Отток Клиентов Телеком компании

Описание проекта

Оператор связи «Ниединогоразрыва.ком» хочет научиться прогнозировать отток клиентов. Если выяснится, что пользователь планирует уйти, ему будут предложены промокоды и специальные условия. Команда оператора собрала персональные данные о некоторых клиентах, информацию об их тарифах и договорах.

Описание услуг
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.

Описание данных

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.

Описание столбцов

- Customer ID - Уникальный идентификатор пользователя
- Gender - Половая принадлежность пользователя
- Begin Date - Дата начала пользования услугами.
- End Date - Дата окончания пользования услугами

- Type - Тип оплаты (ежемесячный, годовой и т.д.
- Paperless Billing - Факт выставления счета на электронную почту
- Payment Method - Способ оплаты
- Monthly Charges - Ежемесячные траты на услуги
- Total Charges - Всего потрачено на услуги
- Dependents - Наличие иждевенцев
- Senior Citizen - Наличие пенсионного статуса по возрасту
- Partner - Наличие супруга(и)
- Multiple Lines - Наличие возможности ведения параллельных линий во время звонка
- Internet Service - Тип подключения интернета (digital subscriber line (через телефонную линию) / Fiber optic(оптоволоконный кабель))
- Devise Protection - Наличие антивирусаа
- Online Security - Наличие услуги блокировки небезопасных сайтов
- Tech Support - Наличие выделенной линии технической поддержки
- Online Backup - Наличие облачного хранилища для резервного копирования данных
- Streaming TV - Стриминговое телевидение
- Streaming Movies - Каталог фильмов

Цель проекта

Построить модель для прогноза оттока клиентов. Разобраться в факторах и причинах прекращения пользования услугами компании

Загрузка и изучение данных

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import re

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder, StandardScaler
from sklearn.metrics import roc_auc_score, confusion_matrix, accuracy_score, roc_c
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
# from imblearn.over_sampling import SMOTENC

from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier

import phik

pd.set_option("display.max_columns", None)
pd.set_option("display.max_colwidth", None)
warnings.filterwarnings("ignore")
```

In [2]:
```
try:
    personal = pd.read_csv("D:/practicum/datasets_for_projects/final_provider/perso
except:
    personal = pd.read_csv("/datasets/final_provider/personal.csv")

try:
    phone = pd.read_csv("D:/practicum/datasets_for_projects/final_provider/phone.cs
except:
    phone = pd.read_csv("/datasets/final_provider/phone.csv")

try:
    internet = pd.read_csv("D:/practicum/datasets_for_projects/final_provider/inter
except:
    internet = pd.read_csv("/datasets/final_provider/internet.csv")

try:
    contract = pd.read_csv("D:/practicum/datasets_for_projects/final_provider/contr
except:
    contract = pd.read_csv("/datasets/final_provider/contract.csv")
```

In [3]:
```
print("\n____PERSONAL____\n")
personal.info()
display(personal.head())
personal.describe(include="all")
```

```
____PERSONAL____

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   customerID     7043 non-null   object
 1   gender         7043 non-null   object
 2   SeniorCitizen  7043 non-null   int64 
 3   Partner        7043 non-null   object
 4   Dependents     7043 non-null   object
dtypes: int64(1), object(4)
memory usage: 275.2+ KB
```

	customerID	gender	SeniorCitizen	Partner	Dependents
0	7590-VHVEG	Female	0	Yes	No
1	5575-GNVDE	Male	0	No	No
2	3668-QPYBK	Male	0	No	No
3	7795-CFOCW	Male	0	No	No
4	9237-HQITU	Female	0	No	No

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents
count	7043	7043	7043.000000	7043	7043
unique	7043	2	NaN	2	2
top	7590-VHVEG	Male	NaN	No	No
freq	1	3555	NaN	3641	4933
mean	NaN	NaN	0.162147	NaN	NaN
std	NaN	NaN	0.368612	NaN	NaN
min	NaN	NaN	0.000000	NaN	NaN
25%	NaN	NaN	0.000000	NaN	NaN
50%	NaN	NaN	0.000000	NaN	NaN
75%	NaN	NaN	0.000000	NaN	NaN
max	NaN	NaN	1.000000	NaN	NaN

In [4]:
```
personal.duplicated().sum()
```

Out[4]:
```
0
```

Вывод по таблице Personal

- В таблице 5 столбцов и 7043 строки
- Пропусков и дубликатов нет
- Все столбцы содержат бинрные категориальные переменные
- Столбец SeniorCitizen имеет целочисленный тип данных, в конечном итоге мы приведем все бинарные признаки к числовым значениям
- Названия столбцов в околоверблюжьем регистре

In [5]:
```
print("\n____PHONE____\n")
phone.info()
display(phone.head())
phone.describe()
```

```
____PHONE____

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6361 entries, 0 to 6360
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   customerID     6361 non-null   object
 1   MultipleLines  6361 non-null   object
dtypes: object(2)
memory usage: 99.5+ KB
```

	customerID	MultipleLines
0	5575-GNVDE	No
1	3668-QPYBK	No
2	9237-HQITU	No
3	9305-CDSKC	Yes
4	1452-KIOVK	Yes

Out[5]:

	customerID	MultipleLines
count	6361	6361
unique	6361	2
top	5575-GNVDE	No
freq	1	3390

In [6]:
```
phone.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
phone['customerID']. isin (personal['customerID']). value_counts ()
```

Out[7]:
```
True    6361
Name: customerID, dtype: int64
```

Вывод по таблице Phone

- Таблица содержит меньше записей чем предыдущая
- в предыдущей таблице присутствуют все уникальные идентификаторы пользователей
- регистр снова не в порядке
- помимо идентификатора присутствует один бинарный признак строкового типа
- пропусков и дубликатов нет

In [8]:
```
print("\n____INTERNET____\n")
internet.info()
display(internet.head())
internet.describe()
```

```
____INTERNET____

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5517 entries, 0 to 5516
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   customerID        5517 non-null   object
 1   InternetService   5517 non-null   object
 2   OnlineSecurity    5517 non-null   object
 3   OnlineBackup      5517 non-null   object
 4   DeviceProtection  5517 non-null   object
 5   TechSupport       5517 non-null   object
 6   StreamingTV       5517 non-null   object
 7   StreamingMovies   5517 non-null   object
dtypes: object(8)
memory usage: 344.9+ KB
```

	customerID	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Stre
0	7590-VHVEG	DSL	No	Yes	No	No	
1	5575-GNVDE	DSL	Yes	No	Yes	No	
2	3668-QPYBK	DSL	Yes	Yes	No	No	
3	7795-CFOCW	DSL	Yes	No	Yes	Yes	
4	9237-HQITU	Fiber optic	No	No	No	No	

Out[8]:

	customerID	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
count	5517	5517	5517	5517	5517	5517
unique	5517	2	2	2	2	2
top	7590-VHVEG	Fiber optic	No	No	No	No
freq	1	3096	3498	3088	3095	3473

In [9]:
```
internet.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
internet['customerID']. isin (personal['customerID']). value_counts ()
```

Out[10]:
```
True    5517
Name: customerID, dtype: int64
```

Вывод по таблице Internet

- 7 столбцов помимо идентификатора. все бинарные, строковые
- строк в таблице еще меньше
- все идентификаторы присутсвуют в большой таблице
- пропусков и дубликатов нет

- названия столбцов аналогично

In [11]:
```
print("\n____CONTRACT____\n")
contract.info()
display(contract.head(10))
contract.describe()
```

```
____CONTRACT____

<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 
 5   PaymentMethod     7043 non-null   object 
 6   MonthlyCharges    7043 non-null   float64
 7   TotalCharges      7043 non-null   object 
dtypes: float64(1), object(7)
memory usage: 440.3+ KB
```

	customerID	BeginDate	EndDate	Type	PaperlessBilling	PaymentMethod	MonthlyCharges
0	7590-VHVEG	2020-01-01	No	Month-to-month	Yes	Electronic check	29.85
1	5575-GNVDE	2017-04-01	No	One year	No	Mailed check	56.95
2	3668-QPYBK	2019-10-01	2019-12-01 00:00:00	Month-to-month	Yes	Mailed check	53.85
3	7795-CFOCW	2016-05-01	No	One year	No	Bank transfer (automatic)	42.30
4	9237-HQITU	2019-09-01	2019-11-01 00:00:00	Month-to-month	Yes	Electronic check	70.70
5	9305-CDSKC	2019-03-01	2019-11-01 00:00:00	Month-to-month	Yes	Electronic check	99.65
6	1452-KIOVK	2018-04-01	No	Month-to-month	Yes	Credit card (automatic)	89.10
7	6713-OKOMC	2019-04-01	No	Month-to-month	No	Mailed check	29.75
8	7892-POOKP	2017-07-01	2019-11-01 00:00:00	Month-to-month	Yes	Electronic check	104.80
9	6388-TABGU	2014-12-01	No	One year	No	Bank transfer (automatic)	56.15

Out[11]:

	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

In [12]:
```
contract.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
contract['customerID']. isin (personal['customerID']). value_counts ()
```

Out[13]:
```
True    7043
Name: customerID, dtype: int64
```

Вывод по таблице Contract

- строк столько же, сколько и в первой таблице, все идентификаторы совпадают
- два столбца с датами и один с числами в строковом формате
- один числовой столбец с неприрывной величиной в нужном формате, остальные категориальные
- признак EndDate имеет помимо дат значение No. Из него сгенерируем целевойпризнак (ушедшие/оставшиеся клиенты)
- Остальное аналогично с прочими таблицами

План!

1) Объединить таблицы
2) Привести названия столбцов к змеиному регистру
3) Сгенерировать целевой признак на основе $End\ Date$
4) Привести данные в столбцах с датами к типу $datetime$
5) Привести $Total\ Charges$ к типу $float$
6) Сгенерировать признаки длительности пользования услугами $Duration$ в годах и месяцах при помощи дат
7) Заполнить пропуски в столбцах с телефонными и интернет услугами
8) Удалить столбец $Customer\ ID$ как не несущий полезной информации
9) Удалить столбцы $Begin\ Date$ и $End\ Date$ для предотвращения утечки целевого признака при обучении
10) Сгенерировать пару признаков на основе сочетаний и количества интернет услуг
11) Отделить целевой признак от остальных (разбить выборку на $features$ и $target$)
12) Разбить закодированные признаки на тренировочную $train$ и тестовую $test$

выборки

13) Закодировать признаки двумя способами: $One\ Hot\ Encoding$ и $OrdinalEncoding$

14) Произвести кросс-валидацию с подбором гиперпараметров при помощи $GridSearchCV$ для разных моделей

15) Повторить то же действие, применив $SMOTENC$ для балансировки классов в несбалансированных признаках тренировочных данных

16) Выбрать лучшую модель посмотреть на вклад признаков в работу модели припомощи атрибута $feature\_importances\_$

17) Удалить лишние признаки при наличии таковых

18) Проверить как повлияло удаление признаков на качество и время обучения

19) Обучить модель на общем наборе тренировочных данных, сделать предсказания на тестовой выборке, посчитать итоговый $ROC-AUC$

20) Пораскинуть мозгами и навтыкать везде красивых осмысленных графиков

Вопросы

1. Данные в таблице phone означают, что некоторые могут пользоваться несколькими линиями (Yes) или одной (No), а отсутствие данных говорит об отсутствии подключения?
2. По интернету аналогично. Отсутствие данных означает отсутствие подключения?

Предобработка и анализ

In [14]:
```
# Объединяем таблицы
df = personal.merge(contract, how="left").merge(phone, how="left").merge(internet,
df.head()
```

Out[14]:

	customerID	gender	SeniorCitizen	Partner	Dependents	BeginDate	EndDate	Type	Paperle
0	7590-VHVEG	Female	0	Yes	No	2020-01-01	No	Month-to-month	
1	5575-GNVDE	Male	0	No	No	2017-04-01	No	One year	
2	3668-QPYBK	Male	0	No	No	2019-10-01	2019-12-01 00:00:00	Month-to-month	
3	7795-CFOCW	Male	0	No	No	2016-05-01	No	One year	
4	9237-HQITU	Female	0	No	No	2019-09-01	2019-11-01 00:00:00	Month-to-month	

In [15]:
```
def to_snake(df):
    """Функция приводит названия
    столбцов к змеиному регистру"""
    new_names = []
```

```
    for col in df.columns:
        name = ""
        for i in range(len(col)):
            if col[i-1].isupper():
                name +=(col[i].lower())
            elif col[i].isupper():
                name += ("_" + col[i].lower())
            else:
                name += col[i]
        new_names.append(name.lstrip("_"))
    return new_names
```

In [16]:
```
# наводим порядок в названиях столбцов
df.columns = to_snake(df)
```

In [17]:
```
df.columns
```

Out[17]:
```
Index(['customer_id', 'gender', 'senior_citizen', 'partner', 'dependents',
       'begin_date', 'end_date', 'type', 'paperless_billing', 'payment_method',
       'monthly_charges', 'total_charges', 'multiple_lines',
       'internet_service', 'online_security', 'online_backup',
       'device_protection', 'tech_support', 'streaming_tv',
       'streaming_movies'],
      dtype='object')
```

Переименуем столбец **type** в **payment_type** чтобы оно выражало суть

In [18]:
```
df = df.rename({"type":'payment_type'}, axis=1)
df.columns
```

Out[18]:
```
Index(['customer_id', 'gender', 'senior_citizen', 'partner', 'dependents',
       'begin_date', 'end_date', 'payment_type', 'paperless_billing',
       'payment_method', 'monthly_charges', 'total_charges', 'multiple_lines',
       'internet_service', 'online_security', 'online_backup',
       'device_protection', 'tech_support', 'streaming_tv',
       'streaming_movies'],
      dtype='object')
```

Еще раз взглянем на форматы данных и пропуски

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   customer_id        7043 non-null   object 
 1   gender             7043 non-null   object 
 2   senior_citizen     7043 non-null   int64  
 3   partner            7043 non-null   object 
 4   dependents         7043 non-null   object 
 5   begin_date         7043 non-null   object 
 6   end_date           7043 non-null   object 
 7   payment_type       7043 non-null   object 
 8   paperless_billing  7043 non-null   object 
 9   payment_method     7043 non-null   object 
 10  monthly_charges    7043 non-null   float64
 11  total_charges      7043 non-null   object 
 12  multiple_lines     6361 non-null   object 
 13  internet_service   5517 non-null   object 
 14  online_security    5517 non-null   object 
 15  online_backup      5517 non-null   object 
 16  device_protection  5517 non-null   object 
 17  tech_support       5517 non-null   object 
 18  streaming_tv       5517 non-null   object 
 19  streaming_movies   5517 non-null   object 
dtypes: float64(1), int64(1), object(18)
memory usage: 1.1+ MB
```

Сгенерируем целевой признак из признака с датой ухода. Положительным классом будут ушедшие клиенты

In [20]:
```
# генерируем целевой признак
df["exited"] = (df["end_date"] != "No") * 1
```

In [21]:
```
# приводим даты к типу datetime
df["begin_date"] = pd.to_datetime(df["begin_date"], format = "%Y-%m")
df.loc[df["end_date"] == "No", "end_date"] = "2020-02-01"
df["end_date"] = pd.to_datetime(df["end_date"], format = "%Y-%m")
```

Попробуем привести столбец total_charges к типу числа с плавающей точкой

In [22]:
```
df["total_charges"] = pd.to_numeric(df["total_charges"], errors="ignore")
df["total_charges"].dtype
```

Out[22]:
```
dtype('O')
```

Не получилось, значит в данных есть не числовые значения. Найдем их

In [23]:
```
result = []
for i in df["total_charges"]:
    matches = re.findall(r"[^\d.]", i )
    if len(matches) > 0:
        result.append(i)
result
```

Out[23]:
```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

Посмотрим на строки с пробелами вместо значений

In [24]:
```
df[df["total_charges"] == " "]
```

Out[24]:

	customer_id	gender	senior_citizen	partner	dependents	begin_date	end_date	payment_t
488	4472-LVYGI	Female	0	Yes	Yes	2020-02-01	2020-02-01	Two y
753	3115-CZMZD	Male	0	No	Yes	2020-02-01	2020-02-01	Two y
936	5709-LVOEQ	Female	0	Yes	Yes	2020-02-01	2020-02-01	Two y
1082	4367-NUYAO	Male	0	Yes	Yes	2020-02-01	2020-02-01	Two y
1340	1371-DWPAZ	Female	0	Yes	Yes	2020-02-01	2020-02-01	Two y
3331	7644-OMVMY	Male	0	Yes	Yes	2020-02-01	2020-02-01	Two y
3826	3213-VVOLG	Male	0	Yes	Yes	2020-02-01	2020-02-01	Two y
4380	2520-SGTTA	Female	0	Yes	Yes	2020-02-01	2020-02-01	Two y
5218	2923-ARZLG	Male	0	Yes	Yes	2020-02-01	2020-02-01	One y
6670	4075-WKNIU	Female	0	Yes	Yes	2020-02-01	2020-02-01	Two y
6754	2775-SEFEE	Male	0	No	Yes	2020-02-01	2020-02-01	Two y

Очевидно, пропуски в столбце с суммарными затратами связаны с тем, что клиенты пришли в текущем месяце.

Заполним пропуски помесячной платой

In [25]:
```
# заполняем
df.loc[df["total_charges"] == " ", "total_charges"] = df["monthly_charges"]
```

In [26]:
```
# меняем тип данных
df["total_charges"] = pd.to_numeric(df["total_charges"], errors="ignore")
df["total_charges"].dtype
```

Out[26]:
```
dtype('float64')
```

In [27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   customer_id        7043 non-null   object
 1   gender             7043 non-null   object
 2   senior_citizen     7043 non-null   int64
 3   partner            7043 non-null   object
 4   dependents         7043 non-null   object
 5   begin_date         7043 non-null   datetime64[ns]
 6   end_date           7043 non-null   datetime64[ns]
 7   payment_type       7043 non-null   object
 8   paperless_billing  7043 non-null   object
 9   payment_method     7043 non-null   object
 10  monthly_charges    7043 non-null   float64
 11  total_charges      7043 non-null   float64
 12  multiple_lines     6361 non-null   object
 13  internet_service   5517 non-null   object
 14  online_security    5517 non-null   object
 15  online_backup      5517 non-null   object
 16  device_protection  5517 non-null   object
 17  tech_support       5517 non-null   object
 18  streaming_tv       5517 non-null   object
 19  streaming_movies   5517 non-null   object
 20  exited             7043 non-null   int32
dtypes: datetime64[ns](2), float64(2), int32(1), int64(1), object(15)
memory usage: 1.4+ MB
```

все типы данных приведены к нужному формату.

Сгенерируем признаки с полным количеством лет и месяцев пользования услугами компании

In [28]:
```
df["dur_months"] = (df["end_date"].dt.year - df["begin_date"].dt.year)*12 + df["en
df["dur_years"] = df["dur_months"] // 12
```

In [29]:
```
# смотрим на пропуски
df.isna().sum()
```

Out[29]:
```
customer_id             0
gender                  0
senior_citizen          0
partner                 0
dependents              0
begin_date              0
end_date                0
payment_type            0
paperless_billing       0
payment_method          0
monthly_charges         0
total_charges           0
multiple_lines        682
internet_service     1526
online_security      1526
online_backup        1526
device_protection    1526
tech_support         1526
streaming_tv         1526
streaming_movies     1526
exited                  0
dur_months              0
dur_years               0
dtype: int64
```

Заполним все пропуски в столбцах связанных с интернет услугами нулями, а так же заменим Yes на единицы, а No на нули

In [30]:
```
internet = ["online_security", "online_backup", "device_protection", "tech_support
df[internet] = (df[internet] == "Yes") * 1
```

In [31]:
```
df.isna().sum()
```

Out[31]:
```
customer_id             0
gender                  0
senior_citizen          0
partner                 0
dependents              0
begin_date              0
end_date                0
payment_type            0
paperless_billing       0
payment_method          0
monthly_charges         0
total_charges           0
multiple_lines        682
internet_service     1526
online_security         0
online_backup           0
device_protection       0
tech_support            0
streaming_tv            0
streaming_movies        0
exited                  0
dur_months              0
dur_years               0
dtype: int64
```

Осталось 2 столбца. Способ интернет подключения и наличие/отсутствие нескольких телефонных линий.

In [32]:
```
# смотрим какие есть значения в стлбце с интернет подключением
df["internet_service"].value_counts(dropna=False)
```

Out[32]:
```
Fiber optic    3096
DSL            2421
NaN            1526
Name: internet_service, dtype: int64
```

Так как пропуски говорят об отсутствии подключения, в столбце internet_service заполним их новым значением "No"

In [33]:
```
df["internet_service"] = df["internet_service"].fillna("No")
```

In [34]:
```
# смотрим значения в столбце multiple_lines
df["multiple_lines"].value_counts(dropna=False)
```

Out[34]:
```
No     3390
Yes    2971
NaN     682
Name: multiple_lines, dtype: int64
```

заменим значения в столбце multiple_lines, чтобы добавить третье значение. "Yes" на "Multiple", "No" на "One", и заполним пропуски значением "No"

In [35]:
```
df["multiple_lines"] = df["multiple_lines"].replace({"Yes" : "Multiple", "No" : "O
df["multiple_lines"].head(10)
```

Out[35]:
```
0          No
1         One
2         One
3          No
4         One
5    Multiple
6    Multiple
7          No
8    Multiple
9         One
Name: multiple_lines, dtype: object
```

In [36]:
```
# смотрим на результаты
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7043 entries, 0 to 7042
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   customer_id        7043 non-null   object        
 1   gender             7043 non-null   object        
 2   senior_citizen     7043 non-null   int64         
 3   partner            7043 non-null   object        
 4   dependents         7043 non-null   object        
 5   begin_date         7043 non-null   datetime64[ns]
 6   end_date           7043 non-null   datetime64[ns]
 7   payment_type       7043 non-null   object        
 8   paperless_billing  7043 non-null   object        
 9   payment_method     7043 non-null   object        
 10  monthly_charges    7043 non-null   float64       
 11  total_charges      7043 non-null   float64       
 12  multiple_lines     7043 non-null   object        
 13  internet_service   7043 non-null   object        
 14  online_security    7043 non-null   int32         
 15  online_backup      7043 non-null   int32         
 16  device_protection  7043 non-null   int32         
 17  tech_support       7043 non-null   int32         
 18  streaming_tv       7043 non-null   int32         
 19  streaming_movies   7043 non-null   int32         
 20  exited             7043 non-null   int32         
 21  dur_months         7043 non-null   int64         
 22  dur_years          7043 non-null   int64         
dtypes: datetime64[ns](2), float64(2), int32(7), int64(3), object(9)
memory usage: 1.4+ MB
```

Пропуски заполнены. Осталось избавиться от неинформативных столбцов и столбцов, которые могут спровоцировать утечку целевого признака

In [37]:
```
# удаляем
df = df.drop(["customer_id", "begin_date", "end_date"], axis=1)
```

Создадим еще два признака:

- наличие/отсутствие подключения интернета, ибо тип подключения может быть и не важен, а наличие - более важный признак
- и что-то вроде рейтинга активности использования услуг. каждая из интернет услуг будет единицей, наличие интернета - единица, а телефонное подключение сделаем в виде рейтинга. 0 - отсутствие, 1 - одна линия, 2 - несколько линий. и посчитаем сумму

In [38]:
```
df["phone"] = df["multiple_lines"].replace({"Multiple" : 2, "One" : 1, "No" : 0})
df["internet"] = (df["internet_service"] != "No") * 1
df["num_of_services"] = (
    df["phone"]
    + df["internet"]
    + df["online_security"]
    + df["online_backup"]
    + df["device_protection"]
    + df["tech_support"]
    + df["streaming_tv"]
    + df["streaming_movies"]
)
```

```
df = df.drop("phone", axis=1)
df.head()
```

Out[38]:

	gender	senior_citizen	partner	dependents	payment_type	paperless_billing	payment_method
0	Female	0	Yes	No	Month-to-month	Yes	Electronic check
1	Male	0	No	No	One year	No	Mailed check
2	Male	0	No	No	Month-to-month	Yes	Mailed check
3	Male	0	No	No	One year	No	Bank transfer (automatic)
4	Female	0	No	No	Month-to-month	Yes	Electronic check

отделим целевой признак

In [39]:
```
target = df["exited"]
features = df.drop("exited", axis=1)
```

In [40]:
```
# создаем списки с названиями категориальных и количественных признаков
categorical = ["gender",
               "senior_citizen",
               "partner",
               "dependents",
               "payment_type",
               "paperless_billing",
               "payment_method",
               "multiple_lines",
               "internet_service",
               "online_security",
               "online_backup",
               "device_protection",
               "tech_support",
               "streaming_tv",
               "streaming_movies",
               "internet"
              ]
numeric = ["monthly_charges",
           "total_charges",
           "dur_months",
           "dur_years",
           "num_of_services"
          ]
```

Упорядочим колонки. сначала категориальные, потом количественные

In [41]:
```
features = features[categorical+numeric]
features.head()
```

Out[41]:

	gender	senior_citizen	partner	dependents	payment_type	paperless_billing	payment_method
0	Female	0	Yes	No	Month-to-month	Yes	Electronic check
1	Male	0	No	No	One year	No	Mailed check
2	Male	0	No	No	Month-to-month	Yes	Mailed check
3	Male	0	No	No	One year	No	Bank transfer (automatic)
4	Female	0	No	No	Month-to-month	Yes	Electronic check

In [42]:

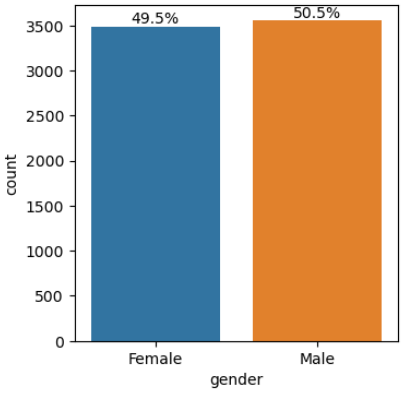
```
# визуализируем распределения
def dist_visualizer(cols):
    """Функция принимает на вход список названий столбцов и
    визуализирует распределение каждого признака и распределение
    его же по классам целевого признака"""
    for col in cols:
        if col in categorical:
            fig = plt.figure(figsize=(13,5))
            plt.suptitle(col, x=0.45, y=1.1, size=31)
            if len(df[col].unique()) > 2:
                fig.add_subplot(1, 2, 1)
                if len(df[col].unique()) > 3:
                    ax = sns.countplot(data=df, x=col, order=df[col].unique())
                    ax.set_xticklabels(ax.get_xticklabels(),rotation = 7)
                    for c in ax.containers:
                        labels = [f'{h/df.exited.count()*100:0.1f}%' if (h := v.ge
                        ax.bar_label(c, labels=labels, label_type='edge')
                else:
                    fig = plt.figure(figsize=(13,4))
                    plt.suptitle(col, x=0.45, y=1.1, size=26)
                    fig.add_subplot(1, 3, 1)
                ax = sns.countplot(data=df, x=col, order=df[col].unique())
                for c in ax.containers:
                    labels = [f'{h/df.exited.count()*100:0.1f}%' if (h := v.get_height
                    ax.bar_label(c, labels=labels, label_type='edge')
                fig.add_subplot(1, 2, 2)
                if len(df[col].unique()) > 3:
                    ax = sns.countplot(data=df, x=col, hue='exited', order=df[col].uni
                    ax.set_xticklabels(ax.get_xticklabels(),rotation = 7)
                    for c in ax.containers:
                        labels = [f'{h/df.exited.count()*100:0.1f}%' if (h := v.get_he
                        ax.bar_label(c, labels=labels, label_type='edge')
                else:
                    ax = sns.countplot(data=df, x=col, hue='exited', order=df[col].uni
                    for c in ax.containers:
                        labels = [f'{h/df.exited.count()*100:0.1f}%' if (h := v.get_he
                        ax.bar_label(c, labels=labels, label_type='edge')
                plt.tight_layout()
            else:
                bins = 10
                fig = plt.figure(figsize=(13,4))
                plt.suptitle(col, x=0.45, y=1, size=22)
                if len(df[col].unique()) < 10:
                    bins = len(df[col].unique())
                fig.add_subplot(1, 3, 1)
                sns.barplot(data=df, x="exited", y=col)
                fig.add_subplot(1, 2, 2)
```

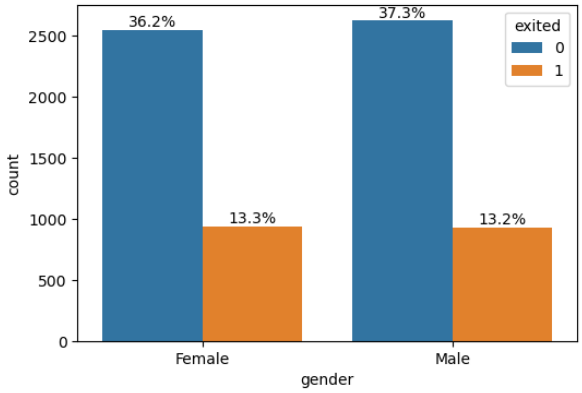
```
            sns.histplot(data=df, x=col, bins=bins)
    plt.show()
```

In [43]:
```
dist_visualizer(features.columns)
```

```
<Figure size 1300x500 with 0 Axes>
```

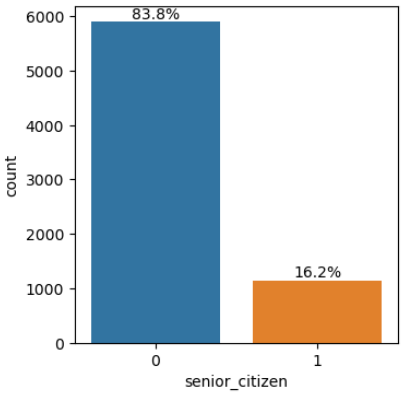
gender

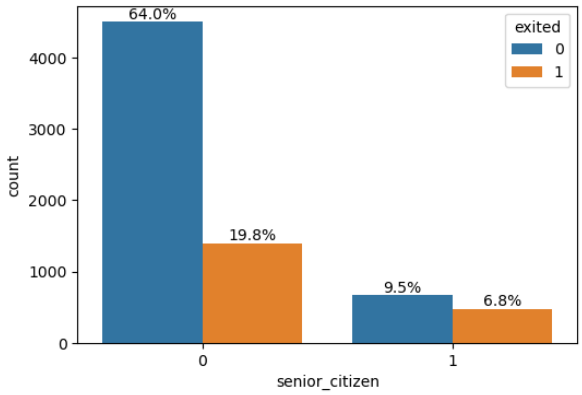





```
<Figure size 1300x500 with 0 Axes>
```

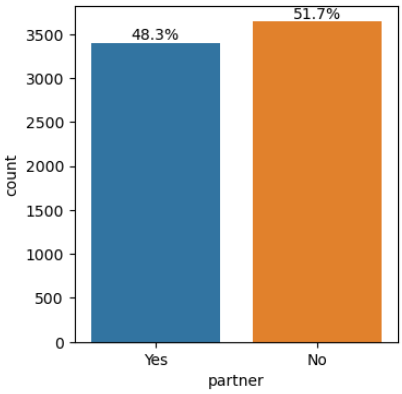
senior_citizen

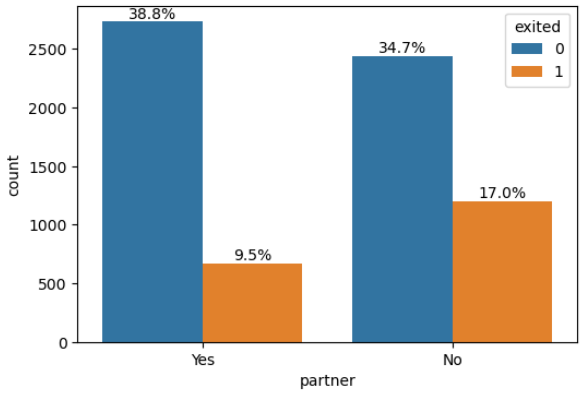





```
<Figure size 1300x500 with 0 Axes>
```

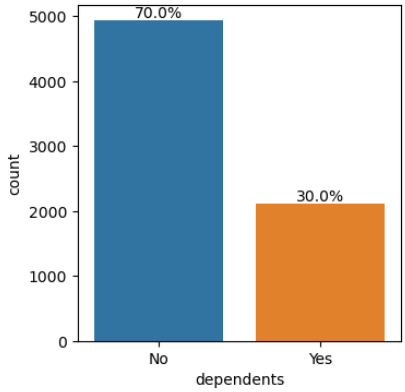
partner

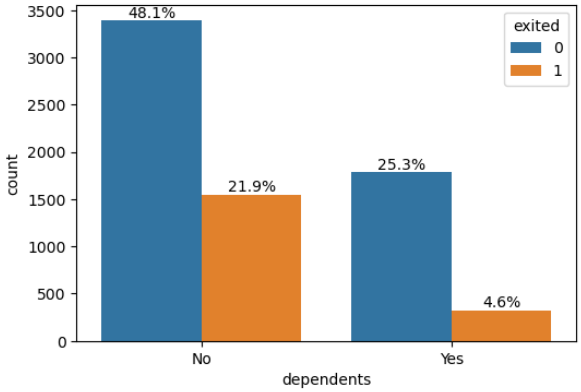




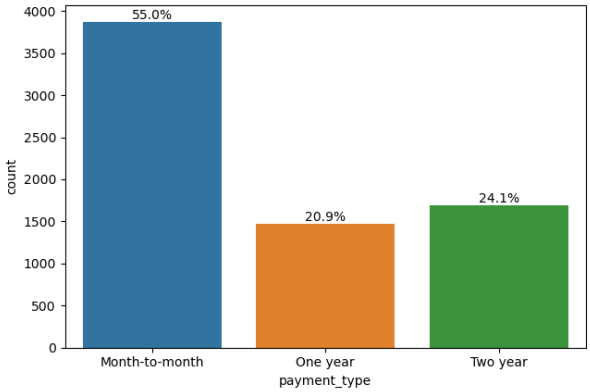

```
<Figure size 1300x500 with 0 Axes>
```

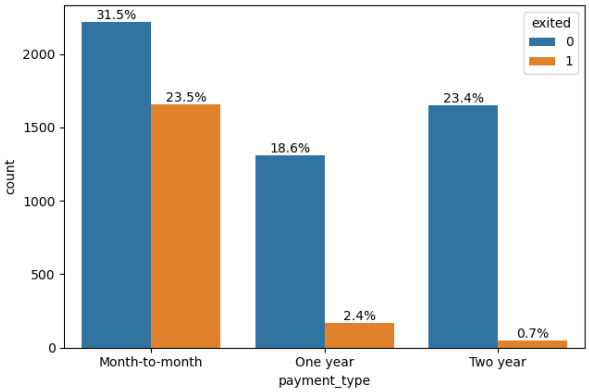
dependents





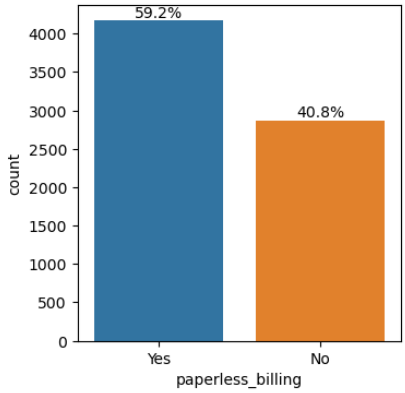
payment_type

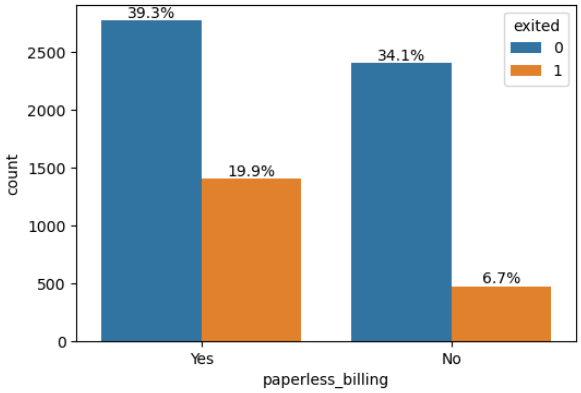




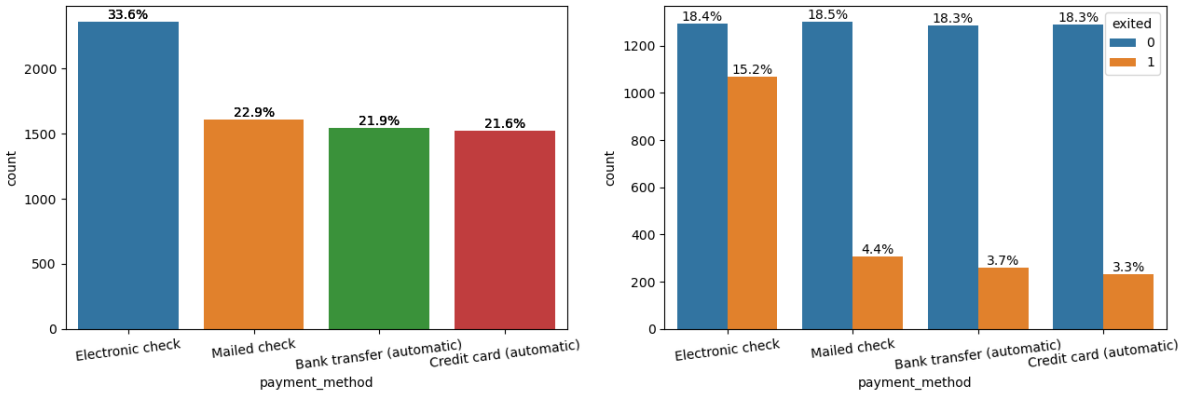

```
<Figure size 1300x500 with 0 Axes>
```

paperless_billing

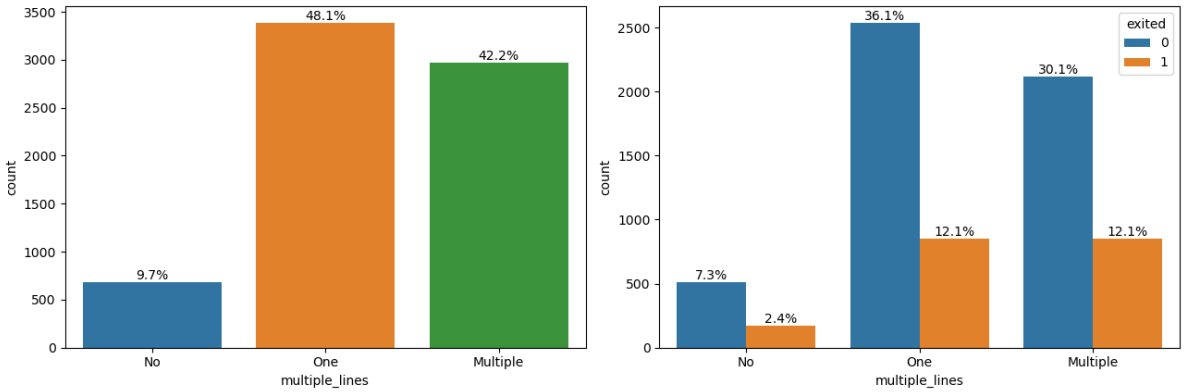




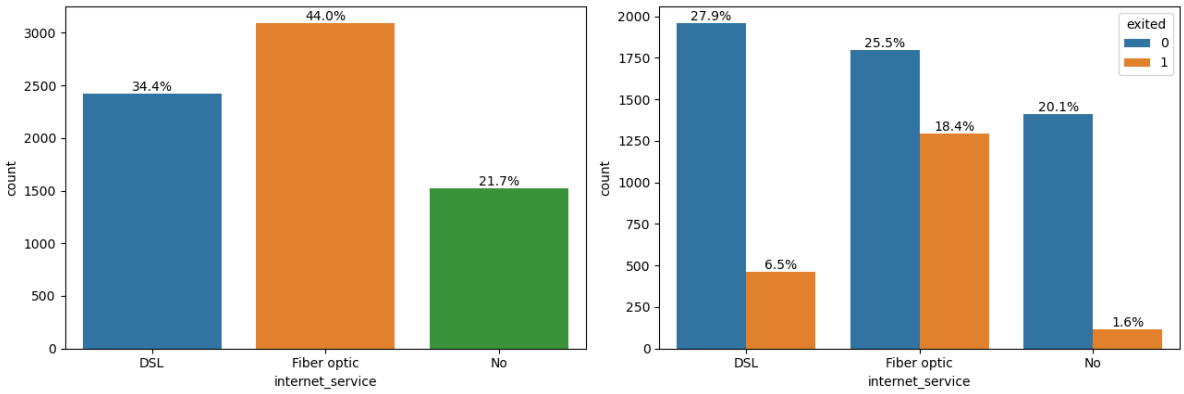
payment_method



multiple_lines

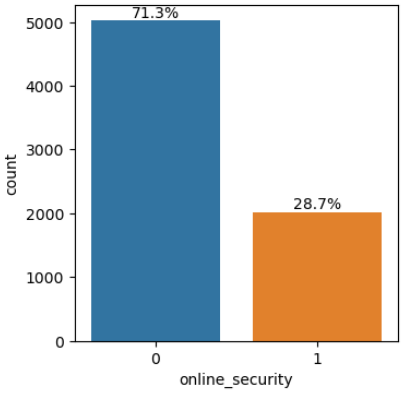


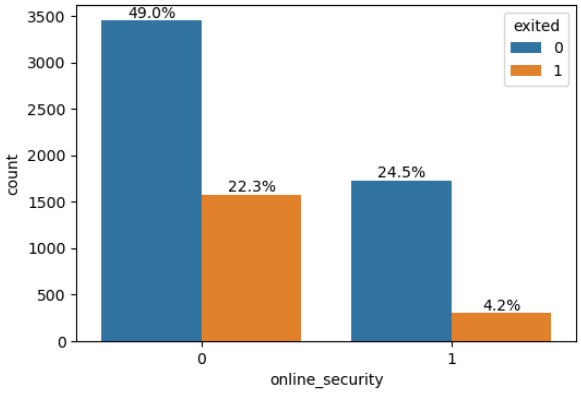
internet_service



<Figure size 1300x500 with 0 Axes>

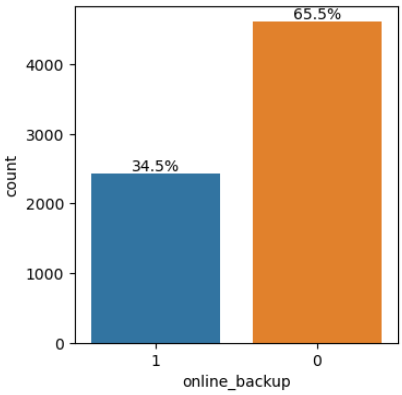
online_security

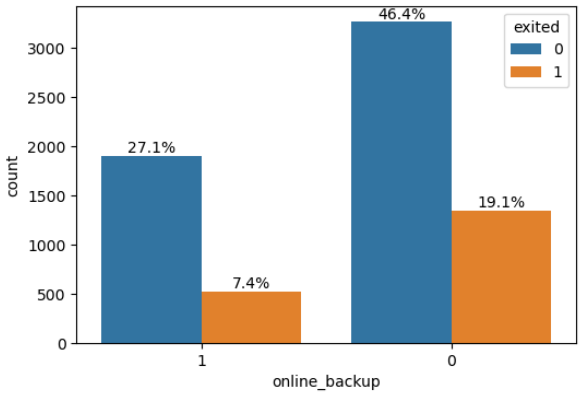




<Figure size 1300x500 with 0 Axes>

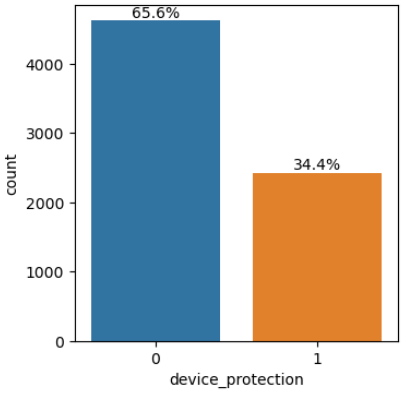
online_backup

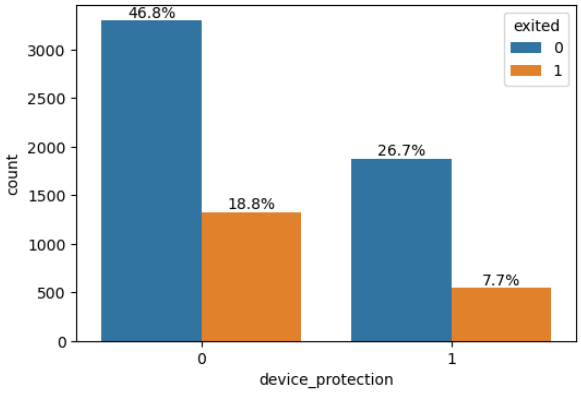




<Figure size 1300x500 with 0 Axes>

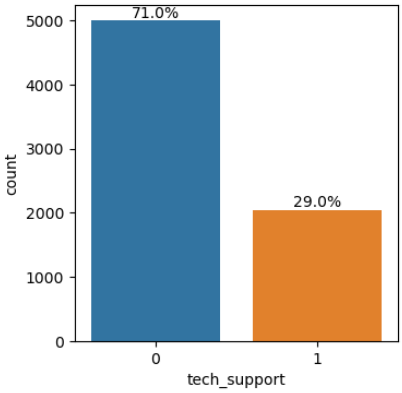
device_protection

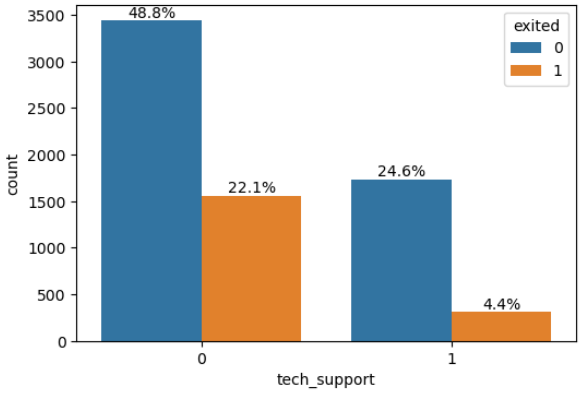




<Figure size 1300x500 with 0 Axes>

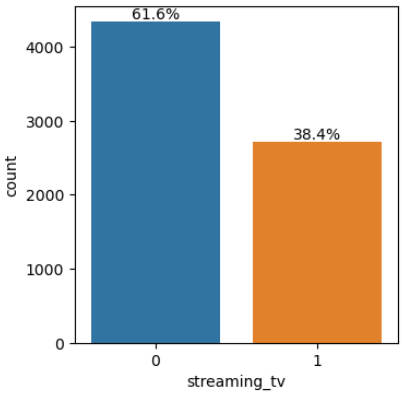
tech_support

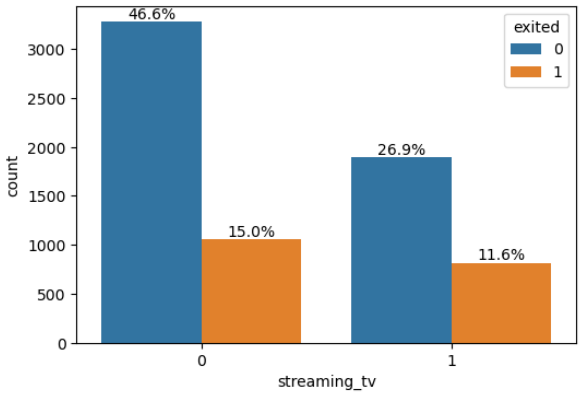





```
<Figure size 1300x500 with 0 Axes>
```

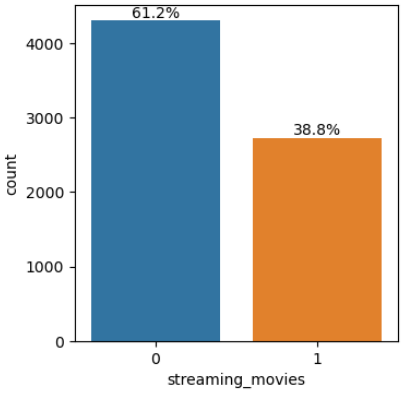
streaming_tv

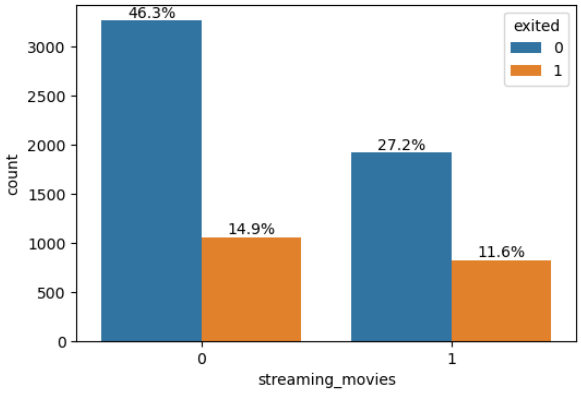





```
<Figure size 1300x500 with 0 Axes>
```

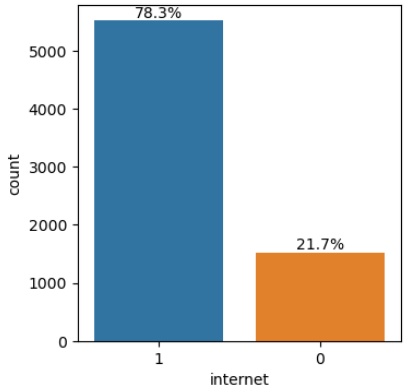
streaming_movies

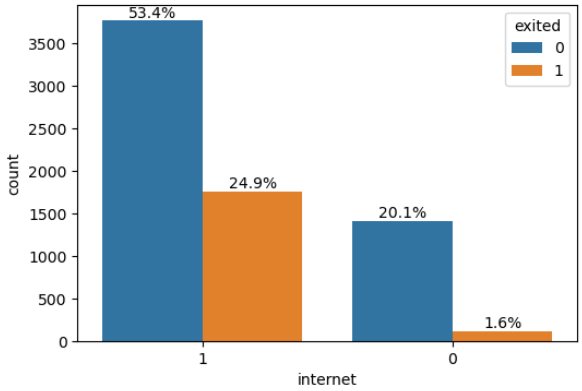




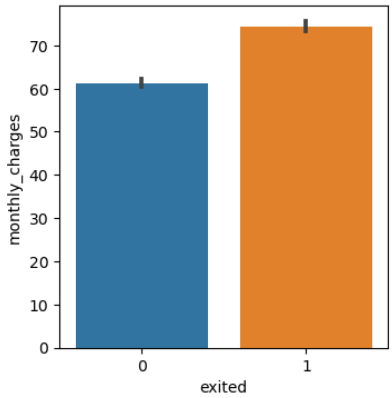

```
<Figure size 1300x500 with 0 Axes>
```

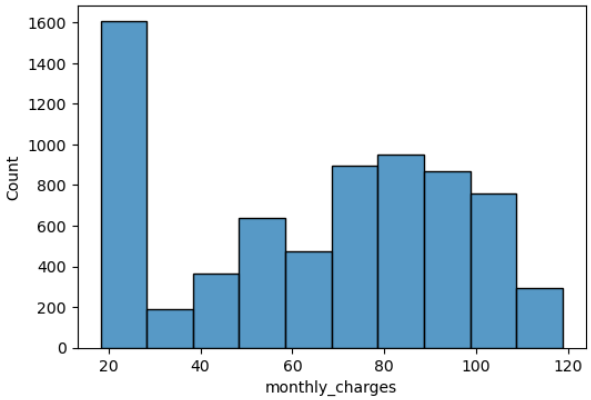
internet



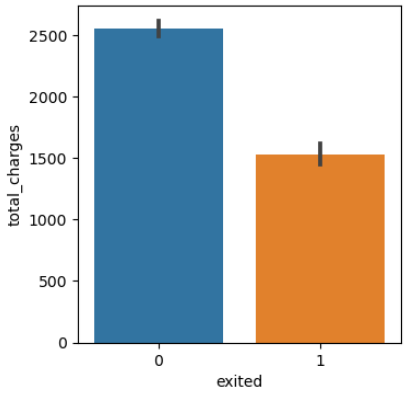


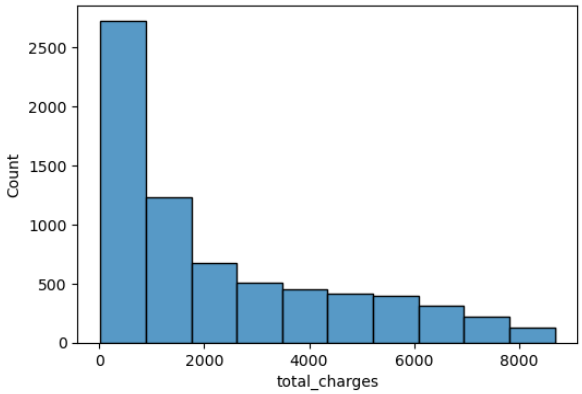
monthly_charges



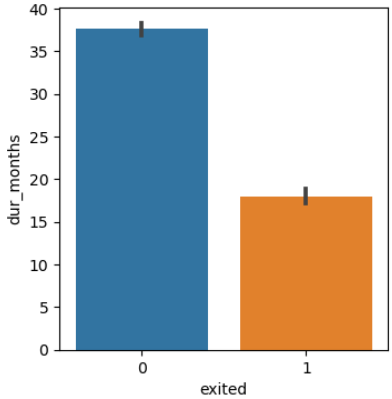


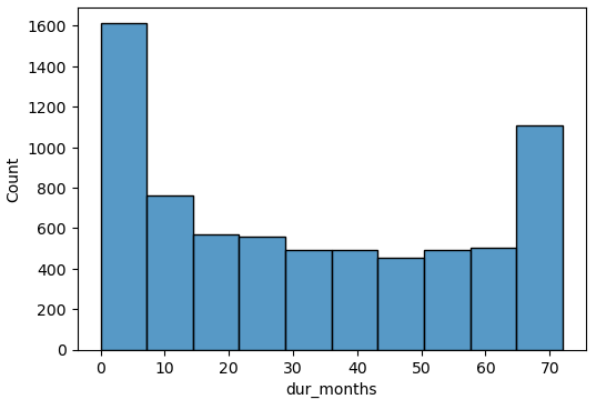
total_charges

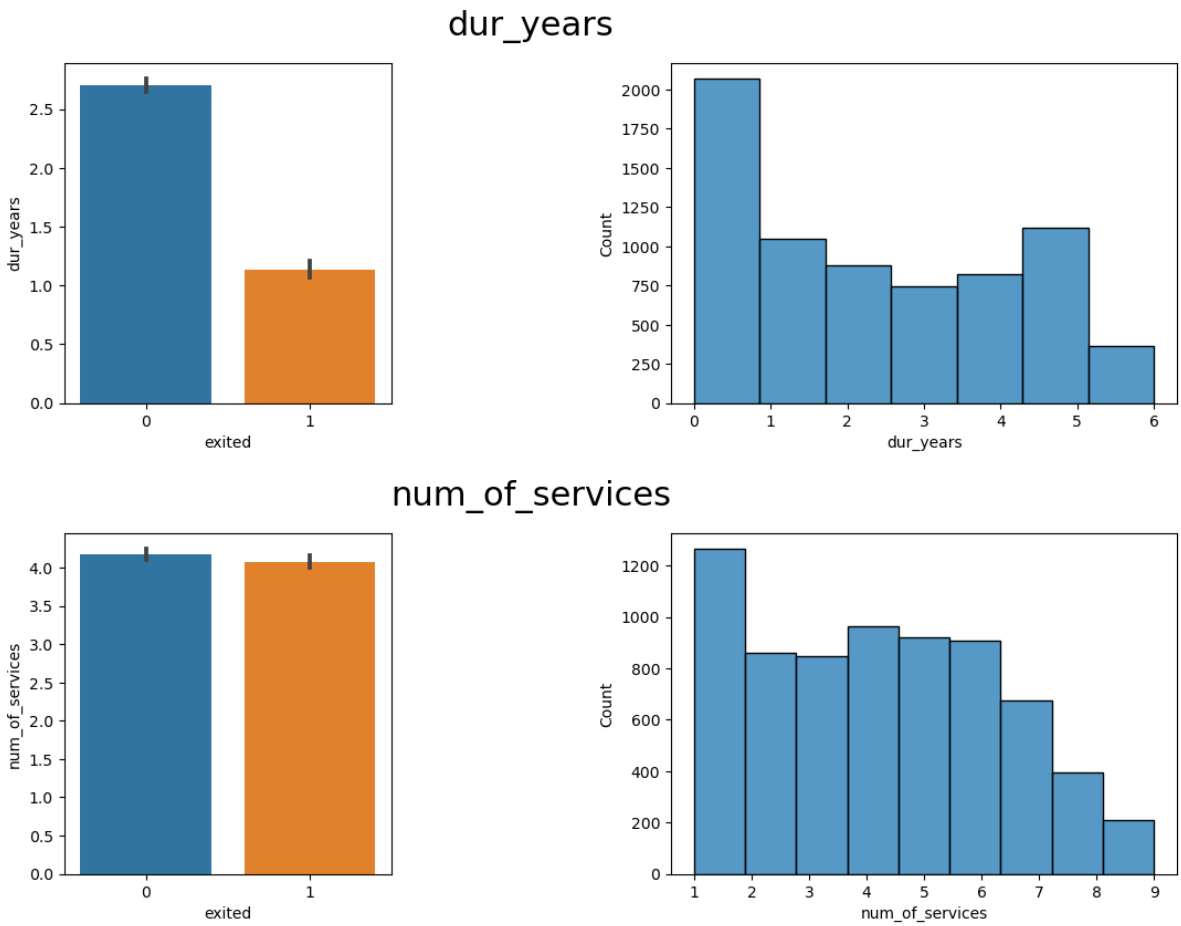




dur_months







Выводы по распределению признаков

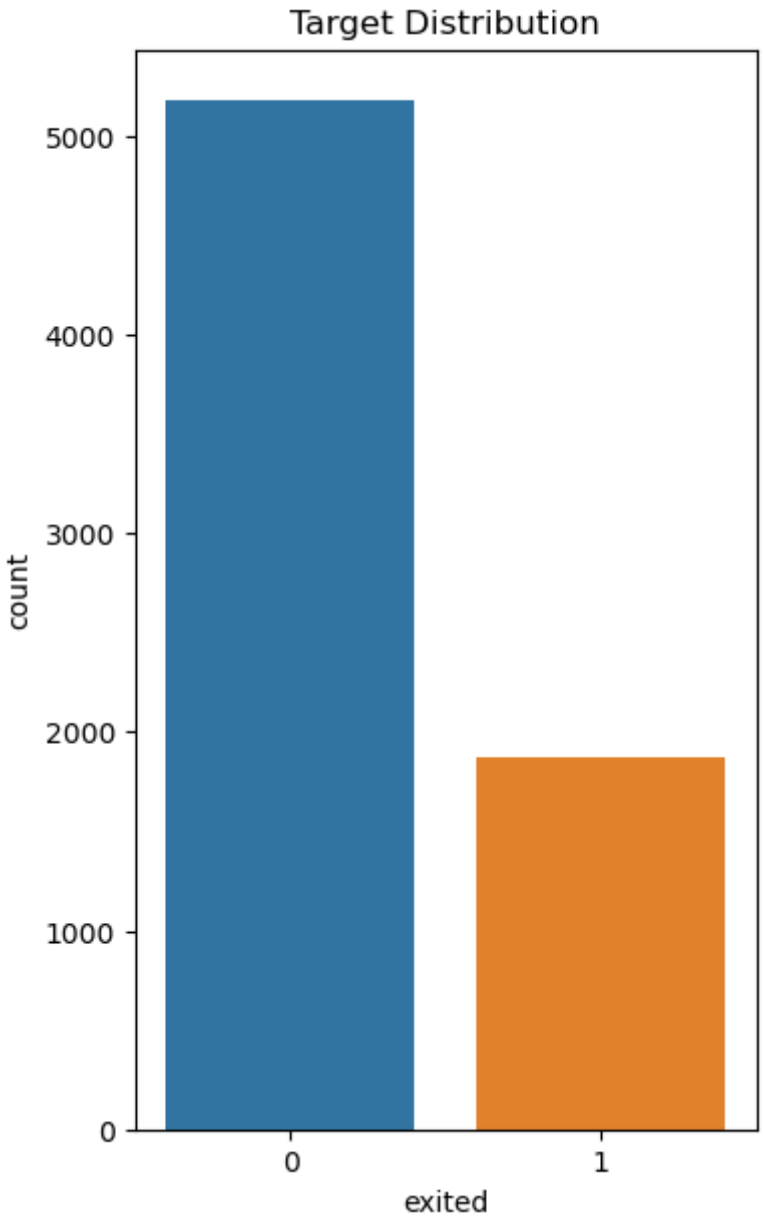
- По половому признаку клиенты распределены равномерно. распределение целевого признака по половой принадлежности так же равномерно
- Клиенты пенсионного возраста уходят значительно чаще, но их всего 16 процентов.
- По семейному положению распределение равномерное. Среди не состоящих в браке процент ушедших больше, чем среди состоящих
- Клиентов с иждевенцами 30 процентов и они уходят реже
- Половина клиентов платит за услуги помесячно, более половины из них уходит. Чем выше срок абонентской платы, тем ниже процент уходящих клиентов
- более стабильны клиенты предпочитающие бумажные чеки их 40 процентов от выборки
- по способу оплаты перевешивают клиенты предпочитающие электронные платежи они же и уходят чаще других
- по наличию нескольких телефонных линий не видно примечательных закономерностей
- клиенты с оптоволоконным кабелем уходят значительно чаще, чем с цифровой абонентской линией, а реже всего уходят те, кто не пользуется интернетом вообще
- У трети клиентов подключена услуга блокировки небезопасных сайтов и уходят они значительно реже прочих (стабильность!)
- В точности такая же ситуация с теми, кто пользуется тех-поддержкой

- С прочимми интернет-услугами ситуация похожа. Те кто ими пользуются уходят реже, кроме стримингового телевидения и каталога фильмов. там ситуация противоположная
- среди ушедших клиентов средняя месячная абонентская плата выше, а среднее значение общих затрат почти в 2 раза ниже
- в среднем человекоактивность оценивается в четверочку из 9) Как среди ушедших, так и среди оставшихся
- Диапазоны значений количественных признаков сильно различаются. Потребуется StandardScaler.

Присутствует дисбаланс классов в большинстве категориальных признаков. Посмотрим на распределение целевого признака

In [44]:
```
plt.figure(figsize=(4, 7))
plt.title("Target Distribution")
sns.countplot(data=df, x='exited', order =df["exited"].unique() )
```

Out[44]:
```
<AxesSubplot:title={'center':'Target Distribution'}, xlabel='exited', ylabel='coun
t'>
```

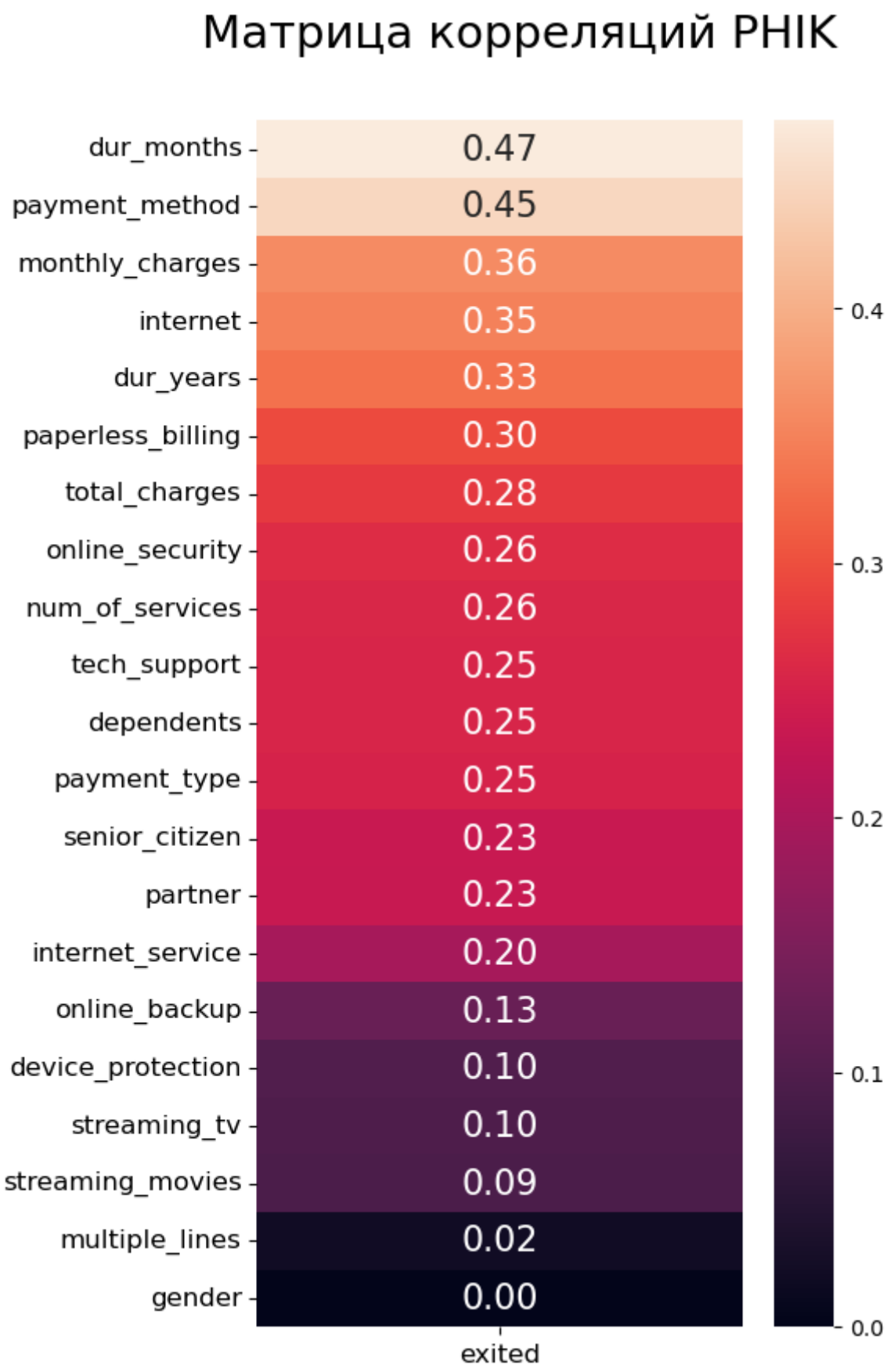


Классы целевого признака так же не сбалансированы

В ходе исследования для устранения дисбаланса классов был произведен ресэмплинг при помощи SMOTENC. Влияние на результаты было слабым и неоднозначным, в связи с чем было принято решение не использовать его в окончательном варианте

In [45]:
```
# Определяем индексы нелинейной корреляции и записываем в переменную
corr = df.phik_matrix()
# Визуализируем
plt.figure(figsize=(5,10))
sns.heatmap((corr[["exited"]]
             .drop("exited", axis=0)
             .sort_values(by="exited", ascending=False)),
             annot=True,
             fmt=".2f",
             annot_kws={'fontsize': 16}
            )
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.title('Матрица корреляций PHIK', size=21, x=.54, y=1.05)
plt.show()
```

```
interval columns not set, guessing: ['senior_citizen', 'monthly_charges', 'total_c
harges', 'online_security', 'online_backup', 'device_protection', 'tech_support',
'streaming_tv', 'streaming_movies', 'exited', 'dur_months', 'dur_years', 'interne
t', 'num_of_services']
```



наибольшие значения нелинейной корреляции с целевым признаком у признаков длительности пользования услугами, метода оплаты, ежемесячной оплаты и наличия интернет подключения.

Подбор модели

In [46]:
```
# создаем список для результатов всех моделей
final_list = []
def everything_maker(model_name, model, params, features, target, encoder="ordinal

    """Функция принимает на вход имя модели, модель, гиперпараметры и обучающие
    данные, а так же имя кодировщика категориальных данных,производит
    кроссвалидацию с подбором параметров и возвращает модель и
    параметры модели с наибольшим значением метрики ROC-AUC"""
    result_list = []    #     список с результатами кроссвалидации
#     разбиваем выборку на трейн и тест
    X_train, X_test, y_train, y_test = train_test_split(features,
                                                        target,
                                                        test_size=0.25,
                                                        random_state=281122,
                                                        stratify=(target)) #стратифика
    if encoder == "ohe":
        cat_transformer = OneHotEncoder() # drop="first")
    elif encoder == "ordinal":
        cat_transformer = OrdinalEncoder()
    else:
        print("Choose transformer!")
        return
#     для количественных признаков используем стандарт скеллер по умолчанию
    num_transformer = StandardScaler()
#     создаем пайплайн для кодирования
    col_transformer = ColumnTransformer(
        [
            ("cat", cat_transformer, categorical),
            ("num", num_transformer, numeric)
        ])
#     создаем пайплайн для грид сёрч
    pl = Pipeline(
        [
            ("transformer", col_transformer),
            ("model", model)
        ])
#     кроссвалидация с подбором параметров
    grid_search = GridSearchCV(pl, params, cv=5, scoring="roc_auc", n_jobs=-1)
    model = grid_search.fit(X_train, y_train)
    best_params = str(model.best_params_)                    # параметры лучшей м
    best_score = model.best_score_.round(4)                  # значение ROC-AUC д
    best_model = model.best_estimator_                       # лучшая модель
    best_model.fit(X_train, y_train)                         # обучаем модель
    accuracy = best_model.score(X_train, y_train).round(4)   # добавляем к списку
#     добавляем результаты в таблицу
    result_list.append(model_name)
    result_list.append(best_params)
    result_list.append(encoder)
    result_list.append(best_score)
    result_list.append(accuracy)
#     добавляем таблицу в общую таблицу
    final_list.append(result_list)
    print(result_list)
    if i_am_cheater:
        print(f"\nROC-AUC на тесте: {(roc_auc_score(y_test, (best_model[1].predict_
    return X_test, y_test, best_model
```

Дерево Решений

In [47]:
```
tree = DecisionTreeClassifier(random_state=281122, class_weight="balanced")
tree_hyper = {"model__max_depth" : range(1, 20), "model__min_samples_leaf" : range
```

In [48]:
```
%%time
X_test, y_test, best_ord_tree = everything_maker("Decision Tree", tree, tree_hyper
X_test, y_test, best_ohe_tree = everything_maker("Decision Tree", tree, tree_hyper
```

```
['Decision Tree', "{'model__max_depth': 6, 'model__min_samples_leaf': 48}", 'ordin
al', 0.8189, 0.744]
['Decision Tree', "{'model__max_depth': 8, 'model__min_samples_leaf': 59}", 'ohe',
0.8258, 0.749]
CPU times: total: 6.84 s
Wall time: 22 s
```

Случайный Лес

In [49]:
```
forest = RandomForestClassifier(random_state=281122, class_weight="balanced")
forest_hyper = {"model__max_depth": range(1, 15), "model__n_estimators": range(60,
```

In [50]:
```
%%time
X_test, y_test, best_ord_forest = everything_maker("Random Forest", forest, forest_
X_test, y_test, best_ohe_forest = everything_maker("Random Forest", forest, forest_
```

```
['Random Forest', "{'model__max_depth': 7, 'model__n_estimators': 67}", 'ordinal',
0.8368, 0.7772]
['Random Forest', "{'model__max_depth': 7, 'model__n_estimators': 62}", 'ohe', 0.8
378, 0.7732]
CPU times: total: 8.38 s
Wall time: 1min 7s
```

LightGBM

In [51]:
```
lgbm = LGBMClassifier(random_state=281122, silent=True, class_weight="balanced")
lgbm_hyper = {"model__max_depth": range(1, 10), "model__n_estimators": range(25, 2(
```

In [52]:
```
%%time
X_test, y_test, best_ord_lgbm = everything_maker("LightGBM", lgbm, lgbm_hyper, fea
X_test, y_test, best_ohe_lgbm = everything_maker("LightGBM", lgbm, lgbm_hyper, fea
```

```
['LightGBM', "{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estim
ators': 100}", 'ordinal', 0.8388, 0.7463]
['LightGBM', "{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estim
ators': 100}", 'ohe', 0.8404, 0.7471]
CPU times: total: 9.06 s
Wall time: 37.7 s
```

CatBoost

In [53]:
```
cat = CatBoostClassifier(random_state=281122, silent=True, auto_class_weights="Bal
cat_hyper = {"model__max_depth": range(1, 5), "model__iterations": range(156, 190,
```

In [54]:
```
%%time
X_test, y_test, best_ord_cat = everything_maker("CatBoost", cat, cat_hyper, featur
X_test, y_test, best_ohe_cat = everything_maker("CatBoost", cat, cat_hyper, featur
```

```
['CatBoost', "{'model__iterations': 171, 'model__learning_rate': 0.05, 'model__max
_depth': 4}", 'ordinal', 0.8415, 0.756]
['CatBoost', "{'model__iterations': 177, 'model__learning_rate': 0.07, 'model__max
_depth': 2}", 'ohe', 0.8418, 0.7418]
CPU times: total: 21.8 s
Wall time: 4min 25s
```

Попробуем произвести кроссвалидацию моделей градиентного бустинга с их стандартными встроенными кодировщиками

CatBoost (встроенный кодировщик)

In [55]:
```
%%time

X_train, X_test, y_train, y_test = train_test_split(features,
                                                    target,
                                                    test_size=0.25,
                                                    random_state=281122,
                                                    stratify=(target))

cat_cat_feat = CatBoostClassifier(random_state=281122, silent=True, auto_class_wei
catboost_hyper = {"max_depth": range(1, 5), "iterations": range(162, 170), "learni
best = []

grid_search = GridSearchCV(cat_cat_feat, catboost_hyper, cv=5, scoring="roc_auc",
catboost = grid_search.fit(X_train, y_train)
best_params = str(catboost.best_params_)                    # параметры лучшей мо
best_score = catboost.best_score_.round(4)                  # значение ROC-AUC дл
best_catboost = catboost.best_estimator_                       # лучшая модель
best_catboost.fit(X_train, y_train)                         # обучаем модель
accuracy = best_catboost.score(X_train, y_train).round(4)    # добавляем к списку
#     добавляем результаты в таблицу
best.append("CatBoost")
best.append(best_params)
best.append("cat_features")
best.append(best_score)
best.append(accuracy)
#     добавляем таблицу в общуютаблицу
final_list.append(best)
print(best)
```

```
['CatBoost', "{'iterations': 165, 'learning_rate': 0.08, 'max_depth': 3}", 'cat_fe
atures', 0.8413, 0.7527]
CPU times: total: 14.2 s
Wall time: 4min 24s
```

LightGBM (встроенный кодировщик)

In [56]:
```
lgbm_X_train = X_train.copy()
lgbm_X_test = X_test.copy()
for col in lgbm_X_train[categorical]:
    lgbm_X_train[col] = lgbm_X_train[col].astype('category')
for col in lgbm_X_test[categorical]:
    lgbm_X_test[col] = lgbm_X_test[col].astype('category')
lgbm_X_train.info()
lgbm_X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5282 entries, 4215 to 1800
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   gender             5282 non-null   category
 1   senior_citizen     5282 non-null   category
 2   partner            5282 non-null   category
 3   dependents         5282 non-null   category
 4   payment_type       5282 non-null   category
 5   paperless_billing  5282 non-null   category
 6   payment_method     5282 non-null   category
 7   multiple_lines     5282 non-null   category
 8   internet_service   5282 non-null   category
 9   online_security    5282 non-null   category
 10  online_backup      5282 non-null   category
 11  device_protection  5282 non-null   category
 12  tech_support       5282 non-null   category
 13  streaming_tv       5282 non-null   category
 14  streaming_movies   5282 non-null   category
 15  internet           5282 non-null   category
 16  monthly_charges    5282 non-null   float64 
 17  total_charges      5282 non-null   float64 
 18  dur_months         5282 non-null   int64   
 19  dur_years          5282 non-null   int64   
 20  num_of_services    5282 non-null   int64   
dtypes: category(16), float64(2), int64(3)
memory usage: 332.2 KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1761 entries, 1558 to 5527
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype   
---  ------             --------------  -----   
 0   gender             1761 non-null   category
 1   senior_citizen     1761 non-null   category
 2   partner            1761 non-null   category
 3   dependents         1761 non-null   category
 4   payment_type       1761 non-null   category
 5   paperless_billing  1761 non-null   category
 6   payment_method     1761 non-null   category
 7   multiple_lines     1761 non-null   category
 8   internet_service   1761 non-null   category
 9   online_security    1761 non-null   category
 10  online_backup      1761 non-null   category
 11  device_protection  1761 non-null   category
 12  tech_support       1761 non-null   category
 13  streaming_tv       1761 non-null   category
 14  streaming_movies   1761 non-null   category
 15  internet           1761 non-null   category
 16  monthly_charges    1761 non-null   float64 
 17  total_charges      1761 non-null   float64 
 18  dur_months         1761 non-null   int64   
 19  dur_years          1761 non-null   int64   
 20  num_of_services    1761 non-null   int64   
dtypes: category(16), float64(2), int64(3)
memory usage: 112.1 KB
```

In [57]:

```
%%time
best = []
lgbm = LGBMClassifier(random_state=281122, silent=True, class_weight="balanced")
lightgbm_hyper = {"max_depth": range(1, 6), "n_estimators": range(25, 400, 25), "l

grid_search = GridSearchCV(lgbm, lightgbm_hyper, cv=5, scoring="roc_auc", n_jobs=-
lightgbm = grid_search.fit(lgbm_X_train, y_train)
```

```
best_params = str(lightgbm.best_params_)                          # параметры лучшей м
best_score = lightgbm.best_score_.round(4)                        # значение ROC-AUC д
best_lgbm = lightgbm.best_estimator_                              # лучшая модель
best_lgbm.fit(lgbm_X_train, y_train)                              # обучаем модель
accuracy = best_lgbm.score(lgbm_X_train, y_train).round(4)        # добавляем к списку
#     добавляем результаты в таблицу
best.append("LightGBM")
best.append(best_params)
best.append("category_dtype")
best.append(best_score)
best.append(accuracy)
#     добавляем таблицу в общую таблицу
final_list.append(best)
print(best)
```

```
['LightGBM', "{'learning_rate': 0.06, 'max_depth': 2, 'n_estimators': 150}", 'cate
gory_dtype', 0.84, 0.7461]
CPU times: total: 4.19 s
Wall time: 34.6 s
```

Логистическая Регрессия

In [58]:
```
log_reg = LogisticRegression(random_state=281122, class_weight="balanced")
lr_hyper = {"model__penalty": ['l1', 'l2', 'elasticnet'],
                "model__tol": [0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001],
                "model__max_iter": [10, 25, 50, 100, 200],
                "model__solver": ["liblinear"]}
```

In [59]:
```
%%time
X_test, y_test, best_ohe_log_reg = everything_maker("Logistic Regression", log_reg
```

```
['Logistic Regression', "{'model__max_iter': 25, 'model__penalty': 'l1', 'model__s
olver': 'liblinear', 'model__tol': 1e-06}", 'ohe', 0.834, 0.7416]
CPU times: total: 2.77 s
Wall time: 6.93 s
```

In [60]:
```
# финальная таблица с характеристиками лучших моделей после кроссвалидации
finl = pd.DataFrame(final_list, columns=("model_name", "model_hyperparameters", "e

finl.sort_values("roc_auc_score", ascending=False)
```

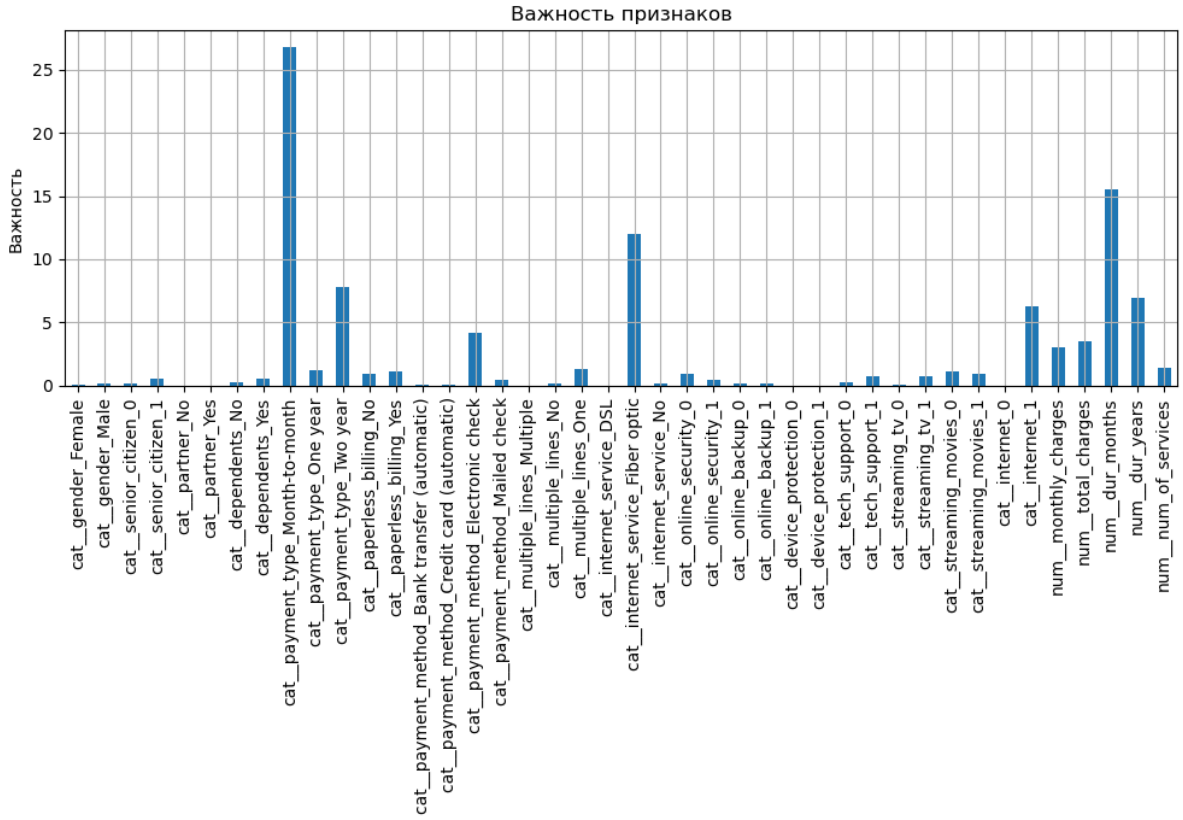
Out[60]:

	model_name	model_hyperparameters	encoder	roc_auc_score	train_accuracy
7	CatBoost	{'model__iterations': 177, 'model__learning_rate': 0.07, 'model__max_depth': 2}	ohe	0.8418	0.7418
6	CatBoost	{'model__iterations': 171, 'model__learning_rate': 0.05, 'model__max_depth': 4}	ordinal	0.8415	0.7560
8	CatBoost	{'iterations': 165, 'learning_rate': 0.08, 'max_depth': 3}	cat_features	0.8413	0.7527
5	LightGBM	{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estimators': 100}	ohe	0.8404	0.7471
9	LightGBM	{'learning_rate': 0.06, 'max_depth': 2, 'n_estimators': 150}	category_dtype	0.8400	0.7461
4	LightGBM	{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estimators': 100}	ordinal	0.8388	0.7463
3	Random Forest	{'model__max_depth': 7, 'model__n_estimators': 62}	ohe	0.8378	0.7732
2	Random Forest	{'model__max_depth': 7, 'model__n_estimators': 67}	ordinal	0.8368	0.7772
10	Logistic Regression	{'model__max_iter': 25, 'model__penalty': 'l1', 'model__solver': 'liblinear', 'model__tol': 1e-06}	ohe	0.8340	0.7416
1	Decision Tree	{'model__max_depth': 8, 'model__min_samples_leaf': 59}	ohe	0.8258	0.7490
0	Decision Tree	{'model__max_depth': 6, 'model__min_samples_leaf': 48}	ordinal	0.8189	0.7440

лучшие результаты у катбуста с обоими типами кодирования категориальных признаков (OHE и Ordinal) результаты почти идентичны. С Ordinal немного выше значение accuracy. Будем считать эту модель лучшей в виду незначительности отрыва модели закодированной OHE. Посмотрим на важность вклада каждого признака в предсказания двух лучших моделей

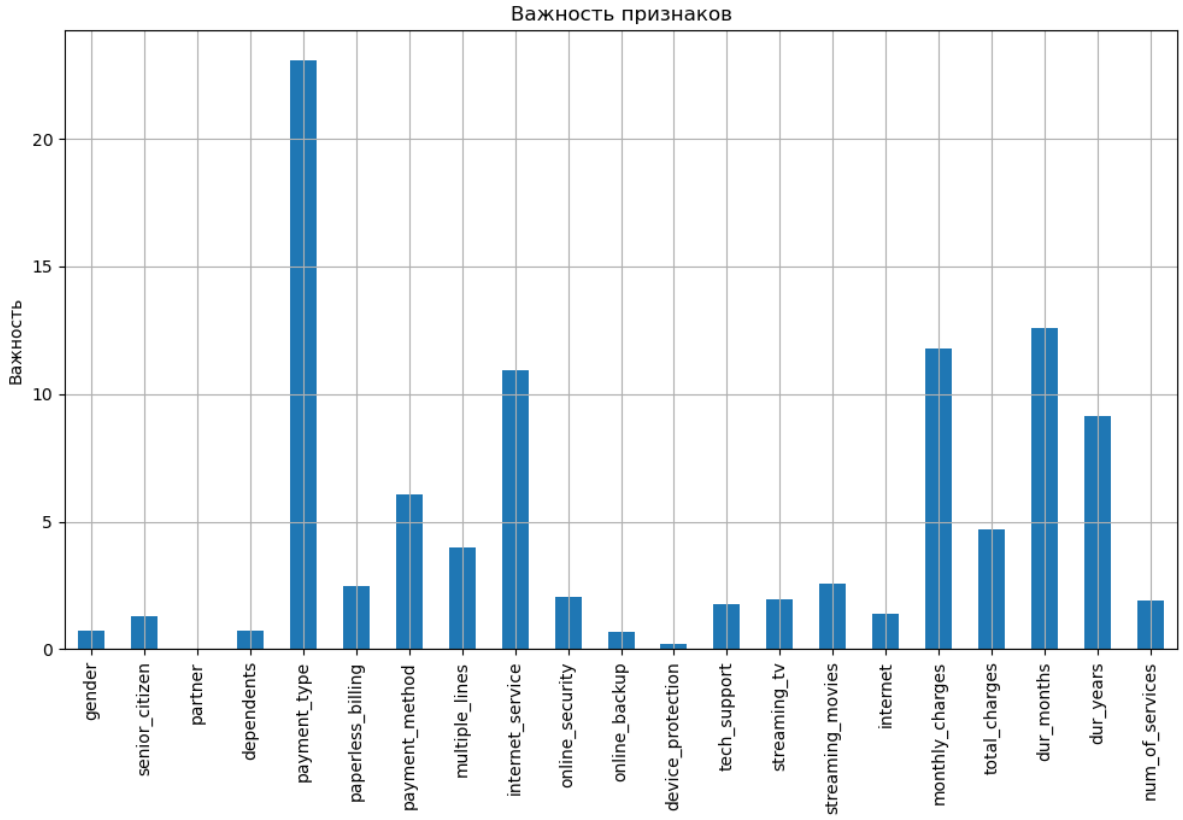
In [61]:
```
# смотрим на важность признаков с кодированием OHE
importance = pd.Series(best_ohe_cat[1].feature_importances_,
                       best_ohe_cat[0].get_feature_names_out())

fig, ax = plt.subplots(figsize=(10,7))
importance.plot.bar(ax=ax)
ax.set_title("Важность признаков")
ax.set_ylabel('Важность')
fig.tight_layout()
plt.grid()
plt.show()
```



In [62]:
```
# смотрим на важность признаков с кодированием Ordinal
importance = pd.Series(best_ord_cat[1].feature_importances_,
                       features.columns)

fig, ax = plt.subplots(figsize=(10,7))
importance.plot.bar(ax=ax)
ax.set_title("Важность признаков")
ax.set_ylabel('Важность')
fig.tight_layout()
plt.grid()
plt.show()
```



наибольший вклад в предсказания внесли признаки связанные с длительностью пользования услугами, типом оплаты, типом интернет подключения и месячной стоимостью услуг. попробуем удалить признаки, внесшие наименьший вклад, и посмотрим на качество модели на валидации

In [63]:
```
categorical = [
    "senior_citizen",
    "payment_type",
    "paperless_billing",
    "payment_method",
    "multiple_lines",
    "internet_service",
    "online_security",
    "tech_support",
    "streaming_tv",
    "streaming_movies",
    "internet"
]
numeric = [
    'monthly_charges',
    'total_charges',
    'dur_months',
    'dur_years',
    'num_of_services'
]
droped_features=features[categorical+numeric]
```

CatBoost без части признаков

In [64]:
```
%%time
drop_X_test, drop_y_test, best_droped_ord_cat = everything_maker("CatBoost", cat, 
drop_X_test, drop_y_test, best_droped_ohe_cat = everything_maker("CatBoost", cat, 
```

```
['CatBoost', "{'model__iterations': 186, 'model__learning_rate': 0.04, 'model__max
_depth': 4}", 'ordinal', 0.8412, 0.7438]
['CatBoost', "{'model__iterations': 177, 'model__learning_rate': 0.06, 'model__max
_depth': 2}", 'ohe', 0.8414, 0.7378]
CPU times: total: 18.5 s
Wall time: 3min 27s
```

метрики остались практически без изменений. время работы так же сократилось незначительно. оставим первоначальный вариант

Тестирование лучшей модели

In [65]:
```
# кодируем и сравниваем тестовую выборку до и после кодирования
display(X_test.head())
X_testt = best_ord_cat[0].transform(X_test)
pd.DataFrame(X_testt, columns=features.columns).head()
```

	gender	senior_citizen	partner	dependents	payment_type	paperless_billing	payment_metl
1558	Female	0	No	No	Month-to-month	Yes	Credit c (automa
3020	Male	0	No	Yes	One year	No	Credit c (automa
3845	Male	0	No	No	Month-to-month	Yes	Electronic ch
423	Male	0	Yes	Yes	One year	No	Mailed ch
95	Female	0	No	No	Month-to-month	Yes	Electronic ch

Out[65]:

	gender	senior_citizen	partner	dependents	payment_type	paperless_billing	payment_method
0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
1	1.0	0.0	0.0	1.0	1.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0	1.0	2.0
3	1.0	0.0	1.0	1.0	1.0	0.0	3.0
4	0.0	0.0	0.0	0.0	0.0	1.0	2.0

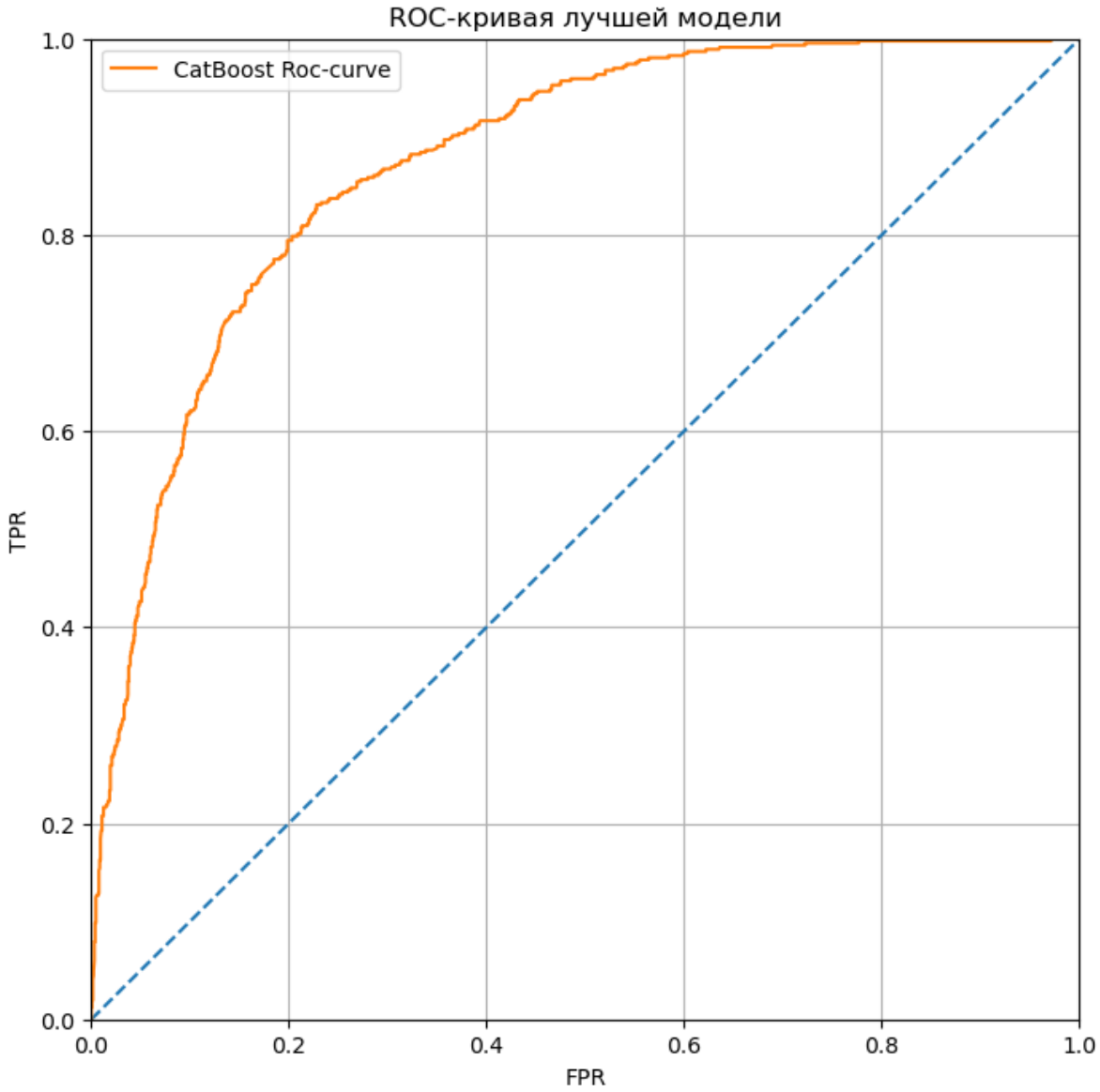
кодирование прошло корректно

In [74]:
```
# вероятности предсказания положительного класса для каждого значения целевого при
pos_pred = best_ord_cat[1].predict_proba(X_testt)[:, 1]
pos_pred
# Выводим значения ROC-AUC
print('ROC-AUC лучшей модели на тестовой выборке:', roc_auc_score(y_test, pos_pred
```

```
ROC-AUC лучшей модели на тестовой выборке: 0.8733
```

In [68]:
```
# визуализируем ROC-кривую
plt.figure(figsize=(8, 8))
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.title('ROC-кривая лучшей модели')
plt.xlabel('FPR')
```

```
plt.ylabel('TPR')
plt.grid()
plt.plot([0, 1], [0, 1], linestyle='--')
fpr, tpr, thresholds = roc_curve(y_test, pos_pred)
plt.plot(fpr, tpr, label='CatBoost Roc-curve')
plt.legend()
plt.show()
```



Посмотрим на значения точности, полноты и Ф1-меры на разных порогах классификации

In [69]:
```
for threshold in np.arange(0, 1, 0.1):
    predicted = pos_pred > threshold
    precision = precision_score(y_test, predicted)
    recall = recall_score(y_test, predicted)
    f1 = f1_score(y_test, predicted)


    print("threshold = {:.2f} | precision = {:.3f}, recall = {:.3f}, F1 = {:.3f}".
        threshold, precision, recall, f1))
```

```
threshold = 0.00 | precision = 0.265, recall = 1.000, F1 = 0.419
threshold = 0.10 | precision = 0.345, recall = 0.991, F1 = 0.512
threshold = 0.20 | precision = 0.392, recall = 0.974, F1 = 0.559
threshold = 0.30 | precision = 0.437, recall = 0.938, F1 = 0.596
threshold = 0.40 | precision = 0.478, recall = 0.891, F1 = 0.622
threshold = 0.50 | precision = 0.547, recall = 0.839, F1 = 0.663
threshold = 0.60 | precision = 0.619, recall = 0.752, F1 = 0.679
threshold = 0.70 | precision = 0.697, recall = 0.591, F1 = 0.640
threshold = 0.80 | precision = 0.770, recall = 0.351, F1 = 0.482
threshold = 0.90 | precision = 0.886, recall = 0.084, F1 = 0.153
```

лучшее значение Ф1 дает порог 0.6, что странно учитывая балансировку весов классов при обучении модели

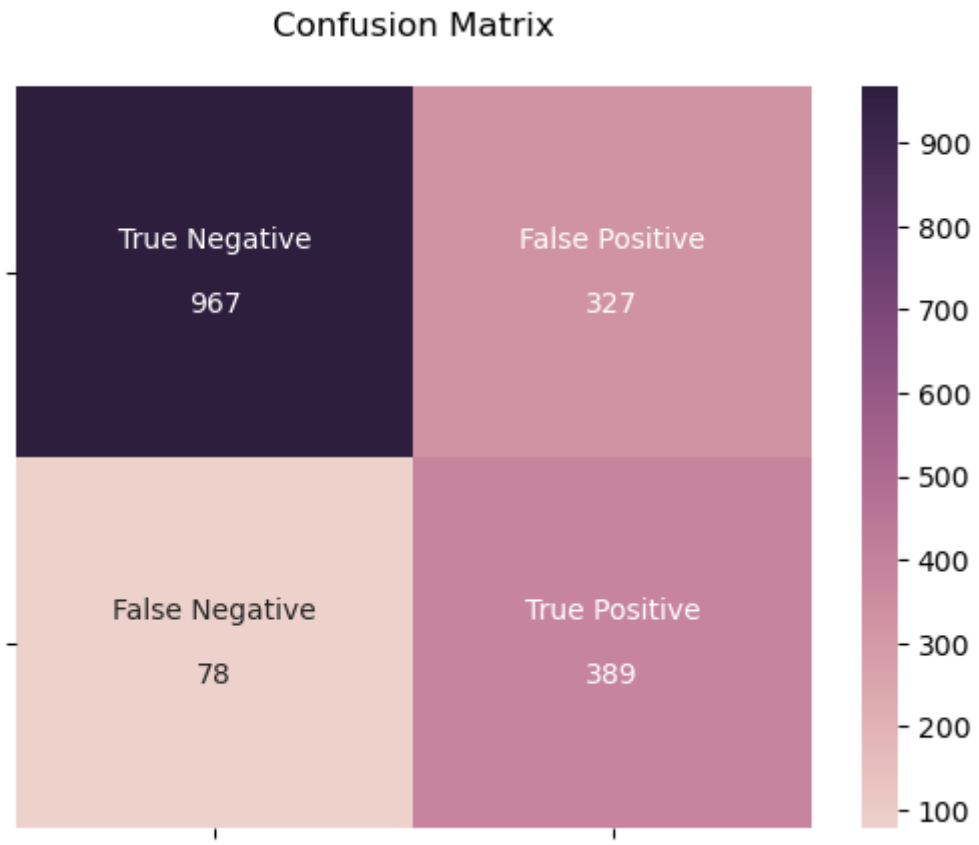
In [70]:
```
predictions = best_ord_cat[1].predict(X_testt)
acc = accuracy_score(y_test, predictions)
print(f"Значение accuracy лучшей модели на тестовой выборке: {acc}")
```

```
Значение accuracy лучшей модели на тестовой выборке: 0.7700170357751278
```

In [71]:
```
# построим матрицу ошибок
cm = confusion_matrix(y_test, predictions)
tn, fp, fn, tp = confusion_matrix(y_test, predictions).ravel()
tn, fp, fn, tp
```

Out[71]:
```
(967, 327, 78, 389)
```

In [72]:
```
# визуализируем
cm_named = np.core.defchararray.add(np.array([["True Negative\n\n", "False Positive
                                               ["False Negative\n\n", "True Positive
fig, ax = plt.subplots()
ax = sns.heatmap(cm, annot=cm_named, fmt="", cmap=sns.cubehelix_palette(as_cmap=Tru
ax.set(xticklabels=[])
ax.set(yticklabels=[])
ax.set_title("Confusion Matrix\n")
plt.show()
```



ВЫВОД

Получилась очень красивая матрица ошибок

Отчет

Отток клиентов телеком компании

Цель исследования:

Изучить информацию о пользователях телеком компании. Построить модель для прогноза оттока клиентов. Разобраться в факторах и причинах прекращения пользования услугами компании.

Знакомство с данными

В нашем распоряжении было 4 таблицы:

- С персональными данными клиентов (7043 строки),
- с характеристиками их контрактов (7043 строки),
- с характеристиками интернет подключения и подключенных услуг (5517 строк),
- с информацией о подключении услуги нескольких телефонных линий (6361 строка)

Проверка уникальных ID в таблицах с с большим количеством строк подтвердила, что они совпадают, а в остальных таблицах отсутствие данных связано с отсутствием у того или иного клиента телефонного или интернет подключения. Пропуски и дубликаты не были обнаружены.

Анализ и предобработка

- Данные 4 таблиц были объединены в одну путем присоединения меньших к большим без потери данных, в результате образовались пропуски.
- Из столбца с датой ухода end_date был сгенерирован целевой признак $exited$
- Встолбцах $begin\_date$ и end_date данные были приведены к типу $datetime$ для того, чтобы сгенерировать признаки длительности пользования услугами
- Так же был изменен тип данных в столбце с общими тратами клиентов $total\_charges$, в результате чего обнаружились пропуски в этом столбце, связянные с тем, что клиенты подключились в текущем месяце, они были заполнены данными из столбца с суммой помесячной платы $monthly\_charges$
- Далее были сгенерированы признаки длительности пользования услугами в полных годах и в месяцах dur_years и dur_months
- Пропуски в столбцах с бинарными признаками подключенных интернет услуг $online\_security$, $online\_backup$, $device\_protection$, $tech\_support$, $streaming\_tv$, $streaming\_movies$ были заполнены значением " No " и приведены к целочисленному типу данных
- Пропуски в столбце с типом интернет подключения $internet\_service$ были заполнены новым уникальным значением означающим отсутствие подключения " No "
- В столбце с информацией о наличии нескольких телефонных линий были полностью заменены значения на " $Multiple$ ", " One ", и " No ", таким образом образовалось 3 группы: несколько линий, одна линия и отсутствие подключения.
- Затем исходя из предположения, что наличие/отсутствие интернета может играть значительно большую роль нежели тип подключения был создан бинарный признак $internet$ с информацией о наличии/отсутствии интернет подключения
- Так же был создан признак с общим рейтингом активности использования услуг интернета и телефонии $num\_of\_services$
- Далее были удалены не несущий предсказательного смысла признак $customer\_id$ и признаки способные спровоцировать утечку целевого признака $begin\_date$ и end_date
- В завершении этапа иссследовательского анализа были проанализированы распределения признаков и распределения в разбивке по классам целевого признака, так же проанализирована нелинейная корреляция с целевым признаком при помощи библиотеки $Phik$

Подбор модели

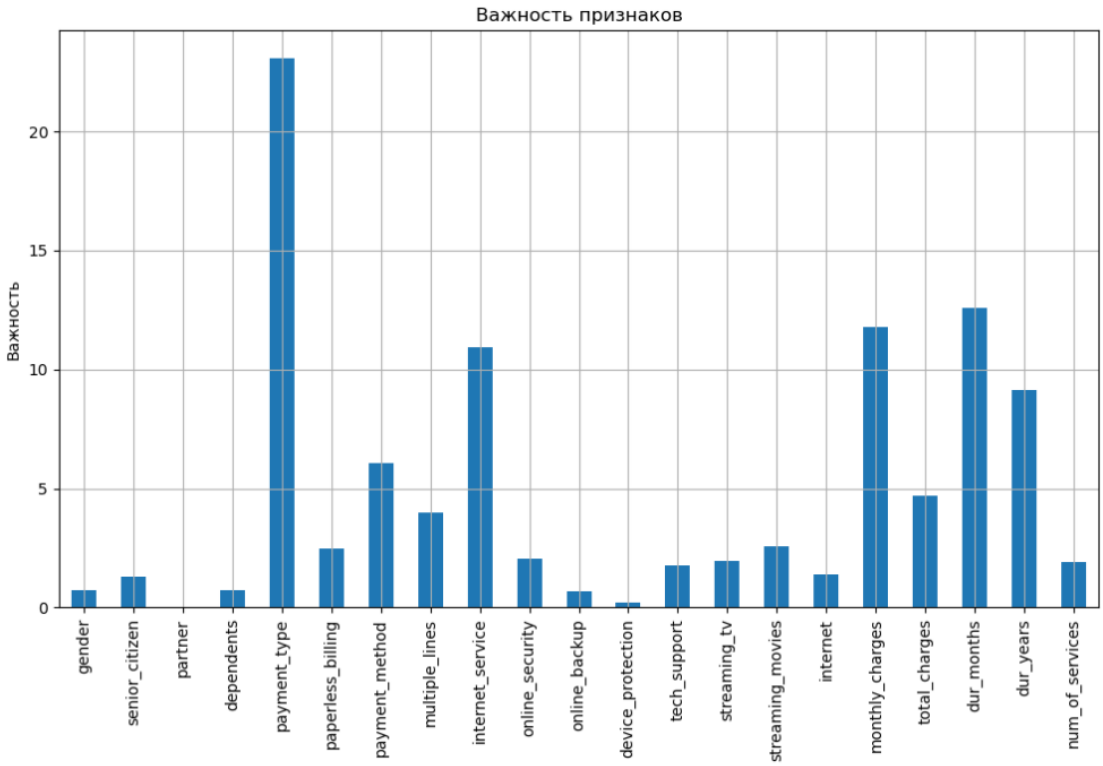
- Была выполнена кросс-валидация с подбором гиперпараметров на выборке из всех имеющихся признаков. Категориальные признаки были закодированы двумя способами прямым с помощью $OneHotEncoder$ и порядковым кодированием с

помощью $OrdinalEncoder$, количественные признаки были масштабированы с помощью $StandardScaler$, в связи с большой разницей в диапазонах величин

- Кроссвалидация выполнялась для 5 моделей $DecisionTreeClassifier$, $RandomForestClassifier$, $LightGBMClassifier$, $CatBoostClassifier$ и $LogisticRegression$.
- Лучшие результаты показала модель $CatBoostClassifier$ с порядковой кодировкой и гиперпараметрами: ***'iterations': 171, 'learning_rate': 0.05, 'max_depth': 4'***

	model_name	model_hyperparameters	encoder	roc_auc_score	train_accuracy
7	CatBoost	{'model__iterations': 177, 'model__learning_rate': 0.07, 'model__max_depth': 2}	ohe	0.8418	0.7418
6	CatBoost	{'model__iterations': 171, 'model__learning_rate': 0.05, 'model__max_depth': 4}	ordinal	0.8415	0.7560
8	CatBoost	{'iterations': 165, 'learning_rate': 0.08, 'max_depth': 3}	cat_features	0.8413	0.7527
5	LightGBM	{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estimators': 100}	ohe	0.8404	0.7471
9	LightGBM	{'learning_rate': 0.06, 'max_depth': 2, 'n_estimators': 150}	category_dtype	0.8400	0.7461
4	LightGBM	{'model__learning_rate': 0.1, 'model__max_depth': 2, 'model__n_estimators': 100}	ordinal	0.8388	0.7463
3	Random Forest	{'model__max_depth': 7, 'model__n_estimators': 62}	ohe	0.8378	0.7732
2	Random Forest	{'model__max_depth': 7, 'model__n_estimators': 67}	ordinal	0.8368	0.7772
10	Logistic Regression	{'model__max_iter': 25, 'model__penalty': 'l1', 'model__solver': 'liblinear', 'model__tol': 1e-06}	ohe	0.8340	0.7416
1	Decision Tree	{'model__max_depth': 8, 'model__min_samples_leaf': 59}	ohe	0.8258	0.7490
0	Decision Tree	{'model__max_depth': 6, 'model__min_samples_leaf': 48}	ordinal	0.8189	0.7440

- Для лучшей модели был произведен анализ важности вклада признаков в предсказание



- Затем в целях оптимизации были удалены малозначимые признаки и проведена контрольная кросс-валидация. В результате показатель целевой метрики незначительно упал, а время обучения сократилось так же незначительно и было принято решение оставить все признаки

Проверка лучшей модели

- Была выполнена проверка лучшей модели на тестовой выборке. Итоговое значение целевой метрики $ROC-AUC:\ 0.8733$, значение точности ($accuracy$) $:\ 0.77$

- В конечную версию проекта не вошел этап кросс-валидации с ресемплингом с помощью $SMOTENC$. Это связано с тем, что он не имеет метода transform, в связи с чем не подходит для пайплайна(если конечно я правильно понял мысль, которую юпитер пытался до меня донести). Поэтому ресемплинг был сделан перед пайплайном, что дало переобучение на тренировочных данных и как результат более низкий скор на тестовых

Итоговая модель:

```
CatBoostClassifier(random_state=281122, silent=True,
                   auto_class_weights="Balanced",
                   max_depth=2,
                   learning_rate=0.1,
                   iteraations=212)
```

```
ROC-AUC: 0.8733
```