```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

```
!gdown 1ERwQ5odiK1Zvi1LtjpkzCMUswYsAX8_K  # train.csv
!gdown 1fGw_-RFwvn_LEdt91Jq-7A-wzG6mmH8r  # test.csv
!gdown 199Mt4OYZNaelT83U-HGDsEYs2YcUGQ6y  # submission.csv
```

```
Downloading...
From: https://drive.google.com/uc?id=1ERwQ5odiK1Zvi1LtjpkzCMUswYsAX8_K
To: /content/train.csv
100% 664k/664k [00:00<00:00, 19.0MB/s]
Downloading...
From: https://drive.google.com/uc?id=1fGw_-RFwvn_LEdt91Jq-7A-wzG6mmH8r
To: /content/test.csv
100% 218k/218k [00:00<00:00, 21.8MB/s]
Downloading...
From: https://drive.google.com/uc?id=199Mt4OYZNaelT83U-HGDsEYs2YcUGQ6y
To: /content/submission.csv
100% 14.7k/14.7k [00:00<00:00, 22.8MB/s]
```

```
data = pd.read_csv('./train.csv')
```

```
# Числовые признаки
num_cols = [
    'ClientPeriod',
    'MonthlySpending',
    'TotalSpent'
]

# Категориальные признаки
cat_cols = [
    'Sex',
    'IsSeniorCitizen',
    'HasPartner',
    'HasChild',
    'HasPhoneService',
    'HasMultiplePhoneNumbers',
    'HasInternetService',
    'HasOnlineSecurityService',
    'HasOnlineBackup',
    'HasDeviceProtection',
    'HasTechSupportAccess',
    'HasOnlineTV',
    'HasMovieSubscription',
    'HasContractPhone',
    'IsBillingPaperless',
    'PaymentMethod'
]
```

```
feature_cols = num_cols + cat_cols
target_col = 'Churn'
```

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ClientPeriod	5282.0	32.397009	24.550326	0.00	9.0000	29.0	55.00	72.00
MonthlySpending	5282.0	64.924754	30.176464	18.25	35.4625	70.4	90.05	118.75
IsSeniorCitizen	5282.0	0.159409	0.366092	0.00	0.0000	0.0	0.00	1.00
Churn	5282.0	0.262022	0.439776	0.00	0.0000	0.0	1.00	1.00

```
data.head(10)
```

	ClientPeriod	MonthlySpending	TotalSpent	Sex	IsSeniorCitizen	HasPartne
0	55	19.50	1026.35	Male	0	Ye
1	72	25.85	1872.2	Male	0	Ye
2	1	75.90	75.9	Male	0	N
3	32	79.30	2570	Female	1	Ye
4	60	115.25	6758.45	Female	0	Ye
5	25	19.80	475.2	Female	0	N
6	27	90.15	2423.4	Female	0	Ye
7	1	45.70	45.7	Male	0	N
8	50	105.95	5341.8	Male	0	Ye
9	72	61.20	4390.25	Male	0	N

```
nan_counts = data.isna().sum()

fig, ax = plt.subplots(figsize=(10, 4))
ax.axis('tight')
ax.axis('off')

table = ax.table(cellText=[nan_counts.values],
                 rowLabels=['Количество NaN'],
                 colLabels=nan_counts.index,
```

```
                    cellLoc='center',
                    loc='center')

table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1, 2)

plt.title('Количество NaN значений по столбцам')
plt.show()
```

Показать скрытые выходные данные

```
empty_string_counts = data.applymap(lambda x: isinstance(x, str) and x.strip()

print("Количество пустых строк:")
print(empty_string_counts)
```

Показать скрытые выходные данные

```
mask = data['TotalSpent'].apply(lambda x: isinstance(x, str) and x.strip() == '
data.loc[mask, 'TotalSpent'] = 0
```

```
data['TotalSpent'] = pd.to_numeric(data['TotalSpent'])
```

```
q_string_counts = data.applymap(lambda x: isinstance(x, str) and x.strip() == '

print("Количество пустых строк:")
print(q_string_counts)
```

Показать скрытые выходные данные

```
data_origin = data.copy()
```

Начнём с категориальных признаков. Пойдём по порядку расположения в таблице.

```
value_counts = data['ClientPeriod'].value_counts()

plt.figure(figsize=(10, 6))
plt.bar(value_counts.index, value_counts.values, color='skyblue', alpha=0.7)
plt.title('Распределение Периода клиентов')
plt.xlabel('Период')
plt.ylabel('Количество клиентов')
plt.grid(axis='y', alpha=0.3)

plt.show()
```

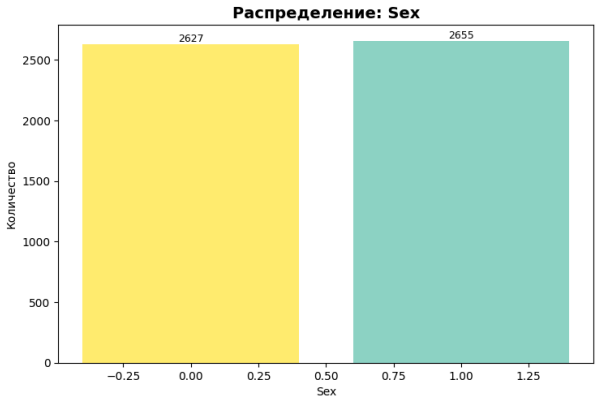
Показать скрытые выходные данные

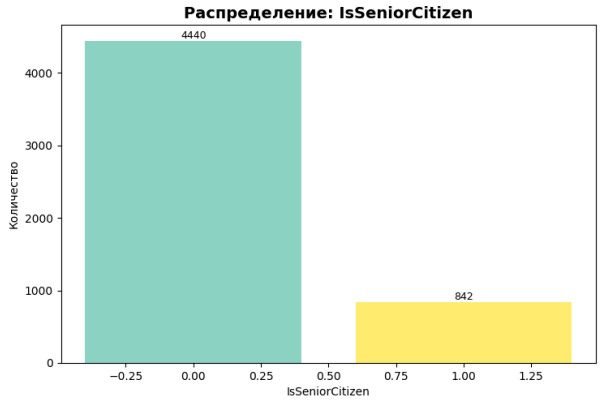
У ClientPeriod сильный перекос влево, то есть нормализовать логарифмом или корнем или возведением в квадрат бессмысленно (возможно станет хуже). Оставлю как есть

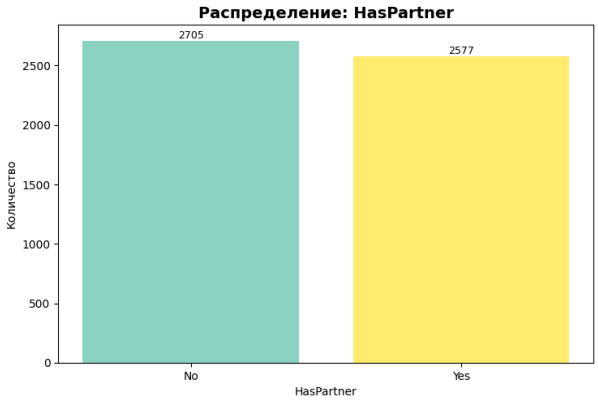
```
data['Sex'] = data['Sex'].replace({'Male':1, 'Female':0})
```

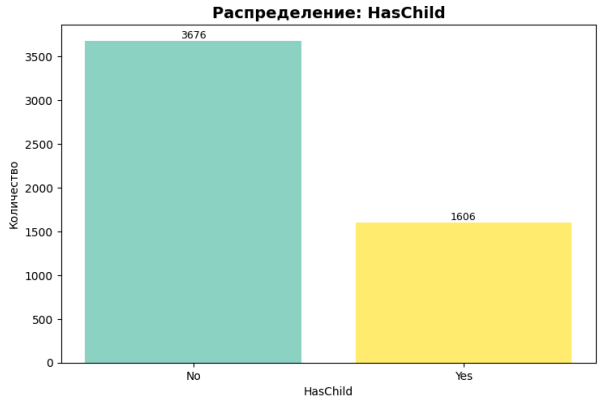
```
/tmp/ipython-input-4038858295.py:1: FutureWarning: Downcasting behavior in `repl
  data['Sex'] = data['Sex'].replace({'Male':1, 'Female':0})
```

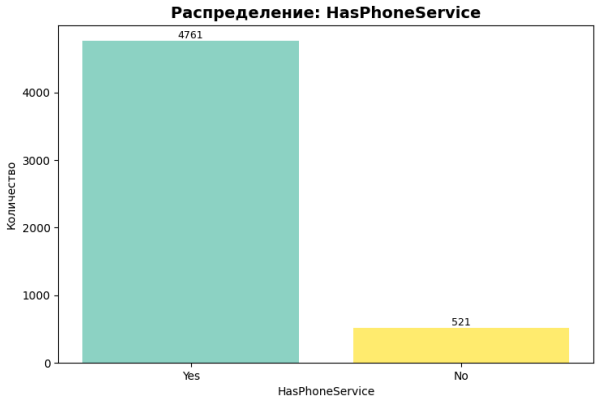
```
n_cols = 2
n_rows = (len(cat_cols) + n_cols - 1) // n_cols

fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 5*n_rows))
axes = axes.flatten() if n_rows > 1 else [axes]

for i, col in enumerate(cat_cols):
    if i < len(axes):
        value_counts = data[col].value_counts()

        bars = axes[i].bar(value_counts.index, value_counts.values,
                           color=plt.cm.Set3(np.linspace(0, 1, len(value_counts)

        axes[i].set_title(f'Распределение: {col}', fontsize=14, fontweight='bol
        axes[i].set_xlabel(col)
        axes[i].set_ylabel('Количество')

        if len(value_counts) > 5:
            axes[i].tick_params(axis='x', rotation=45)

        for bar in bars:
            height = bar.get_height()
            axes[i].text(bar.get_x() + bar.get_width()/2., height + 0.1,
                         f'{int(height)}', ha='center', va='bottom', fontsize=9)

plt.tight_layout()
plt.show()
```

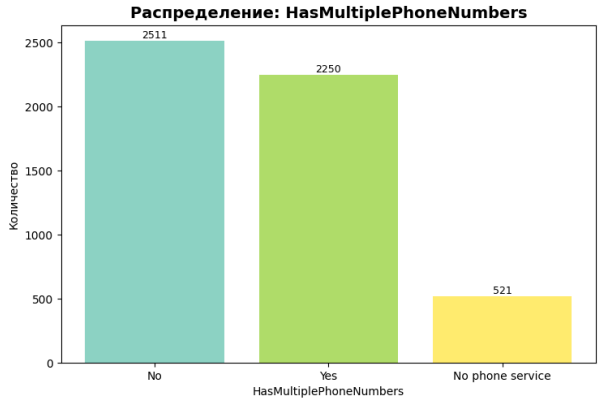

Распределение: Sex
Количество
Sex
2627
2655

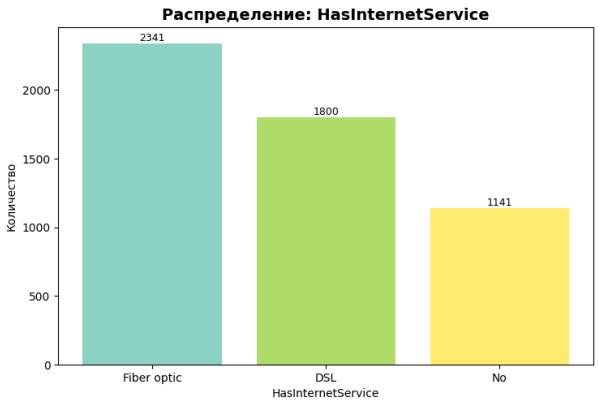

Распределение: IsSeniorCitizen
Количество
IsSeniorCitizen
4440
842

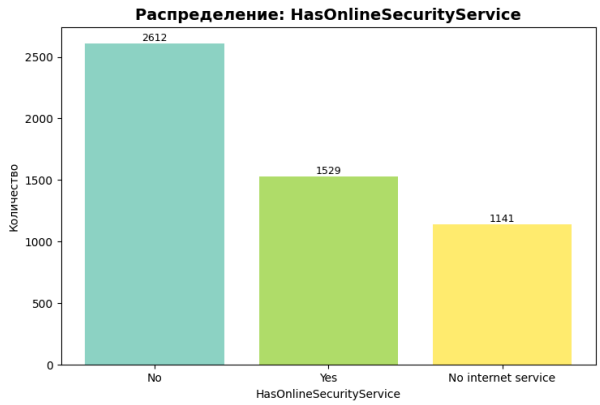

Распределение: HasPartner
Количество
HasPartner
No
Yes
2705
2577

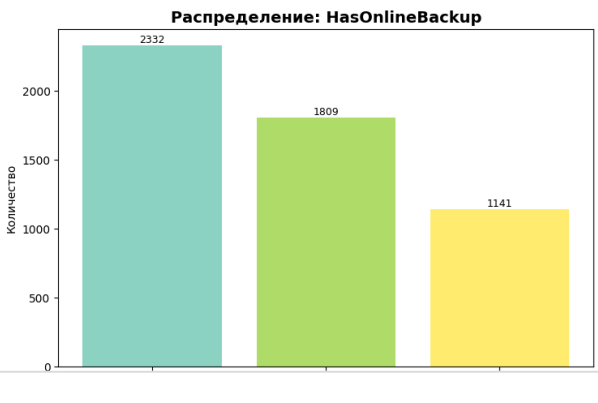

Распределение: HasChild
Количество
HasChild
No
Yes
3676
1606

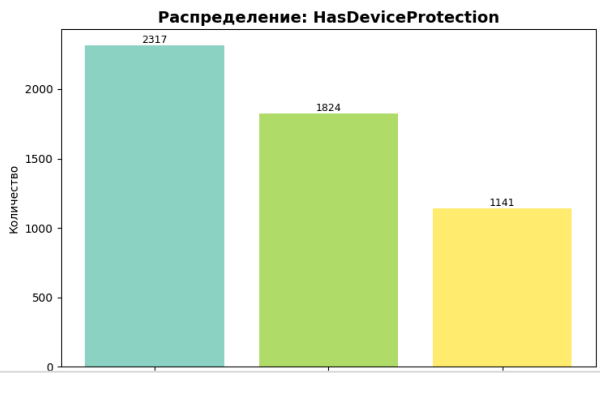

Распределение: HasPhoneService
Количество
HasPhoneService
Yes
No
4761
521


Распределение: HasMultiplePhoneNumbers
Количество
HasMultiplePhoneNumbers
No
Yes
No phone service
2511
2250
521

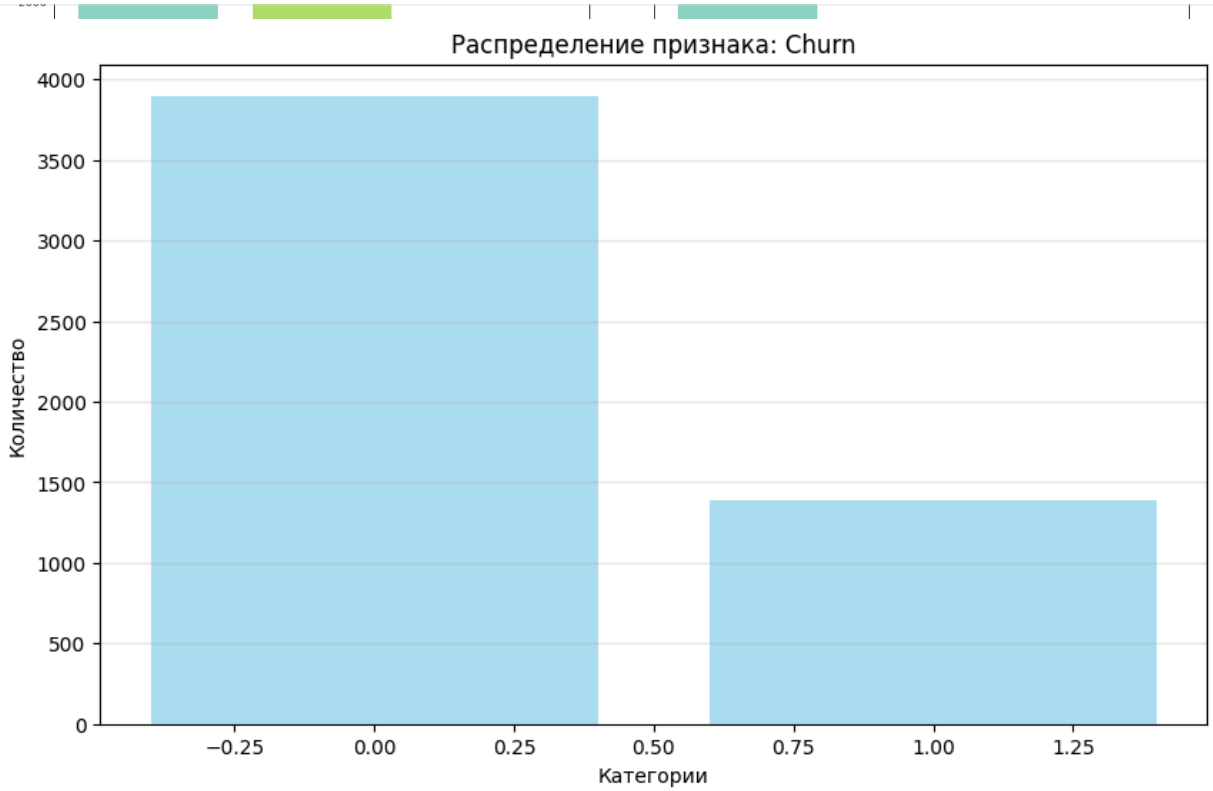

Распределение: HasInternetService
Количество
HasInternetService
Fiber optic
DSL
No
2341
1800
1141


Распределение: HasOnlineSecurityService
Количество
HasOnlineSecurityService
No
Yes
No internet service
2612
1529
1141


Распределение: HasOnlineBackup
Количество
2332
1809
1141


Распределение: HasDeviceProtection
Количество
2317
1824
1141

```
value_counts = data[target_col].value_counts()

# Строим график
plt.figure(figsize=(10, 6))
plt.bar(value_counts.index, value_counts.values, color='skyblue', alpha=0.7)
plt.title('Распределение признака: Churn')
plt.xlabel('Категории')
plt.ylabel('Количество')
plt.grid(axis='y', alpha=0.3)
plt.show()
```



Классы Таргета несбаланисированны. В категориальных признаках с >2 классами по большому счёту везде сбалансированное распределение - нет каких то сильных выбросов, поэтому можно просто воспользоваться Ohe. А вот в двух бинарных признаках есть несбалансированное распределение. Поэтому на обучении воспользуюсь class_weight='balanced'.

```
to_num_columns = ['HasPartner', 'HasChild', 'HasPhoneService', 'IsBillingPaperl

for col in to_num_columns:
    if col in data.columns:
        data[col] = data[col].replace({'Yes': 1, 'No': 0})
```

```
data.head()
```

Показать скрытые выходные данные

```
data.info()
```

Показать скрытые выходные данные

```
new_categ_cols = ["HasMultiplePhoneNumbers", "HasInternetService", "HasOnlineSe
```

```
from sklearn.preprocessing import StandardScaler, RobustScaler, LabelEncoder, C
ohe = OneHotEncoder(sparse_output=False, drop='first')

encoded_array = ohe.fit_transform(data[new_categ_cols])
encoded_columns = ohe.get_feature_names_out(new_categ_cols)
data_encoded = pd.DataFrame(encoded_array, columns=encoded_columns, index=data.

numerical_columns = data.select_dtypes(include=['int64', 'float64']).columns[:-
data_final = pd.concat([data[numerical_columns], data_encoded], axis=1)
```

```
data_final = pd.concat([data_final, data[target_col]], axis=1)
data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5282 entries, 0 to 5281
Data columns (total 31 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   ClientPeriod                                  5282 non-null   int64
 1   MonthlySpending                               5282 non-null   float64
 2   TotalSpent                                    5282 non-null   float64
 3   Sex                                           5282 non-null   int64
 4   IsSeniorCitizen                               5282 non-null   int64
 5   HasPartner                                    5282 non-null   int64
 6   HasChild                                      5282 non-null   int64
 7   HasPhoneService                               5282 non-null   int64
 8   IsBillingPaperless                            5282 non-null   int64
 9   HasMultiplePhoneNumbers_No phone service      5282 non-null   float64
 10  HasMultiplePhoneNumbers_Yes                   5282 non-null   float64
 11  HasInternetService_Fiber optic                5282 non-null   float64
 12  HasInternetService_No                         5282 non-null   float64
 13  HasOnlineSecurityService_No internet service  5282 non-null   float64
 14  HasOnlineSecurityService_Yes                  5282 non-null   float64
 15  HasOnlineBackup_No internet service           5282 non-null   float64
 16  HasOnlineBackup_Yes                           5282 non-null   float64
 17  HasDeviceProtection_No internet service       5282 non-null   float64
 18  HasDeviceProtection_Yes                       5282 non-null   float64
 19  HasTechSupportAccess_No internet service      5282 non-null   float64
 20  HasTechSupportAccess_Yes                      5282 non-null   float64
 21  HasOnlineTV_No internet service               5282 non-null   float64
 22  HasOnlineTV_Yes                               5282 non-null   float64
 23  HasMovieSubscription_No internet service      5282 non-null   float64
 24  HasMovieSubscription_Yes                      5282 non-null   float64
 25  HasContractPhone_One year                     5282 non-null   float64
```

```
 26  HasContractPhone_Two year                    5282 non-null   float64
 27  PaymentMethod_Credit card (automatic)        5282 non-null   float64
 28  PaymentMethod_Electronic check               5282 non-null   float64
 29  PaymentMethod_Mailed check                   5282 non-null   float64
 30  Churn                                        5282 non-null   int64
dtypes: float64(23), int64(8)
memory usage: 1.2 MB
```

```
import seaborn as sns
target_correlations = data_final.corr()[['Churn']].sort_values('Churn', ascendi

print("Корреляции с таргетной переменной:")
print(target_correlations)

plt.figure(figsize=(8, 10))
sns.heatmap(target_correlations, annot=True, cmap='RdYlBu', center=0, fmt='.2f'
plt.title('Корреляции признаков с таргетной переменной')
plt.tight_layout()
plt.show()
```

Показать скрытые выходные данные

Применение линейных моделей

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, RobustScaler, LabelEncoder, C
from sklearn.pipeline import make_pipeline
```

```
data_final.describe().T
```

Показать скрытые выходные данные

```
cols_to_scale = ["ClientPeriod", "MonthlySpending", "TotalSpent"]

scaler = StandardScaler()

data_scaled = data_final.copy()
data_scaled[cols_to_scale] = scaler.fit_transform(data_final[cols_to_scale])
```

```
X_train, X_val, y_train, y_val = train_test_split(data_scaled.drop('Churn',axis
```

```
Cs = [100, 10, 1, 0.1, 0.01, 0.001]
model = LogisticRegressionCV(
    Cs=Cs,
    cv=5,
    scoring='roc_auc',
    random_state=42,
    max_iter=1000,
```

```
    refit=True,
    class_weight='balanced',
    penalty='l2',
)

model.fit(X_train, y_train)

print(f"Лучший параметр C: {model.C_[0]}")
print(f"Все tested C: {model.Cs_}")
print(f"Средние ROC-AUC для каждого C: {model.scores_[1].mean(axis=0)}")
```

```
Лучший параметр C: 10.0
Все tested C: [1.e+02 1.e+01 1.e+00 1.e-01 1.e-02 1.e-03]
Средние ROC-AUC для каждого C: [0.84851283 0.84879276 0.84836086 0.84746824 0.84
```

Лучший C = 10

```
best_log_model = LogisticRegressionCV(
    Cs=[10],
    cv=5,
    scoring='roc_auc',
    random_state=42,
    max_iter=1000,
    refit=True,
    class_weight='balanced',
    penalty='l2',
)

best_log_model.fit(X_train, y_train)
print(f"Средние ROC-AUC: {best_log_model.scores_[1]}")
```

```
Средние ROC-AUC: [[0.82964961]
 [0.83634992]
 [0.85520362]
 [0.86195469]
 [0.86034228]]
```

```
np.mean(best_log_model.scores_[1])
```

```
np.float64(0.8487000266875959)
```

0.8487000266875959

При C=10, class_weight='balanced'.

Применение градиентного бустинга

```
X_train_origin, X_val_origin, y_train_origin, y_val_origin = train_test_split(c
                                                             train_size=0.8,
                                                             random_state=42)
```

```
!pip install catboost
```

Показать скрытые выходные данные

```
import catboost

boosting_model = catboost.CatBoostClassifier(n_estimators=200,
                                             cat_features=cat_cols,
                                             eval_metric='AUC')

boosting_model.fit(X_train_origin, y_train_origin)

y_val_predicted = boosting_model.predict_proba(X_val_origin)[:, 1]
```

Показать скрытые выходные данные

```
val_auc = roc_auc_score(y_val_origin, y_val_predicted)
print(val_auc)
```

```
0.8228013224850159
```

Со стандартными параметрами roc_auc вышел 0.82 (что меньше, чем у логистической регрессии)

```
boosting_model = catboost.CatBoostClassifier(n_estimators=300,
                                             depth=4,
                                             learning_rate = 0.03,
                                             cat_features=cat_cols,
                                             eval_metric='AUC')
boosting_model.fit(X_train_origin, y_train_origin)

y_val_predicted = boosting_model.predict_proba(X_val_origin)[:, 1]
val_auc = roc_auc_score(y_val_origin, y_val_predicted)
print(val_auc)
```

Показать скрытые выходные данные

0.8327453569878372

Я немного времени подбирал вручную параметры n_estimators (100,200,300,400), learning_rate (0.01,0.03,0.05,0.1) и depth (3,4,5,7,8) и пришел к выводу, что лучше параметров, чем n_estimators=300,depth=4,learning_rate = 0.03 я не получил.

Обработаю тестовые данные

```
X_test = pd.read_csv('./test.csv')
X_test.info()
```

Показать скрытые выходные данные

```
empty_string_counts_test = X_test.applymap(lambda x: isinstance(x, str) and x.s

print("Количество пустых строк:")
print(empty_string_counts_test)
```

Показать скрытые выходные данные

```
mask_test = X_test['TotalSpent'].apply(lambda x: isinstance(x, str) and x.strip
X_test.loc[mask_test, 'TotalSpent'] = 0
X_test['TotalSpent'] = pd.to_numeric(X_test['TotalSpent'])
```

```
X_test.info()
```

Показать скрытые выходные данные

```
X_new_test = X_test.copy()
```

```
X_new_test['Sex'] = X_new_test['Sex'].replace({'Male':1, 'Female':0})

for col in to_num_columns:
    if col in X_new_test.columns:
        X_new_test[col] = X_new_test[col].replace({'Yes': 1, 'No': 0})
```

Показать скрытые выходные данные

```
encoded_array_test = ohe.transform(X_new_test[new_categ_cols])
data_encoded_test = pd.DataFrame(encoded_array_test, columns=encoded_columns, i

test_numerical_columns = X_new_test.select_dtypes(include=['int64', 'float64'])
data_final_test = pd.concat([X_new_test[test_numerical_columns], data_encoded_t
```

```
data_test_scaled = data_final_test.copy()
```

```
data_test_scaled[cols_to_scale] = scaler.transform(data_final_test[cols_to_scal
```

Предсказания

```
best_catboost_model = boosting_model
```

```
submission = pd.read_csv('./submission.csv')
```

```
submission['Churn'] = best_catboost_model.predict_proba(X_test)[:, 1]
submission.to_csv('./my_submission.csv', index=False)
```

```
best_logistic_model = best_log_model
```

```
submission['Churn'] = best_logistic_model.predict_proba(data_test_scaled)[:, 1]
submission.to_csv('./my_submission2.csv', index=False)
```

ИТОГО:

Лучшая модель на train - LogisticRegression (0.84606)