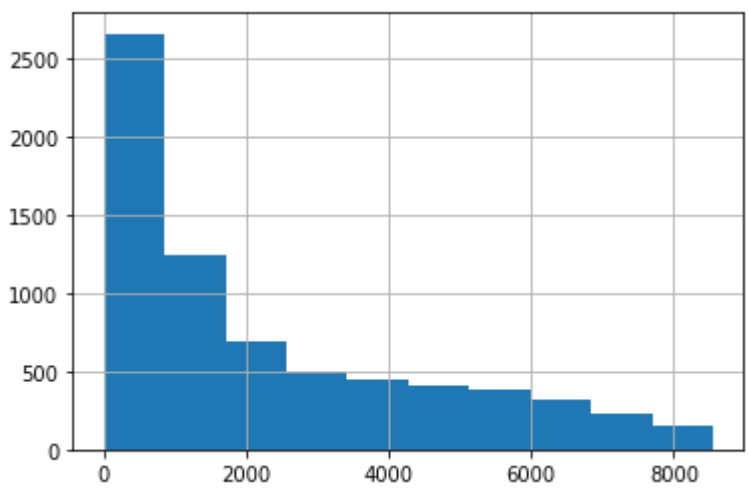
```
1 data['totalsumm'] = data['MonthlyCharges']*(data['duration_month'])
```

In [52]:

```
1 data['totalsumm'].hist()
```

Out[52]:

```
<AxesSubplot:>
```



In [53]:

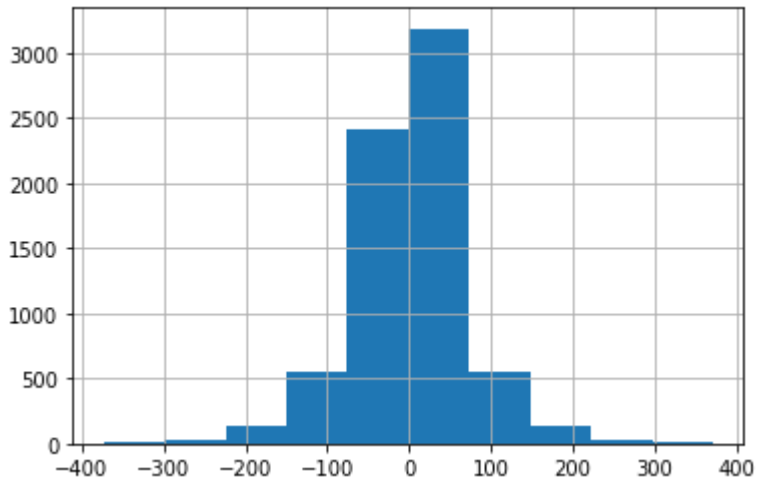
```
data['wallet'] = data['totalsumm']-data['TotalCharges']
```

In [54]:

```
data['wallet'].hist()
#сколько денег останется на счету в конце месяца
```

Out[54]:

<AxesSubplot:>



In [55]:

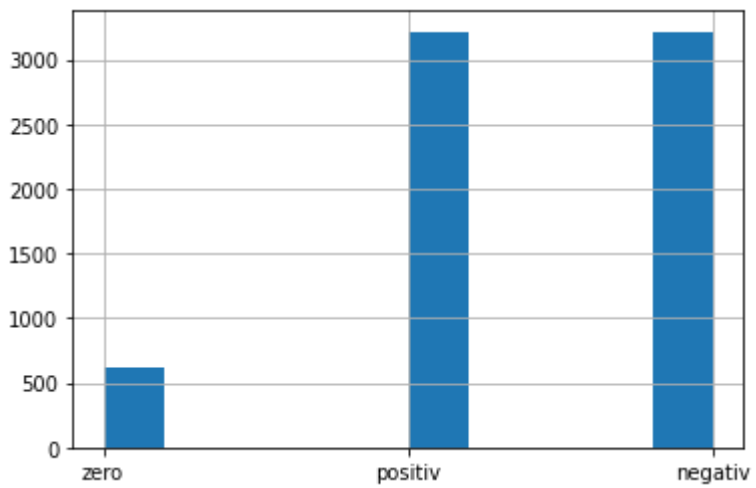
```
data.loc[data.wallet<0,'wallet_category'] = 'negativ'
data.loc[data.wallet>0,'wallet_category'] = 'positiv'
data.loc[data.wallet==0,'wallet_category'] = 'zero'
data = data.drop(['TotalCharges','wallet'],axis =1) #какой баланс на конец месяц
```

In [56]:

```
data.wallet_category.hist()
```

Out[56]:

<AxesSubplot:>



In [57]:

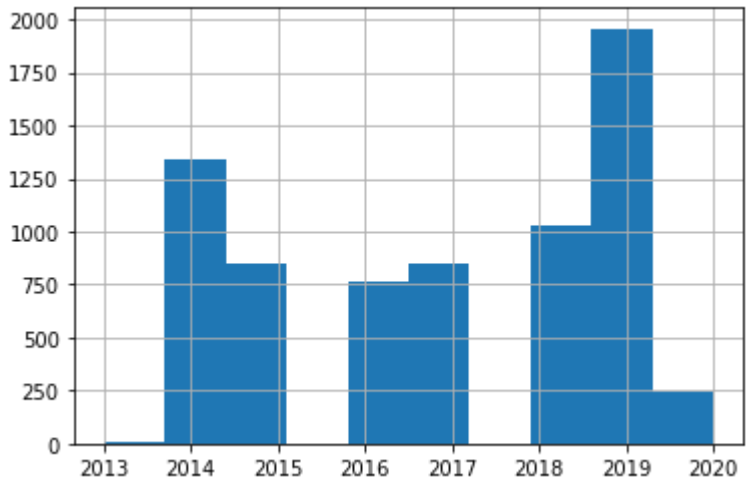
```
data['year'] = data.BeginDate.dt.year
data['month'] = data.BeginDate.dt.month
```

In [58]:

```
data.year.hist()
```

Out[58]:

<AxesSubplot:>

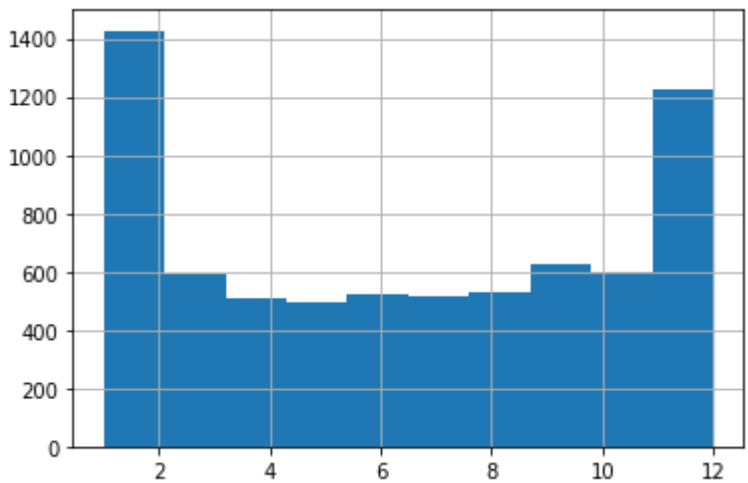


In [59]:

```
data.month.hist()
```

Out[59]:

<AxesSubplot:>



Больше всего контрактов заключено в 2019 году. Чаще всего контракты заключают в январе и в декабре

In [60]:

```
data.index = data.BeginDate
data = data.sort_index()
```

In [61]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 7043 entries, 2013-10-01 to 2020-02-01
Data columns (total 29 columns):
 #   Column            Non-Null Count  Dtype         
---  ------            --------------  -----         
 0   BeginDate         7043 non-null   datetime64[ns]
 1   EndDate           7043 non-null   datetime64[ns]
 2   Type              7043 non-null   object        
 3   PaperlessBilling  7043 non-null   object        
 4   PaymentMethod     7043 non-null   object        
 5   MonthlyCharges    7043 non-null   float64       
 6   InternetService   7043 non-null   object        
 7   OnlineSecurity    7043 non-null   object        
 8   OnlineBackup      7043 non-null   object        
 9   DeviceProtection  7043 non-null   object        
 10  TechSupport       7043 non-null   object        
 11  StreamingTV       7043 non-null   object        
 12  StreamingMovies   7043 non-null   object        
 13  gender            7043 non-null   object        
 14  SeniorCitizen     7043 non-null   int64         
 15  Partner           7043 non-null   object        
 16  Dependents        7043 non-null   object        
 17  MultipleLines     7043 non-null   object        
 18  Monthly           7043 non-null   float64       
 19  duration          7043 non-null   int64         
 20  duration_year     7043 non-null   int64         
 21  duration_month    7043 non-null   int64         
 22  services          7043 non-null   float64       
 23  continiue         7043 non-null   float64       
 24  target            7043 non-null   int64         
 25  totalsumm         7043 non-null   float64       
 26  wallet_category   7043 non-null   object        
 27  year              7043 non-null   int64         
 28  month             7043 non-null   int64         
dtypes: datetime64[ns](2), float64(5), int64(7), object(15)
memory usage: 1.6+ MB
```

Разделить выборки

In [62]:

```
train, test = train_test_split(data, test_size=0.25, random_state = 261222)
```

In [63]:

```
train.loc[train.EndDate>train.index.max(),'EndDate'] = train.index.max()
```

In [64]:

```
train['duration_month'] = (train['EndDate'].dt.month - train['BeginDate'].dt.mon
12*(train['EndDate'].dt.year - train['BeginDate'].dt.year) #количество месяцев м
train['duration'] = (train['EndDate'] - train['BeginDate']).dt.days #количество
train['duration_year'] = (train['EndDate'] - train['BeginDate']).dt.days//365 #к
train['totalsumm'] = train['MonthlyCharges']*(train['duration_month']) #всего по

```

In [65]:

```
train = train.drop(['month','year'],axis=1)
test = test.drop(['month','year'],axis=1)
```

In [66]:

```
train_target = train['target']
train_features = train.reset_index(drop=True)
train_features = train_features.drop(['target','BeginDate'],axis = 1)
test_target = test['target']
test_features = test.reset_index(drop=True)
test_features = test_features.drop(['target','BeginDate'],axis = 1)
train_target.shape, test_target.shape, train_features.shape, test_features.shape
```

Out[66]:

```
((5282,), (1761,), (5282, 25), (1761, 25))
```

In [67]:

```
train = train.drop(['EndDate','BeginDate'],axis = 1)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5282 entries, 2014-06-01 to 2018-04-01
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Type              5282 non-null   object 
 1   PaperlessBilling  5282 non-null   object 
 2   PaymentMethod     5282 non-null   object 
 3   MonthlyCharges    5282 non-null   float64
 4   InternetService   5282 non-null   object 
 5   OnlineSecurity    5282 non-null   object 
 6   OnlineBackup      5282 non-null   object 
 7   DeviceProtection  5282 non-null   object 
 8   TechSupport       5282 non-null   object 
 9   StreamingTV       5282 non-null   object 
 10  StreamingMovies   5282 non-null   object 
 11  gender            5282 non-null   object 
 12  SeniorCitizen     5282 non-null   int64  
 13  Partner           5282 non-null   object 
 14  Dependents        5282 non-null   object 
 15  MultipleLines     5282 non-null   object 
 16  Monthly           5282 non-null   float64
 17  duration          5282 non-null   int64  
 18  duration_year     5282 non-null   int64  
 19  duration_month    5282 non-null   int64  
 20  services          5282 non-null   float64
 21  continiue         5282 non-null   float64
 22  target            5282 non-null   int64  
 23  totalsumm         5282 non-null   float64
 24  wallet_category   5282 non-null   object 
dtypes: float64(5), int64(5), object(15)
memory usage: 1.0+ MB
```

Проверить мультиколлениарность признаков, масштабировать признаки (для логистической регрессии)

In [68]:

```
1 data.corr()
```

Out[68]:

	MonthlyCharges	SeniorCitizen	Monthly	duration	duration_year	duration_month	services
MonthlyCharges	1.000000	0.220173	0.994087	0.247754	0.244850	0.247900	0.730354
SeniorCitizen	0.220173	1.000000	0.219036	0.016514	0.013833	0.016567	0.078300
Monthly	0.994087	0.219036	1.000000	0.244394	0.241438	0.244538	0.724627
duration	0.247754	0.016514	0.244394	1.000000	0.989045	0.999999	0.536886
duration_year	0.244850	0.013833	0.241438	0.989045	1.000000	0.989145	0.532530
duration_month	0.247900	0.016567	0.244538	0.999999	0.989145	1.000000	0.536964
services	0.730354	0.078300	0.724627	0.536886	0.532530	0.536964	1.000000
continiue	0.255670	0.179578	0.253664	0.168319	0.151833	0.168367	0.126462
target	0.193356	0.150889	0.191763	-0.352673	-0.342196	-0.352229	-0.098570
totalsumm	0.651566	0.103261	0.646403	0.826499	0.821914	0.826568	0.775639

In [69]:

```
1 train_lin = train.drop(['target'],axis=1)
2 train_lin = train.drop(['Monthly','target','services','duration_month','duration
```

In [70]:

```
1 train_lin.corr()
```

Out[70]:

	MonthlyCharges	SeniorCitizen	duration	continiue
MonthlyCharges	1.000000	0.227370	0.257047	0.259387
SeniorCitizen	0.227370	1.000000	0.032451	0.182047
duration	0.257047	0.032451	1.000000	0.175392
continiue	0.259387	0.182047	0.175392	1.000000

In [71]:

```
1 train_lin.phik_matrix()
```

```
interval columns not set, guessing: ['MonthlyCharges', 'SeniorCitize
n', 'duration', 'continiue']
```

Out[71]:

	Type	PaperlessBilling	PaymentMethod	MonthlyCharges	InternetService	C
Type	1.000000	0.095966	0.274828	0.381613	0.497042	
PaperlessBilling	0.095966	1.000000	0.374631	0.477321	0.236988	
PaymentMethod	0.274828	0.374631	1.000000	0.403969	0.325018	
MonthlyCharges	0.381613	0.477321	0.403969	1.000000	0.917326	
InternetService	0.497042	0.236988	0.325018	0.917326	1.000000	
OnlineSecurity	0.154625	0.000000	0.276708	0.538608	0.237589	
OnlineBackup	0.099234	0.204977	0.291209	0.629326	0.229983	
DeviceProtection	0.133614	0.169166	0.311081	0.672396	0.232733	
TechSupport	0.175956	0.067637	0.279979	0.578571	0.240397	
StreamingTV	0.070463	0.350069	0.388339	0.836150	0.272662	
StreamingMovies	0.071150	0.327957	0.391509	0.833675	0.274052	
gender	0.000000	0.012755	0.007508	0.000000	0.000000	
SeniorCitizen	0.078251	0.228156	0.281912	0.313582	0.161129	
Partner	0.177408	0.000000	0.224054	0.193202	0.000000	
Dependents	0.147795	0.169951	0.220683	0.182506	0.105384	
MultipleLines	0.067634	0.266215	0.336867	0.677493	0.226850	
duration	0.665219	0.000000	0.375264	0.414489	0.030832	
continiue	0.604586	0.208793	0.223440	0.336336	0.327275	
wallet_category	0.462666	0.000000	0.152025	0.256886	0.114086	

In [72]:

```
1 train_lin = train_lin.drop(['Type','InternetService'],axis=1)
```

In [73]:

```
train_lin = pd.get_dummies(train_lin, drop_first=True)
train_lin
```

Out[73]:

BeginDate	MonthlyCharges	SeniorCitizen	duration	continiue	PaperlessBilling_Yes	PaymentMetl card (a
2014-06-01	53.00	1	2071	3.0	1	
2020-01-01	19.20	0	31	1.0	0	
2014-02-01	109.55	0	2191	3.0	1	
2018-06-01	51.80	0	610	20.0	1	
2017-05-01	88.60	0	1006	33.0	0	
...	...	...	...	...	...	
2016-11-01	100.00	0	1187	4.0	0	
2018-12-01	18.80	0	427	1.0	0	
2014-06-01	73.00	0	2071	6.0	1	
2019-06-01	24.40	0	245	1.0	0	
2018-04-01	56.25	0	671	2.0	1	

5282 rows × 20 columns

In [74]:

```
1 train_lin
```

Out[74]:

BeginDate	MonthlyCharges	SeniorCitizen	duration	continiue	PaperlessBilling_Yes	PaymentMetl card (
2014-06-01	53.00	1	2071	3.0	1	
2020-01-01	19.20	0	31	1.0	0	
2014-02-01	109.55	0	2191	3.0	1	
2018-06-01	51.80	0	610	20.0	1	
2017-05-01	88.60	0	1006	33.0	0	
...	...	...	...	...	...	
2016-11-01	100.00	0	1187	4.0	0	
2018-12-01	18.80	0	427	1.0	0	
2014-06-01	73.00	0	2071	6.0	1	
2019-06-01	24.40	0	245	1.0	0	
2018-04-01	56.25	0	671	2.0	1	

5282 rows × 20 columns

In [75]:

```
train_lin = train_lin.reset_index().drop('BeginDate', axis = 1)
train_lin
```

Out[75]:

	MonthlyCharges	SeniorCitizen	duration	continiue	PaperlessBilling_Yes	PaymentMethod_C card (autom
0	53.00	1	2071	3.0	1	
1	19.20	0	31	1.0	0	
2	109.55	0	2191	3.0	1	
3	51.80	0	610	20.0	1	
4	88.60	0	1006	33.0	0	
...	...	...	...	...	...	
5277	100.00	0	1187	4.0	0	
5278	18.80	0	427	1.0	0	
5279	73.00	0	2071	6.0	1	
5280	24.40	0	245	1.0	0	
5281	56.25	0	671	2.0	1	

5282 rows × 20 columns

In [76]:

```
scaler = StandardScaler()
scaler.fit(train_lin)
train_features_lin_scaled = scaler.transform(train_lin)
```

Обучение дерева решений

In [77]:

```
train_features_tree = pd.get_dummies(train_lin)
```

In [78]:

```
1 train_features_tree
```

Out[78]:

	MonthlyCharges	SeniorCitizen	duration	continiue	PaperlessBilling_Yes	PaymentMethod_C card (autom
0	53.00	1	2071	3.0	1	
1	19.20	0	31	1.0	0	
2	109.55	0	2191	3.0	1	
3	51.80	0	610	20.0	1	
4	88.60	0	1006	33.0	0	
...	...	...	...	...	...	
5277	100.00	0	1187	4.0	0	
5278	18.80	0	427	1.0	0	
5279	73.00	0	2071	6.0	1	
5280	24.40	0	245	1.0	0	
5281	56.25	0	671	2.0	1	

5282 rows × 20 columns

In [79]:

```
1 model = RandomForestClassifier()
2 predict  = cross_val_score(model, train_features_tree, train_target, cv=3, scori
3 print(predict.mean())
4
```

```
0.8405317032972562
```

In [80]:

```
1 parametrs = {'max_depth': range(175,176,1),
2              'min_samples_split': [2,12,0.1],
3               'n_estimators': range(70,80,4),
4               'max_features':['log2'],
5               }
6 grid = GridSearchCV(model, parametrs, cv=3, scoring = 'roc_auc', return_train_sc
7 grid.fit(train_features_tree, train_target)
8 print(grid.best_score_)
9 print(grid.best_estimator_)
```

```
0.8562920991798825
RandomForestClassifier(max_depth=175, max_features='log2', min_samples
_split=12,
                       n_estimators=78)
```

DecisionTreeClassifier

In [81]:

```
model = DecisionTreeClassifier(class_weight='balanced')
predict  = cross_val_score(model, train_features_tree, train_target, cv=3, scori
print(predict.mean())

```

```
0.7096936495769824
```

In [82]:

```
parametrs = {'max_depth': range(10,150,10),
                     'min_samples_split': [0.8,2,12,0.4]
                     }

grid = GridSearchCV(model, parametrs, cv=3, scoring = 'roc_auc', return_train_sc
grid.fit(train_features_tree, train_target)
print(grid.best_score_)
print(grid.best_estimator_)


```

```
0.8063870498042981
DecisionTreeClassifier(class_weight='balanced', max_depth=10,
                       min_samples_split=12)
```

LogisticRegression

In [83]:

```
model = LogisticRegression()

predict  = cross_val_score(model, train_features_lin_scaled, train_target, cv=3,
print(predict.mean())
```

```
0.8368322866771973
```

catboost

In [84]:

```
model = CatBoostClassifier(custom_loss = 'AUC')
parametrs = {
             'iterations': [100],
            'thread_count': range(1,4),
             'depth': range(2,8),
             'learning_rate': [0.3,0.5,0.8]
                    }

grid = GridSearchCV(model, parametrs, cv=3, scoring = 'roc_auc', return_train_sc
grid.fit(train_features_lin_scaled, train_target)

print(grid.best_score_)
print(grid.best_params_)


```

```
0:      learn: 0.5887423        total: 53.1ms   remaining: 5.25s
1:      learn: 0.5190752        total: 54.3ms   remaining: 2.66s
2:      learn: 0.4845298        total: 55.5ms   remaining: 1.79s
3:      learn: 0.4715725        total: 56.6ms   remaining: 1.36s
4:      learn: 0.4606145        total: 57.5ms   remaining: 1.09s
5:      learn: 0.4465877        total: 58.3ms   remaining: 914ms
6:      learn: 0.4416092        total: 59.6ms   remaining: 791ms
7:      learn: 0.4333219        total: 60.6ms   remaining: 697ms
8:      learn: 0.4284804        total: 61.5ms   remaining: 622ms
9:      learn: 0.4260062        total: 62.4ms   remaining: 561ms
10:     learn: 0.4222950        total: 63.2ms   remaining: 511ms
11:     learn: 0.4206211        total: 64.2ms   remaining: 471ms
12:     learn: 0.4177323        total: 65ms     remaining: 435ms
13:     learn: 0.4114478        total: 65.9ms   remaining: 405ms
14:     learn: 0.4033778        total: 66.8ms   remaining: 378ms
15:     learn: 0.4020143        total: 67.6ms   remaining: 355ms
16:     learn: 0.3949917        total: 68.5ms   remaining: 334ms
17:     learn: 0.3937107        total: 69.3ms   remaining: 316ms
18:     learn: 0.3913771        total: 70.2ms   remaining: 299ms
```

In [85]:

```
grid.best_params_,grid.best_score_
```

Out[85]:

```
({'depth': 3, 'iterations': 100, 'learning_rate': 0.5, 'thread_count':
1},
 0.9057345599744021)
```

catboost проверяем на тестовой выборке

In [86]:

```
test_features = test_features.drop(['EndDate','Monthly','services','duration_mor
                                    'Type','InternetService',
                                    'totalsumm'],axis=1)
```

In [87]:

```
test_features = pd.get_dummies(test_features , drop_first=True)

```

In [88]:

```
train_lin.columns.sort_values()==test_features.columns.sort_values()
```

Out[88]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True])
```

In [89]:

```
test_features_scaled = scaler.transform(test_features)
```

In [90]:

```
test_features_scaled.shape, train_features_lin_scaled.shape,
```

Out[90]:

```
((1761, 20), (5282, 20))
```

In [91]:

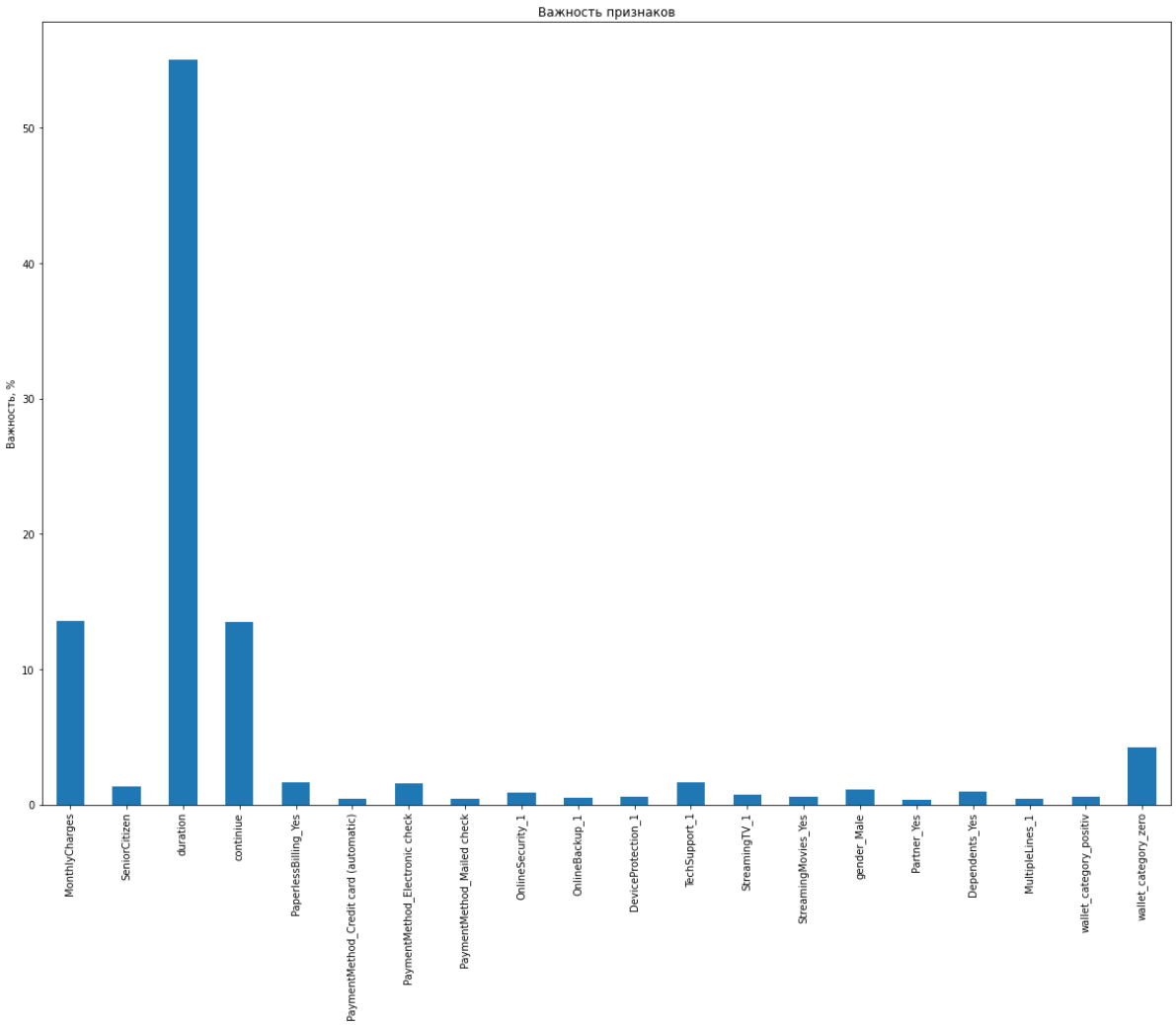
```
model = CatBoostClassifier(custom_loss = 'Logloss',
                           iterations=100,
                           thread_count=1,
                           depth=5,
                           learning_rate=0.3)
model.fit(train_features_lin_scaled, train_target)
predict=model.predict_proba(test_features_scaled)

print(roc_auc_score(test_target, predict[:,1]))

```

```
0:      learn: 0.5446715        total: 1.77ms   remaining: 175ms
1:      learn: 0.4826870        total: 4.39ms   remaining: 215ms
2:      learn: 0.4571751        total: 7.17ms   remaining: 232ms
3:      learn: 0.4388132        total: 9.76ms   remaining: 234ms
4:      learn: 0.4231454        total: 12.4ms   remaining: 235ms
5:      learn: 0.4098427        total: 14.9ms   remaining: 234ms
6:      learn: 0.3994925        total: 17.6ms   remaining: 233ms
7:      learn: 0.3942937        total: 20.4ms   remaining: 235ms
8:      learn: 0.3911535        total: 23.1ms   remaining: 234ms
9:      learn: 0.3869546        total: 25.7ms   remaining: 231ms
10:     learn: 0.3848560        total: 28.4ms   remaining: 230ms
11:     learn: 0.3819866        total: 30.4ms   remaining: 223ms
12:     learn: 0.3754721        total: 32.7ms   remaining: 219ms
13:     learn: 0.3726929        total: 34.4ms   remaining: 212ms
14:     learn: 0.3697428        total: 36.4ms   remaining: 206ms
15:     learn: 0.3678783        total: 38.4ms   remaining: 201ms
16:     learn: 0.3662372        total: 40.3ms   remaining: 197ms
17:     learn: 0.3650882        total: 41.9ms   remaining: 191ms
18:     learn: 0.3640207        total: 43.7ms   remaining: 186ms
19:     learn: 0.3626985        total: 45.7ms   remaining: 183ms
20:     learn: 0.3602438        total: 47.4ms   remaining: 178ms
21:     learn: 0.3540522        total: 49.2ms   remaining: 174ms
22:     learn: 0.3526720        total: 50.9ms   remaining: 170ms
23:     learn: 0.3513257        total: 52.8ms   remaining: 167ms
24:     learn: 0.3504148        total: 54.6ms   remaining: 164ms
25:     learn: 0.3497710        total: 56.3ms   remaining: 160ms
26:     learn: 0.3478311        total: 57.9ms   remaining: 157ms
27:     learn: 0.3456446        total: 59.6ms   remaining: 153ms
28:     learn: 0.3442848        total: 61.2ms   remaining: 150ms
29:     learn: 0.3369934        total: 62.9ms   remaining: 147ms
30:     learn: 0.3350257        total: 64.5ms   remaining: 143ms
31:     learn: 0.3326926        total: 66.1ms   remaining: 141ms
32:     learn: 0.3295370        total: 67.8ms   remaining: 138ms
33:     learn: 0.3280078        total: 69.4ms   remaining: 135ms
34:     learn: 0.3266860        total: 71.2ms   remaining: 132ms
35:     learn: 0.3249891        total: 72.7ms   remaining: 129ms
36:     learn: 0.3234777        total: 74.3ms   remaining: 127ms
37:     learn: 0.3201338        total: 76.1ms   remaining: 124ms
38:     learn: 0.3193763        total: 77.9ms   remaining: 122ms
39:     learn: 0.3180443        total: 79.7ms   remaining: 119ms
40:     learn: 0.3166662        total: 81.4ms   remaining: 117ms
41:     learn: 0.3144200        total: 83.4ms   remaining: 115ms
42:     learn: 0.3126926        total: 85.6ms   remaining: 113ms
43:     learn: 0.3104078        total: 87.9ms   remaining: 112ms
44:     learn: 0.3074574        total: 89.7ms   remaining: 110ms
45:     learn: 0.3061767        total: 91.6ms   remaining: 107ms
46:     learn: 0.3032652        total: 93.4ms   remaining: 105ms
47:     learn: 0.3019929        total: 95.3ms   remaining: 103ms
48:     learn: 0.2985114        total: 97.2ms   remaining: 101ms
49:     learn: 0.2973227        total: 99.2ms   remaining: 99.2ms
50:     learn: 0.2955237        total: 101ms    remaining: 97.1ms
51:     learn: 0.2938850        total: 104ms    remaining: 96ms
52:     learn: 0.2911880        total: 106ms    remaining: 94.4ms
53:     learn: 0.2897035        total: 109ms    remaining: 93.1ms
54:     learn: 0.2887861        total: 112ms    remaining: 91.3ms
55:     learn: 0.2879118        total: 114ms    remaining: 89.5ms
56:     learn: 0.2872666        total: 116ms    remaining: 87.4ms
57:     learn: 0.2858940        total: 118ms    remaining: 85.2ms
58:     learn: 0.2833503        total: 120ms    remaining: 83.2ms
59:     learn: 0.2820078        total: 122ms    remaining: 81.1ms
60:     learn: 0.2812318        total: 123ms    remaining: 78.9ms
```

```
61:     learn: 0.2795461        total: 125ms    remaining: 76.8ms
62:     learn: 0.2781669        total: 128ms    remaining: 75.1ms
63:     learn: 0.2752394        total: 131ms    remaining: 73.5ms
64:     learn: 0.2746163        total: 133ms    remaining: 71.6ms
65:     learn: 0.2727407        total: 135ms    remaining: 69.7ms
66:     learn: 0.2706232        total: 138ms    remaining: 67.9ms
67:     learn: 0.2678948        total: 141ms    remaining: 66.1ms
68:     learn: 0.2664334        total: 143ms    remaining: 64.4ms
69:     learn: 0.2654163        total: 146ms    remaining: 62.4ms
70:     learn: 0.2638457        total: 147ms    remaining: 60.2ms
71:     learn: 0.2630642        total: 149ms    remaining: 58ms
72:     learn: 0.2621632        total: 152ms    remaining: 56.1ms
73:     learn: 0.2616504        total: 154ms    remaining: 54ms
74:     learn: 0.2597020        total: 156ms    remaining: 52ms
75:     learn: 0.2588061        total: 158ms    remaining: 50ms
76:     learn: 0.2561684        total: 161ms    remaining: 48.2ms
77:     learn: 0.2546679        total: 164ms    remaining: 46.4ms
78:     learn: 0.2534955        total: 167ms    remaining: 44.5ms
79:     learn: 0.2516913        total: 170ms    remaining: 42.5ms
80:     learn: 0.2510843        total: 173ms    remaining: 40.5ms
81:     learn: 0.2494686        total: 175ms    remaining: 38.5ms
82:     learn: 0.2483772        total: 178ms    remaining: 36.4ms
83:     learn: 0.2475652        total: 180ms    remaining: 34.3ms
84:     learn: 0.2469184        total: 183ms    remaining: 32.3ms
85:     learn: 0.2460527        total: 186ms    remaining: 30.2ms
86:     learn: 0.2453345        total: 188ms    remaining: 28.1ms
87:     learn: 0.2447194        total: 190ms    remaining: 25.9ms
88:     learn: 0.2432978        total: 191ms    remaining: 23.7ms
89:     learn: 0.2423603        total: 193ms    remaining: 21.5ms
90:     learn: 0.2403611        total: 195ms    remaining: 19.3ms
91:     learn: 0.2390078        total: 197ms    remaining: 17.1ms
92:     learn: 0.2384974        total: 198ms    remaining: 14.9ms
93:     learn: 0.2376199        total: 200ms    remaining: 12.8ms
94:     learn: 0.2369635        total: 202ms    remaining: 10.6ms
95:     learn: 0.2356211        total: 203ms    remaining: 8.48ms
96:     learn: 0.2346228        total: 205ms    remaining: 6.34ms
97:     learn: 0.2334620        total: 207ms    remaining: 4.22ms
98:     learn: 0.2321748        total: 209ms    remaining: 2.11ms
99:     learn: 0.2314451        total: 211ms    remaining: 0us
0.9192943679845196
```

In [92]:

```
cat_iris_imp = pd.Series(model.get_feature_importance(),
                         test_features.columns)
fig, ax = plt.subplots(figsize=(16,14))
cat_iris_imp.plot.bar(ax=ax)
ax.set_title("Важность признаков")
ax.set_ylabel('Важность, %')
fig.tight_layout()
```



важные признаки
1. продолжительность дней/месяцев,
2. количество продлений,
3. Месячная оплата, оплата электронным чеком, нулевой баланс на конец месяца

Отчет

Какие пункты плана были выполнены, а какие — нет. Почему?

> Были выполненны все пункты плана

Какие трудности возникли и как вы их преодолели?

> Труднее всего мне было заметить, что часть параметров влияет на утечку данных. Тилид подсказал в чем проблема.

Какие ключевые шаги в решении задачи вы выделили?

> Проработка плана работы
> Анализ и предобработка полученных данных
> Проверка гипотез
> Обучение модели
> Анализ результатов

Какие признаки использовали для обучения модели и какая предобработка этих признаков была выполнена?

> Monthly Charges, Senior Citizen, duration, continue, PaperLessbilling, PaymentMethod, OnlineSecurity, DeviceProtection, TechSupport, streamingTV, StreamingMovies, gender, Parthner, Dependens, MultipleLines, wallet_category

> Можно выделить 3 важных признака: Monthly Charges, Monthly Charges,continue

> Предобработка: заполнение пропусков, проверка на выбросы, ошибки, аномалии и дубли, кодирование и масштабирование категориальных признаков, генерация синтетических признаков (количество дней использования, количество продлений, баланс на конец месяца).

Какая ваша итоговая модель, какие у неё гиперпараметы и какое качество?

> Catboost
> custom_loss = 'Logloss',
> iterations=100,
> thread_count=1,
> depth=5,
> learning_rate=0.3

> roc_auc 0.92 на тестовой и 0.91 на обучающей выборке

Сводная таблица моделей

In [103]:

```
table = pd.DataFrame([['Рандомный лес',0.86],
['Дерево решений',0.81],
['Логистическая регрессия', 0.84],
['Градиентный бустинг', 0.91]])
table.columns = ['model','roc-auc']
table
```

Out[103]:

	model	roc-auc
0	Рандомный лес	0.86
1	Дерево решений	0.81
2	Логистическая регрессия	0.84
3	Градиентный бустинг	0.91