# Описание

В данном исследовании нам предоставленны данные о клиентах фитнес-клуба. В предоставленныхданных указан факт прекращения клиентом использования услуг фитнес-клуба.

Нам необходимо выявить закономерности в данных, которые могут указать на причины прекращения клиентом посещения, и подготовить план действий по удержанию клиентов.

Описание данных «Культурист-датасаентист» предоставил сведения в csv-файлах. Заказчик подготовил данные, которые содержат данные на месяц до оттока и факт оттока на определённый месяц. Набор данных включает следующие поля:

Churn — факт оттока в текущем месяце; Данные клиента за предыдущий до проверки факта оттока месяц:

gender — пол; Near_Location — проживание или работа в районе, где находится фитнес-центр; Partner — сотрудник компании-партнёра клуба (сотрудничество с компаниями, чьи сотрудники могут получать скидки на абонемент — в таком случае фитнес-центр хранит информацию о работодателе клиента); Promo_friends — факт первоначальной записи в рамках акции «приведи друга» (использовал промо-код от знакомого при оплате первого абонемента); Phone — наличие контактного телефона; Age — возраст; Lifetime — время с момента первого обращения в фитнес-центр (в месяцах). Информация на основе журнала посещений, покупок и информация о текущем статусе абонемента клиента:

Contract_period — длительность текущего действующего абонемента (месяц, 3 месяца, 6 месяцев, год); Month_to_end_contract — срок до окончания текущего действующего абонемента (в месяцах); Group_visits — факт посещения групповых занятий; Avg_class_frequency_total — средняя частота посещений в неделю за все время с начала действия абонемента; Avg_class_frequency_current_month — средняя частота посещений в неделю за предыдущий месяц; Avg_additional_charges_total — суммарная выручка от других услуг фитнес-центра: кафе, спорт-товары, косметический и массажный салон.

# Импорт и анализ

### импорт

Ввод [1]:
```
import pandas as pd

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import roc_curve


import seaborn as sns
import matplotlib.pyplot as plt

import numpy as np
import datetime as dt
import plotly.express as px
from plotly import graph_objects as go
from plotly.subplots import make_subplots
```

Ввод [2]:
```
df=pd.read_csv('D:\download\проект\gym_churn.csv')
```

### анализ данных

Ввод [3]:
```
display(df.sample(10))
```

| | gender | Near_Location | Partner | Promo_friends | Phone | Contract_period | Group_visits | Age | Avg_additional_charges_total | Month_to_end_contract | Lifetime |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **2329** | 0 | 1 | 1 | 1 | 1 | 6 | 0 | 33 | 223.685108 | 6.0 | 4 |
| **3789** | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 29 | 48.345310 | 1.0 | 3 |
| **3658** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 24 | 2.039813 | 1.0 | 0 |
| **3605** | 1 | 1 | 0 | 0 | 1 | 6 | 0 | 25 | 155.134007 | 6.0 | 3 |
| **3688** | 0 | 1 | 1 | 0 | 1 | 6 | 1 | 27 | 35.425845 | 6.0 | 2 |
| **1734** | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 27 | 23.896375 | 1.0 | 1 |
| **455** | 1 | 1 | 1 | 1 | 1 | 6 | 1 | 31 | 124.321476 | 6.0 | 2 |
| **1564** | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 29 | 32.922315 | 1.0 | 1 |
| **3874** | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 29 | 2.373369 | 1.0 | 1 |
| **3705** | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 29 | 60.885977 | 1.0 | 5 |

Ввод [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   gender                             4000 non-null   int64  
 1   Near_Location                      4000 non-null   int64  
 2   Partner                            4000 non-null   int64  
 3   Promo_friends                      4000 non-null   int64  
 4   Phone                              4000 non-null   int64  
 5   Contract_period                    4000 non-null   int64  
 6   Group_visits                       4000 non-null   int64  
 7   Age                                4000 non-null   int64  
 8   Avg_additional_charges_total       4000 non-null   float64
 9   Month_to_end_contract              4000 non-null   float64
 10  Lifetime                           4000 non-null   int64  
 11  Avg_class_frequency_total          4000 non-null   float64
 12  Avg_class_frequency_current_month  4000 non-null   float64
 13  Churn                              4000 non-null   int64  
dtypes: float64(4), int64(10)
memory usage: 437.6 KB
```

В предоставленных данных имеется 4000 строк, пустых значений нет. Имена столбцов можно было-бы привести к нижнему регистру, но если это не принципиальное требование, то я оставлю так. Мне это не мещает. Столбцы Avg_class_frequency_total, Avg_class_frequency_current_month и Month_to_end_contract можно привести к целому типу данных, т.к. количество месяцев и количество посещений как-бы целые числа, но поскольку это среднее количество посещений, то оно может быть дробным. Оставлю как есть. Проверим на полные дубли

Ввод [5]:
```
print(df.duplicated().sum())
```

```
0
```

Дубликатов нет. Посмотрим на минимальные и максимальные значения. Столбцы, в которых данные принимают значения только 0 или 1 я убрал.

Ввод [6]:
```
df[['Age','Avg_additional_charges_total','Month_to_end_contract','Lifetime','Avg_class_frequency_total','Avg_class_frequency_current
```

Out[6]:

| | Avg_additional_charges_total | Lifetime | Avg_class_frequency_current_month | Avg_class_frequency_total | Month_to_end_contract | Age |
|---|---|---|---|---|---|---|
| **count** | 4000.000000 | 4000.000000 | 4000.000000 | 4000.000000 | 4000.000000 | 4000.000000 |
| **mean** | 146.943728 | 3.724750 | 1.767052 | 1.879020 | 4.322750 | 29.184250 |
| **std** | 96.355602 | 3.749267 | 1.052906 | 0.972245 | 4.191297 | 3.258367 |
| **min** | 0.148205 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 18.000000 |
| **25%** | 68.868830 | 1.000000 | 0.963003 | 1.180875 | 1.000000 | 27.000000 |
| **50%** | 136.220159 | 3.000000 | 1.719574 | 1.832768 | 1.000000 | 29.000000 |
| **75%** | 210.949625 | 5.000000 | 2.510336 | 2.536078 | 6.000000 | 31.000000 |
| **max** | 552.590740 | 31.000000 | 6.146783 | 6.023668 | 12.000000 | 41.000000 |

Все значения в рамках разумного, выбросов нет. Посмотрим таким же образом по отдельности на ушедших и оставшихся клиентов.

Ввод [7]:
```
df_ch0=df[df.Churn==0]
df_ch1=df[df.Churn==1]
print(len(df_ch0))
print(len(df_ch1))
```

```
2939
1061
```

Ввод [8]:
```
df_ch0[['Age','Avg_additional_charges_total','Month_to_end_contract','Lifetime','Avg_class_frequency_total','Avg_class_frequency_cur
```

Out[8]:

| | Avg_additional_charges_total | Lifetime | Avg_class_frequency_current_month | Avg_class_frequency_total | Month_to_end_contract | Age |
|---|---|---|---|---|---|---|
| **count** | 2939.000000 | 2939.000000 | 2939.000000 | 2939.000000 | 2939.000000 | 2939.000000 |
| **mean** | 158.445715 | 4.711807 | 2.027882 | 2.024876 | 5.283089 | 29.976523 |
| **std** | 99.801599 | 3.874780 | 1.018994 | 1.016006 | 4.363522 | 3.009933 |
| **min** | 0.171862 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 19.000000 |
| **25%** | 76.920993 | 2.000000 | 1.297021 | 1.283137 | 1.000000 | 28.000000 |
| **50%** | 149.881171 | 4.000000 | 2.046697 | 2.043252 | 6.000000 | 30.000000 |
| **75%** | 224.448274 | 6.000000 | 2.740648 | 2.732944 | 10.000000 | 32.000000 |
| **max** | 552.590740 | 31.000000 | 6.146783 | 6.023668 | 12.000000 | 41.000000 |

Ввод [9]:
```
df_ch1[['Age','Avg_additional_charges_total','Month_to_end_contract','Lifetime','Avg_class_frequency_total','Avg_class_frequency_cur
```

Out[9]:

| | Avg_additional_charges_total | Lifetime | Avg_class_frequency_current_month | Avg_class_frequency_total | Month_to_end_contract | Age |
|---|---|---|---|---|---|---|
| **count** | 1061.000000 | 1061.000000 | 1061.000000 | 1061.000000 | 1061.000000 | 1061.000000 |
| **mean** | 115.082899 | 0.990575 | 1.044546 | 1.474995 | 1.662582 | 26.989632 |
| **std** | 77.696419 | 1.110799 | 0.770237 | 0.694705 | 1.964593 | 2.895163 |
| **min** | 0.148205 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 18.000000 |
| **25%** | 50.629127 | 0.000000 | 0.421337 | 1.010771 | 1.000000 | 25.000000 |
| **50%** | 103.814686 | 1.000000 | 0.979445 | 1.491187 | 1.000000 | 27.000000 |
| **75%** | 165.616858 | 1.000000 | 1.588576 | 1.956438 | 1.000000 | 29.000000 |
| **max** | 425.535220 | 9.000000 | 3.540271 | 3.478646 | 12.000000 | 38.000000 |

Ввод [10]:
```
#df.groupby('Churn').agg({'mean', 'std', 'var'}).T
```

Всего посетителей 4000. Отметка о прекращении у 1061, примерно 30%. Можно отметить, что ушедшие клиенты посещали занятия примерно в 2 раза реже, чем оставшиеся. И до окончания контракты у ушедших в основном 1 месяц.

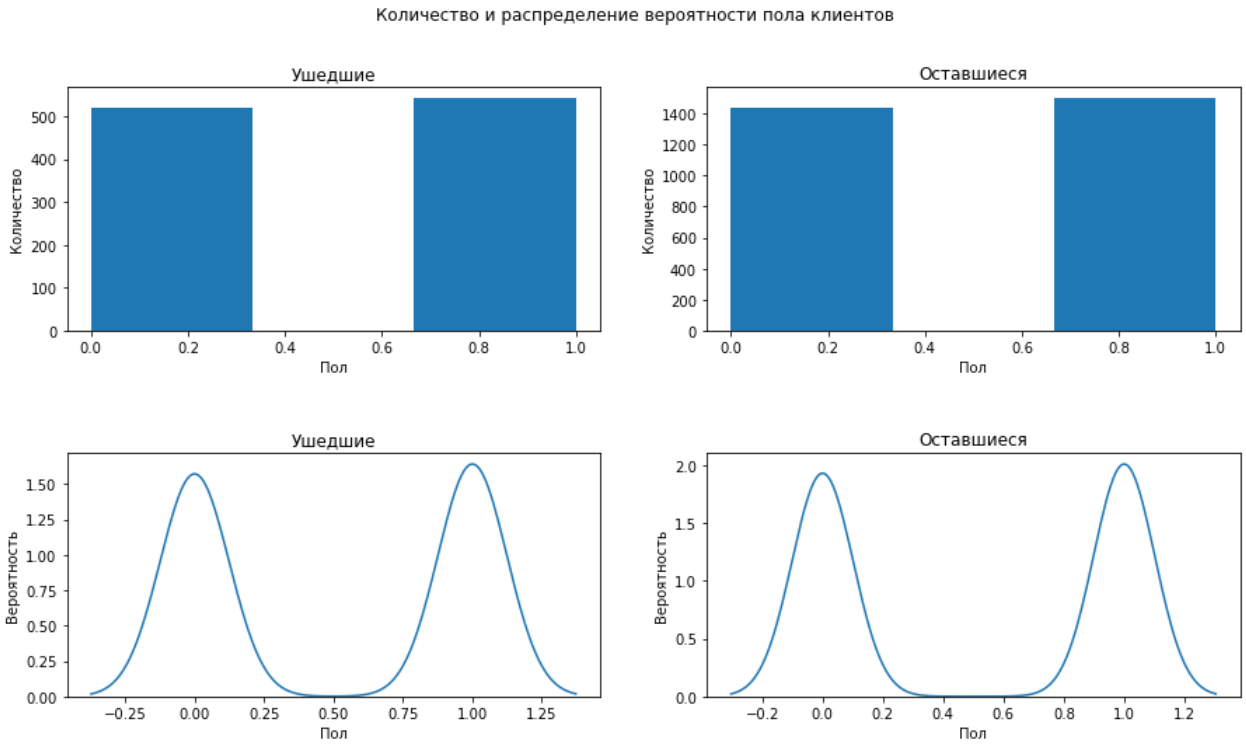
### построение гистограмм

Построим диаграммы количества и распределения вероятности для всех параметров в разрезе ушедших и оставшихся.

Я оформил построение диаграмм в виде отдельной функции, параметрами для которой являются наименование столбца, количество корзин разбиения для построения диаграмм, подпись к оси x, подпись к графику в целом.

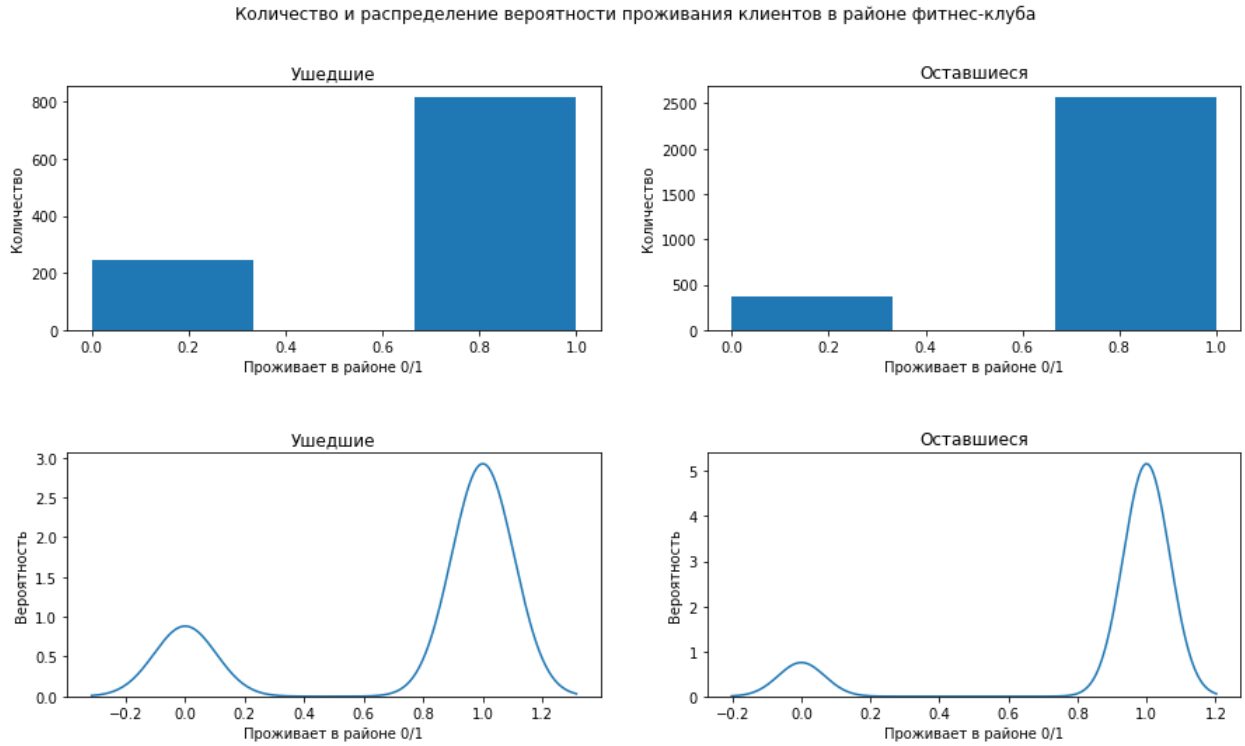
Ввод [11]:
```
def plotgr(ncolumn,bins,x_label,total_label):
    fig, ax = plt.subplots(ncols=2,nrows=2,figsize=(15,8))
    plt.subplots_adjust(hspace=0.5)
    ax[0][0].hist(df[df.Churn==1][ncolumn],bins=bins)
    ax[0][1].hist(df[df.Churn==0][ncolumn],bins=bins)
    ax[0][0].set_title('Ушедшие')
    ax[0][0].set_xlabel(x_label)
    ax[0][0].set_ylabel('Количество')
    ax[0][1].set_title('Оставшиеся')
    ax[0][1].set_xlabel(x_label)
    ax[0][1].set_ylabel('Количество')
    sns.kdeplot(df[df.Churn==1][ncolumn],ax=ax[1][0],legend=False)
    sns.kdeplot(df[df.Churn==0][ncolumn],ax=ax[1][1],legend=False)
    ax[1][0].set_title('Ушедшие')
    ax[1][0].set_xlabel(x_label)
    ax[1][0].set_ylabel('Вероятность')
    ax[1][1].set_title('Оставшиеся')
    ax[1][1].set_xlabel(x_label)
    ax[1][1].set_ylabel('Вероятность')
    fig.suptitle(total_label)
    plt.show()
```

Ввод [12]:
```
plotgr('gender',3,'Пол','Количество и распределение вероятности пола клиентов')
```



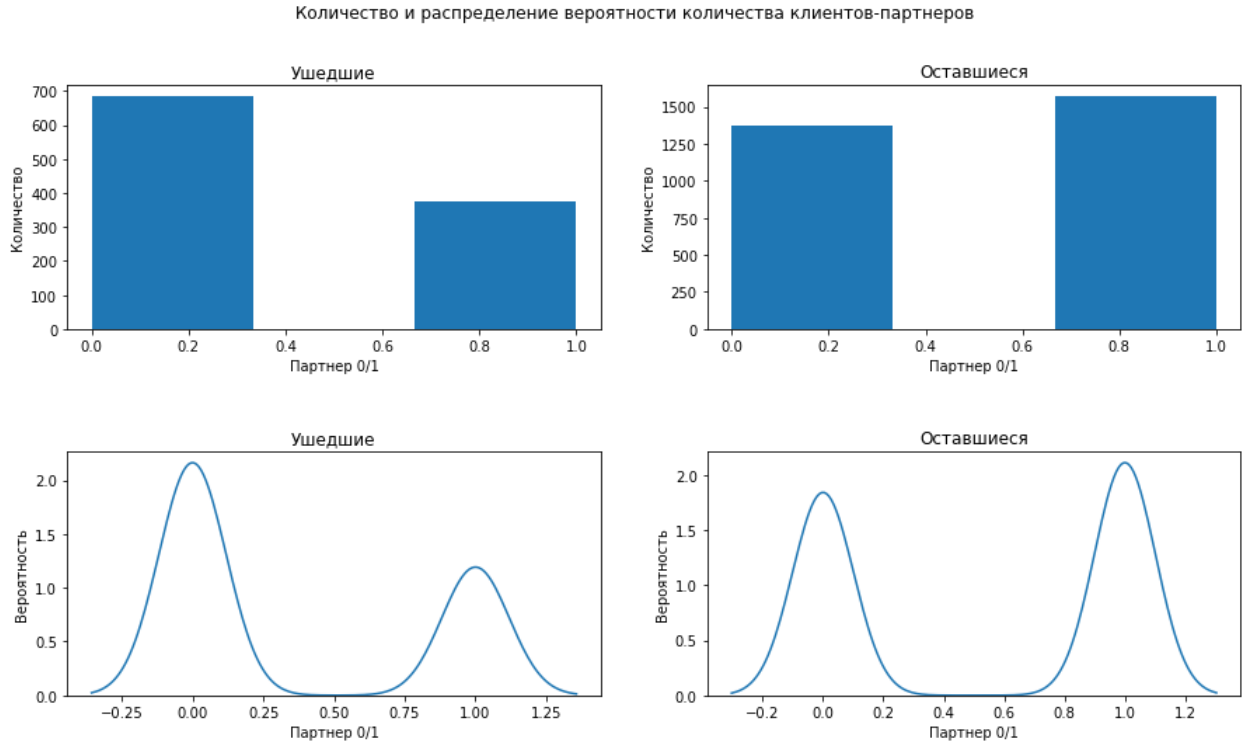
Мужчин и женщин поравну и в ушедших и в оставшихся.

Ввод [13]:
```
plotgr('Near_Location',3,'Проживает в районе 0/1',
       'Количество и распределение вероятности проживания клиентов в районе фитнес-клуба')
```



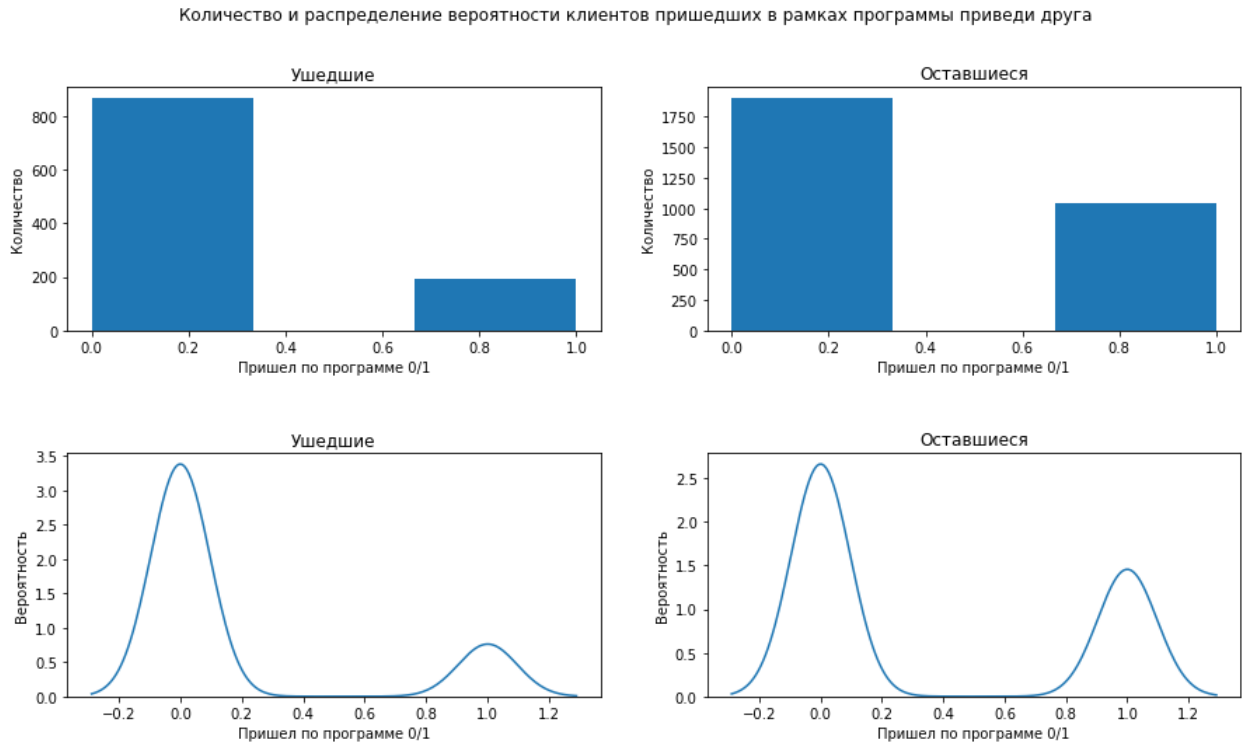

Проживают/работаю не далеко от фитнес центра большинство клиентов, но это никак не влияет на факт оттока.

Ввод [14]:
```
plotgr('Partner',3,'Партнер 0/1','Количество и распределение вероятности количества клиентов-партнеров')
```



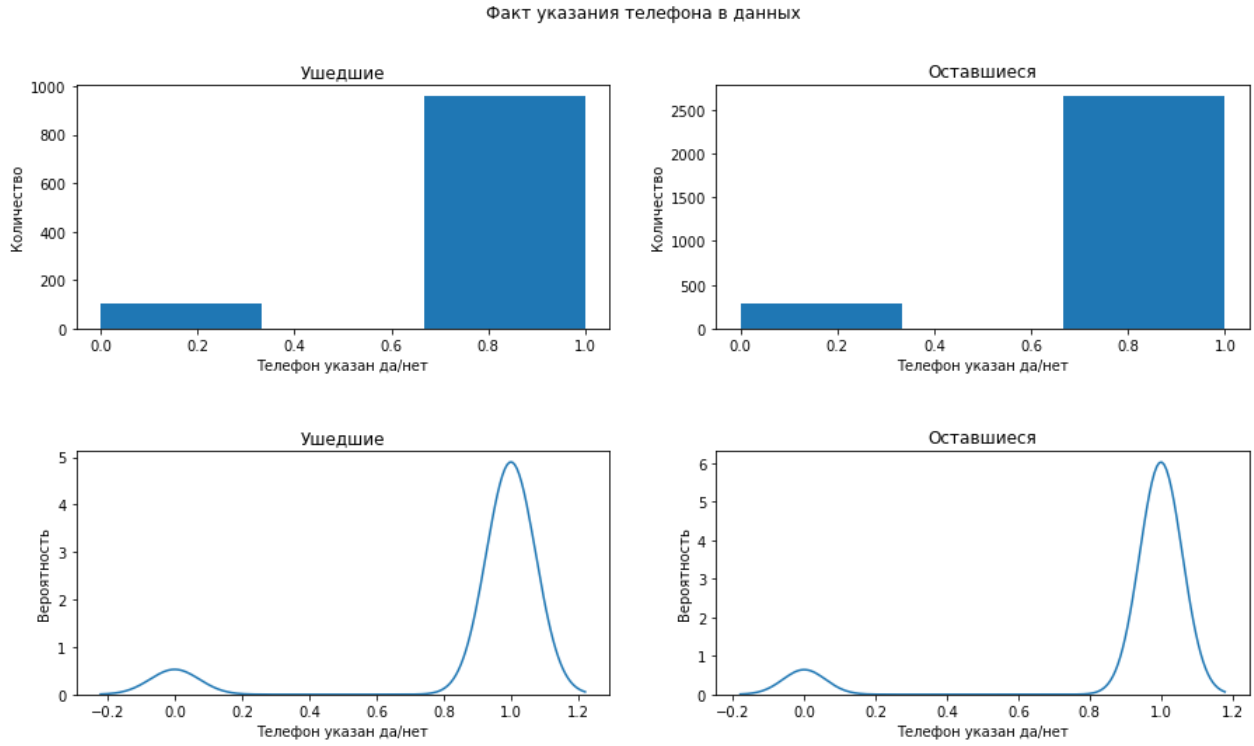

Больше половины оставшихся являются сотрудниками компаний-партнеров. Среди ушедших около половины таковыми не являются.

Ввод [15]:
```
plotgr('Promo_friends',3,'Пришел по программе 0/1',
       'Количество и распределение вероятности клиентов пришедших в рамках программы приведи друга')
```



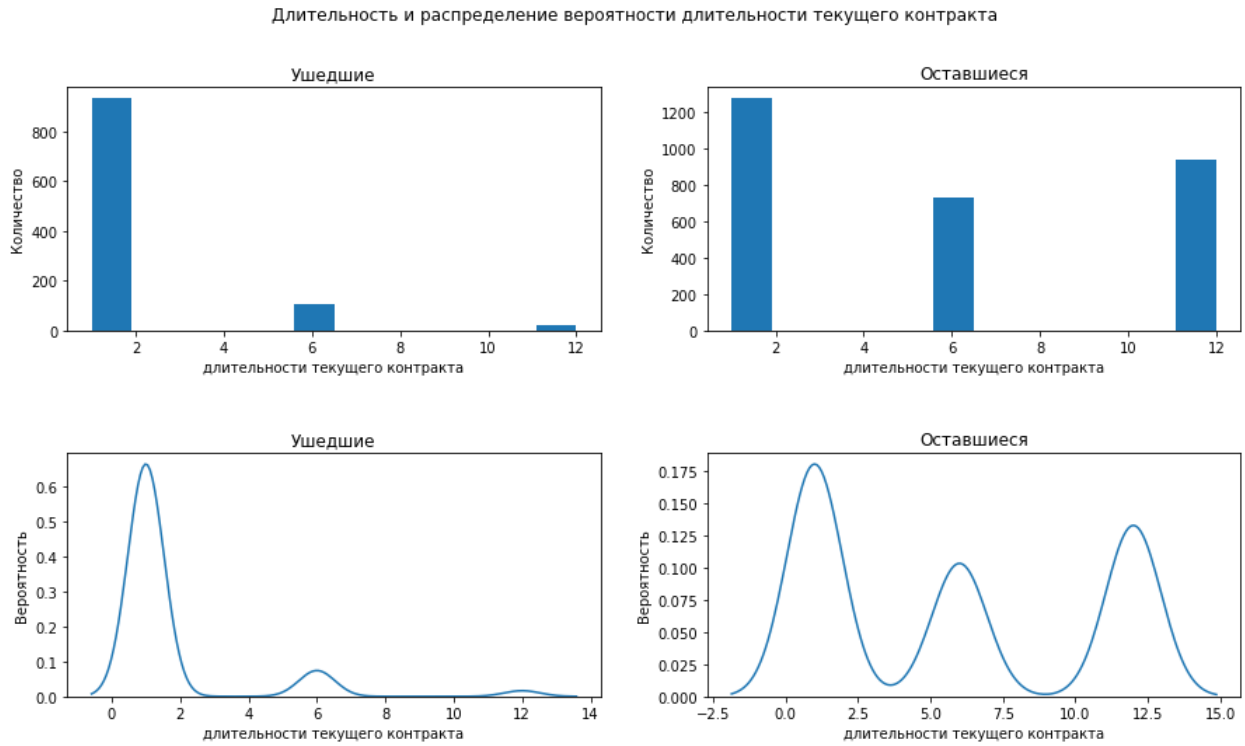

Промо акция "приведи друга" показывает свою эффективность. Среди активных клиентов таковых почти половина, а среди ушедших всего четверть.

Ввод [16]:
```
plotgr('Phone',3,'Телефон указан да/нет','Факт указания телефона в данных')
```



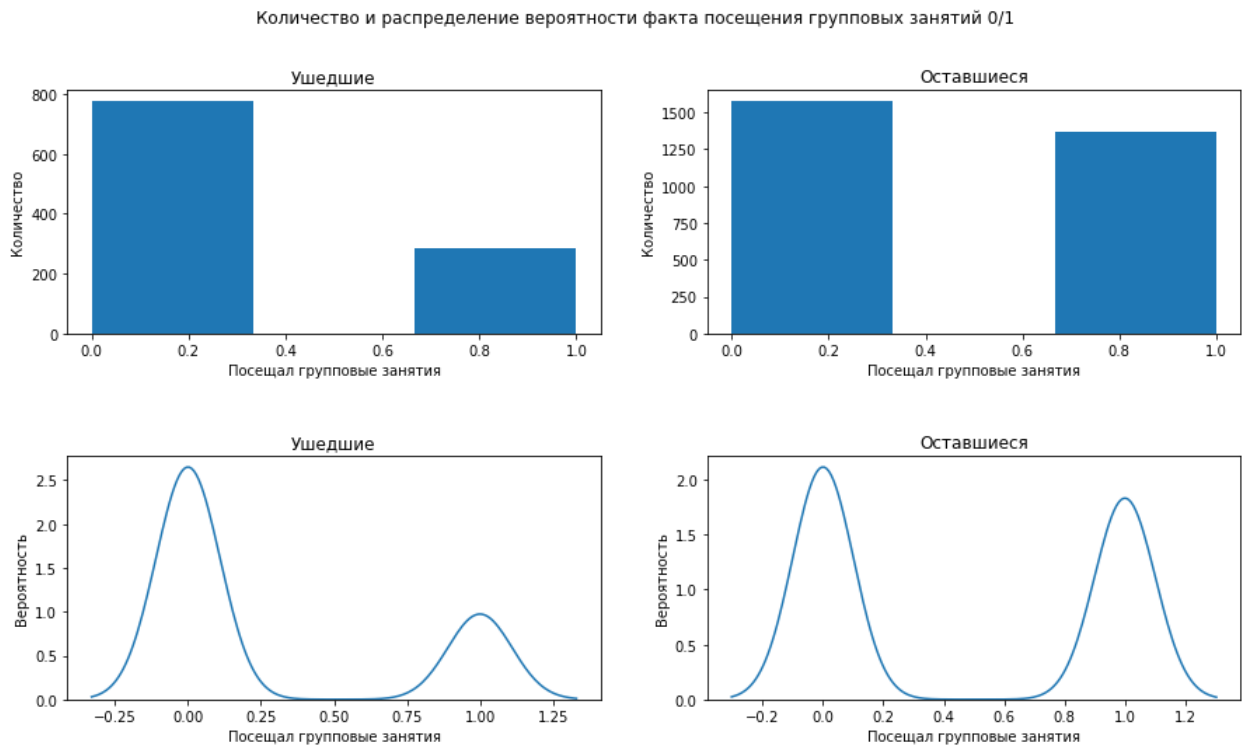

Большинство указывает телефон в контактных данных.

Ввод [17]:
```
plotgr('Contract_period',12,'длительности текущего контракта',
       'Длительность и распределение вероятности длительности текущего контракта')
```



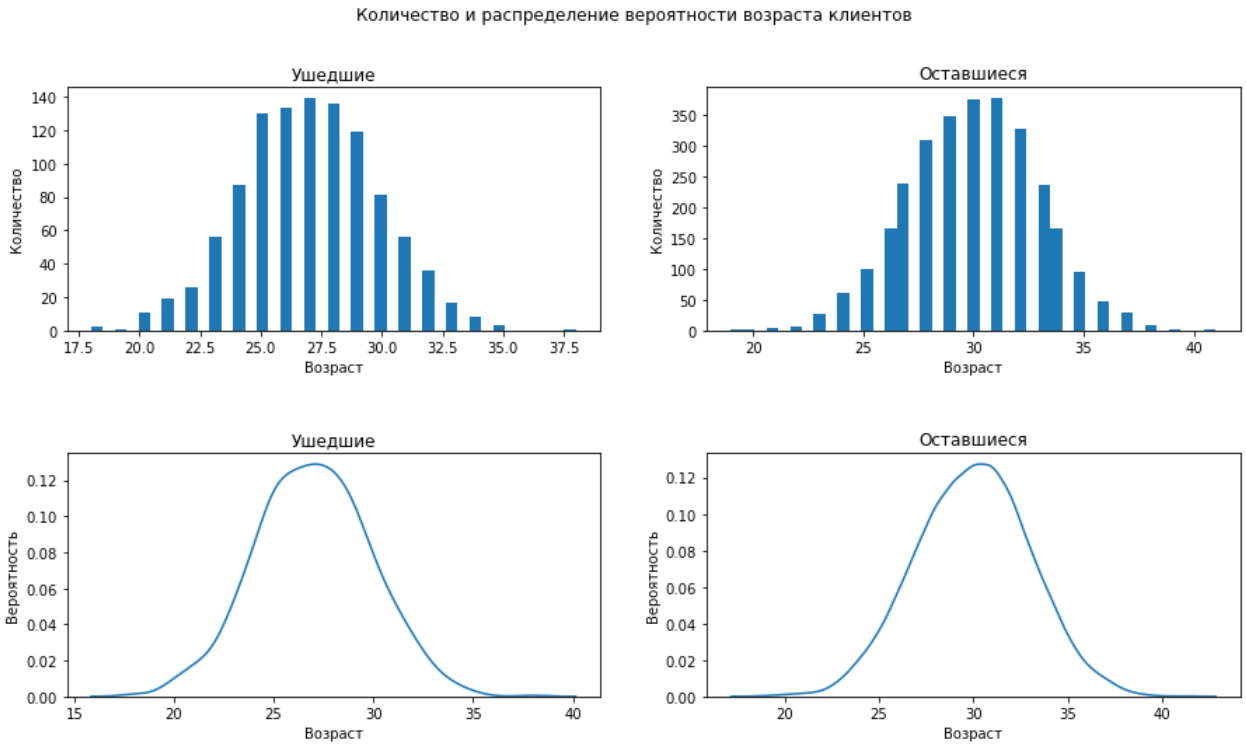

Месячные контракты очень популярны и у ушедших и у оставшихся. Но можно отметить, что очень мало уходит с контрактом больше месяца.

Ввод [18]:
```
plotgr('Group_visits',3,
       'Посещал групповые занятия','Количество и распределение вероятности факта посещения групповых занятий 0/1')
```



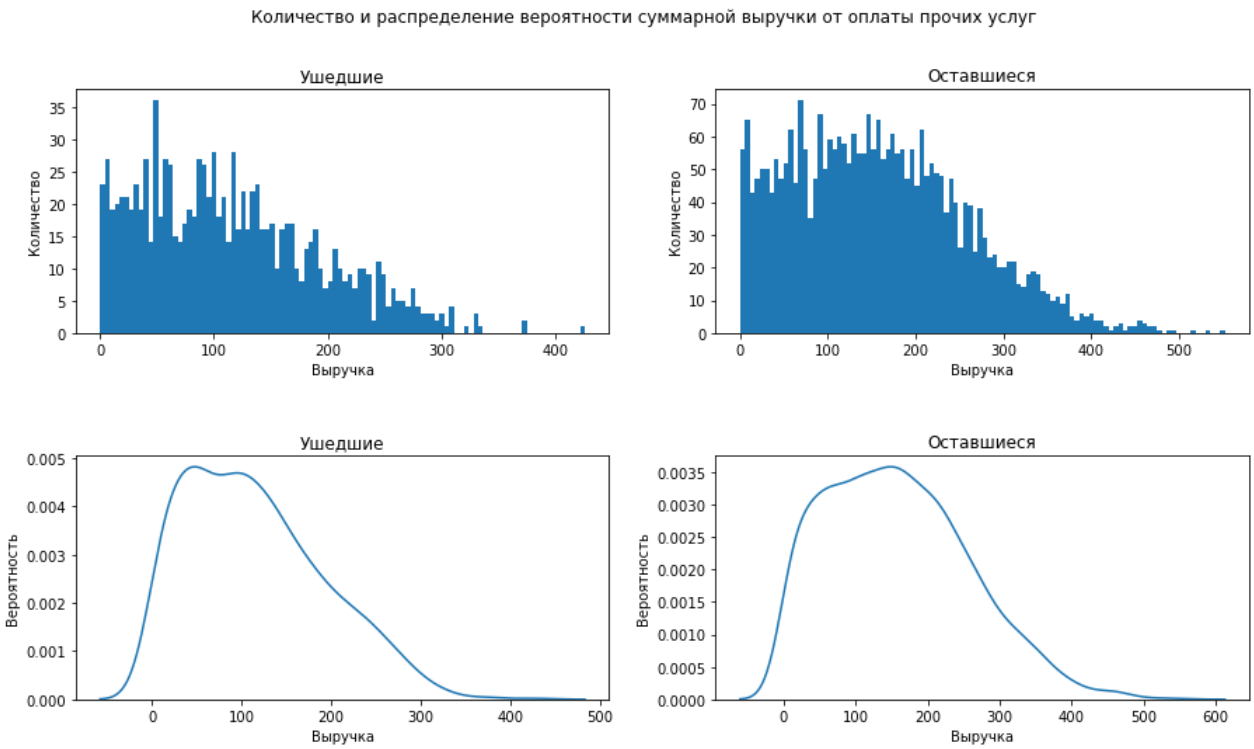

Практически половина из активных клиентов посещают групповые занятия, в то время как среди ушедших таковых около трети. Это параметр явно связан с вероятностью оттока.

Ввод [19]:
```
plotgr('Age',41,'Возраст','Количество и распределение вероятности возраста клиентов')
```



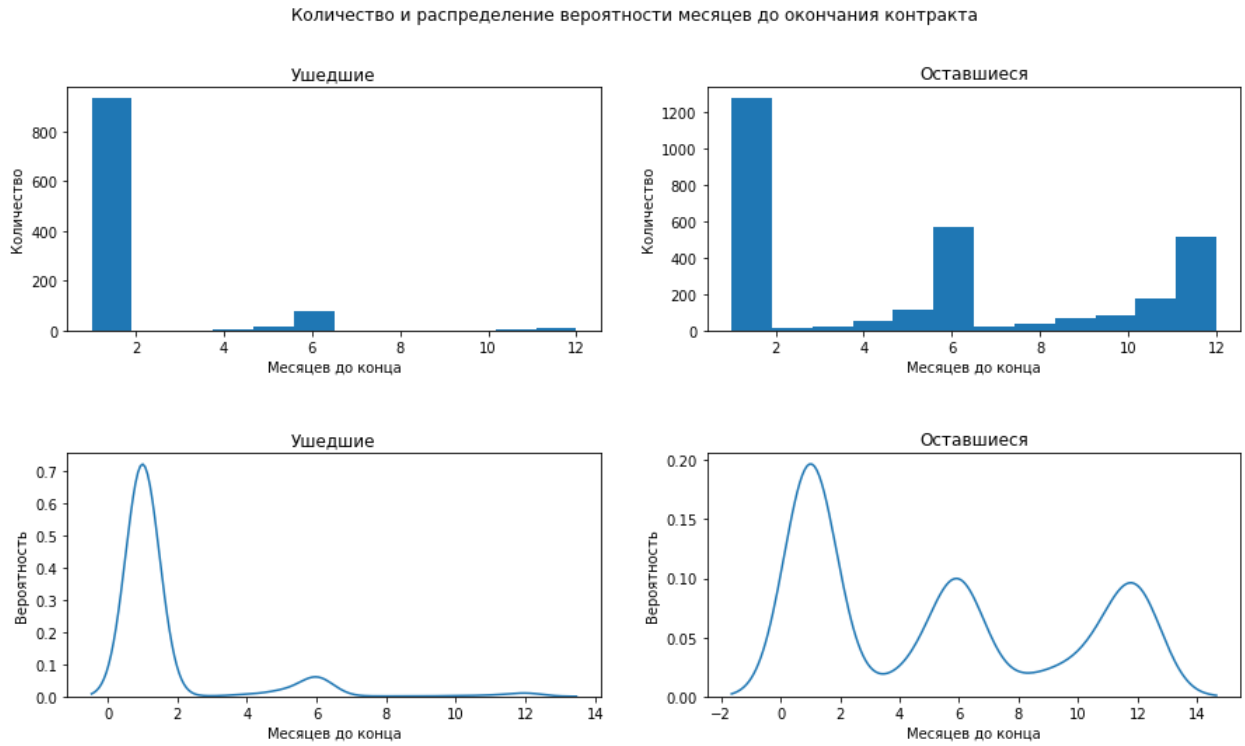

Нормальное распределение возрастов, у ушедших максимум чуть левее 30 лет, к оставшихся чуть правее. Не думаю что это принципиальный параметр.

Ввод [20]:
```
plotgr('Avg_additional_charges_total',100,'Выручка',
       'Количество и распределение вероятности суммарной выручки от оплаты прочих услуг')
```



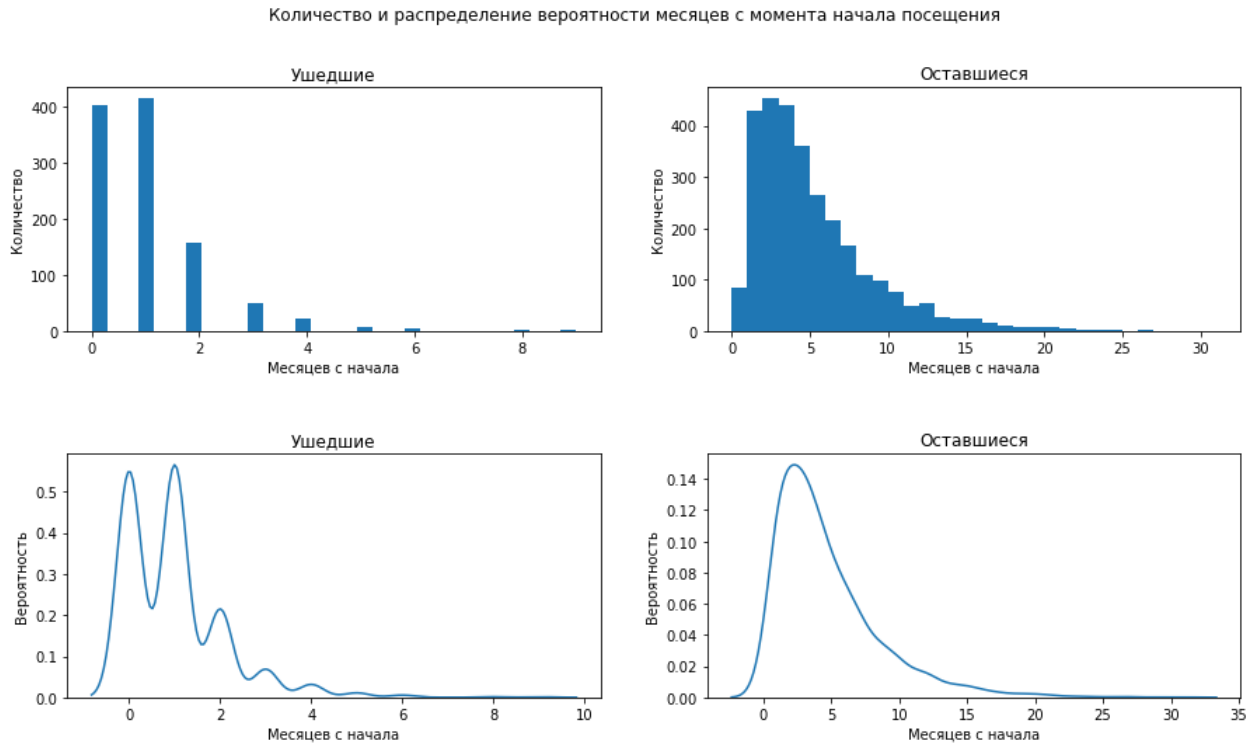

Все совершают мелкие покупки. Видимо в фитнес-центре есть бар.

Ввод [21]:
```
plotgr('Month_to_end_contract',12,'Месяцев до конца','Количество и распределение вероятности месяцев до окончания контракта')
```



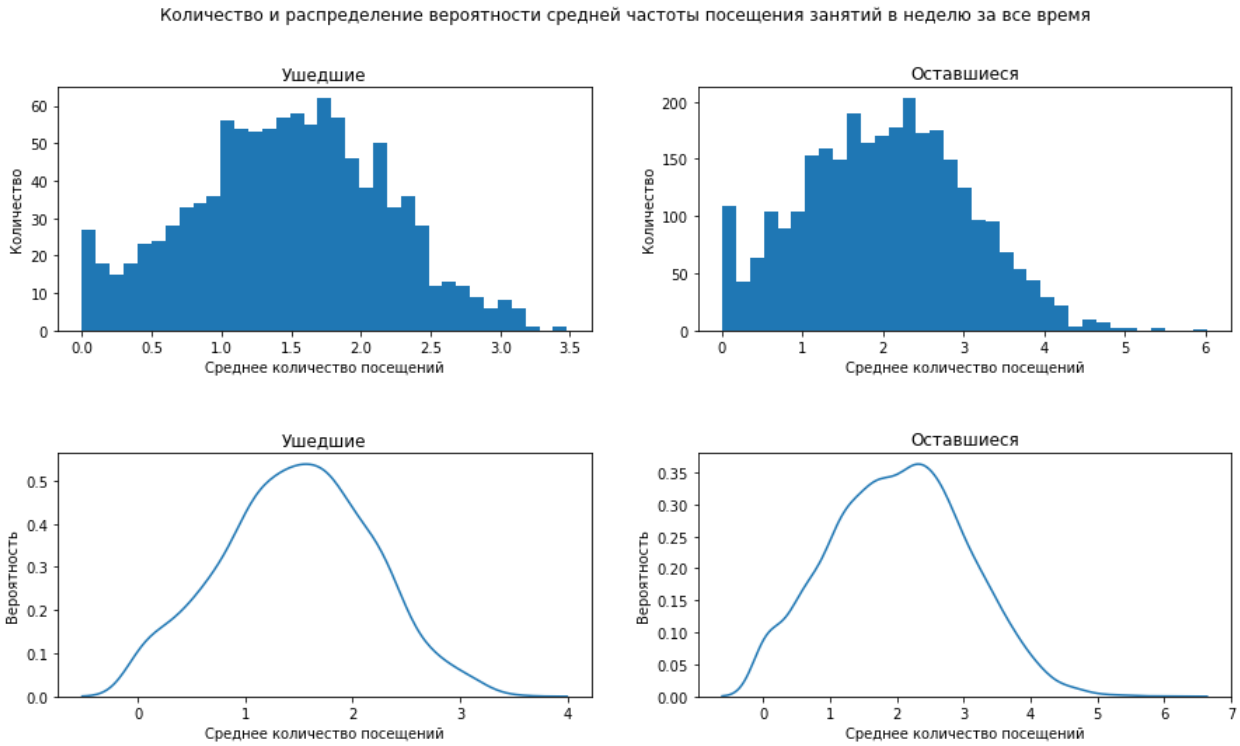

Как мы уже отметили ранее, месячные контракты самые популярные. Особенно у ушедших.

Ввод [22]:
```
plotgr('Lifetime',31,'Месяцев с начала','Количество и распределение вероятности месяцев с момента начала посещения')
```



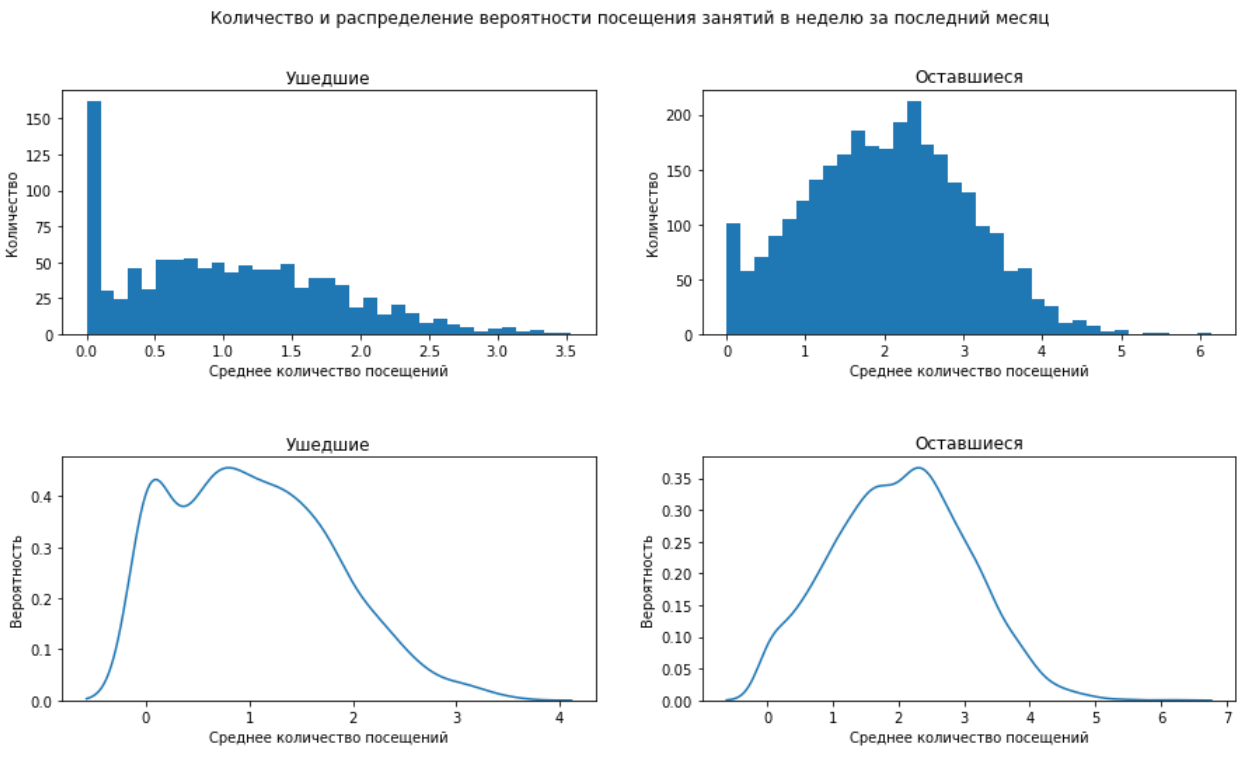

Еще одно подтверждение популярности месячных контрактов. Но хочу заметить, что отсутствие пиков на 6 и 12 месяцев означает, что у фитнес-клуба нет программы скидок за покупку длительных контрактов. Т.е. месячный контракт пользуется у посетителей не просто так, видимо это ценовая политика клуба.

Ввод [23]:
```
plotgr('Avg_class_frequency_total',35,'Среднее количество посещений',
       'Количество и распределение вероятности средней частоты посещения занятий в неделю за все время')
```




Около 3-х посещений в неделю у оставшихся это нормально. А вот 1,5 у ушедших явно тревожный признак.

Ввод [24]:
```
plotgr('Avg_class_frequency_current_month',
       35,
       'Среднее количество посещений',
       'Количество и распределение вероятности посещения занятий в неделю за последний месяц')
```




Как и можно было предположить 0 посещений в последний месяц у ушедших - прямое указание на то, что решение уйти принято.

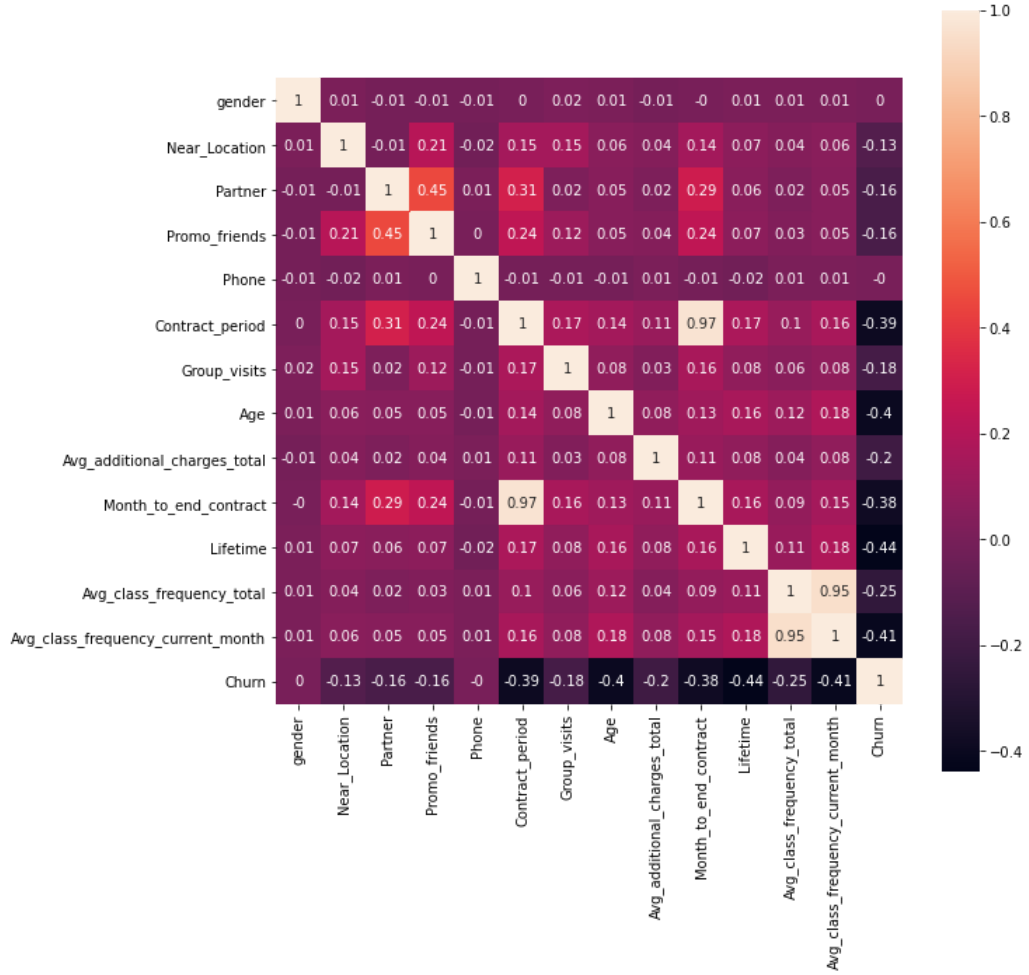
мужчина и женщина - примерно одинаковое количество тех и других; возраст большинства - 26-34 года; большинство проживают или работают в районе, где находится фитнес-центр; сотрудник компании-партнёра клуба - около половины; факт первоначальной записи в рамках акции «приведи друга» - около 30 %; длительность текущего действующего абонемента - в среднем клиенты берут абонемент на 6 мес., но высокий показатель

стандартного отклонения коэффициента вариативности говорит о том, что абонемент на 6 мес. не самый часто покупаемый; факт посещения групповых занятий - почти половина посещает; суммарная выручка от других услуг фитнес-центра: кафе, спорт-товары, косметический и массажный салон - небольшая и колеблется в среднем от 60 до 2000 у.е.; время с момента первого обращения в фитнес-центр (в месяцах) - от 1 до 9 мес.; средняя частота посещений в неделю за все время с начала действия абонемента и за последний месяц - 2.5 - 3 раза;

### анализ корреляции

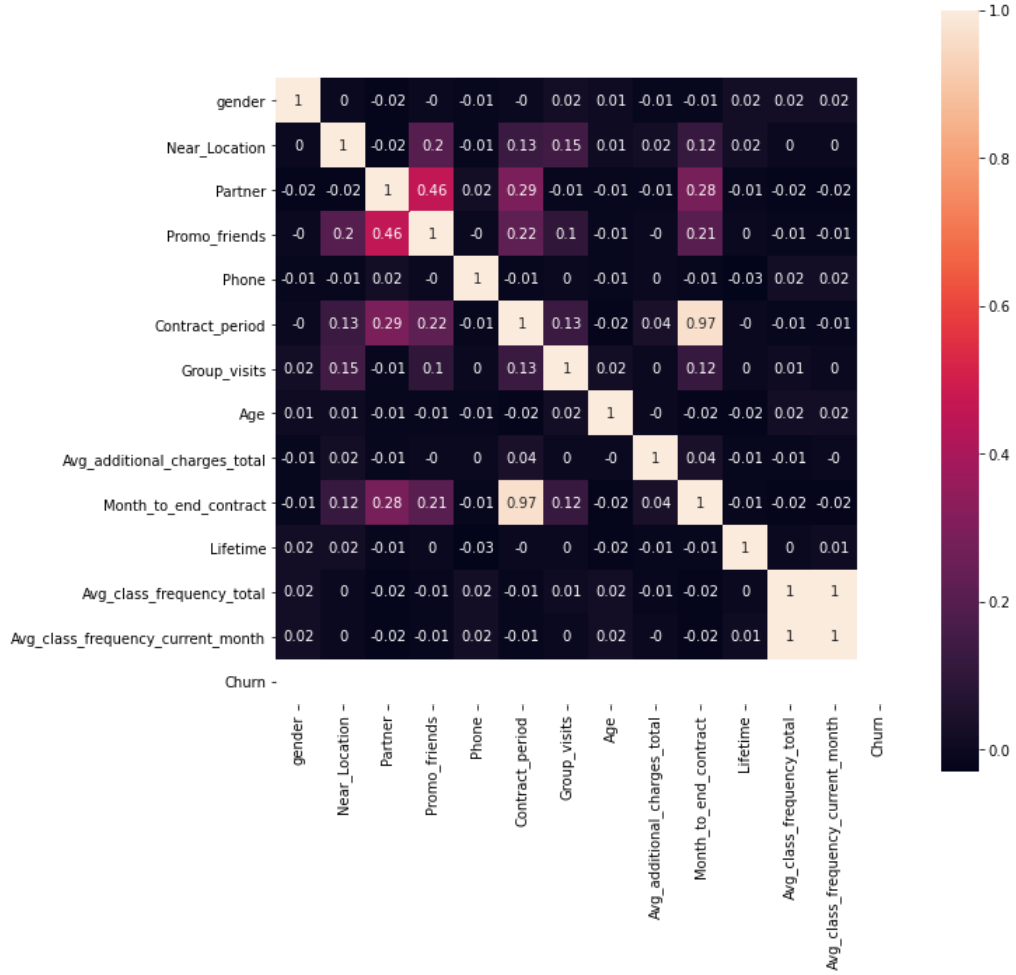
Посчитаем матрицу корреляции и нарисуем тепловую карту сначала в целом, затем только по ушедшим и оставщимся

Ввод [25]:
```
cm = df.corr().round(2)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(cm, annot = True, square=True)
plt.show()
```



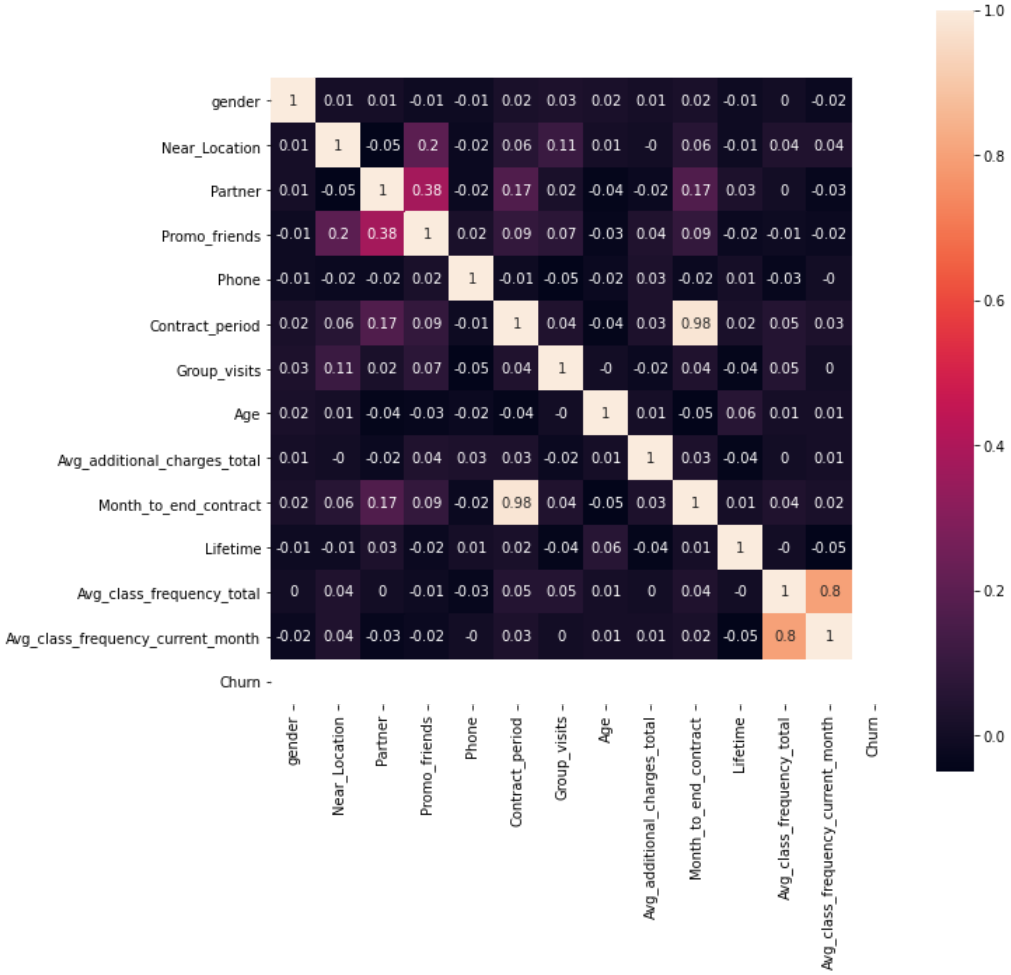

Оставшиеся

Ввод [26]:
```
cm = df_ch0.corr().round(2)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(cm, annot = True, square=True)
plt.show()
```




Ушедшие

Ввод [27]:
```
cm = df_ch1.corr().round(2)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(cm, annot = True, square=True)
plt.show()
```




Четко видно что коррелируют признаки

- Month_to_end_contract и Contract_period
- Avg_class_frequency_current_month и Avg_class_frequency_total

По одному из признаков из каждой пары надо будет удалить

## Построение модели

Построим модель бинарной классификации клиентов, где целевой признак — факт оттока клиента в следующем месяце Удаляем коррелирующие признаки

Ввод [28]:
```
dftmp=df.drop(['Month_to_end_contract', 'Avg_class_frequency_total'],axis=1)
```

Разделяем целевой признак и остальные параметры и делим данные на обучающую и тестовую выборки. Поскольку целевой параметр у нас бинарный, то я применил параметр stratify=y, что-бы получить равномерные обучающие и тестовые выборки для ушедших и оставшихся.

Ввод [29]:
```
X=dftmp.drop('Churn', axis=1)
y=dftmp['Churn']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

Нормализуем данные.

Ввод [30]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_st = scaler.transform(X_train)
X_test_st =  scaler.transform(X_test)
```

Обучаем модель и делаем предсказание методом логистической регрессии.

Ввод [31]:
```
# зададим алгоритм для модели логистической регрессии
lr_model = LogisticRegression(random_state = 0,solver='lbfgs')
# обучим модель
lr_model.fit(X_train_st, y_train)
# воспользуемся уже обученной моделью, чтобы сделать прогнозы
lr_predictions = lr_model.predict(X_test_st)
lr_probabilities = lr_model.predict_proba(X_test_st)[:,1]
```

Обучаем модель и делаем предсказание методом случайного леса.

Ввод [32]:
```
# зададим алгоритм для новой модели на основе алгоритма случайного леса
rf_model = RandomForestClassifier(n_estimators = 100,random_state = 0)
# обучим модель случайного леса
rf_model.fit(X_train_st, y_train)
# воспользуемся уже обученной моделью, чтобы сделать прогнозы
rf_predictions = rf_model.predict(X_test_st)
rf_probabilities = rf_model.predict_proba(X_test_st)[:,1]
```

Оценим метрики для обеих моделей на тестовой выборке

Ввод [33]:
```
def print_all_metrics(y_true, y_pred, y_proba, title):
    print(title)
    print('\tAccuracy: {:.2f}'.format(accuracy_score(y_true, y_pred)))
    print('\tPrecision: {:.2f}'.format(precision_score(y_true, y_pred)))
    print('\tRecall: {:.2f}'.format(recall_score(y_true, y_pred)))
    print('\tF1: {:.2f}'.format(f1_score(y_true, y_pred)))
    print('\tROC_AUC: {:.2f}'.format(roc_auc_score(y_true, y_proba)))
```

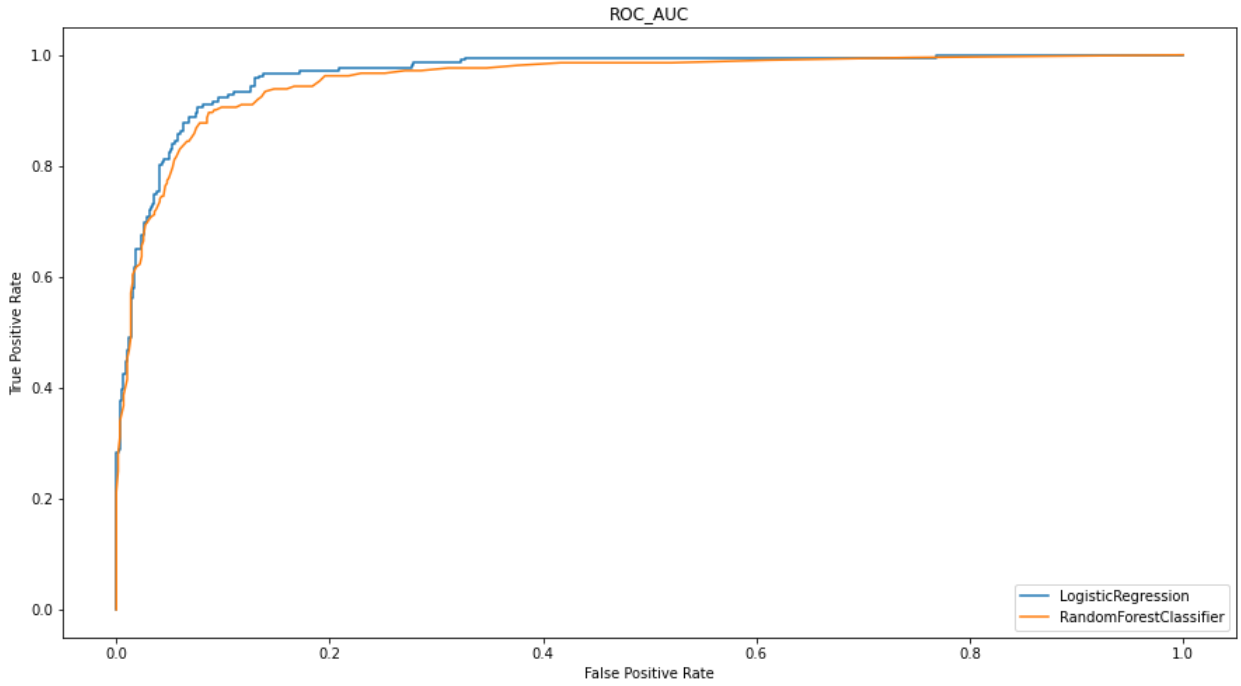
Ввод [34]:
```
# выведем все метрики
print_all_metrics(y_test, lr_predictions, lr_probabilities, 'Метрики для модели логистической регрессии:')
print_all_metrics(y_test, rf_predictions, rf_probabilities, 'Метрики для модели случайного леса:')
```

```
Метрики для модели логистической регрессии:
        Accuracy: 0.92
        Precision: 0.85
        Recall: 0.84
        F1: 0.84
        ROC_AUC: 0.97
Метрики для модели случайного леса:
        Accuracy: 0.91
        Precision: 0.85
        Recall: 0.77
        F1: 0.81
        ROC_AUC: 0.96
```

Модель логистической регрессии дает лучьшие результаты.

Дополнительно посмотим на ROC_AUC

Ввод [35]:
```
fig, ax = plt.subplots(figsize=(15,8))
fpr, tpr, thresh = roc_curve(y_test, lr_probabilities)
plt.plot(fpr,tpr,label="LogisticRegression")
fpr, tpr, thresh = roc_curve(y_test, rf_probabilities)
plt.plot(fpr,tpr,label="RandomForestClassifier")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC_AUC')
plt.legend(loc=0)
plt.show()
```




И здесь мы видим, что логистическая регрессия немного лучьше. По графику метрики ROC_AUC видим, что кривая алгоритма логистической регрессии «выгибается» к верхнему левому углу больше, то есть близка к 1, а значит является более точной. В дальнейшем для прогнозирования можно использовать модель логистической регрессии.

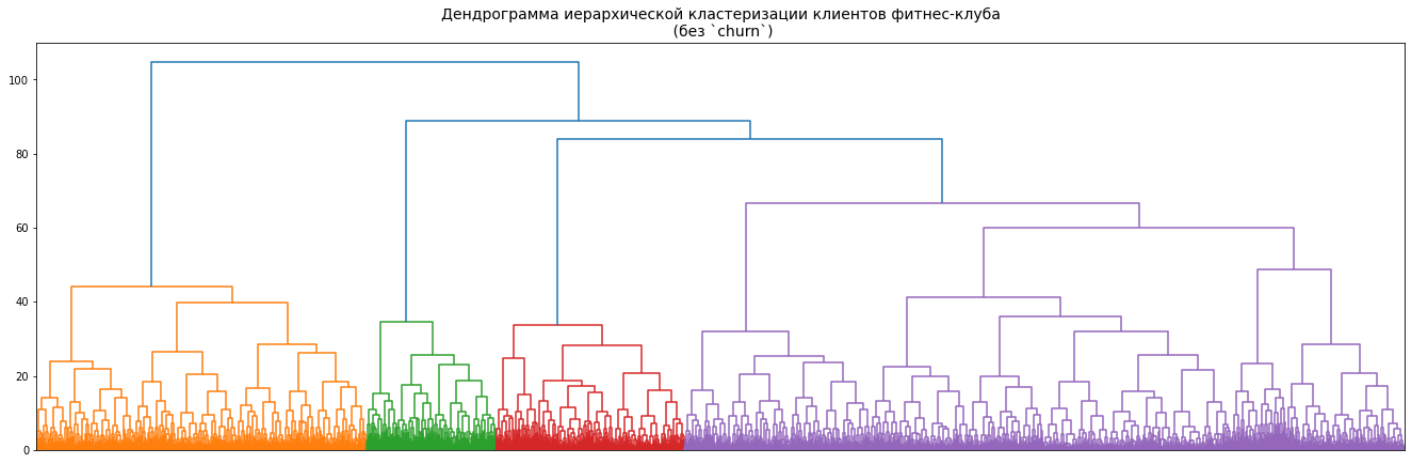
## Кластеризация

Стандартизируем данные

Ввод [36]:
```
df_sc = scaler.fit_transform(df.drop(columns = ['Churn']))
```

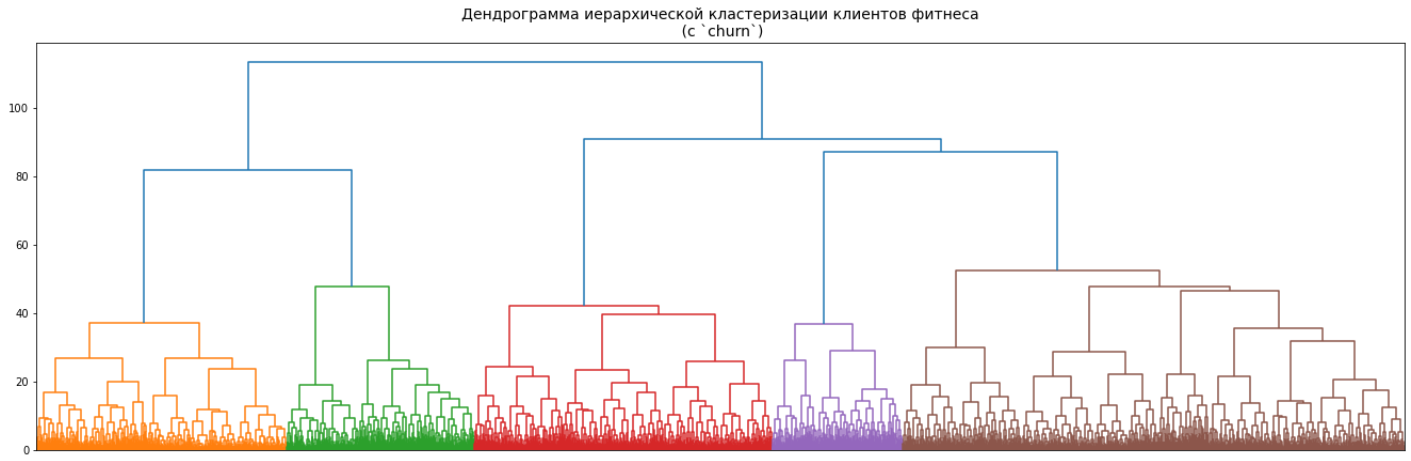
Построим матрицу расстояний и нарисуем дендрограмму без учета столбца Churn. При вызове функции linkage я использовал параметр method='ward', т.к. он дает лучьшие результаты. Я перебрал single,complete,average,weighted,centroid,median и только ward дает приемлимую картину.

Ввод [37]:
```
linked = linkage(df_sc,method = 'ward')
plt.figure(figsize=(23, 7))
dendrogram(linked, orientation='top', no_labels=True)
plt.title('Дендрограмма иерархической кластеризации клиентов фитнес-клуба\n (без `churn`)', size=14)
plt.show()
```




Построим матрицу расстояний и нарисуем дендрограмму с учетом столбца Churn.

Ввод [38]:
```
df_sc = scaler.fit_transform(df)
linked = linkage(df_sc, method = 'ward')
plt.figure(figsize=(23, 7))
dendrogram(linked, orientation='top', no_labels=True)
plt.title('Дендрограмма иерархической кластеризации клиентов фитнеса\n (с `churn`)', size=14)
plt.show()
```




Число кластеров равно 5, что соответсвует предложенному в задании числу.

Обучим модель кластеризации на основании алгоритма K-Means и спрогнозируем кластеры клиентов

Ввод [39]:
```
km = KMeans(n_clusters = 5, random_state=0)
labels = km.fit_predict(df_sc)
```

Занесем данные о кластерах в исходный набор данных

Ввод [40]:
```
df['cluster_km'] = labels
```

Оценим метрикой силуэта качество кластеризации

Ввод [41]:
```
df_sc = scaler.fit_transform(df)
silhouette_score(df_sc, labels)
```

Out[41]:
```
0.17411035347503936
```

Качество кластеризации не очень высокое. Должно быть как можно ближе к 1.

Посмотрим на отток по кластерам

Ввод [42]:
```
df.groupby('cluster_km').agg({'Churn':'mean'})
```

Out[42]:

| cluster_km | Churn |
|---|---|
| 0 | 0.450368 |
| 1 | 0.007823 |
| 2 | 0.961023 |
| 3 | 0.003390 |
| 4 | 0.024709 |

Имеем наибольший отток в кластере 2 и вдвое меньший в кластере 0

Посмотрим на средние значения признаков для кластеров

Ввод [43]:
```
df.groupby('cluster_km').mean().T
```

Out[43]:

| cluster_km | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| gender | 0.503676 | 0.492829 | 0.507917 | 0.521186 | 0.518895 |
| Near_Location | 0.000000 | 0.936115 | 1.000000 | 0.996610 | 0.968023 |
| Partner | 0.459559 | 0.764016 | 0.323995 | 0.282203 | 0.744186 |
| Promo_friends | 0.075368 | 0.534550 | 0.213155 | 0.208475 | 0.526163 |
| Phone | 0.917279 | 0.919166 | 0.903776 | 0.904237 | 0.873547 |
| Contract_period | 2.069853 | 10.447197 | 1.546894 | 1.882203 | 8.859012 |
| Group_visits | 0.211397 | 0.516297 | 0.287454 | 0.452542 | 0.534884 |
| Age | 28.522059 | 29.962190 | 26.926918 | 30.237288 | 29.728198 |
| Avg_additional_charges_total | 132.741216 | 161.853561 | 113.173051 | 157.988625 | 162.907495 |
| Month_to_end_contract | 1.950368 | 9.477184 | 1.500609 | 1.794915 | 8.155523 |
| Lifetime | 2.757353 | 4.833116 | 1.031669 | 4.923729 | 4.411337 |
| Avg_class_frequency_total | 1.706629 | 2.788103 | 1.449720 | 2.203432 | 0.957744 |
| Avg_class_frequency_current_month | 1.515890 | 2.786919 | 1.037592 | 2.207359 | 0.943967 |
| Churn | 0.450368 | 0.007823 | 0.961023 | 0.003390 | 0.024709 |

Почти все клиенты в оттоке попали в кластер № 2: мужчин и женщин в нем поровну; средний возраст - 27 лет; все признаки ниже чем в остальных кластерах, особенно ярковыраженные из них: длительность текущего действующего абонемента, факт посещения групповых занятий, суммарная выручка от других услуг фитнес-центра и средняя частота посещений в неделю за предыдущий месяц.

В кластере № 0 из оттока 45 % клиентов. Отличительной особенностью этого кластера является что все клиенты не проживают/работают рядом с фитнес-центром.

Самые надежные кластеры № 3 затем №1 и №4:

самые возрастные клиенты - 30 лет; самая высокая средняя частота посещений

Нарисуем графики распределения параметров в разрезе кластеров, с разбивкой на ушедших и оставшихся.

Для этого я определю две функции рисовани. Первая для целочисленных параметров, вторая для параметров с непрерывным значением.

Параметры вызова первой функции - имя столбца, подпись оси x, подпись графика, аггрегационная функция.

Параметры вызова второй функции - имя столбца, количество корзин разбиения, подпись оси x, подпись оси y, подпись графика.

Примечание для проверяющего. Я не доволен тем, как я сделал эти графики.

- Во-первых цвета на графиках ушедших и оставшихся - разные. Я хотел, что-бы цвета были одинаковые и была одна легенда на оба графика, но не смог. Не нашел как это можно сделать.
- Во-вторых функция рисования непрерывных данных мне кажется дико переусложненной. Она делит данные по корзинам (по интервалам) методом pd.cut(),затем эти интервалы заносяться в таблицу и производиться группировка с аггрегацией. Таким образом для оси x я получаю список значений равный началу интервалов, а для оси y количество строк, попавших в этот интервал.

Почему я так сделал.

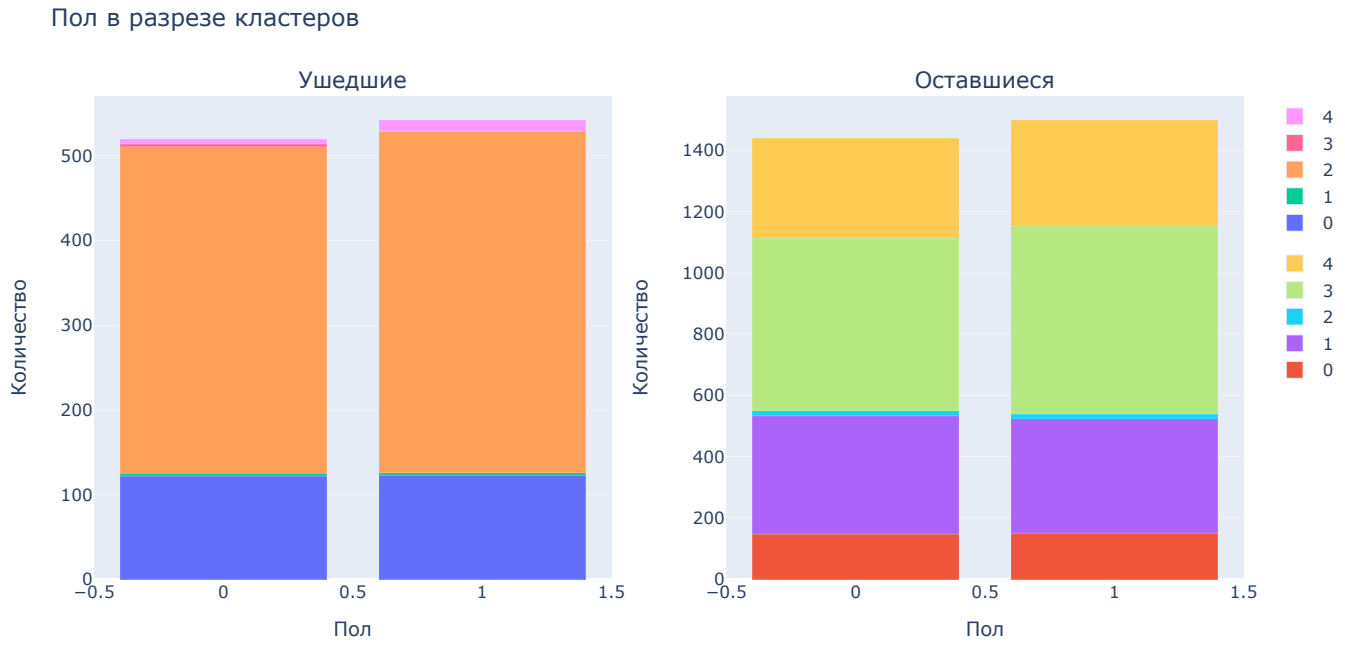
Я не смог заставить matplotlib рисовать staked bar, т.к. для разных значений кластера получалось разное количество данных для отображения. Т.е. для некоторых кластеров, некоторых данных просто не было. И matplotlib выдавал ошибку, что ряды данных должны быть одинаковой длинны.

В seaborn есть удобная функция рисования displot (не путать с distplot) и histplot, но они есть в версии 0.11 а мне в данном случае доступна только 0.9, а в ней этих функций нет.

Ввод [44]:
```
def plot_stack(numcol,x_label,t_label,aggf):
    #dftmp=df.groupby([numcol,'cluster_km','Churn']).agg({numcol:{'Value':aggf}}).reset_index()
    dftmp=df.groupby([numcol,'cluster_km','Churn']).agg(val=(numcol,aggf)).reset_index()
    dftmp.columns=['c1','c2','c3','c4']
    kat=[0,1,2,3,4]
    fig = make_subplots(rows=1, cols=2,subplot_titles=('Ушедшие',  'Оставшиеся'))
    for kat in kat:
        dftmp_c1=dftmp[(dftmp['c2']==kat) & (dftmp['c3']==1)]
        dftmp_c0=dftmp[(dftmp['c2']==kat) & (dftmp['c3']==0)]
        fig.add_trace(go.Bar(x=dftmp_c1['c1'], y=dftmp_c1['c4'], name=kat,legendgroup=1),row=1,col=1)
        fig.add_trace(go.Bar(x=dftmp_c0['c1'], y=dftmp_c0['c4'], name=kat,legendgroup=2),row=1,col=2)
    fig.update_layout(barmode='stack',title_text=t_label)
    fig.update_xaxes(title_text=x_label,row=1,col=1)
    fig.update_xaxes(title_text=x_label,row=1,col=2)
    fig.update_yaxes(title_text='Количество',row=1,col=1)
    fig.update_yaxes(title_text='Количество',row=1,col=2)
    fig.show()
```

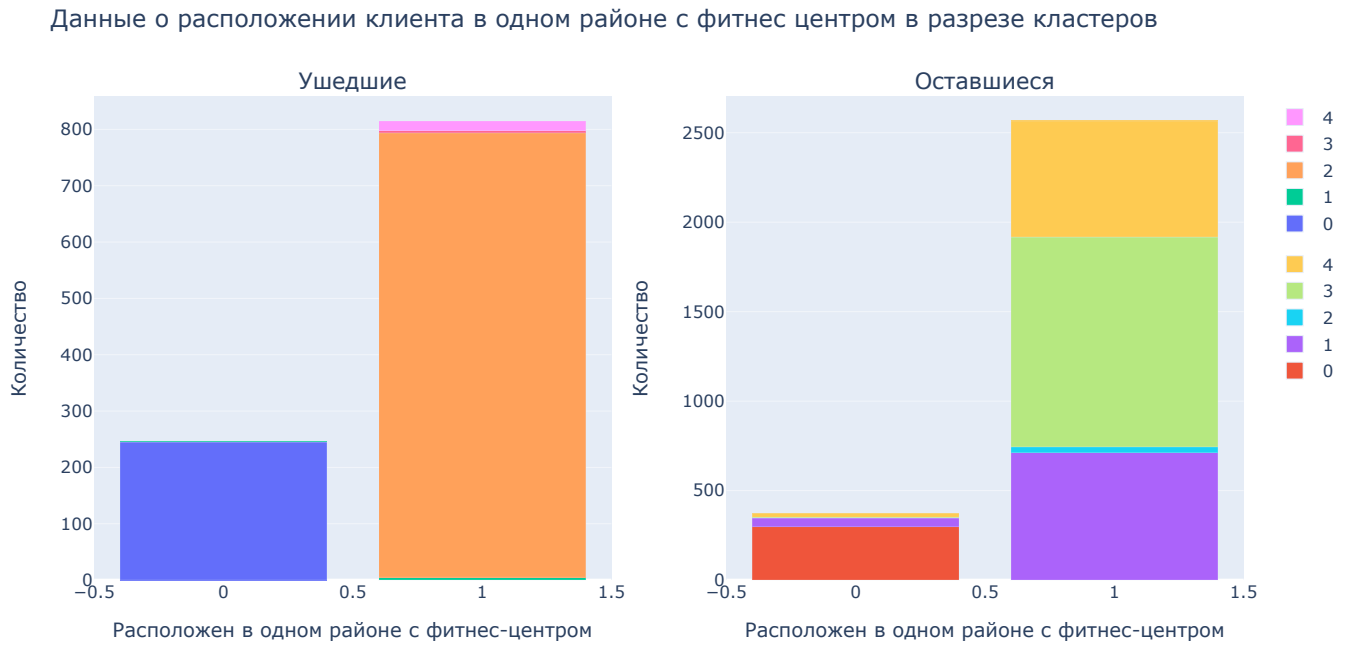
Ввод [45]:
```
def plot_stack_cont(numcol,bins,x_label,y_label,t_label):
    kat=[0,1,2,3,4]
    fig = make_subplots(rows=1, cols=2,subplot_titles=('Ушедшие',  'Оставшиеся'))
    dd=pd.cut(df[numcol],bins=bins,labels=False)
    dd.name='bin'
    dftmp=df.join(dd)
    mv=dftmp[numcol].max()/bins

    for kat in kat:
        dftmp_c1=dftmp[(dftmp['cluster_km']==kat) & (dftmp['Churn']==1)]
        dftmp_c1_g=(dftmp_c1.groupby('bin')[numcol].count())
        x=(dftmp_c1_g.index+1)*mv
        y=dftmp_c1_g.values
        fig.add_trace(go.Bar(x=x, y=y, name=kat,legendgroup=1),row=1,col=1)

        dftmp_c0=dftmp[(df['cluster_km']==kat) & (dftmp['Churn']==0)]
        dftmp_c0_g=(dftmp_c0.groupby('bin')[numcol].count())
        x=(dftmp_c1_g.index+1)*mv
        y=dftmp_c0_g.values
        fig.add_trace(go.Bar(x=x, y=y, name=kat,legendgroup=2),row=1,col=2)

    fig.update_layout(barmode='stack',title_text=t_label)
    fig.update_xaxes(title_text=x_label,row=1,col=1)
    fig.update_xaxes(title_text=x_label,row=1,col=2)
    fig.update_yaxes(title_text=y_label,row=1,col=1)
    fig.update_yaxes(title_text=y_label,row=1,col=2)
    fig.show()
```

Ввод [46]:
```
plot_stack('gender','Пол','Пол в разрезе кластеров','count')
```



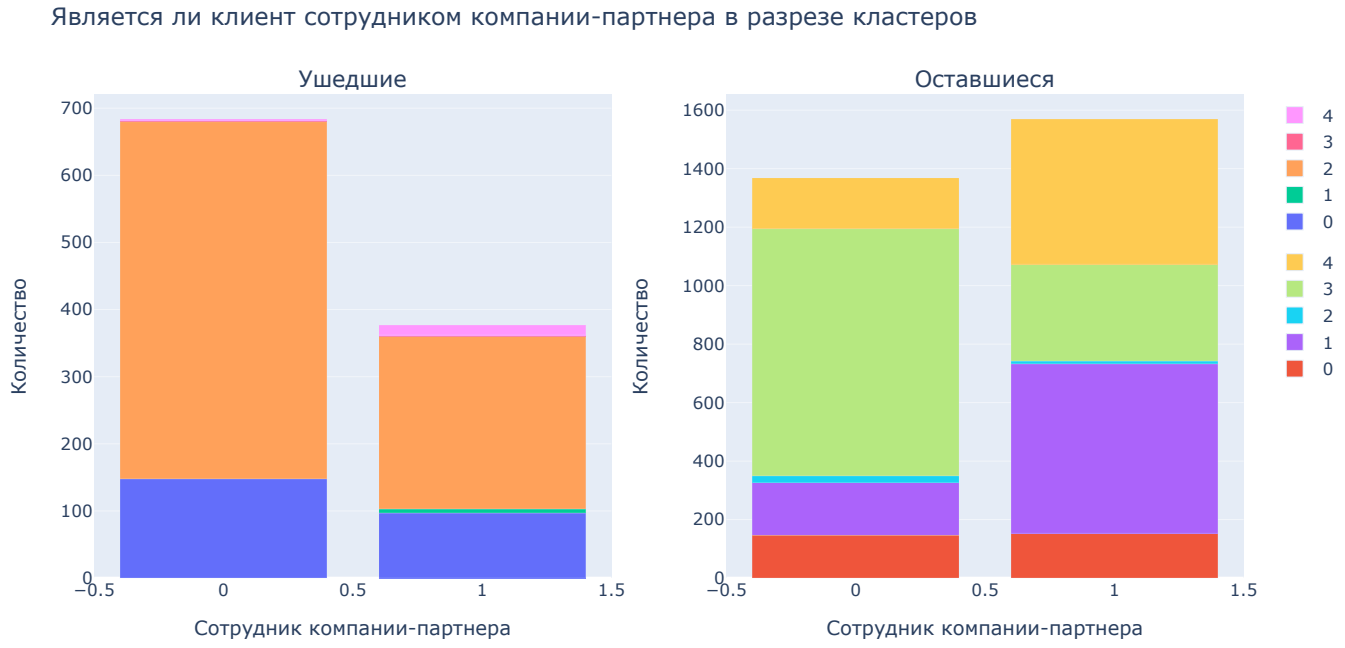

Как мы видели ранее пол не влияет на факт ухода. И среди ушедших преобладает кластер №2.

Ввод [47]:
```
plot_stack('Near_Location','Расположен в одном районе с фитнес-центром',
           'Данные о расположении клиента в одном районе с фитнес центром в разрезе кластеров','count')
```



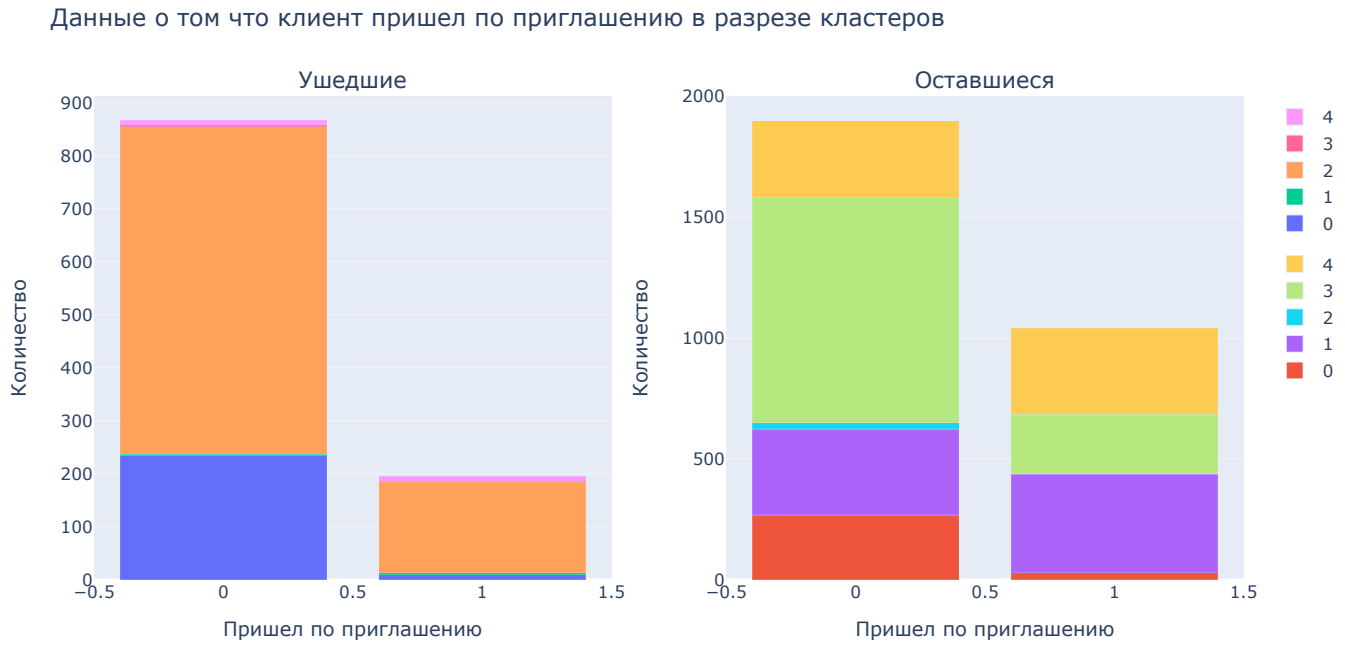

Среди ушедших кластер 2 полностью жил/работал в одном районе с фитнес-центром, а кластер 0 полностью нет.

Ввод [48]:
```
plot_stack('Partner','Сотрудник компании-партнера',
           'Является ли клиент сотрудником компании-партнера в разрезе кластеров','count')
```



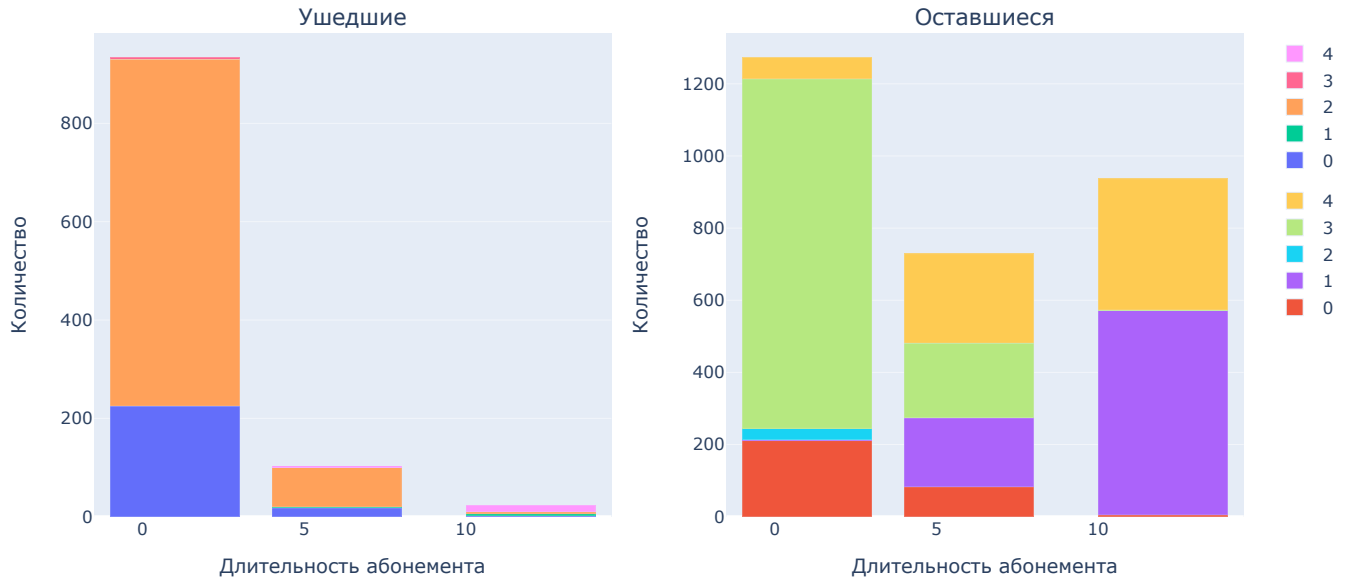

Уходят скорее не сотрудники компании-партнера.

Ввод [49]:
```
plot_stack('Promo_friends','Пришел по приглашению','Данные о том что клиент пришел по приглашению в разрезе кластеров','count')
```



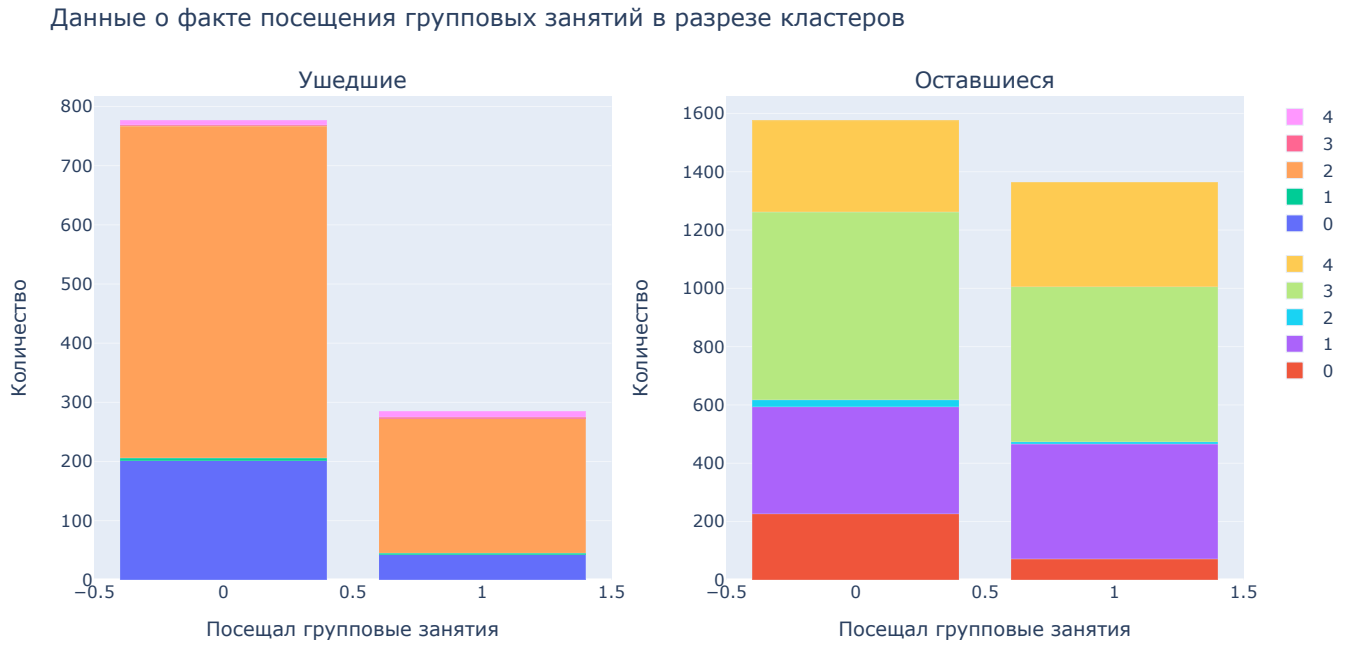

Как мы уже видели, уходят клиенты пришедшие не в рамках акции.

Ввод [50]:
```
plot_stack('Contract_period','Длительность абонемента','Длительность абонемента в разрезе кластеров','count')
```



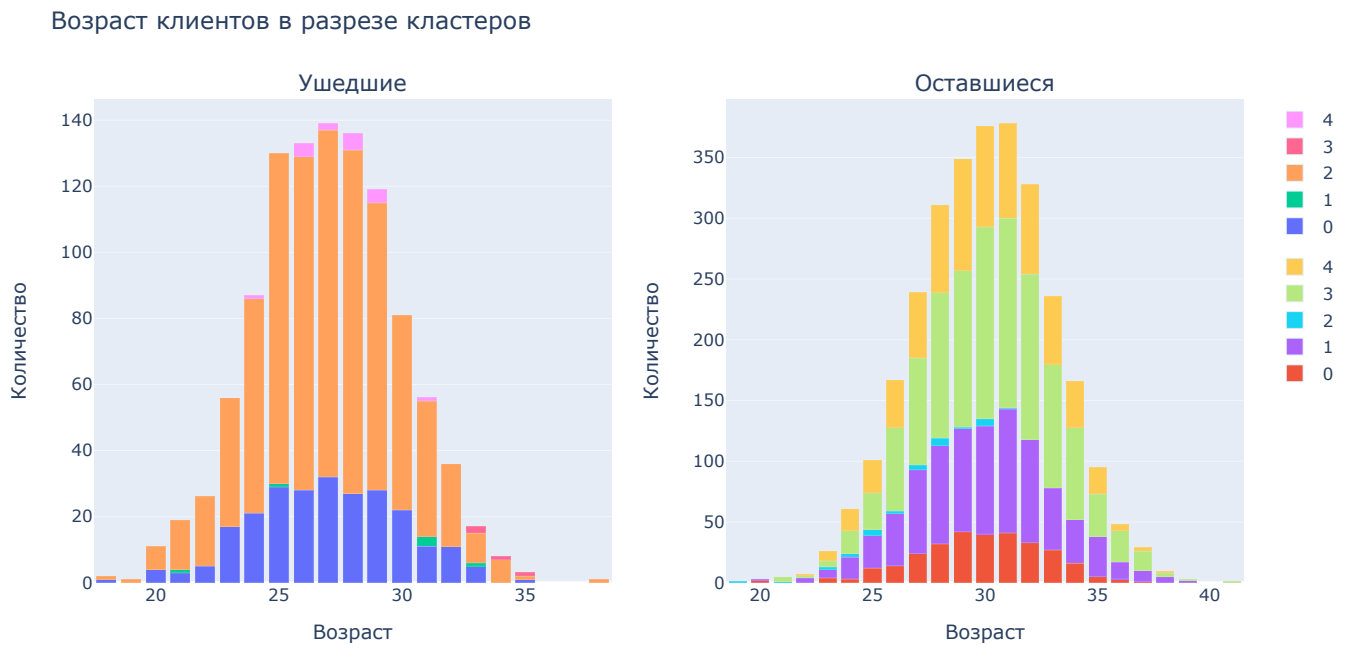

Интересно, что среди оставшихся кластер 3, самые лояльные клиенты тоже сосредоточенны в коротких контрактах, как и ушедшие.

Ввод [51]:
```
plot_stack('Group_visits','Посещал групповые занятия','Данные о факте посещения групповых занятий в разрезе кластеров','count')
```



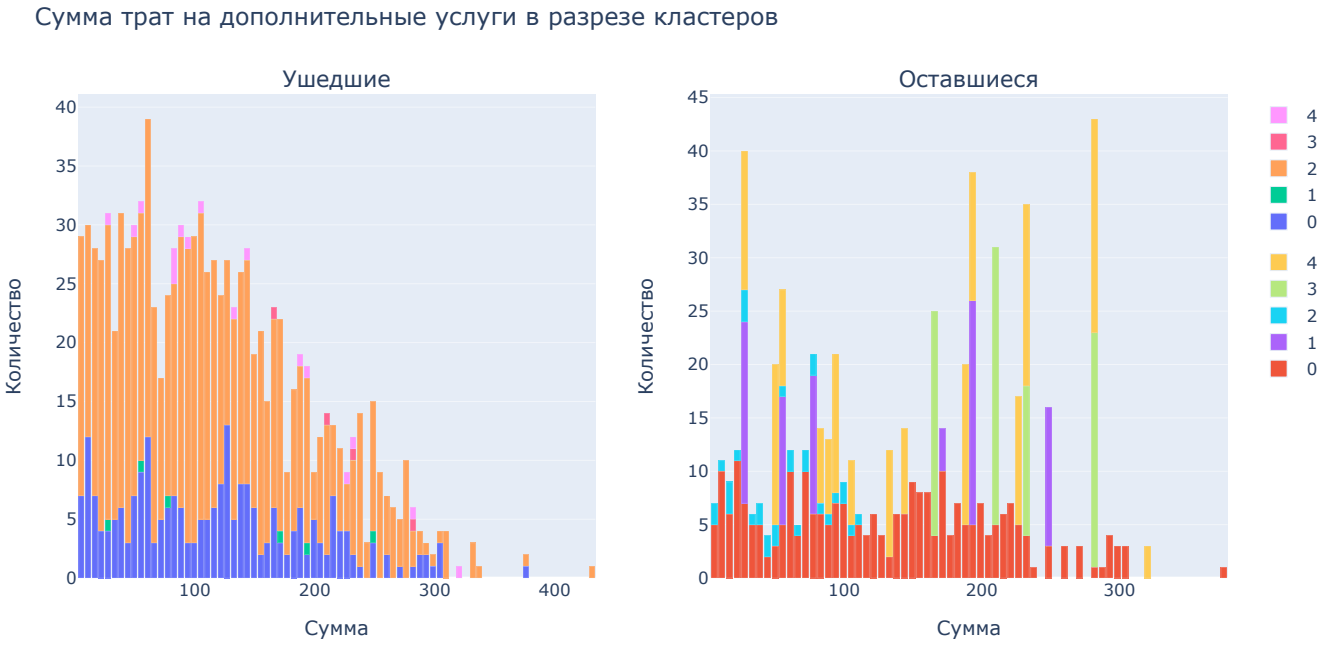

Понятно, сначала игнорировал групповые занятия, затем ушел. В группе оставшихся факт посещения или не посещения групповых занятий распределен по кластерам равномерно.

Ввод [52]:
```
plot_stack('Age','Возраст','Возраст клиентов в разрезе кластеров','count')
```



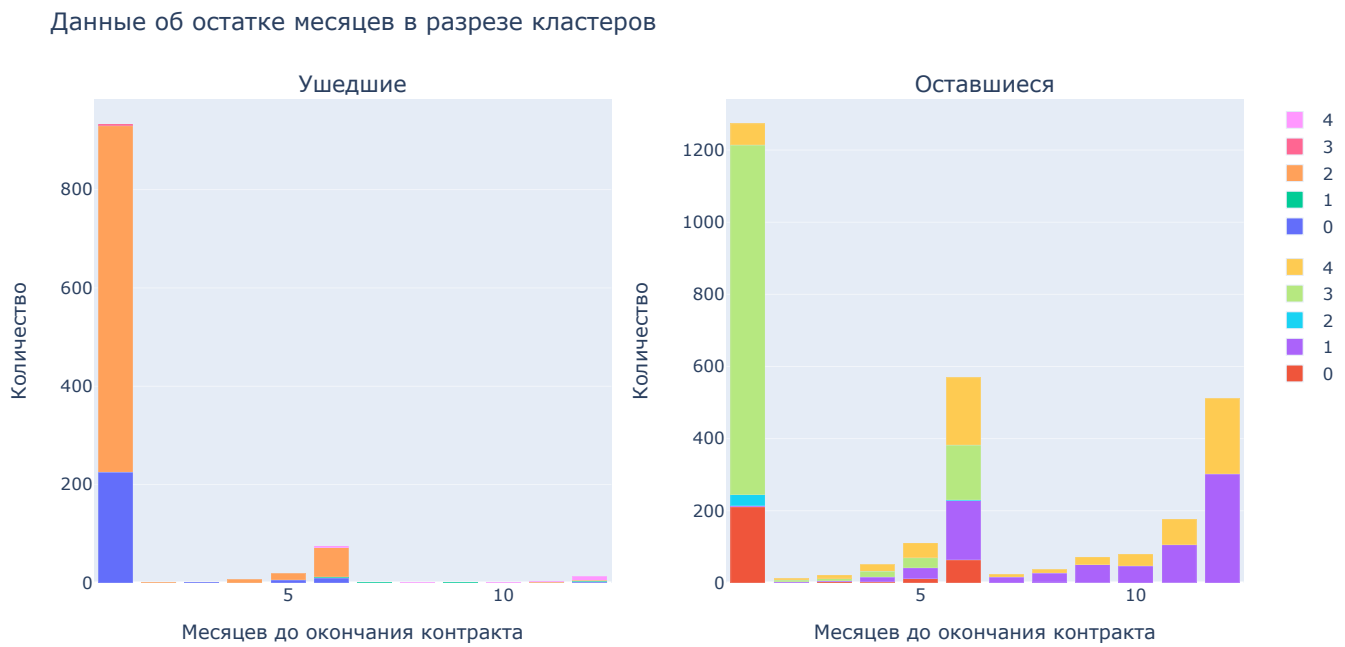

Тут тоже ничего нового, в группе ушедших доминирнуют кластеры 2 и 0, в группе оставшихся более ли менее равномерное распределение.

Ввод [53]:
```
plot_stack_cont('Avg_additional_charges_total',100,'Сумма','Количество',
                'Сумма трат на дополнительные услуги в разрезе кластеров')
```



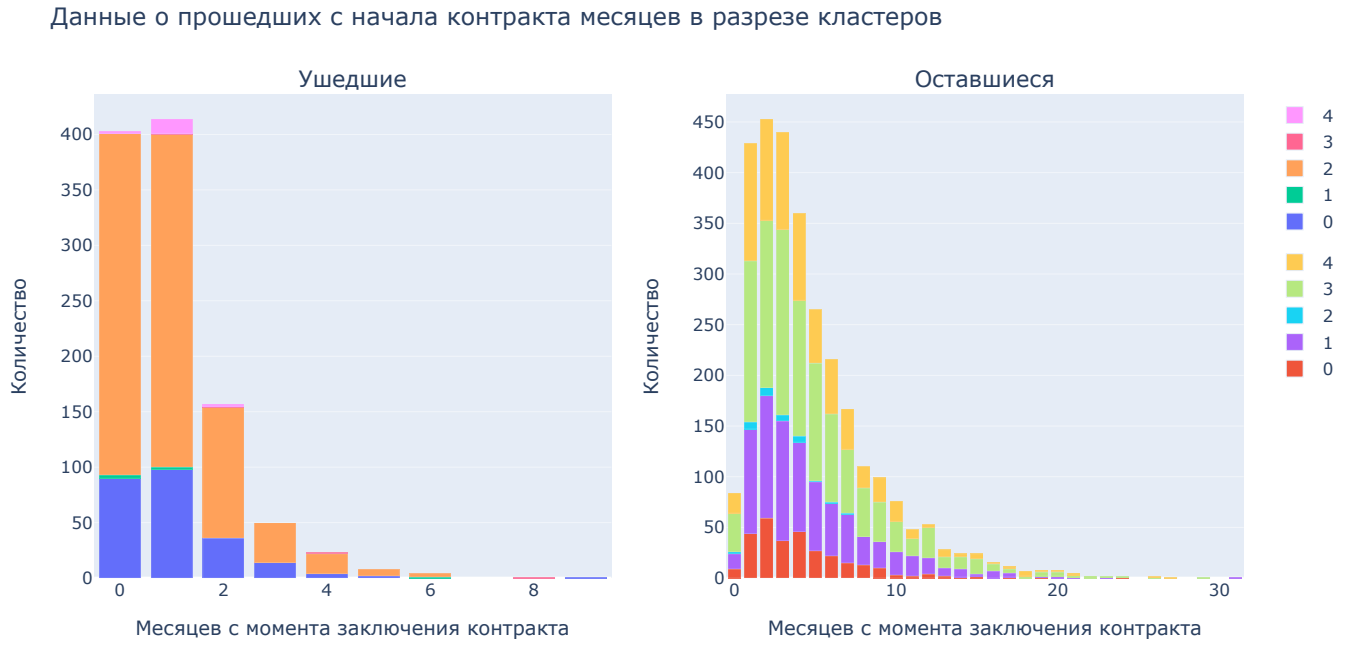

Этот график я построил, отобразив по оси у количество фактов покупки, а не общую сумму. И здесь мы можем видеть, что ушедшие совершали, по количеству, больше покупок, чем осташиеся. Т.е. они приходили в фитнес-зал кофе выпить в баре.

Ввод [54]:
```
plot_stack('Month_to_end_contract','Месяцев до окончания контракта','Данные об остатке месяцев в разрезе кластеров','count')
```

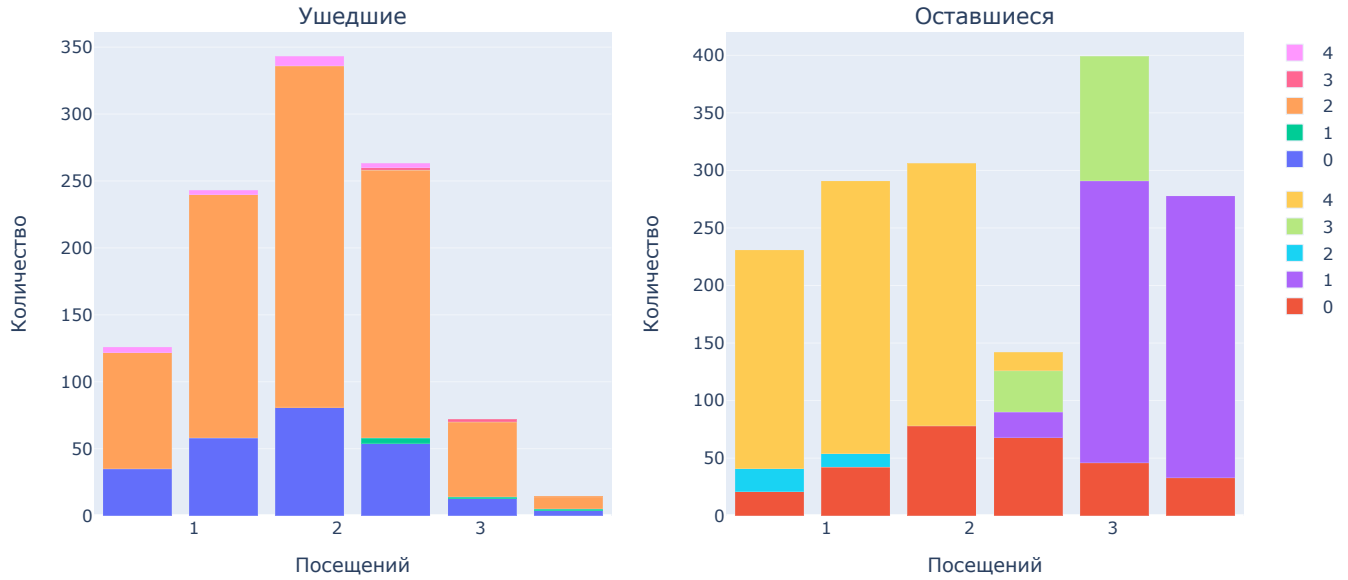



И опять, самый лояльный среди оставшихся кластер №3 сосредоточен в коротких контрактах.

Ввод [55]:
```
plot_stack('Lifetime','Месяцев с момента заключения контракта',
           'Данные о прошедших с начала контракта месяцев в разрезе кластеров','count')
```

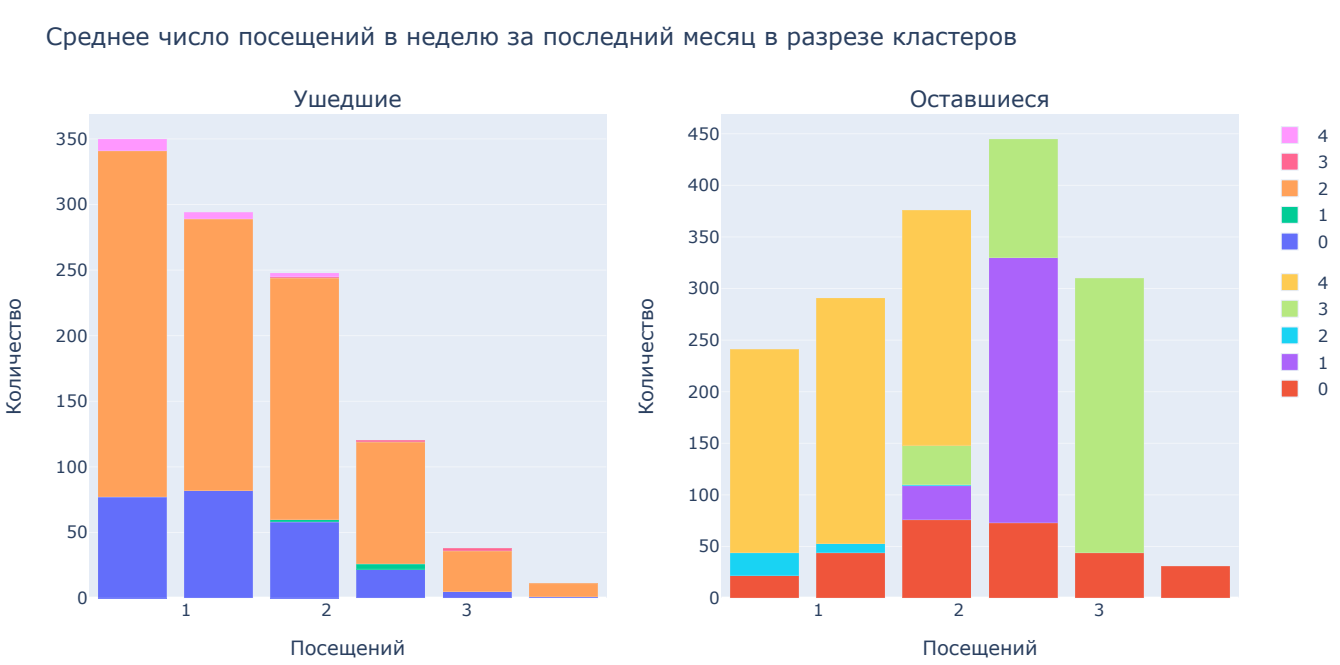



Ввод [56]:
```
plot_stack_cont('Avg_class_frequency_total',10,'Посещений','Количество',
                'Среднее число посещений в неделю за весь период в разрезе кластеров')
```




Если по ушедшим все как обычно, то по оставщимся можно отметить что самые лояльные кластеры 3 и 1 сосредоточенны в числе посещений от 3 и больше

Ввод [57]:
```
plot_stack_cont('Avg_class_frequency_current_month',10,'Посещений','Количество',
                'Среднее число посещений в неделю за последний месяц в разрезе кластеров')
```




Понятно, что если большинство ушедших попали в кластеры 2 и 0, то и отображении данных в разрезе кластеров для ушедших будут преобладать эти два кластера. Но для оставшихся разбиение по кластерам не очень информативное. Очевидно, что разные параметры имеют максимальную популярность для разных кластеров, но картина не очень четкая. Однозначно ничего нельзя сказать.

Ввод [ ]:
```
```

Ввод [58]:
```
display(df[df.cluster_km==2].describe())
```

| | gender | Near_Location | Partner | Promo_friends | Phone | Contract_period | Group_visits | Age | Avg_additional_charges_total | Month_to_end_ |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 821.000000 | 821.0 | 821.000000 | 821.000000 | 821.000000 | 821.000000 | 821.000000 | 821.000000 | 821.000000 | 821 |
| **mean** | 0.507917 | 1.0 | 0.323995 | 0.213155 | 0.903776 | 1.546894 | 0.287454 | 26.926918 | 113.173051 | 1 |
| **std** | 0.500242 | 0.0 | 0.468283 | 0.409786 | 0.295078 | 1.661415 | 0.452851 | 2.844682 | 77.951173 | 1 |
| **min** | 0.000000 | 1.0 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 18.000000 | 0.148205 | 1 |
| **25%** | 0.000000 | 1.0 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 25.000000 | 49.278551 | 1 |
| **50%** | 1.000000 | 1.0 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 27.000000 | 100.543784 | 1 |
| **75%** | 1.000000 | 1.0 | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 29.000000 | 162.873354 | 1 |
| **max** | 1.000000 | 1.0 | 1.000000 | 1.000000 | 1.000000 | 12.000000 | 1.000000 | 38.000000 | 425.535220 | 12 |

Ввод [ ]:
```
```

Ввод [ ]:
```
```

Ввод [ ]:
```
```

Ввод [ ]:
```
```

Ввод [ ]:
```
```

## Выводы

В работе были проведены исследовательский анализ данных, построены две модели прогнозирования оттока клиентов и произведена кластеризация клиентов для определения поведенческих групп клиентов и отличительных признаков клиентов из оттока.

В исследовательском анализе данных мы отметили, что факторами оттока являются:

- приобретение абонемента на месячный срок;
- большинство клиентов не посещают групповые занятия;
- основное распределение трат в диапазоне от 0 до 150 у.е., далее идет на снижение и совсем редко превышает 400 у.е.;
- у основной массы до конца абонемента остается 1 мес.;
- распределение продолжительности посещения в основном сосредоточено на отметке в 1-2 месяца, далее спад;
- отсутствие специальных льготных условий обслуживания: компания-партнер или акция "приведи друга".

На основе данных мы обучили две модели по алгоритмам логистической регрессии и случайного леса. Модель логистической регрессии на валидационной выборке выдала наилучшие метрики. В дальнейшем для прогнозирования оттока клиентов можно применять именно ее.

Также все клиенты были поделены на кластеры с помощью алгоритмов агломеративной иерархической кластеризации и K-Means. Дополнительно мы отметили, что:

- количество дополнительных трат в 2 раза ниже показателей по лояльным клиентам;
- "срок жизни" клиентов до 5 месяцев, но в большинстве случаев они ходят в клуб 1-2 месяца;
- посещения около 2,5 раза в неделю, в последний месяц снижаются до 0;
- мужчин и женщин поровну;
- средний возраст - около 27 лет.

Рекомендуемые меры по удержанию клиентов:

- изменение ценовой политики, способствующее приобретению новыми клиентам абонементы на длительный срок (от 6 месяцев);
- проведение промо-акций по групповым занятиям;
- анализ и внедрение товаров и услуг с более высоким спросом, а возможно и пересмотр ценовой политики на доп. товары и услуги. Можно убрать из ассортимента дешевые товары, превращающие фитнес-клуб в кофейню, но с другой стороны доход такие клиенты приносят, и если их не станет, общий доход может упасть;
- развитие спецпрограмм для новых клиентов.

Ввод [ ]: