Маркетинговая оптимизация

Next best offer

Белый Олег. Стажировка по математической оптимизации GlowByte

Этапы выполнения задания

- **Этап 1.** Построение моделей склонности клиентов для пар продукт - канал
- **Этап 2.** Оптимизация

Этап 1

Подготовка данных и построение моделей склонности клиентов

Выбранная модель классификации – логистическая регрессия

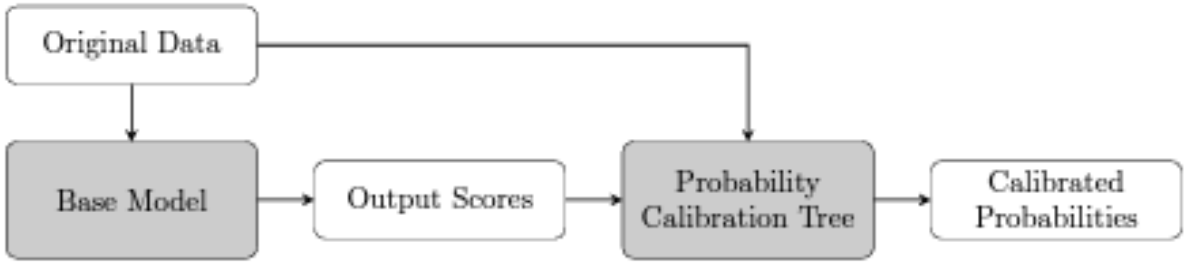
Целевая метрика - AUC-ROC

Пример модели “кредит – звонок”

```
X_train = df.drop('target',axis=1)
y_train = df.target
X_test = dt.drop('target',axis=1)
y_test = dt.target

#нормализация данных
mms = MinMaxScaler()
mms.fit(X_train)
X_train = pd.DataFrame(mms.transform(X_train), columns = X_train.columns)
X_test = pd.DataFrame(mms.transform(X_test), columns = X_test.columns)

parameters = [{'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']},
              {'penalty':['none', 'l2']},
              {'C':[0.001, 0.01, 0.1, 1, 10, 100]}]

clf = LogisticRegression(max_iter=250)

grid_search = GridSearchCV(estimator = clf,
                           param_grid = parameters,
                           scoring = 'roc_auc',
                           cv = 5,
                           verbose=0,
                           n_jobs=-1)

grid_search.fit(X_train,y_train)
best_logreg = grid_search.best_estimator_

dt['score'] = best_logreg.predict_proba(X_test)[:,1]
```

Вероятностная калибровка моделей

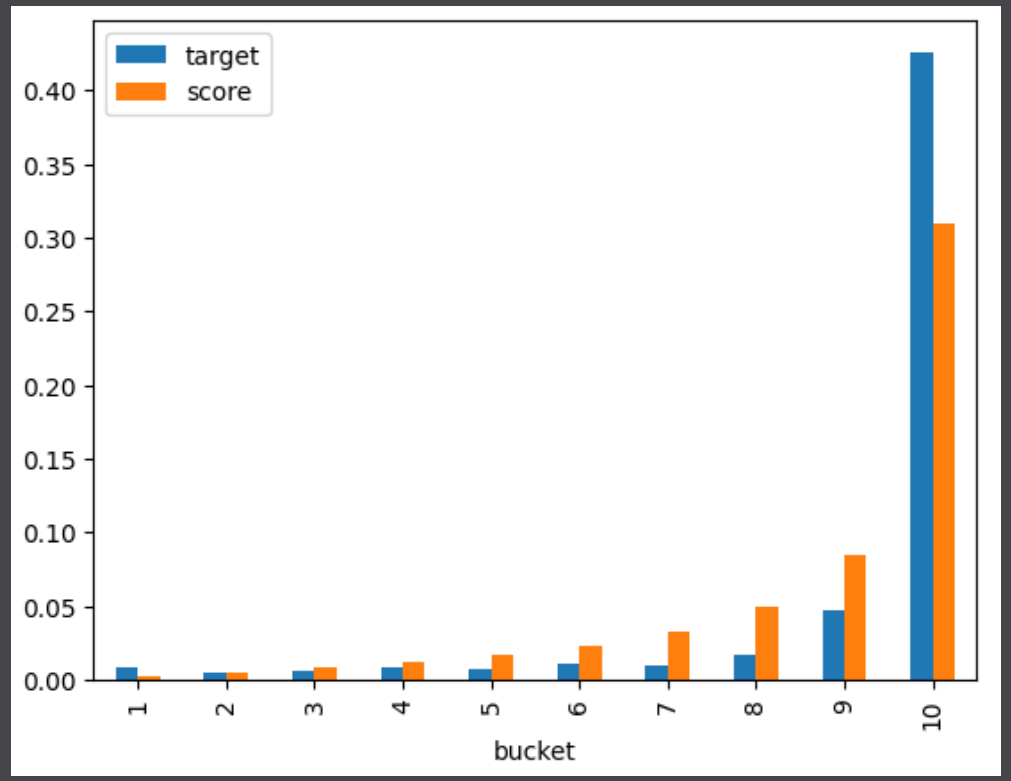
Выбранный метод : “Probability Calibration Trees”



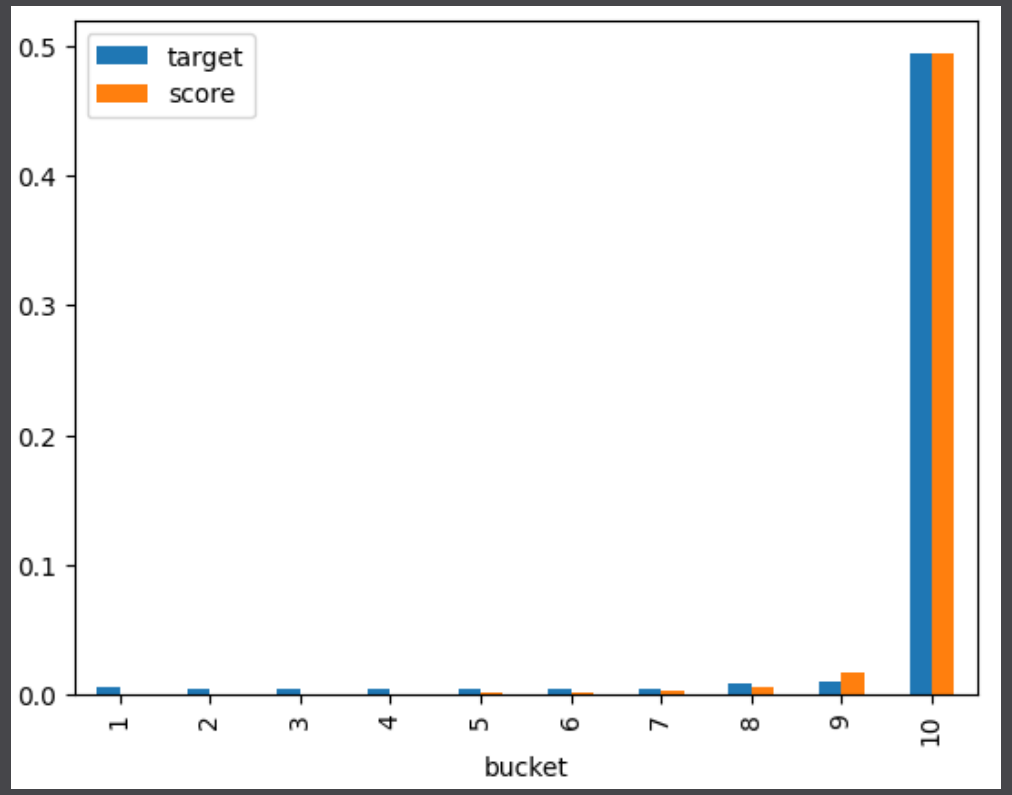
Численная оценка калибровки - **Expected Calibration Error (ECE)**

$$\mathrm{ECE} = \frac{1}{m} \sum_{B \in \{B\}} \left| \sum_{x \in B} b_k(x) - \sum_{x \in B} I[y(x) = k] \right|$$

модель склонности клиентов пары “кредит – звонок”:



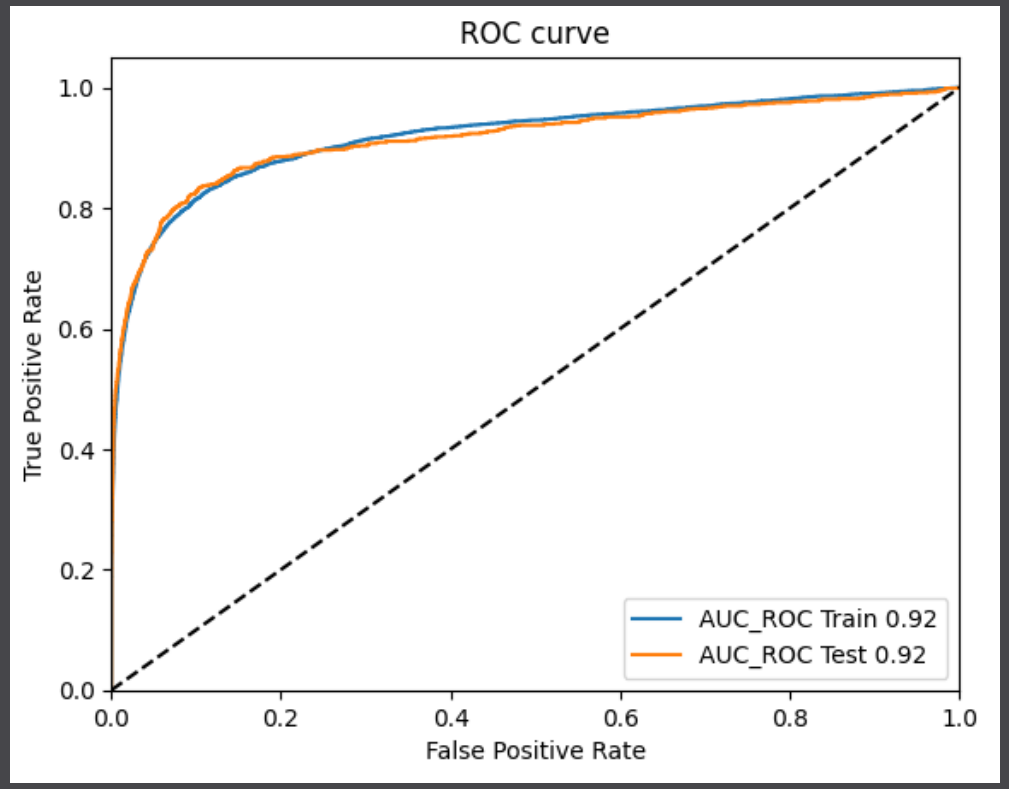
до калибровки
ECE = 0.0244



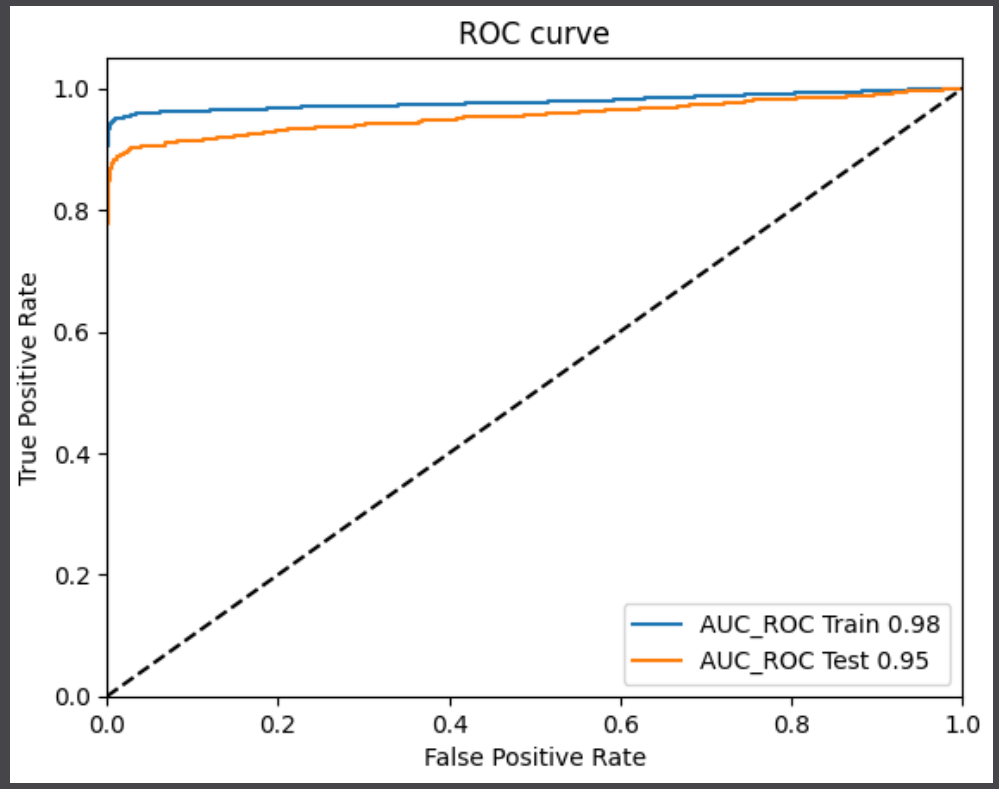
после калибровки
ECE = 0.0037

Распределение прогнозной вероятности и фактического отклика на отложенной выборке до и после калибровки

модель склонности клиентов пары “кредит – звонок”:



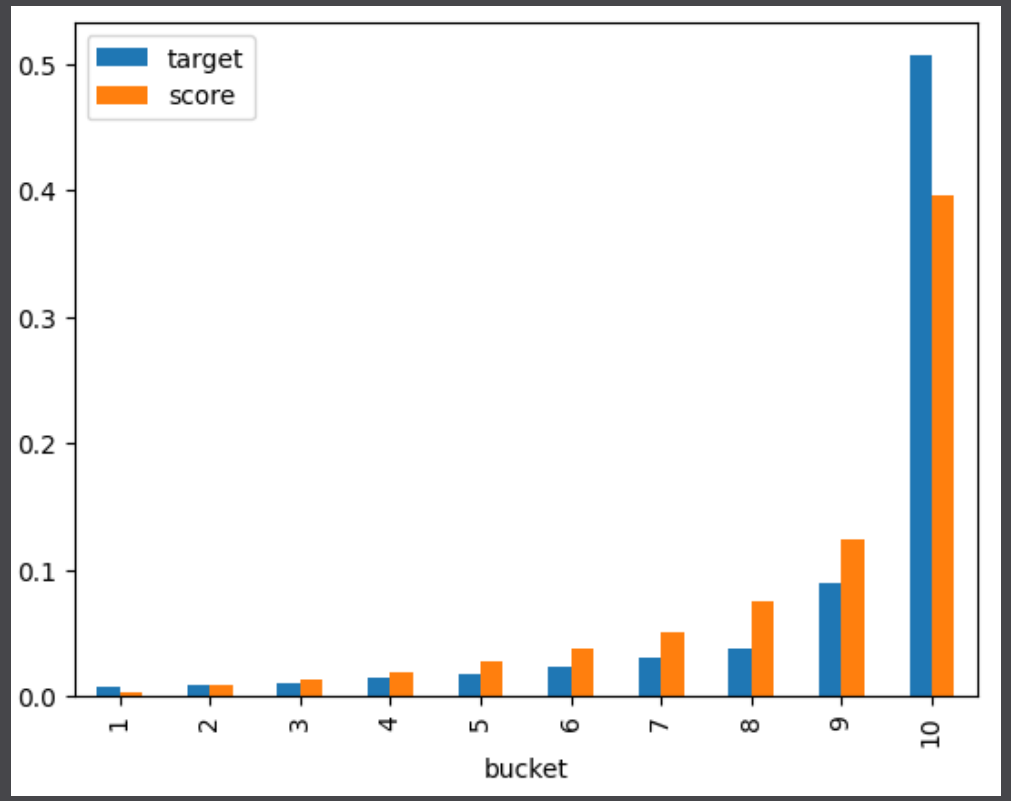
до калибровки



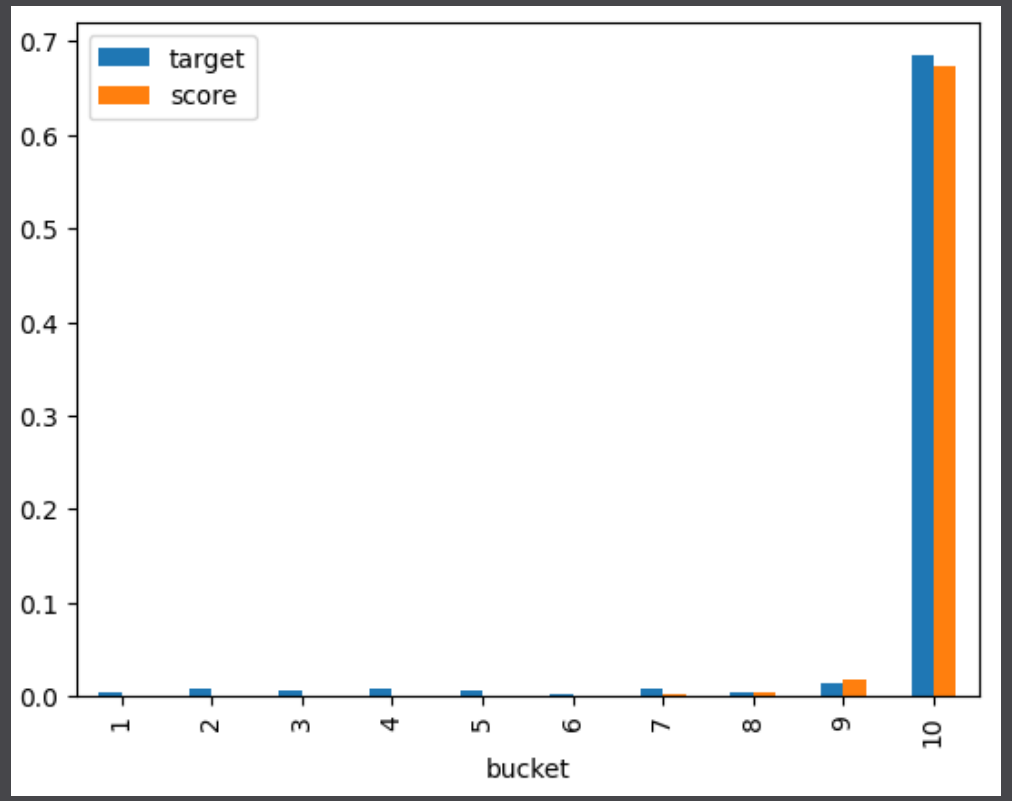
после калибровки

ROC-AUC модели до и после калибровки

модель склонности клиентов пары “кредитная карта – звонок”:



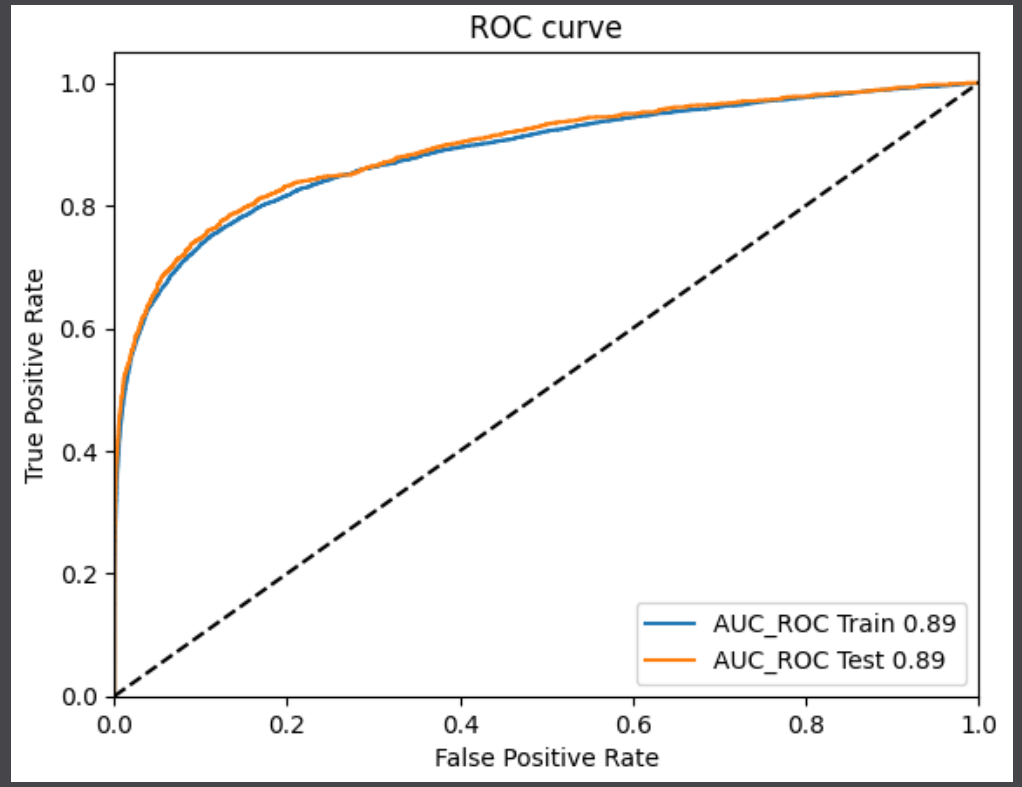
до калибровки
ECE = 0.0239



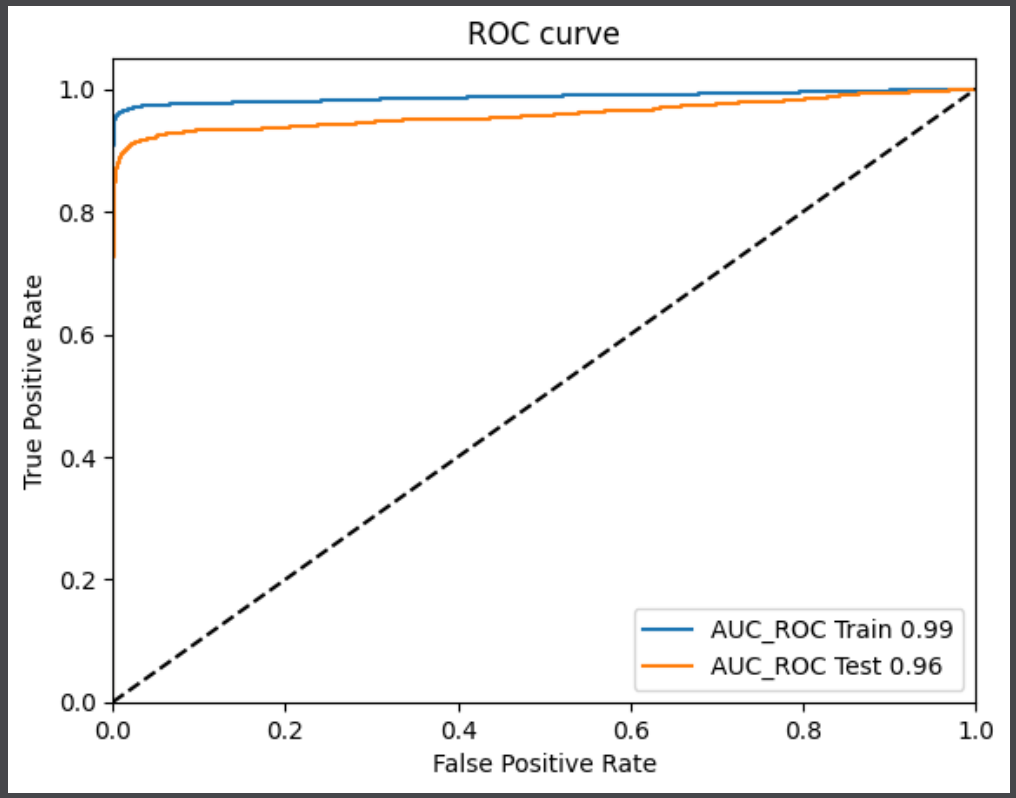
после калибровки
ECE = 0.0056

Распределение прогнозной вероятности и фактического отклика на отложенной выборке до и после калибровки

модель склонности клиентов пары “кредитная карта – звонок”:



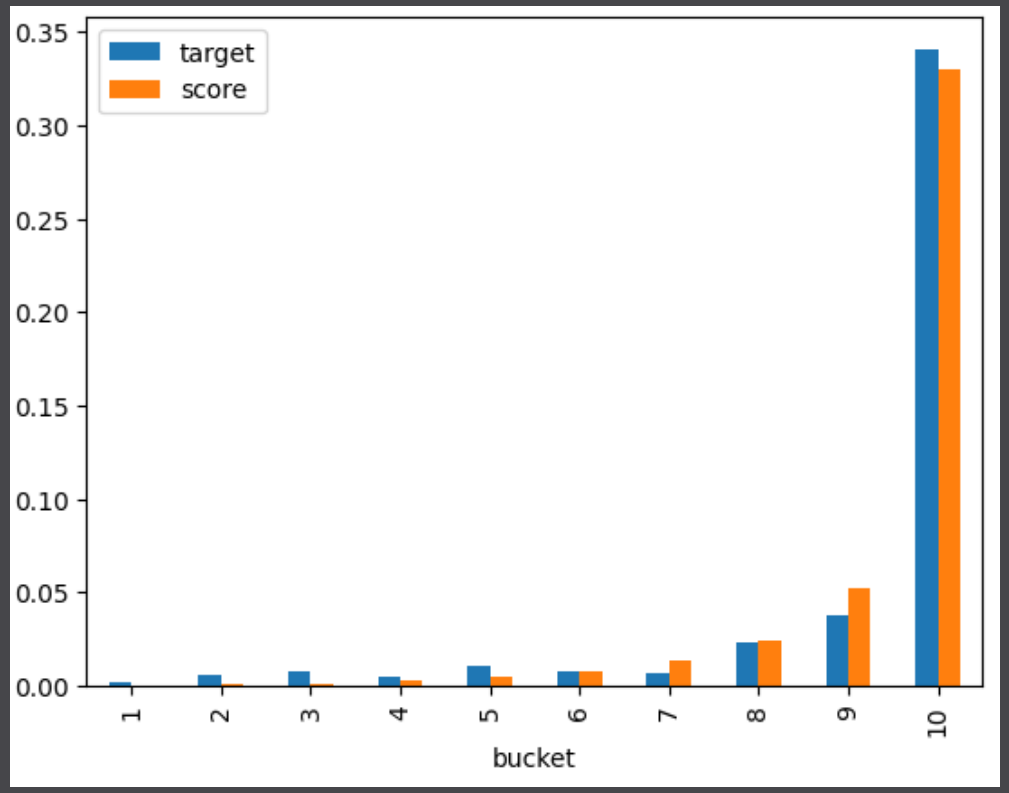
до калибровки



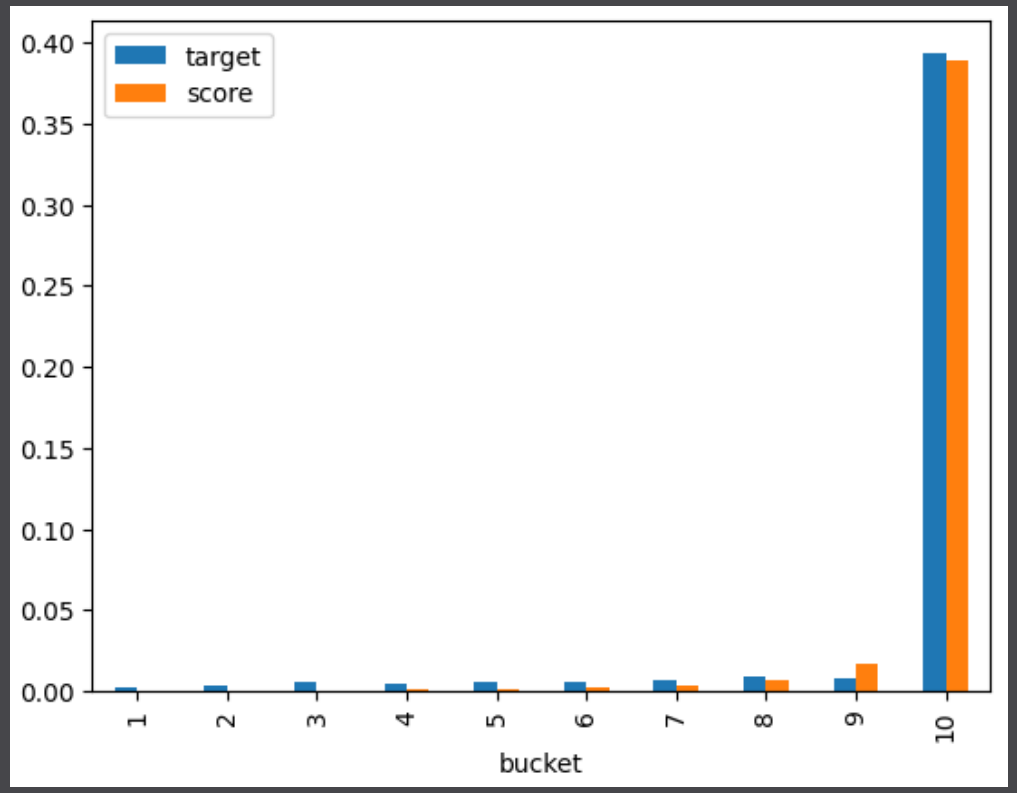
после калибровки

ROC-AUC модели до и после калибровки

модель склонности клиентов пары “кредитная карта – смс”:



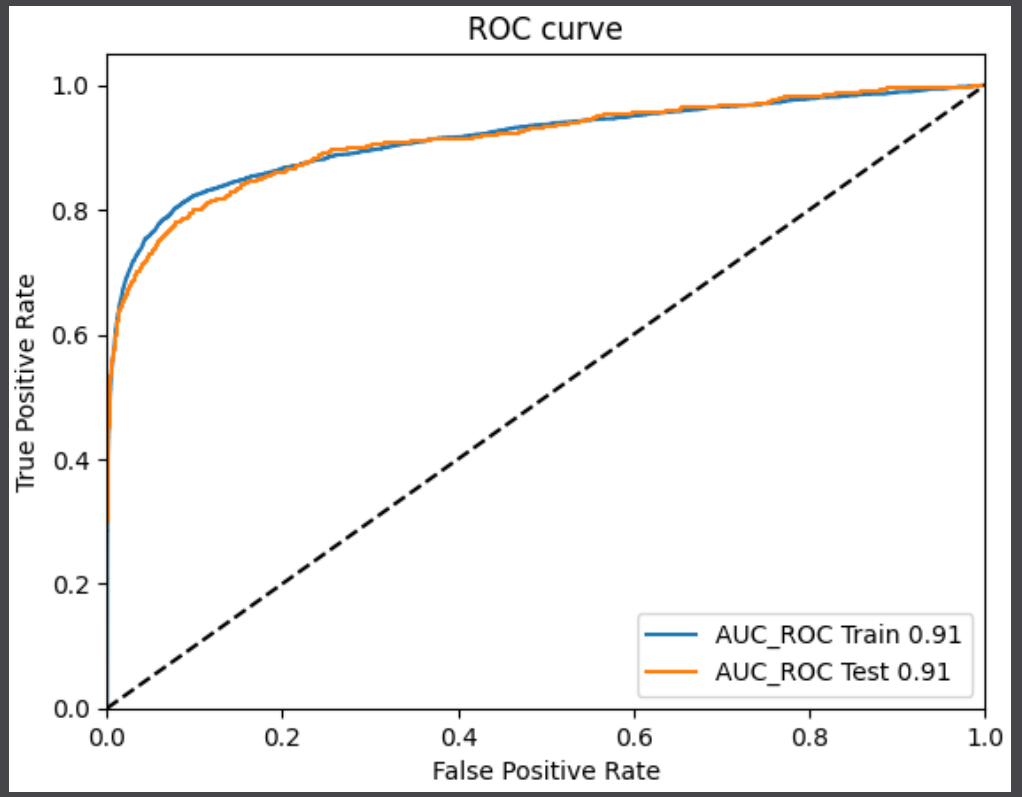
до калибровки
ECE = 0.0055



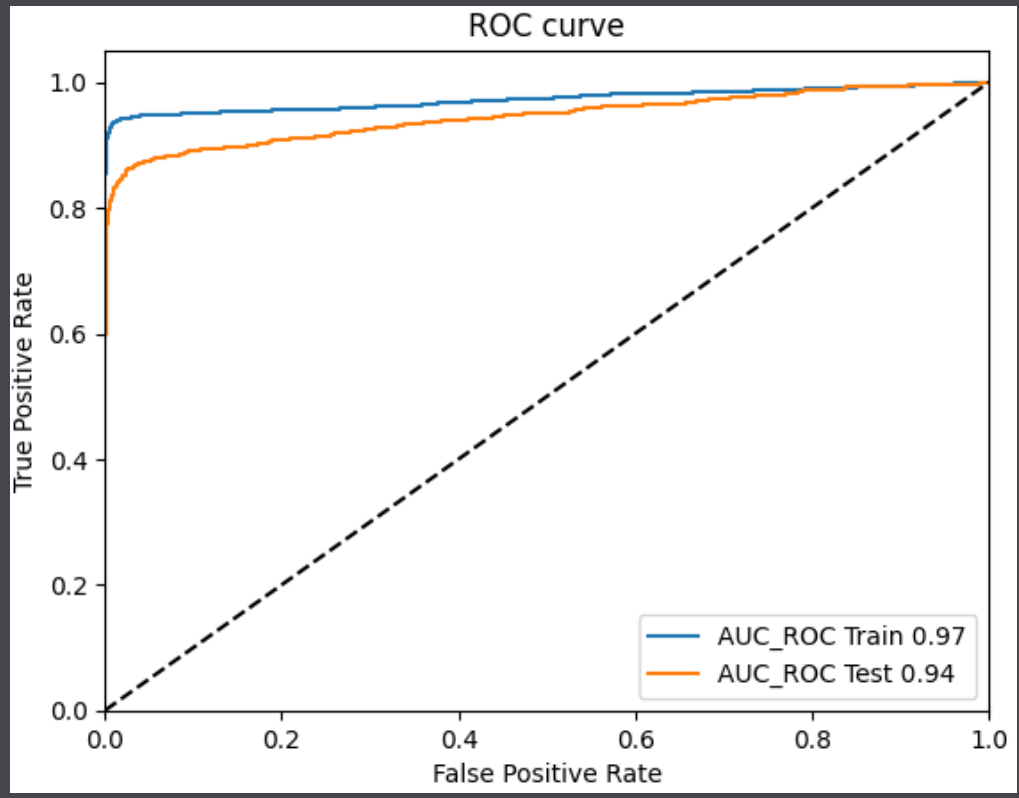
после калибровки
ECE = 0.0043

Распределение прогнозной вероятности и фактического отклика на отложенной выборке до и после калибровки

модель склонности клиентов пары “кредитная карта – смс”:



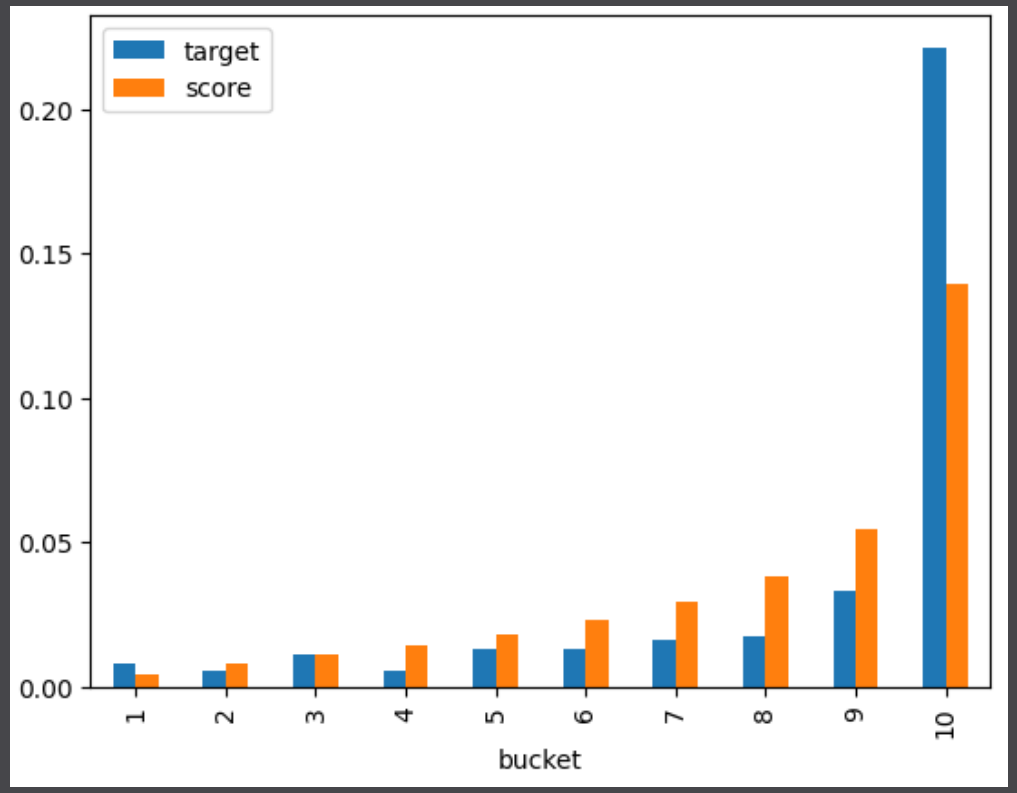
до калибровки



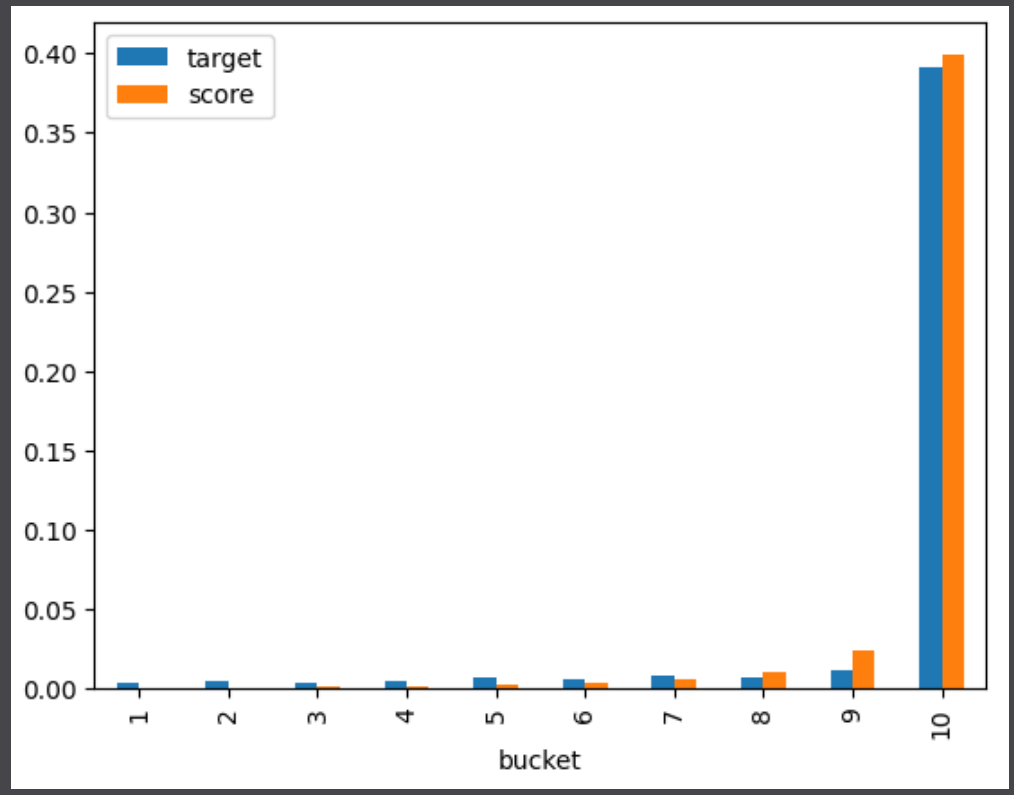
после калибровки

ROC-AUC модели до и после калибровки

модель склонности клиентов пары “кредит – смс”:



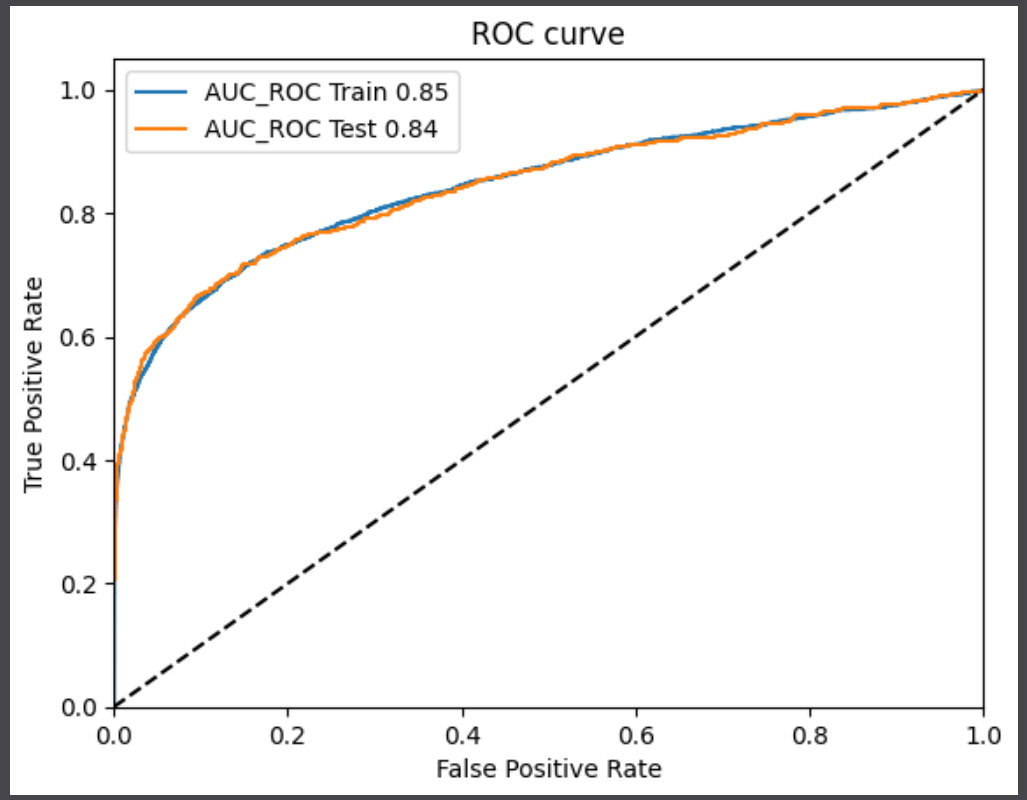
до калибровки
ECE = 0.0168



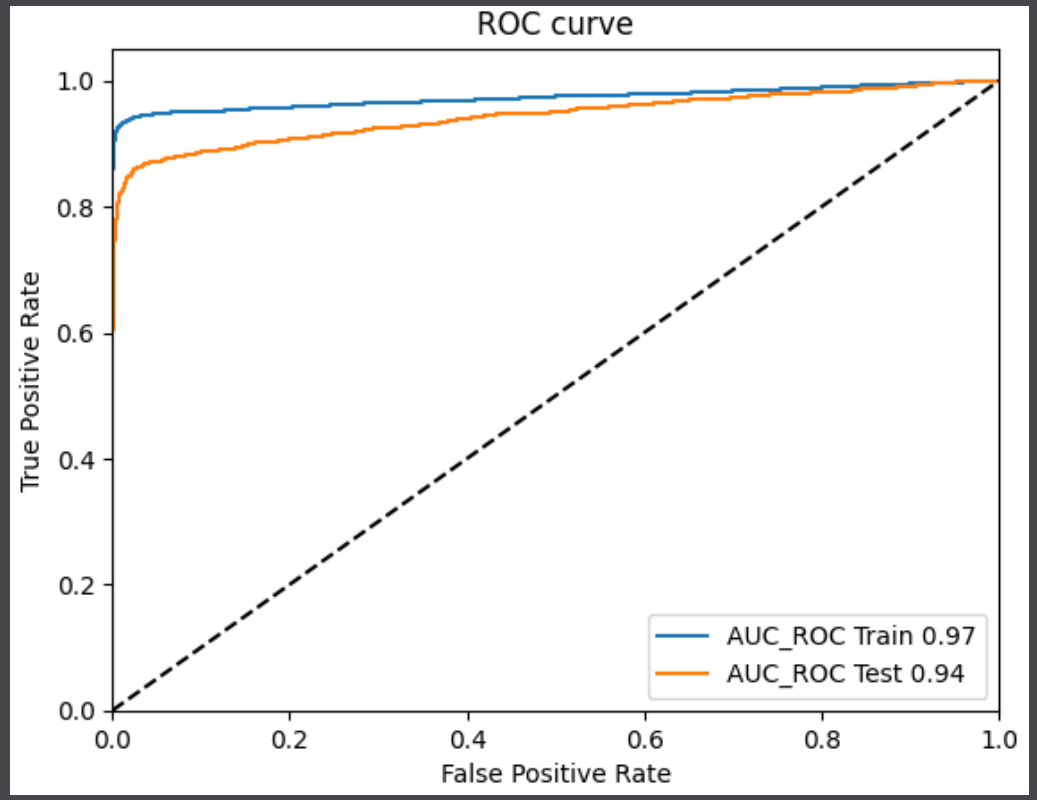
после калибровки
ECE = 0.0046

Распределение прогнозной вероятности и фактического отклика на отложенной выборке до и после калибровки

модель склонности клиентов пары “кредит – смс”:



до калибровки



после калибровки

ROC-AUC модели до и после калибровки

Трудности:

- При log prob трансформации вероятностей, полученных от xgboost во время калибровки методом Probability Calibration Trees возникала ошибка, когда предсказанная вероятность была равна единице
- После калибровки модель становилась переобученной

Решение:

- Заменил все вероятности, равные единице, на 0.9999

```
def transform(x):
    np.place(x, x==1, 0.9999)
    return np.log(x/(1-x))
```

- Настройкой параметров модели (learning_rate, max_depth, test_size) удалось снизить разность между roc-auc на train и test до допустимых значений 0.03

Этап 2

Подготовка данных

```
df=df1.merge(df2,how='outer').merge(df3,how='outer').merge(df4,how='outer')
df=df.loc[:,['client_id','product','channel','score']].sort_values(by=['client_id','product','channel'])
df['client_id']=df['client_id']+1
df.reset_index(drop=True,inplace=True)
df
```

	client_id	product	channel	score
0	1	credit	call	0.999383
1	1	credit	sms	0.000081
2	1	credit_card	call	0.000171
3	1	credit_card	sms	0.000011
4	2	credit	call	0.000756
...	...	...	...	...
79995	19999	credit_card	sms	0.000010
79996	20000	credit	call	0.006400
79997	20000	credit	sms	0.003038
79998	20000	credit_card	call	0.014361
79999	20000	credit_card	sms	0.000219

80000 rows × 4 columns

Построение оптимизационной модели

Инструментарий: Pyomo + GLPK

```
def optimize(frame: pd.DataFrame, channel_limits: dict) -> list:

    ds = frame.copy()

    #создание модели
    model = pyomo.ConcreteModel('model')

    #вектор бинарных переменных задачи
    model.x = pyomo.Var(range(ds.shape[0]), domain=pyomo.Binary, initialize=0)

    #вектор вероятностей
    P = list(ds.score)

    #целевая функция
    obj_expr = sum(P[i] * model.x[i] for i in model.x)
    model.obj = pyomo.Objective(expr=obj_expr, sense=pyomo.maximize)

    # объявление ограничений
    model.c = pyomo.ConstraintList()

    #ограничения на количество коммуникаций в каждом канале
    for channel in ds.channel.unique():
        model.c.add(sum(model.x[i] for i in list(ds[ds.channel==channel].index)) <= channel_limits[channel])

    #ограничения на количество продуктов для каждого клиента (не более одного продукта на клиента)
    for client in ds.client_id.unique():
        model.c.add(sum(model.x[i] for i in list(ds[ds.client_id==client].index)) <= 1)

    solver = pyomo.SolverFactory('glpk')
    results = solver.solve(model,timelimit=500)
```

Результаты

	client_id	product	channel	score	optimal_decision
0	1	credit	call	0.999383	1.0
1	1	credit	sms	0.000081	0.0
2	1	credit_card	call	0.000171	0.0
3	1	credit_card	sms	0.000011	0.0
4	2	credit	call	0.000756	0.0
...	...	...	...	...	...
79995	19999	credit_card	sms	0.000010	0.0
79996	20000	credit	call	0.006400	0.0
79997	20000	credit	sms	0.003038	0.0
79998	20000	credit_card	call	0.014361	0.0
79999	20000	credit_card	sms	0.000219	0.0

80000 rows × 5 columns

Решение по каждому клиенту

		client_cnt
channel	**product**	
call	**credit**	1860
	credit_card	2140
sms	**credit**	4439
	credit_card	2561

Распределение продуктов в каналах

Доп. информация

```
Problem:
- Name: unknown
  Lower bound: 3117.85426959603
  Upper bound: 3117.85426959603
  Number of objectives: 1
  Number of constraints: 20002
  Number of variables: 80000
  Number of nonzeros: 160000
  Sense: maximize
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 1
      Number of created subproblems: 1
  Error rc: 0
  Time: 44.48198080062866
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Выводы:

- Таблица распределения продуктов в каналах показала, что количество клиентов, которым модель посчитала нужным предложить продукт, удовлетворяет ограничениям по каждому из каналов.

- Таблица решений по каждому из клиентов, в свою очередь, позволяет убедиться, что ограничения на количество продуктов для каждого клиента также соблюдены, таким образом, решение является допустимым.

- Получившееся оптимальное значение целевой функции не превышает оценку “сверху” - среднюю вероятность в 10-х бакетах каждой модели, умноженную на общее количество клиентов, которым возможно сделать предложение по продукту.

Оптимизация целевой функции с учётом доходности

Внесённые изменения

```python
def optimize(frame: pd.DataFrame, channel_limits: dict, with_profit: bool) -> list:

    ds = frame.copy()

    #создание модели
    model = pyomo.ConcreteModel('model')

    #вектор бинарных переменных задачи
    model.x = pyomo.Var(range(ds.shape[0]), domain=pyomo.Binary, initialize=0)

    #вектор вероятностей
    P = list(ds.score)

    if(with_profit):
        #формирование колонки доходности по каждому клиенту в зависимости от пары "продукт-канал"
        ds['profit']=10000
        ds.loc[ds['product']=='credit_card','profit']=13000
        ds['profit'] = ds['profit'].astype(float)
        ds.loc[ds['channel']=='call','profit']-=50
        ds.loc[ds['channel']=='sms','profit']-=1.5
        #вектор доходности
        D = list(ds['profit'])
        #целевая функция с доходностью
        obj_expr = sum(D[i] * P[i] * model.x[i] for i in model.x)
    else:
        #целевая функция без доходности
        obj_expr = sum(P[i] * model.x[i] for i in model.x)
```

Результаты

	client_id	product	channel	score	optimal_decision
0	1	credit	call	0.999383	1.0
1	1	credit	sms	0.000081	0.0
2	1	credit_card	call	0.000171	0.0
3	1	credit_card	sms	0.000011	0.0
4	2	credit	call	0.000756	0.0
...	...	...	...	...	...
79995	19999	credit_card	sms	0.000010	0.0
79996	20000	credit	call	0.006400	0.0
79997	20000	credit	sms	0.003038	0.0
79998	20000	credit_card	call	0.014361	0.0
79999	20000	credit_card	sms	0.000219	0.0

80000 rows × 5 columns

Решение по каждому клиенту

		client_cnt
channel	**product**	
call	**credit**	1743
	credit_card	2257
sms	**credit**	3571
	credit_card	3429

Распределение продуктов в каналах

Доп. информация

```
Problem:
- Name: unknown
  Lower bound: 37071954.9039166
  Upper bound: 37071954.9039166
  Number of objectives: 1
  Number of constraints: 20002
  Number of variables: 80000
  Number of nonzeros: 160000
  Sense: maximize
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 1
      Number of created subproblems: 1
  Error rc: 0
  Time: 44.488093852996826
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Сравнение полученных решений

Количество общих предложений

```
print(sum(np.array(optimal_decisions_1) * np.array(optimal_decisions_2)))
```

```
10002.0
```

Случаи, когда предложения двух решений различаются

```
df.loc[df['optimal_decision_1']!=df['optimal_decision_2']]
```

	client_id	product	channel	score	optimal_decision_1	optimal_decision_2
221	56	credit	sms	0.791216	1.0	0.0
223	56	credit_card	sms	0.762277	0.0	1.0
309	78	credit	sms	0.892266	1.0	0.0
311	78	credit_card	sms	0.760747	0.0	1.0
313	79	credit	sms	0.984992	1.0	0.0
...	...	...	...	...	...	...
79707	19927	credit_card	sms	0.007731	0.0	1.0
79837	19960	credit	sms	0.024074	1.0	0.0
79839	19960	credit_card	sms	0.019187	0.0	1.0
79913	19979	credit	sms	0.999544	1.0	0.0
79915	19979	credit_card	sms	0.998253	0.0	1.0

1996 rows × 6 columns

1

channel	product	client_cnt
call	credit	1743
	credit_card	2257
sms	credit	3571
	credit_card	3429

2

channel	product	client_cnt
call	credit	1860
	credit_card	2140
sms	credit	4439
	credit_card	2561

Распределение продуктов в каналах с учётом доходности (1) и без (2)

$\Delta_{12} = 117$,
$\Delta_{12} = 868$

Вывод:

Сравнение таблиц распределений продуктов в каналах и таблиц решений по клиентам показало, что бóльшая часть предложений, полученных двумя оптимизационными моделями совпадает; различия же обусловлены следующим: в модели, учитывающей доходность, когда вероятность отклика по кредитной карте одного и того же клиента ниже вероятности отклика по кредиту, то ему всё равно предлагается кредитная карта, т.к. доходность от неё выше

.

Обратная связь:

К ограничениям можно было бы добавить ограничение в виде общего депозита на каналы связи

.