1. Подготовка данных

In [1]:

```
import pandas as pd
data = pd.read_csv('/datasets/Churn.csv', index_col='RowNumber')
print(data.info())
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 13 columns):
CustomerId         10000 non-null int64
Surname            10000 non-null object
CreditScore        10000 non-null int64
Geography          10000 non-null object
Gender             10000 non-null object
Age                10000 non-null int64
Tenure             9091 non-null float64
Balance            10000 non-null float64
NumOfProducts      10000 non-null int64
HasCrCard          10000 non-null int64
IsActiveMember     10000 non-null int64
EstimatedSalary    10000 non-null float64
Exited             10000 non-null int64
dtypes: float64(3), int64(7), object(3)
memory usage: 1.1+ MB
None
```

Out[1]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMe
1	15634602	Hargrave	619	France	Female	42	2.0	0.00	1	1	
2	15647311	Hill	608	Spain	Female	41	1.0	83807.86	1	0	
3	15619304	Onio	502	France	Female	42	8.0	159660.80	3	1	
4	15701354	Boni	699	France	Female	39	1.0	0.00	2	0	
5	15737888	Mitchell	850	Spain	Female	43	2.0	125510.82	1	1	

In [2]:

```
print(('Медиана по выборке вышедших клиентов -- {}\nМедиана по выборке оставшихся клиентов -- {}\nСреднее по выбо
рке вышедших клиентов -- {}\nСреднее по выборке оставшихся клиентов -- {}')
    .format(data[data['Exited']==0]['Tenure'].median(),
    data[data['Exited']==1]['Tenure'].median(),
    data[data['Exited']==0]['Tenure'].mean(),
    data[data['Exited']==1]['Tenure'].mean())
     )
data.loc[(data['Tenure'].isna()==True), 'Tenure'] = 5
```

```
Медиана по выборке вышедших клиентов -- 5.0
Медиана по выборке оставшихся клиентов -- 5.0
Среднее по выборке вышедших клиентов -- 5.022246787342822
Среднее по выборке оставшихся клиентов -- 4.901833872707659
```

Провел небольшой анализ (не мог решить, использовать медиану или среднее арифметическое, также нужно было понять какое значение характерно для каждого из классов), оказалось, что данные в столбце Tenure распределены нормально, причем выборки по ушедшим и по оставшимся клиентам в некотором смысле эквивалентны (их средние равны). Поэтому можно заполнить все пропуски медианным значением -- 5.

Вывод:
Обнаружил несколько столбцов с нуждающимися в преобразовании категориальными данными:

- Geography
- Gender
- Tenure
- NumOfProducts

То, что я отнес последние два признака к категориальным, является тонким моментом, готов внести коррективы в случае достаточной аргументации, но на мой взгляд, основным критерием, позволяющим считать конкретную характеристику количественной -- возможность на ее основе строить какие-то сравнительные суждения, т.е. два объекта с определенной количественной характеристикой могут вступать в некое отношение, например два клиента с признаком Age: один из них может быть старше другого, второй может быть младше первого, или их возрасты могут быть равны. Аналогичные рассуждения можно проводить по признакам CreditScore, Age, Balance и EstimatedSalary. В случае с Tenure или NumOfProduct это сделать невозможно.

Ряд признаков также нужно масштабировать:

- CreditScore
- Age
- Balance
- EstimatedSalary

Помимо этого, следует исключить из исследования столбцы, используемые для идентификации клиента:

- CustomerId
- Surname

In [4]:

```
from sklearn.preprocessing import StandardScaler
data[['Tenure', 'NumOfProducts']] = data[['Tenure', 'NumOfProducts']].astype('object')
scaler = StandardScaler()
scaler.fit(data[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']])
data[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']] = scaler.transform(
    data[['CreditScore', 'Age', 'Balance', 'EstimatedSalary']]
    )
pure = pd.get_dummies(data.drop(['CustomerId', 'Surname'], axis=1), drop_first=True)
pure
```

Out[4]:

RowNumber	CreditScore	Age	Balance	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_Germany	Geography_
1	-0.326221	0.293517	-1.225848	1	1	0.021886	1	0	
2	-0.440036	0.198164	0.117350	0	1	0.216534	0	0	
3	-1.536794	0.293517	1.333053	1	0	0.240687	1	0	
4	0.501521	0.007457	-1.225848	0	0	-0.108918	0	0	
5	2.063884	0.388871	0.785728	1	1	-0.365276	0	0	
...	...	...	...	...	...	...	...	...	
9996	1.246488	0.007457	-1.225848	1	0	-0.066419	0	0	
9997	-1.391939	-0.373958	-0.306379	1	1	0.027988	0	0	
9998	0.604988	-0.278604	-1.225848	0	1	-1.008643	1	0	
9999	1.256835	0.293517	-0.022608	1	0	-0.125231	1	1	
10000	1.463771	-1.041433	0.859965	1	0	-1.076370	0	0	

10000 rows × 23 columns

2. Исследование задачи

In [5]:

```
print(pure.Exited.mean())
```

```
0.2037
```

Вывод:

данные не сбалансированы, не знаю, стоит ли совершать явно ненужные действия, но тк это требуется (был план выполнения проекта) я обучу модель и без предварительной балансировки

In [7]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
features, target = pure.drop('Exited', axis=1), pure.Exited
features_train, features_other, target_train, target_other = train_test_split(features, target, test_size=0.4, ra
ndom_state=12345)
features_valid, features_test, target_valid, target_test = train_test_split(features_other, target_other, test_si
ze=0.5, random_state=12345)
model = LogisticRegression(random_state=12345, solver='liblinear')
model.fit(features_train, target_train)
predicted_valid = model.predict(features_valid)
print(f1_score(target_valid, predicted_valid), '-- f1')
probabilities_valid = model.predict_proba(features_valid)
probabilities_one_valid = probabilities_valid[:, 1]
print(roc_auc_score(target_valid, probabilities_one_valid), '-- roc_auc_score')
```

```
0.5061349693251534 -- f1
0.8267788336488848 -- roc_auc_score
```

3. Борьба с дисбалансом

In [8]:

```
model = LogisticRegression(random_state=12345, solver='liblinear', class_weight='balanced')
model.fit(features_train, target_train)
predicted_valid = model.predict(features_valid)
print(f1_score(target_valid, predicted_valid), '-- f1')
probabilities_valid = model.predict_proba(features_valid)
probabilities_one_valid = probabilities_valid[:, 1]
print(roc_auc_score(target_valid, probabilities_one_valid), '-- roc_auc_score')
```

```
0.5623869801084992 -- f1
0.8288732692551976 -- roc_auc_score
```

уже лучше, попробую с помощью изменения параметров добиться требуемого значения f1

In [9]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, max_depth=12, random_state=12345, class_weight='balanced')
model.fit(features_train, target_train)
predicted_valid = model.predict(features_valid)
print(f1_score(target_valid, predicted_valid), '-- f1')
probabilities_valid = model.predict_proba(features_valid)
probabilities_one_valid = probabilities_valid[:, 1]
print(roc_auc_score(target_valid, probabilities_one_valid), '-- roc_auc_score')
```

```
0.6292682926829267 -- f1
0.8534953635093364 -- roc_auc_score
```

4. Тестирование модели

In [10]:

```
model.fit(pd.concat([features_train, features_valid], ignore_index=True), pd.concat([target_train, target_valid],
ignore_index=True))
predicted_test = model.predict(features_test)
print(f1_score(target_test, predicted_test), '-- f1')
probabilities_test = model.predict_proba(features_test)
probabilities_one_test = probabilities_valid[:, 1]
print(roc_auc_score(target_test, probabilities_one_test), '-- roc_auc_score')
```

```
0.6012121212121212 -- f1
0.5061335300140465 -- roc_auc_score
```

Вывод:

roc-auc score сильно упал, полагаю это связано с тем, что данная метрика в отличие от метрики f1 учитывает вероятности классов, а выборка не сбалансирована.

В целом, модель прошла тест успешно