Прогнозирование оттока клиентов

Описание проекта

Сеть фитнес-центров «Культурист-датасаентист» разрабатывает стратегию взаимодействия с пользователями на основе аналитических данных.

Чтобы бороться с оттоком, отдел по работе с клиентами «Культуриста-датасаентиста» перевёл в электронный вид множество анкет пользователей. Ваша задача — провести анализ и подготовить план действий по удержанию клиентов.

Цель проекта

- научиться прогнозировать вероятность оттока (на уровне следующего месяца) для каждого клиента;
- сформировать типичные портреты пользователей: выделить несколько наиболее ярких групп и охарактеризовать их основные свойства;
- проанализировать основные признаки, наиболее сильно влияющие на отток;
- сформулировать основные выводы и разработать рекомендации по повышению качества работы с клиентами:
  - 1) выделить целевые группы клиентов;
  - 2) предложить меры по снижению оттока;
  - 3) определить другие особенности взаимодействия с клиентами.

Описание данных

Заказчик предоставил данные на месяц до оттока и факт оттока на определённый месяц.

Данные хранятся в файле `/datasets/gym_churn.csv`

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.

Оглавление

1. Исследовательский анализ данных (EDA)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
```

In [2]:
```
# Workaround for different OS
try:
    df = pd.read_csv('/datasets/gym_churn.csv')
except:
    df = pd.read_csv('datasets/gym_churn.csv')
```

In [3]:
```
df.head().T
```

Out[3]:

	0	1	2	3	4
gender	1.000000	0.000000	0.000000	0.000000	1.000000
Near_Location	1.000000	1.000000	1.000000	1.000000	1.000000
Partner	1.000000	0.000000	1.000000	1.000000	1.000000
Promo_friends	1.000000	0.000000	0.000000	1.000000	1.000000
Phone	0.000000	1.000000	1.000000	1.000000	1.000000
Contract_period	6.000000	12.000000	1.000000	12.000000	1.000000
Group_visits	1.000000	1.000000	0.000000	1.000000	0.000000
Age	29.000000	31.000000	28.000000	33.000000	26.000000
Avg_additional_charges_total	14.227470	113.202938	129.448479	62.669863	198.362265
Month_to_end_contract	5.000000	12.000000	1.000000	12.000000	1.000000
Lifetime	3.000000	7.000000	2.000000	2.000000	3.000000
Avg_class_frequency_total	0.020398	1.922936	1.859098	3.205633	1.113884
Avg_class_frequency_current_month	0.000000	1.910244	1.736502	3.357215	1.120078
Churn	0.000000	0.000000	0.000000	0.000000	0.000000

Данные прочитаны, но названия столбцов для удобства можно перевести в нижний регистр.

In [4]:
```
df.columns = [x.lower() for x in df.columns]
```

In [5]:
```
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
```

Все типы данных числовые, временных данных нет, преобразования не требуется. Проверим дубликаты и пропуски.

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

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

In [7]:
```
df.isna().mean()
```

Out[7]:
```
gender                             0.0
near_location                      0.0
partner                            0.0
promo_friends                      0.0
phone                              0.0
contract_period                    0.0
group_visits                       0.0
age                                0.0
avg_additional_charges_total       0.0
month_to_end_contract              0.0
lifetime                           0.0
avg_class_frequency_total          0.0
avg_class_frequency_current_month  0.0
churn                              0.0
dtype: float64
```

Дубликатов и пропусков нет.

In [8]:
```
df.shape
```

Out[8]:
```
(4000, 14)
```

Всего 4000 наблюдений для 13 признаков и 1 целевой переменной `churn`. Посмотрим на общее распределение значений.

In [9]:
```
df.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
gender	4000.0	0.510250	0.499957	0.000000	0.000000	1.000000	1.000000	1.000000
near_location	4000.0	0.845250	0.361711	0.000000	1.000000	1.000000	1.000000	1.000000
partner	4000.0	0.486750	0.499887	0.000000	0.000000	0.000000	1.000000	1.000000
promo_friends	4000.0	0.308500	0.461932	0.000000	0.000000	0.000000	1.000000	1.000000
phone	4000.0	0.903500	0.295313	0.000000	1.000000	1.000000	1.000000	1.000000
contract_period	4000.0	4.681250	4.549706	1.000000	1.000000	1.000000	6.000000	12.000000
group_visits	4000.0	0.412250	0.492301	0.000000	0.000000	0.000000	1.000000	1.000000
age	4000.0	29.184250	3.258367	18.000000	27.000000	29.000000	31.000000	41.000000
avg_additional_charges_total	4000.0	146.943728	96.355602	0.148205	68.868830	136.220159	210.949625	552.590740
month_to_end_contract	4000.0	4.322750	4.191297	1.000000	1.000000	1.000000	6.000000	12.000000
lifetime	4000.0	3.724750	3.749267	0.000000	1.000000	3.000000	5.000000	31.000000
avg_class_frequency_total	4000.0	1.879020	0.972245	0.000000	1.180875	1.832768	2.536078	6.023668
avg_class_frequency_current_month	4000.0	1.767052	1.052906	0.000000	0.963003	1.719574	2.510336	6.146783
churn	4000.0	0.265250	0.441521	0.000000	0.000000	0.000000	1.000000	1.000000

Можно отметить, что в целевом показатели `churn` **всего 27% тех, кто перестал посещать заведение**, остальные 73% продолжают посещать. Классы пользователей не сбалансированы, поэтому это стоит учесть при выборе метрик моделей.

По среднему и процентилям можно заключить, что большинство посетителей:

- живут рядом (`near_location`),
- пришли не по акции (`promo_friend`),
- оставили телефон для связи (`phone`),
- заключили контракт на полгода или меньший срок (`contract_period`),
- новички в клубе (`lifetime`),
- среднего возраста около 30 лет (`age`).

Посмотрим на разницу между ушедшими и оставшимися пользователями.

In [10]:
```
df.groupby('churn').mean().T
```

Out[10]:

churn	0	1
gender	0.510037	0.510839
near_location	0.873086	0.768143
partner	0.534195	0.355325
promo_friends	0.353522	0.183789
phone	0.903709	0.902922
contract_period	5.747193	1.728558
group_visits	0.464103	0.268615
age	29.976523	26.989632
avg_additional_charges_total	158.445715	115.082899
month_to_end_contract	5.283089	1.662582
lifetime	4.711807	0.990575
avg_class_frequency_total	2.024876	1.474995
avg_class_frequency_current_month	2.027882	1.044546

Даже без машинного обучения можно отметить, что ушедшие пользователи:

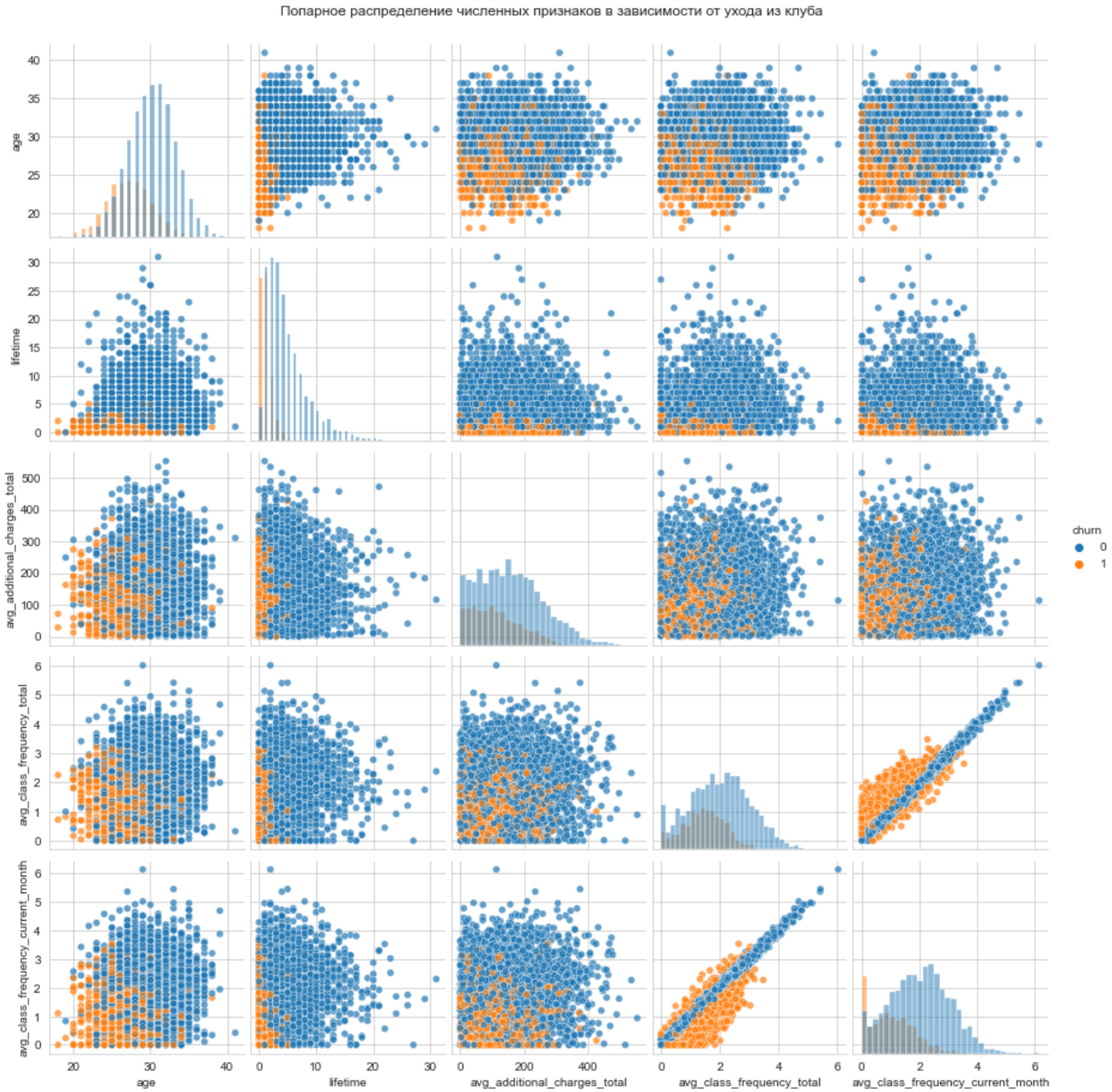
- чуть чаще живут подальше (`near_location`),
- чаще оставшихся посещали клуб без корпоративной программы и "без друга" (`partner` и `promo_friend`),
- заключали короткий договор (`contract_period`),
- немного моложе оставшихся (`age`),
- меньше тратили на дополнительные услуги (`avg_additional_charges_total`),
- новички в клубе (`lifetime`),
- меньше посещали клуб в целом и в текущий месяц (`avg_class_frequency_total`).

Посмотрим на распределение признаков.

In [11]:
```
sns.set_style('whitegrid')
```

In [12]:
```
numeric_cols = ['age',
                'lifetime',
                'avg_additional_charges_total',
                'avg_class_frequency_total',
                'avg_class_frequency_current_month']
```

In [13]:
```
ax = sns.pairplot(df,
             vars=numeric_cols,
             hue='churn',
             diag_kind='hist',
             plot_kws={'alpha': 0.7});
ax.fig.suptitle('Попарное распределение численных признаков в зависимости от ухода из клуба', y=1.02);
```

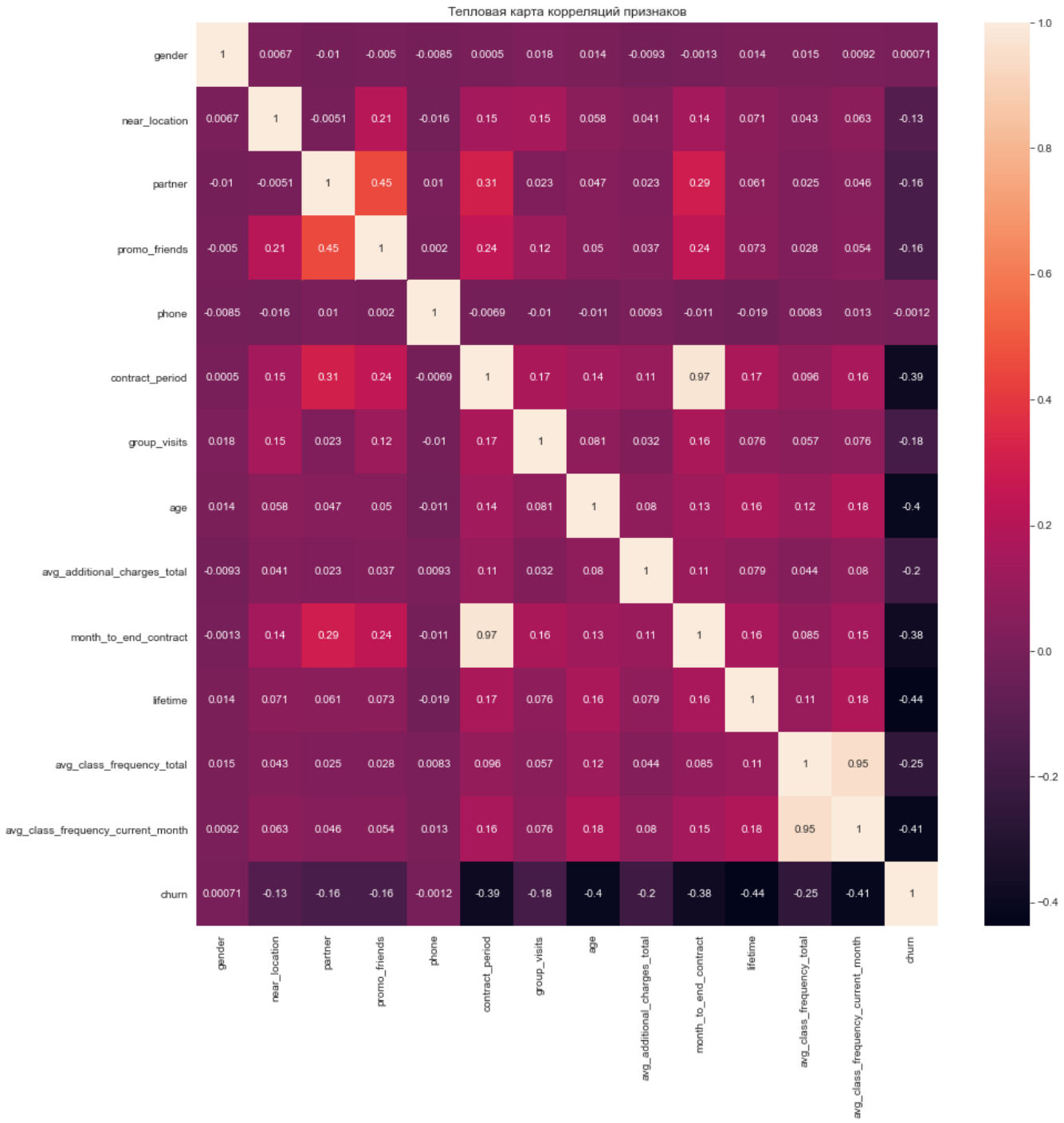


Как видно по распределениям, есть разница в оставшихся и ушедших клиентах по общему времени посещения клуба и возрасту. Признаки частоты посещения за все время и в последний месяц сильно коррелированы. Это следует учесть при выборе модели.

Посмотрим также на корреляцию признаков.

In [14]:
```
corr_matrix = df.corr()
```

In [15]:
```
plt.figure(figsize=(15,15));
plt.title('Тепловая карта корреляций признаков');
sns.heatmap(corr_matrix, annot=True);
plt.show();
```



Как видно, часть признаков для периода `contract_period` и `month_to_end_contract`, `avg_class_frequency_total` и `avg_class_frequency_current_month` коррелированы между собой. В то же время, другой сильной линейной корреляции не прослеживается между признаками.

2. Создание модели прогнозирования оттока клиентов

С точки зрения машинного обучения для прогнозирования оттока нам нужна модель для обучения с учителем, а именно классификации по целевому признаку оттока (`churn` , ушел или нет клиент), а значит **бинарная классификация**.

Разобъем данные на обучающую и валидационную выборки. Так как в данных нет временных рядов, можно это сделать случайным образом.

In [16]:
```
X = df.drop('churn', axis=1)
```

In [17]:
```
y = df['churn']
```

Выделим, например, 20% наблюдений в валидационную выборку.

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [19]:
```
len(X_train), len(X_test)
```

Out[19]:
```
(3200, 800)
```

Для выбора модели сравним результаты моделирания двумя популярными алгоритмами для задача классификации:

- логистическая регрессия
- случайный лес

In [20]:
```
model_lr = LogisticRegression(n_jobs=-1, random_state=42)
```

In [21]:
```
model_forest = RandomForestClassifier(n_jobs=-1, random_state=42)
```

Как мы отметили, классы не сбалансированы (большая часть клиентов не уходит) и часть признаком мультиколлинеарны. Для выбора лучшей метрики расчитаем следующие:

- accuracy
- precision
- recall
- ROC-AUC

In [22]:
```
model_lr.fit(X_train, y_train)
model_forest.fit(X_train, y_train)
```

Out[22]:
```
RandomForestClassifier(n_jobs=-1, random_state=42)
```

Создадим вспомогательную функцию для расчетов метрик.

In [23]:
```
def get_metrics(y_test, y_pred, y_score):
    accuracy_score(y_test, y_pred)
    precision_score(y_test, y_pred)
    recall_score(y_test, y_pred)
    roc_auc_score(y_test, y_score[:, 1])

    print('Accuracy:{:.2f}'.format(accuracy_score(y_test, y_pred)))
    print('Precision:{:.2f}'.format(precision_score(y_test, y_pred)))
    print('Recall:{:.2f}'.format(recall_score(y_test, y_pred)))
    print('ROC-AUC:{:.2f}'.format(roc_auc_score(y_test, y_score[:, 1])))
```

In [24]:
```
get_metrics(y_test, model_lr.predict(X_test), model_lr.predict_proba(X_test))
```

```
Accuracy:0.91
Precision:0.86
Recall:0.77
ROC-AUC:0.96
```

In [25]:
```
get_metrics(y_test, model_forest.predict(X_test), model_forest.predict_proba(X_test))
```

```
Accuracy:0.91
Precision:0.85
Recall:0.78
ROC-AUC:0.96
```

Как видно по метрикам обе модели показывают в целом одинаковый результат (ROC AUC 0.96), но случайный лес показал немногим лучший Recall, чем логистическая регрессия (0.78 против 0.77).

В случае подбора метрики для поиска оттока клиентов важнее не пропустить такого клиента, чем ошибится и принять постоянного клиента за уходящего. В первом случае потери для бизнеса будут значительны (в размере средней выручки с клиента), а во втором - нет, если учесть, что предпринимаемые действия адекватны и не отторгнут постоянного клиента.

С учетом описанного важно ориентироваться на метрику Recall, так как она выше ценит то, что мы не пропустили клиентов 1 класса, а не то, сколько ошибок мы сделали. Таким образом, **модель случайного леса показала себя лучше на основании этой метрики в задаче поиска классификатора для оттока клиентов**.

Для возможности прогнозирования ухода клиентов также полезно узнать какие факторы повлияли больше других на решение по уходу клиентов. Это легко узнать из результатов обучения модели, так как наиболее важные признаки получают больший "вес".

In [26]:
```
pd.DataFrame(index=X.columns,
             data={'fi':model_forest.feature_importances_}).sort_values('fi', ascending=False)
```

Out[26]:

	fi

	fi
lifetime	0.279562
avg_class_frequency_current_month	0.171566
age	0.129747
avg_class_frequency_total	0.127383
avg_additional_charges_total	0.086698
month_to_end_contract	0.073842
contract_period	0.065251
group_visits	0.014624
gender	0.011742
partner	0.011303
promo_friends	0.010726
near_location	0.010471
phone	0.007085

In [27]:
```
pd.DataFrame(index=X.columns,
             data={'fi':model_forest.feature_importances_}).sort_values('fi').plot(kind='barh');
plt.title('Важность признаков для модели случайных деревьев');
```

Как видно, модель выделила следующие признаки:

- общее время клиента в клубе
- частота посещения групповых занятий в прошлом месяце и всего
- возраст
- размер дополнительных трат в клубе
- длительность контракта и насколько он близок к окончанию

3. Кластеризация клиентов

Для выделения отдельных групп клиентов, которые требуют специального внимания, например, ключевые клиенты, основные клиенты и уходящие клиенты, полезно проводить сегментацию (кластеризацию). В этом помогут алгоритмы машинного обучения без учителя. В то же время, для их правильной работы требуется приведение всех признаков в численный формат и нормализация их для возможности рассчета "расстояний" между ними и тем самым выделения сегментов (кластеров).

Стандартизируем данные для подготовки к кластеризации.

In [28]:
```
sc = StandardScaler()
```

In [29]:
```
X_sc = sc.fit_transform(df)
```

In [30]:
```
X_sc
```

Out[30]:
```
array([[ 0.97970588,  0.42788074,  1.02686062, ..., -1.91191971,
        -1.67847198, -0.6008387 ],
```

```
[-1.0207145 ,  0.42788074, -0.973842  , ...,  0.04517569,
  0.1360137 , -0.6008387 ],
[-1.0207145 ,  0.42788074,  1.02686062, ..., -0.02049263,
 -0.02901851, -0.6008387 ],
...,
[ 0.97970588,  0.42788074,  1.02686062, ...,  0.93313947,
  1.01103141, -0.6008387 ],
[-1.0207145 ,  0.42788074,  1.02686062, ..., -0.25604937,
 -0.16225246, -0.6008387 ],
[ 0.97970588, -2.33709981,  1.02686062, ..., -0.79947418,
 -0.69509885, -0.6008387 ]])
```

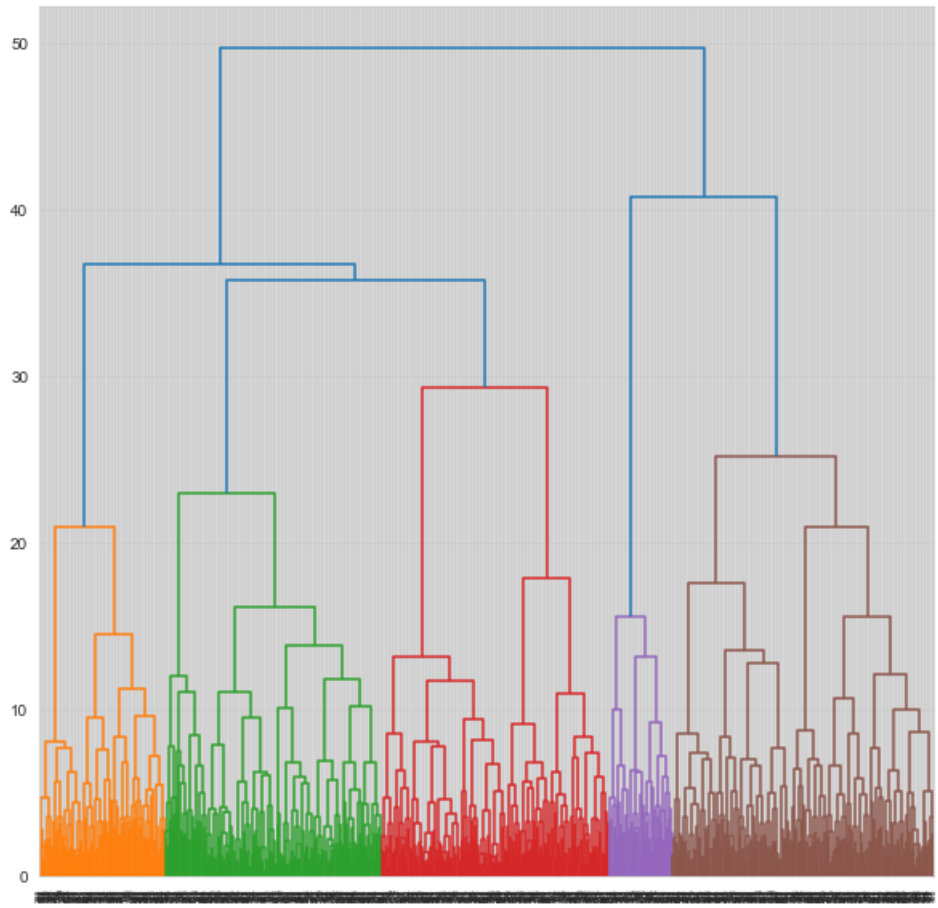
Получили массив признаков, отнормированный по нормальному распределению с центром в 0 и средним 1.

Для оценки возможного разбиения наблюдений на сегменты (кластеры) полезно увидеть как признаки могут разделиться на основе евклидова расстояния между ними. Построим матрицу расстояний (`linkage`). Так как визуализация расстояний между всеми точками может занять много времени, выберем случайным образом 20% данных для этого.

In [31]:
```
linked = linkage(sc.fit_transform(df.sample(frac=0.2, random_state=42)), method = 'ward')
```

Для визуализации расстояний удобно использовать дендрограмму, построим ее.

In [32]:
```
plt.figure(figsize=(10, 10));
dendrogram(linked, orientation='top');
plt.show();
```



Как видно по дендрограмме оптимальным разбиением видятся 5 кластеров (цвета в основании дендрограммы). Разделим всю выборку на 5 кластеров методом K средних (K-Means).

In [33]:
```
km = KMeans(n_clusters = 5)
```

In [34]:
```
clusters = km.fit_predict(X_sc)
```

Добавим разбиение в исходные данные.

In [35]:
```
df['cluster'] = clusters
```

In [36]:
```
df.head().T
```

Out[36]:

	0	1	2	3	4

	0	1	2	3	4
gender	1.000000	0.000000	0.000000	0.000000	1.000000
near_location	1.000000	1.000000	1.000000	1.000000	1.000000
partner	1.000000	0.000000	1.000000	1.000000	1.000000
promo_friends	1.000000	0.000000	0.000000	1.000000	1.000000
phone	0.000000	1.000000	1.000000	1.000000	1.000000
contract_period	6.000000	12.000000	1.000000	12.000000	1.000000
group_visits	1.000000	1.000000	0.000000	1.000000	0.000000
age	29.000000	31.000000	28.000000	33.000000	26.000000
avg_additional_charges_total	14.227470	113.202938	129.448479	62.669863	198.362265
month_to_end_contract	5.000000	12.000000	1.000000	12.000000	1.000000
lifetime	3.000000	7.000000	2.000000	2.000000	3.000000
avg_class_frequency_total	0.020398	1.922936	1.859098	3.205633	1.113884
avg_class_frequency_current_month	0.000000	1.910244	1.736502	3.357215	1.120078
churn	0.000000	0.000000	0.000000	0.000000	0.000000
cluster	0.000000	3.000000	1.000000	3.000000	1.000000

In [37]:
```
df['cluster'].value_counts()
```

Out[37]:
```
3    959
2    941
4    863
1    855
0    382
Name: cluster, dtype: int64
```

В результате получились примерно одинаковые по размеру кластеры, кроме последнего. Проверим средние значения параметров по кластерам.

In [38]:
```
df.groupby('cluster').mean().T
```

Out[38]:

cluster	0	1	2	3	4
gender	0.526178	0.477193	0.506908	0.503650	0.546929
near_location	0.869110	0.824561	0.759830	0.938478	0.844728
partner	0.471204	0.459649	0.341126	0.767466	0.367323
promo_friends	0.308901	0.264327	0.179596	0.563087	0.209733
phone	0.000000	1.000000	0.996812	1.000000	0.998841
contract_period	4.816754	2.761404	1.566419	11.199166	2.676709
group_visits	0.429319	0.385965	0.257173	0.558916	0.436848
age	29.340314	30.028070	26.907545	29.880083	29.988413
avg_additional_charges_total	144.517762	151.348090	114.993470	162.792871	160.879827
month_to_end_contract	4.502618	2.552047	1.513284	10.254432	2.469293
lifetime	3.955497	4.688889	0.975558	4.683003	4.600232
avg_class_frequency_total	1.857525	1.157304	1.448150	2.031016	2.904467
avg_class_frequency_current_month	1.727260	1.155556	1.025562	2.025194	2.912142
churn	0.261780	0.001170	0.997875	0.015641	0.006952

Как видно по средним значениям в **первом классе (0)**:

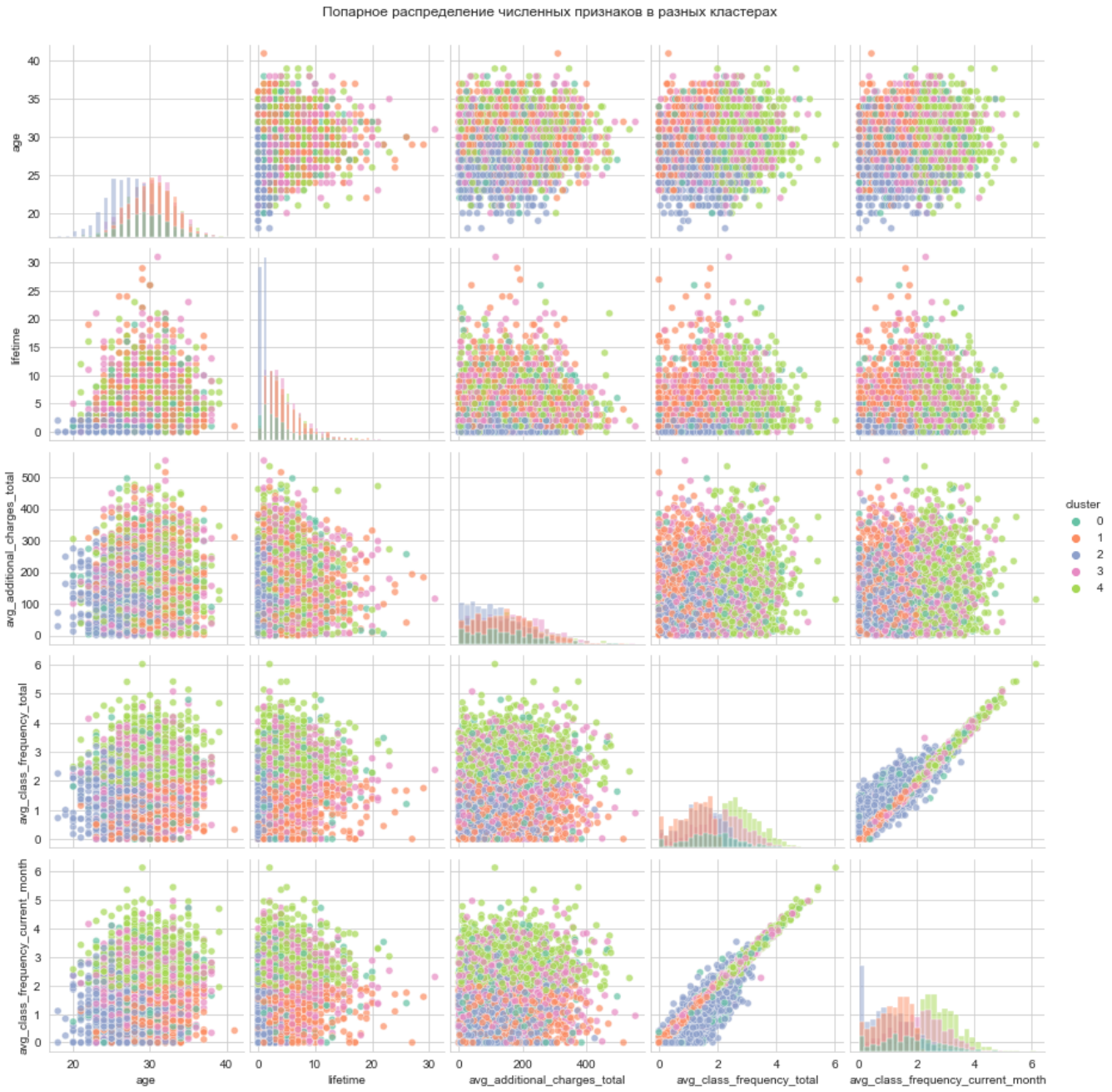
- сосредоточены только уходящие клиенты (среднее `churn` почти 1)
- меньше всего клиентов из компаний-партнеров (доля `Partner` 34%)
- меньше всего клиентов от друзей (`promo_friends` 18%)
- наименьшая средняя продолжительность контракта (`contract_period` в среднем 1.6)

- клиенты меньше других посещают групповые занятия (`Group_visits` 26%)
- меньше всего дополнительных трат (`avg_additional_charges_total` 115)
- меньше всего времени до конца контракта и общий период пользования услугами (`month_to_end_contract` и `lifetime`)
- наименьшая средняя частота посещений в неделю за предыдущий месяц (`avg_class_frequency_current_month` около 1)

Таким образом, в результате кластеризации мы получили подтверждения изначальным предположениям в начале работы без использования машинного обучения, а по сравнению медианных величин.

Для визуализации разбиения на кластеры можно также построить попарные диаграммы рассеяния.

In [39]:
```
ax = sns.pairplot(df,
                  vars=numeric_cols,
                  hue='cluster',
                  diag_kind='hist',
                  palette='Set2',
                  plot_kws={'alpha': 0.7});
ax.fig.suptitle('Попарное распределение численных признаков в разных кластерах', y=1.02);
```



На диаграммах нельзя сразу выделить интересные кластеры, но можно отметить стремление "надежных" клиентов к линейной зависимости посещений групповых занятий всего и в прошлом месяце (`avg_class_frequency_total` и `avg_class_frequency_current_month`), для уходящих клиентов наоборот.

Выделим снова долю уходящих клиентов в каждом кластере.

In [40]:
```
df.groupby('cluster')['churn'].mean().sort_values().reset_index()
```

Out[40]:

	cluster	churn
0	1	0.001170
1	4	0.006952
2	3	0.015641
3	0	0.261780
4	2	0.997875

Кластеризация также помогла выделить самых "надежных" клиентов в кластерах 1 и 2, где средний отток составил менее 1%.

4. Общие выводы и рекомендации

По результатам анализа и проведенного машинного обучения выявились следующие ключевые характеристики уходящих клиентов:

- чаще оставшихся посещали клуб без корпоративной программы и "без друга",
- заключали короткий договор,
- немного моложе оставшихся,
- меньше всего тратили на дополнительные услуги,
- новички в клубе,
- меньше посещали групповые занятия в целом и в прошлый месяц

Для предотвращения ухода клиентов можно выделить следующие основные принципы:

- по возможности предлагать заключение более длительных контрактов или предлагать продление до его окончания (возможно, со скидкой на продление - для того лучше всего оценить стоимость ухода клиента и скидки)
- проводить маркетинговую компанию среди более молодых клиентов
- расширить ассортимент групповых занятий для привлечения большего числа клиентов к их посещению (либо пересмотр стоимости)
- возможно следует расширить программу "приведи друга" с большим поощрением, так как совместное посещение усиливает надежность клиента
- отдельно стоит проработать стоимость дополнительных услуг, так как навязывания допсервисов может наоборот оттолкнуть уходящих клиентов, но им может пригодится что-то иное, что можно совместить с посещением клуба.