Маркетинговая оптимизация

Артём Копань

Содержание

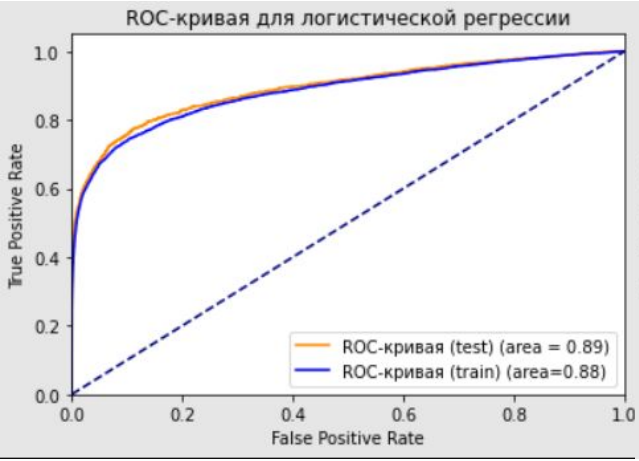
1. Построение моделей машинного обучения
2. Калибровка моделей машинного обучения
3. Решение задачи маркетинговой оптимизации
4. Решение задачи оптимизации прибыли

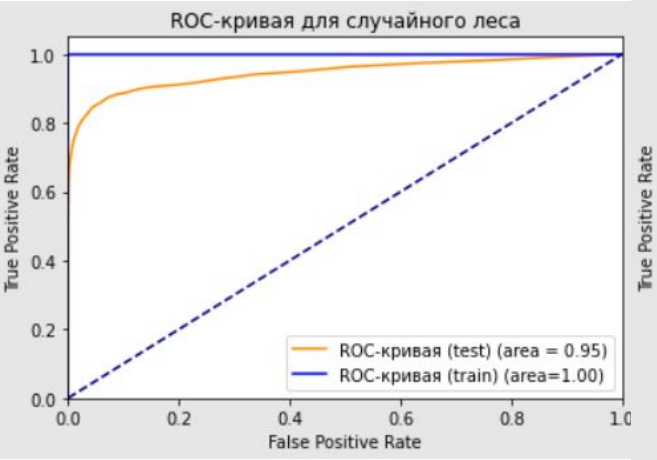
1. Построение моделей машинного обучения

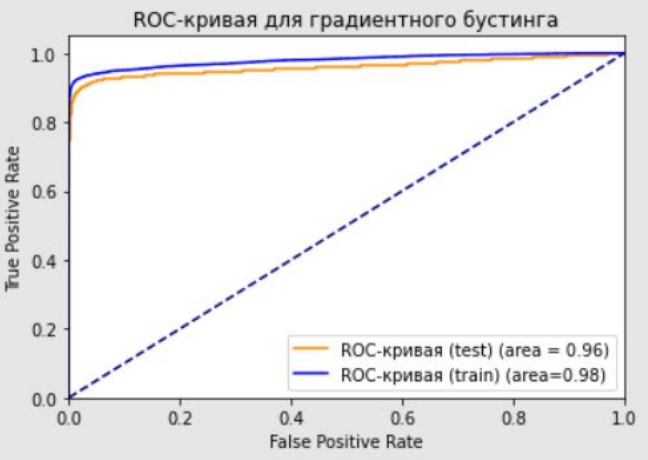
Были обучены следующие модели:

- логистическая регрессия
- случайный лес
- градиентный бустинг на решающих деревьях (catboost)

ROC-AUC


ROC-кривая для логистической регрессии
True Positive Rate
False Positive Rate
ROC-кривая (test) (area = 0.89)
ROC-кривая (train) (area=0.88)


ROC-кривая для случайного леса
True Positive Rate
False Positive Rate
ROC-кривая (test) (area = 0.95)
ROC-кривая (train) (area=1.00)


ROC-кривая для градиентного бустинга
True Positive Rate
False Positive Rate
ROC-кривая (test) (area = 0.96)
ROC-кривая (train) (area=0.98)

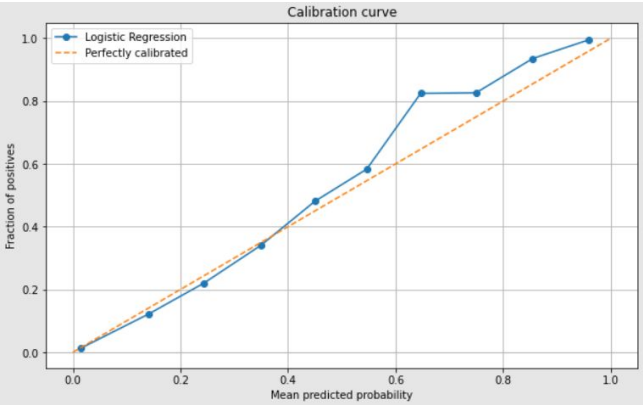
Метрики

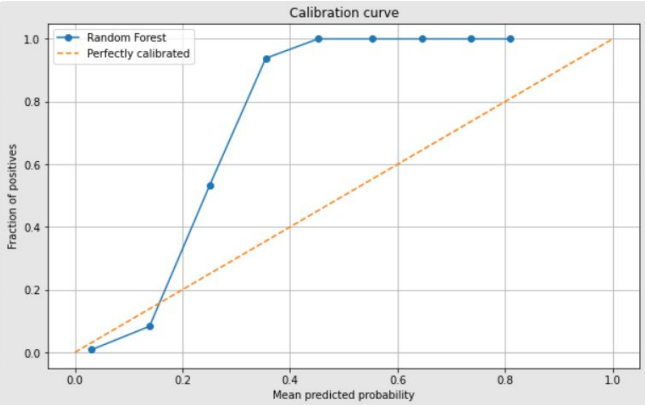
Модель	Accuracy	Precision	Recall	F1
Log. Regression	0.96	0.93	0.74	0.80
Random Forest	0.95	0.97	0.66	0.73
Gradient Boosting	0.98	0.98	0.87	0.92

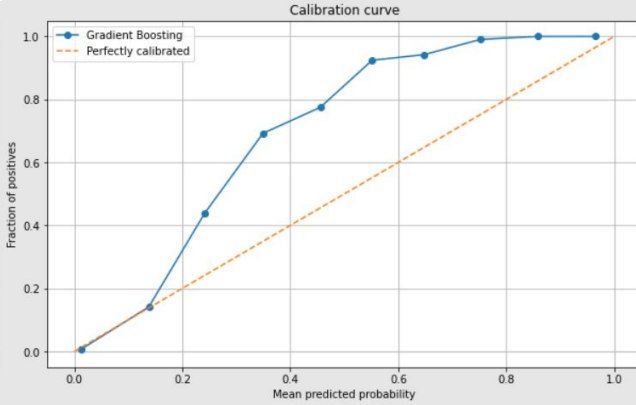
2. Калибровка моделей машинного обучения

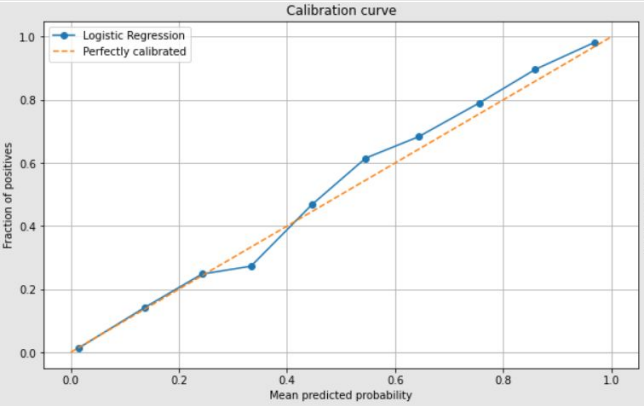
```
isotonic_clf = CalibratedClassifierCV(lr_model, cv=2, method="isotonic")

isotonic_clf.fit(X_test, y_test)
```

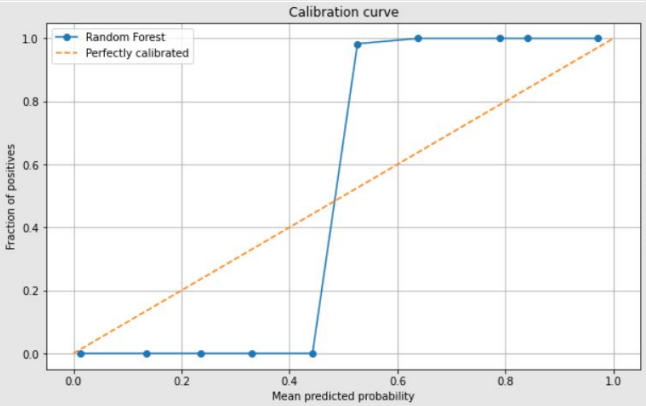
Графики калибровки

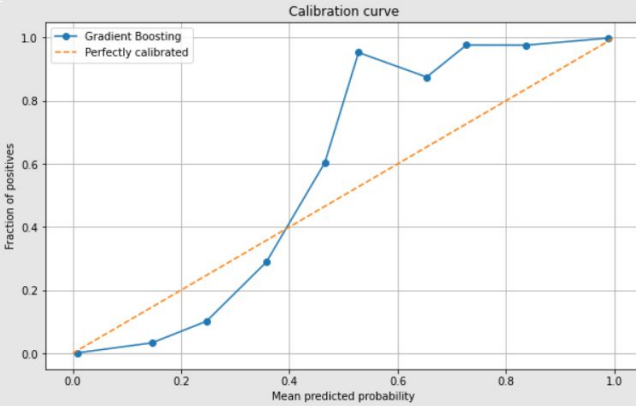












3. Оптимизация

Оптимизация проводилась с помощью библиотеки Pyomo.

Использовались целочисленные солверы cbc и glpk.

Даны вероятности $p_{ij}^{(k)}$, где i, j $\in \{1, 2\}$, i — номер продукта, j — номер канала связи, k — номер клиента.

Нужно максимизировать скалярное произведение (p, x), где $x_{ij}^{(k)}$ — двоичные переменные, обозначающие, предлагаем ли мы клиенту i-й продукт по j-му каналу связи.

Немного кода

```
model.objective = Objective(expr=sum_product(df['score'], model.x), sense=maximize)

model.constraints = ConstraintList()
for channel, limit in channel_limits.items():
    indices = df[df['channel'] == channel].index.tolist()
    model.constraints.add(quicksum(model.x[i] for i in indices) <= limit)

for client_id in df['client_id'].unique():
    indices = df[df['client_id'] == client_id].index.tolist()
    model.constraints.add(quicksum(model.x[i] for i in indices) <= 1)

solver = SolverFactory('glpk') # glpk, cbc
result = solver.solve(model, tee=True)
```

Результаты

glpk:

```
INTEGER OPTIMAL SOLUTION FOUND

Time used:   5.2 secs

obj =   2.938536789e+03
```

channel	product	client_cnt
call	credit	1641
	credit card	2359
sms	credit	3635
	credit card	3365

cbc:

```
Result - Optimal solution found

Time (Wallclock seconds):       33.95

Objective value:                -2938.55780513
```

channel	product	client_cnt
call	credit	1641
	credit card	2359
sms	credit	3635
	credit card	3365

4. Оптимизация прибыли

Дана величина выручки от продажи кредита или кредитной карты и издержки на совершение звонка или отправку СМС.

Выручка (revenue) от кредита $\mathbf{r}_{cr}$ = 10000 руб.

Выручка от кредитной карты $\mathbf{r}_{cc}$ = 13000 руб.

Стоимость (cost) звонка $\mathbf{c}_{call}$ = 50 руб.

Стоимость отправки СМС $\mathbf{c}_{sms}$ = 1.5 руб.

Задача: учитывая величины выручки и издержек, максимизировать маржу (прибыль).

Формула

$$margin = revenue - cost =$$
$$\sum_k p_{11}^{(k)}(r_{cc} - c_{call})x_{11}^{(k)} + \sum_k p_{12}^{(k)}(r_{cc} - c_{sms})x_{12}^{(k)} +$$
$$\sum_k p_{21}^{(k)}(r_{cr} - c_{call})x_{21}^{(k)} + \sum_k p_{22}^{(k)}(r_{cr} - c_{sms})x_{22}^{(k)}$$

Ещё немного кода

```
model.objective = Objective(expr=sum_product(df['margin_score'], model.x), sense=maximize)

model.constraints = ConstraintList()
for channel, limit in channel_limits.items():
    indices = df[df['channel'] == channel].index.tolist()
    model.constraints.add(quicksum(model.x[i] for i in indices) <= limit)

for client_id in df['client_id'].unique():
    indices = df[df['client_id'] == client_id].index.tolist()
    model.constraints.add(quicksum(model.x[i] for i in indices) <= 1)

solver = SolverFactory('glpk') # glpk, cbc
result = solver.solve(model, tee=True)

return [model.x[i].value for i in range(df.shape[0])]
```

Результаты

glpk:

```
INTEGER OPTIMAL SOLUTION FOUND

Time used:   5.2 secs

obj =   3.450885243e+07
```

channel	product	client_cnt
call	credit	1489
	credit card	2511
sms	credit	3176
	credit card	3824

cbc:

```
Result - Optimal solution found

Time (Wallclock seconds):       7.00

Objective value:                -34508852.43226650
```

channel	product	client_cnt
call	credit	1489
	credit card	2511
sms	credit	3176
	credit card	3824

Сравнение решений

Сравним решение задачи оптимизации маржи с решением простой задачи оптимизации.

Простая оптимизация:

channel	product	client_cnt
call	credit	1641
	credit card	2359
sms	credit	3635
	credit card	3365

Оптимизация маржи:

channel	product	client_cnt
call	credit	1489
	credit card	2511
sms	credit	3176
	credit card	3824

Сравнение решений

Количество общих результатов для двух задач: 78730

Количество различающихся результатов для двух задач: 1270

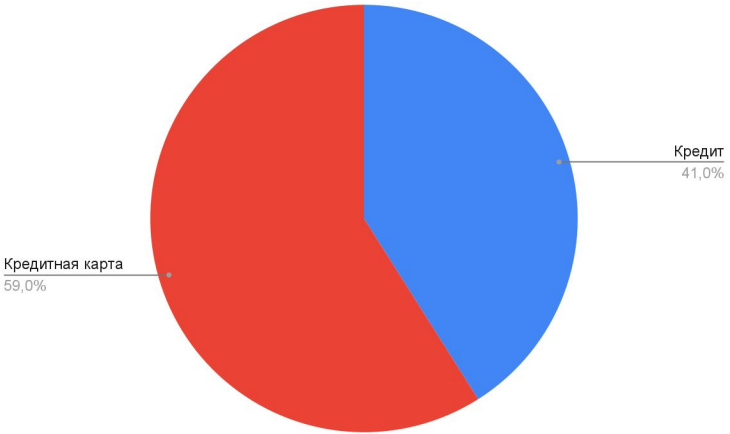
Количество общих сделанных предложений в двух задачах: 10365

Разница в количестве предложенных кредитов и кредитных карт в двух решениях при использовании звонков: 152

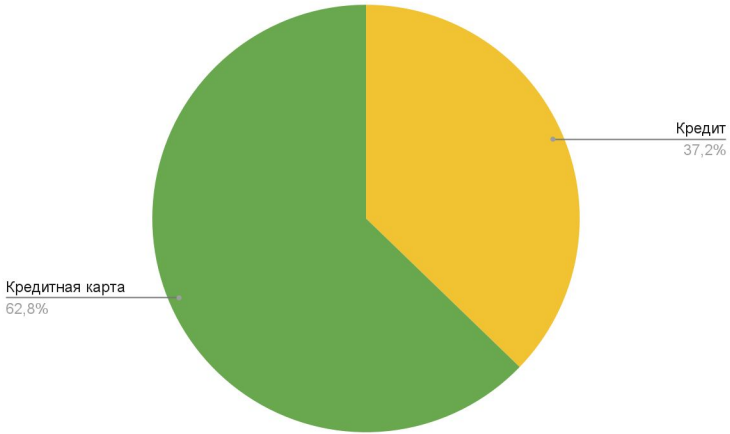
Разница в количестве предложенных кредитов и кредитных карт в двух решениях при использовании SMS: 459

Сравнение решений

Звонки — первая задача

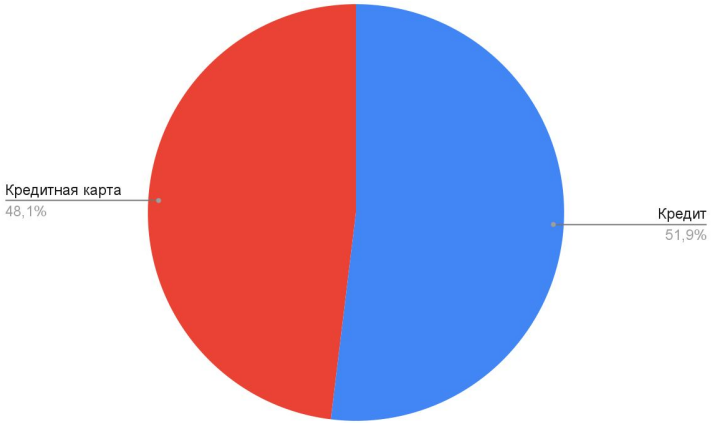


Звонки — вторая задача

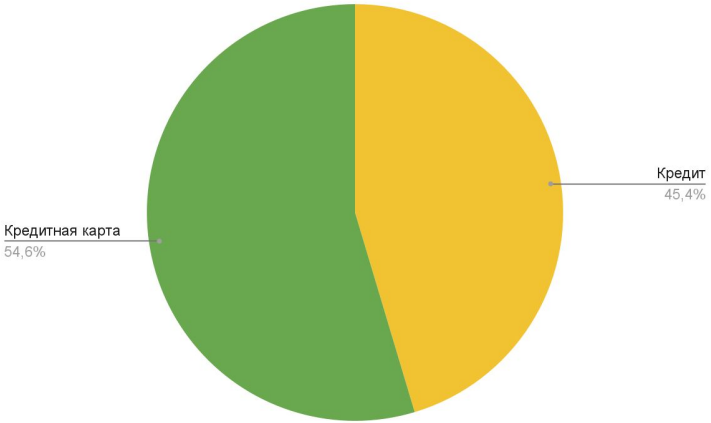


Сравнение решений

SMS — первая задача



SMS — вторая задача



Выводы

- При оптимизации маржи при использовании SMS есть перекос в сторону кредитных карт, при решении обычной задачи — наоборот, в сторону кредитов
- При использовании звонков в обоих задачах количество кредитных карт получается больше
- Большинство предложений — общие и в первой, и во второй задаче, но есть и отличающиеся результаты.