

Образовательная платформа: SkillFactory

Специализация: Data Science

Группа: DST-17

Юнит 5. Проект: "Компьютер говорит «Нет» "

Выполнил: Владимир Юшманов



1. Импорт и объединение данных

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import os
import re
import math
import numpy as np
import pandas as pd

# Загружаем набор собственных функций
import myfunction as mf

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# фиксируйте RANDOM_SEED и версию пакетов, чтобы эксперименты были воспроизводимы:
RANDOM_SEED = 42
!pip freeze > requirements.txt

/kaggle/input/sf-dst-scoring/sample_submission.csv
/kaggle/input/sf-dst-scoring/test.csv
/kaggle/input/sf-dst-scoring/train.csv
```

In [2]:

```
# Читаем датасеты
DATA_DIR = '/kaggle/input/sf-dst-scoring/'
try:
    df_train = pd.read_csv(DATA_DIR + 'train.csv')
    df_test = pd.read_csv(DATA_DIR + 'test.csv')
    sample_submission = pd.read_csv(DATA_DIR + 'sample_submission.csv')
except:
    df_train = pd.read_csv('train.csv')
    df_test = pd.read_csv('test.csv')
    sample_submission = pd.read_csv('sample_submission.csv')

# ВАЖНО! для корректной обработки признаков объединяем трейн и тест в один датасет
df_train['sample'] = 1 # помечаем где у нас трейн
df_test['sample'] = 0 # помечаем где у нас тест

data = df_test.append(df_train, sort=False).reset_index(drop=True) # объединяем

# Выводим сводку о содержании датасета
mf.brief_summary(data, [100,75,75,75,75,75,150])
```

Признак	# заполненных	тип данных	% заполнения	# пропусков	# уникальных	диапазон значений / примеры
client_id	110148	int64	100	0	110148	74835 49000 102650
app_date	110148	object	100	0	120	22MAR2014 25FEB2014 20APR2014
education	109670	object	99.6	478	5	GRD GRD SCH
sex	110148	object	100	0	2	M F F
age	110148	int64	100	0	52	29 32 53
car	110148	object	100	0	2	Y Y N
car_type	110148	object	100	0	2	Y Y

Состав признаков.

Описание:

- **app_date** - дата запроса, временная переменная, требует обработки.
- **education** - уровень образования, категориальная переменная, требует обработки и исправления пропущенных значений.
- **sex** - двоичная переменная, требует обработки.
- **age** - непрерывная переменная, требует обработки.
- **car** - наличие машины, двоичная переменная, требует обработки.
- **car_type** - наличие иномарки, бинарная переменная, требует обработки.
- **decline_app_cnt** - количество отклоненных запросов, непрерывная переменная.
- **good_work** - признак хорошо оплачиваемой работы, бинарная переменная.
- **score_bki** - внутренний рейтинг БКИ (агентство кредитной информации), непрерывная переменная, все значения которой отрицательны.
- **bki_request_cnt** - количество запросов в БКИ (агентство кредитной информации), непрерывная переменная.
- **region_rating** - рейтинг региона, категориальная переменная.
- **home_address** - классификатор домашнего адреса, категориальная переменная.
- **work_address** - классификатор рабочих адресов, категориальная переменная.
- **income** - уровень дохода клиента, непрерывная переменная.
- **sna** - уровень связи с другими клиентами, категориальная переменная.
- **first_time** - сколько времени клиент находится в базе данных, категориальная переменная.
- **foreign_passport** - наличие паспорта, бинарная переменная, требует обработки.
- **default** - по умолчанию в прошлом, бинарная целевая переменная.

In [3]:

```
time_cols = ['app_date']
cat_cols = ['education', 'region_rating', 'home_address', 'work_address', 'sna', 'first_time']
bin_cols = ['sex', 'car', 'car_type', 'good_work', 'foreign_passport']
num_cols = ['age', 'decline_app_cnt', 'score_bki', 'bki_request_cnt', 'income']
```

2. Разведывательный анализ данных (EDA)

2.1. Временной признак

Признак преобразован в формат даты, дополнен числовым признаком периода, между датой добавления и последней датой списка.

Распределение добавленного признака является непрерывным, поэтому он добавлен в список **num_cols**

In [4]:

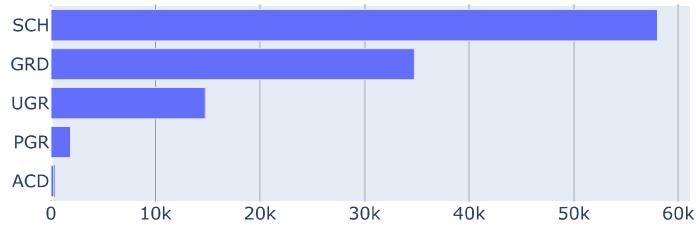
```
data['app_date'] = pd.to_datetime(data['app_date'], format='%d%b%Y')
data_min = min(data['app_date'])
data['days'] = (data['app_date'] - data_min).dt.days.astype('int')
if 'days' not in num_cols : num_cols.append('days')
```

2.2. Категориальные признаки

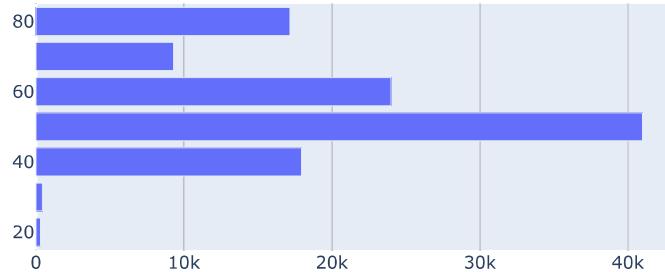
In [5]:

```
for column in cat_cols:
    mf.view_horiz_bar_n_table(data, column)
```

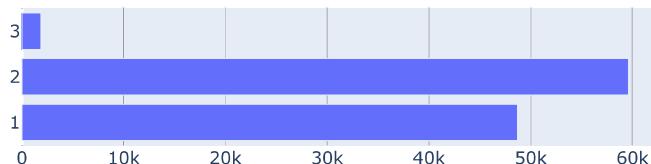
Распределение признака education



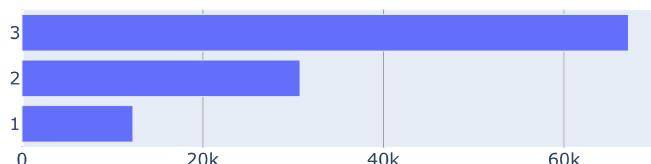
Распределение признака region_rating



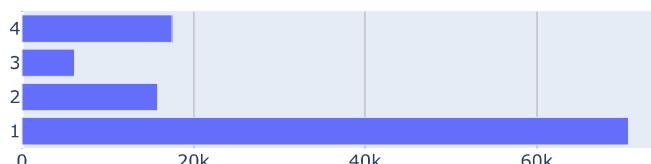
Распределение признака home_address



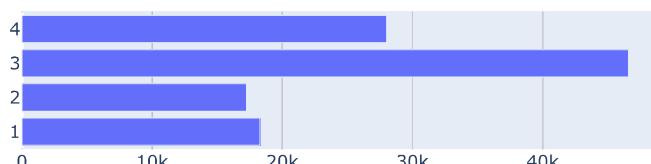
Распределение признака work_address



Распределение признака sna



Распределение признака first_time



Признак **education** нуждается в трансформации в числовой формат. При преобразовании не будем учитывать количественное распределение признаков уровня образования в популяции. Можно будет использовать эту идею при необходимости уточнения модели. Сейчас соответствие уровня образования и числовых признаков выберем исходя из распределения по количеству обращений за кредитами.

```
In [6]:  
education_distrib = {'SCH': 1  
                     , 'GRD': 2  
                     , 'UGR': 3  
                     , 'PGR': 4  
                     , 'ACD': 5}  
  
if 'SCH' in list(data['education']):  
    data['education'] = data['education'].map(education_distrib)
```

Остальные категориальные признаки оставим без изменений.

2.3. Бинарные признаки

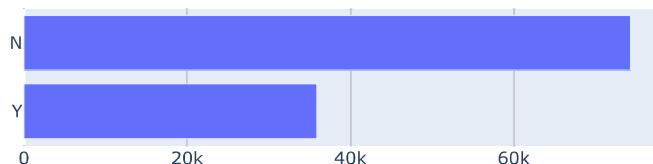
In [7]:

```
for column in bin_cols:
    mf.view_horiz_bar_n_table(data, column)
```

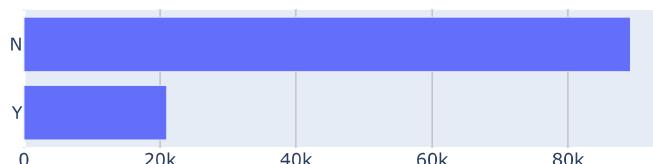
Распределение признака sex



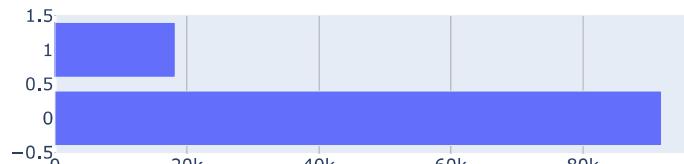
Распределение признака car



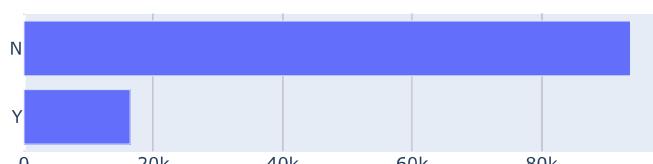
Распределение признака car_type



Распределение признака good_work



Распределение признака foreign_passport



Преимущественно клиентами банка являются женщины.

Большинство клиентов не владеет автотранспортом. Владельцы машин предпочитают транспортные средства отечественного производства.

Среди клиентов банка преобладают люди, работающие на непrestижной работе и не имеющие заграничного паспорта.

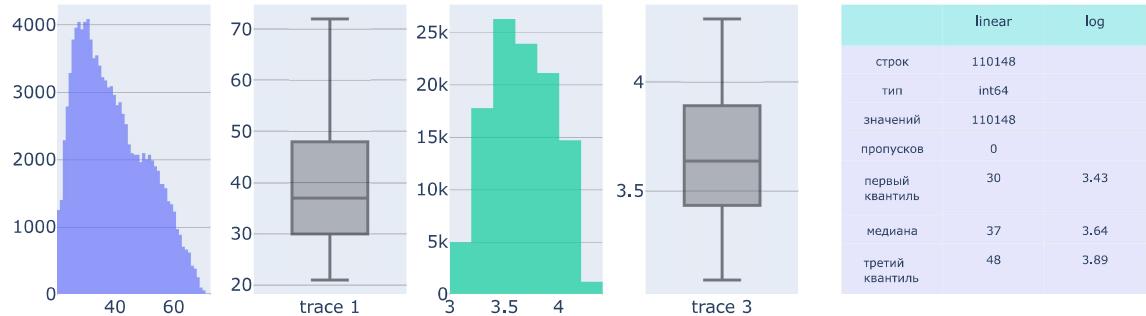
2.4. Признаки с непрерывным распределением

In [8]:

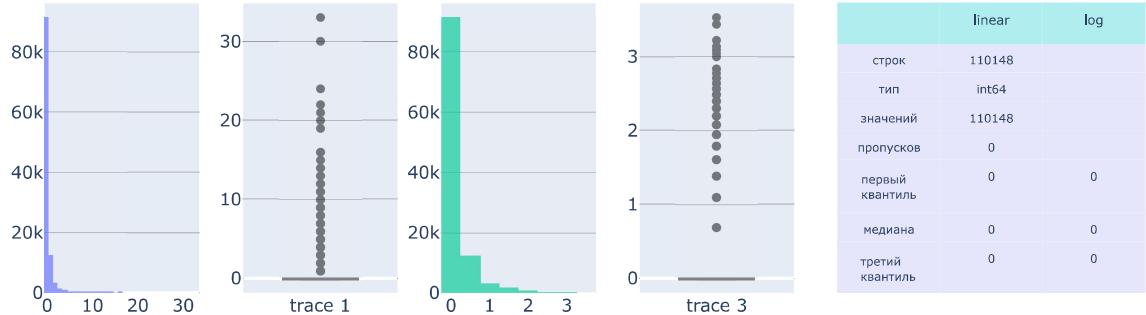
```
# Для последующего логарифмирования преобразуем признак score_bki так, чтобы все его значения
# были положительны. Для этого добавим к значениям целое число, которое не меньше модуля
# минимального значения исходного ряда.

for column in num_cols:
    data = mf.view_histogram_n_boxplot(data, column)
```

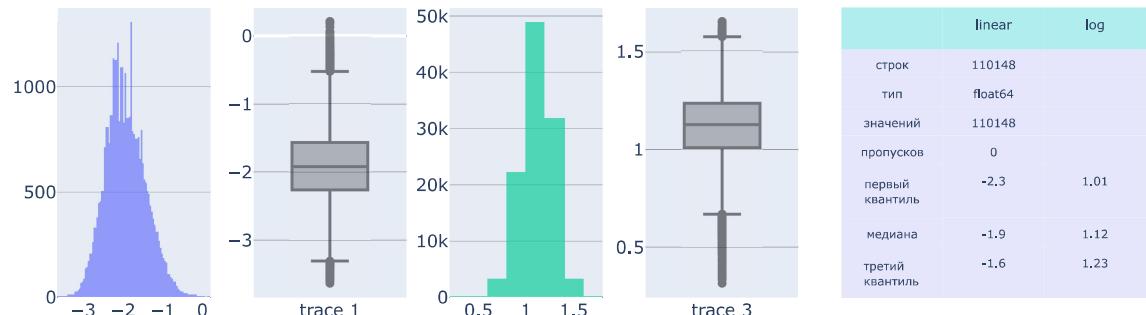
Линейные значения и логарифм признака age, границы и количество выбросов



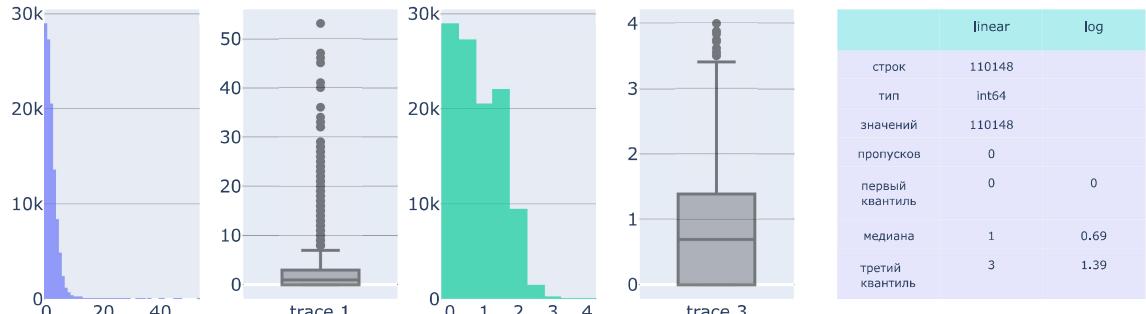
Линейные значения и логарифм признака decline_app_cnt, границы и количество выбросов



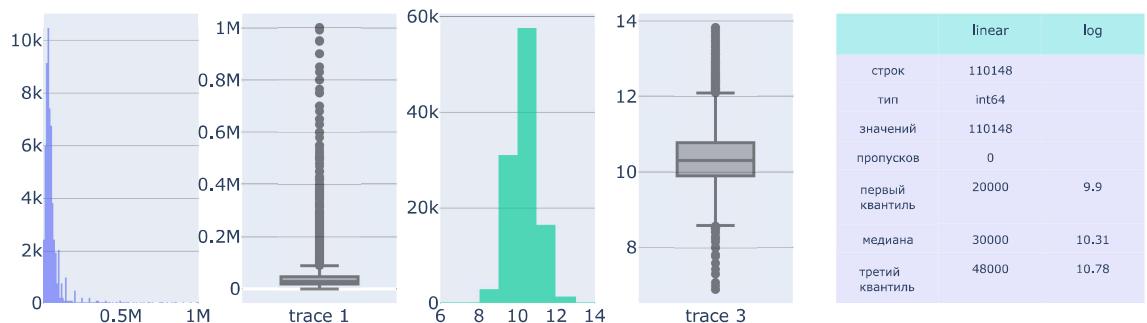
Линейные значения и логарифм признака score_bki, границы и количество выбросов



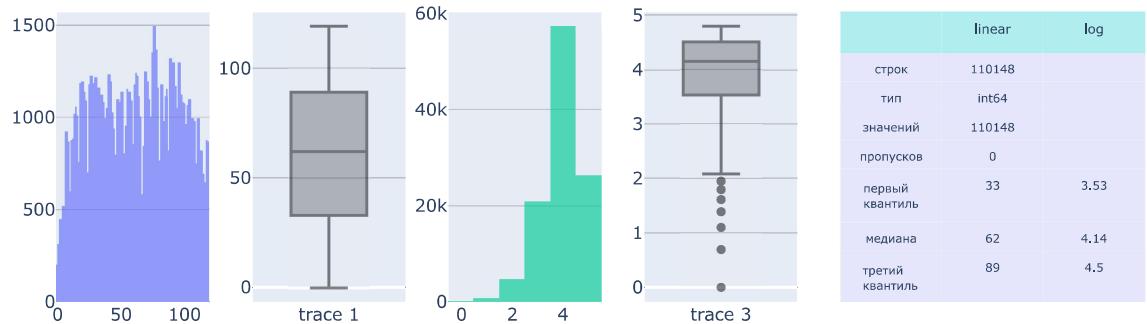
Линейные значения и логарифм признака bki_request_cnt, границы и количество выбросов



Линейные значения и логарифм признака income, границы и количество выбросов



Линейные значения и логарифм признака days, границы и количество выбросов



Проведено исследование различий распределений линейной и логарифмической номинативных признаков. Произведен расчет количества выбросов для линейных и логарифмических значений. Выбраны признаки, логарифмическое преобразование которых позволит сократить количество выбросов.

Признак возраста заемщиков также логарифмирован, поскольку такое представление обеспечивает приближение распределения переменной к нормальному.

In [9]:

```
to_log_cols = ['age', 'decline_app_cnt', 'bki_request_cnt', 'income']
data = mf.get_log_sign(data, to_log_cols)
display(data.head(3))
```

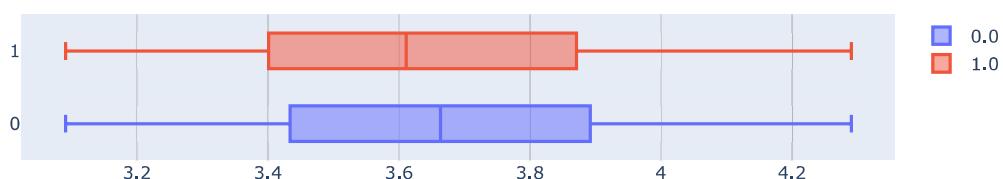
	client_id	app_date	education	sex	age	car	car_type	decline_app_cnt	good_work	score_bki	bki_request_cnt	regi
0	74835	2014-03-22	2.0	M	3.401197	Y	Y	0.000000	0	-2.271884	1.386294	50
1	17527	2014-01-24	1.0	F	3.688879	N	N	1.791759	0	-1.504999	1.098612	50
2	75683	2014-03-23	3.0	M	3.931826	Y	Y	0.000000	0	-1.691339	0.693147	50

Исследование распределения номинативных переменных с учетом влияния на целевую переменную произведем в отношении логарифмированных признаков.

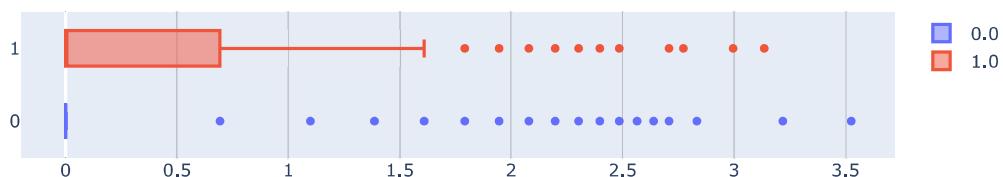
In [10]:

```
for column in num_cols:
    mf.view_sign_influence_on_h_box_plot(data[data['sample']==1], column, 'default')
```

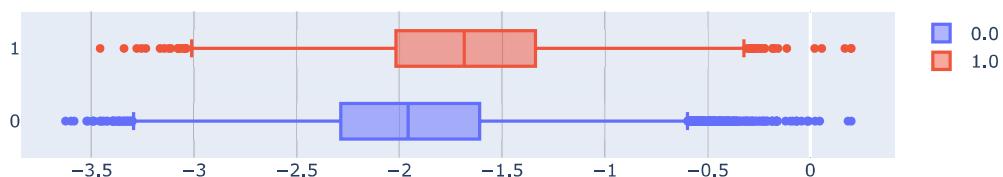
Распределения признака age в зависимости от значения default



Распределения признака decline_app_cnt в зависимости от значения default



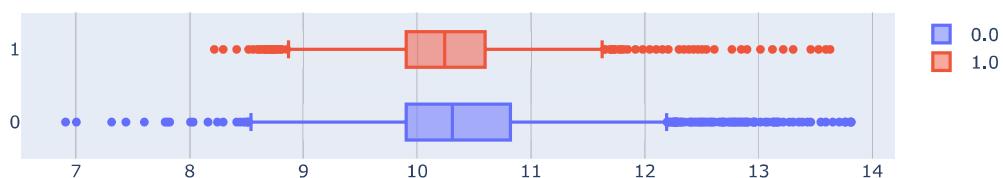
Распределения признака score_bki в зависимости от значения default



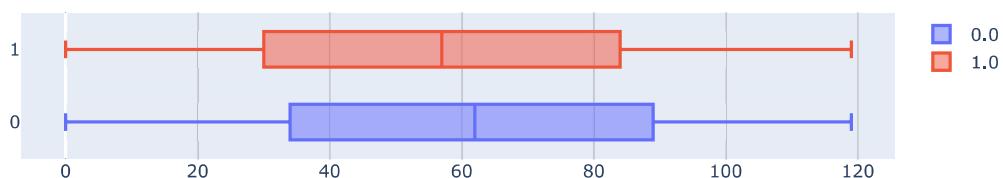
Распределения признака bki_request_cnt в зависимости от значения default



Распределения признака income в зависимости от значения default



Распределения признака days в зависимости от значения default

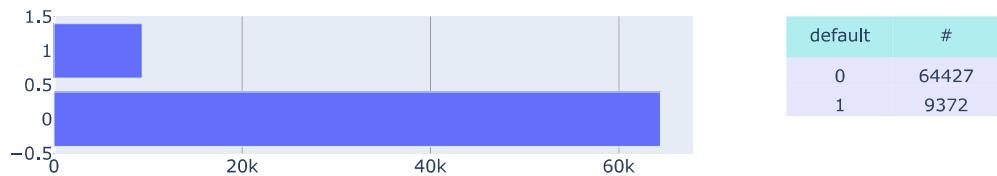


2.5. Целевая переменная

In [11]:

```
mf.view_horiz_bar_n_table(data[data['sample'] == 1], 'default')
```

Распределение признака default



Количество надежных клиентов существенно превышает количество неплательщиков

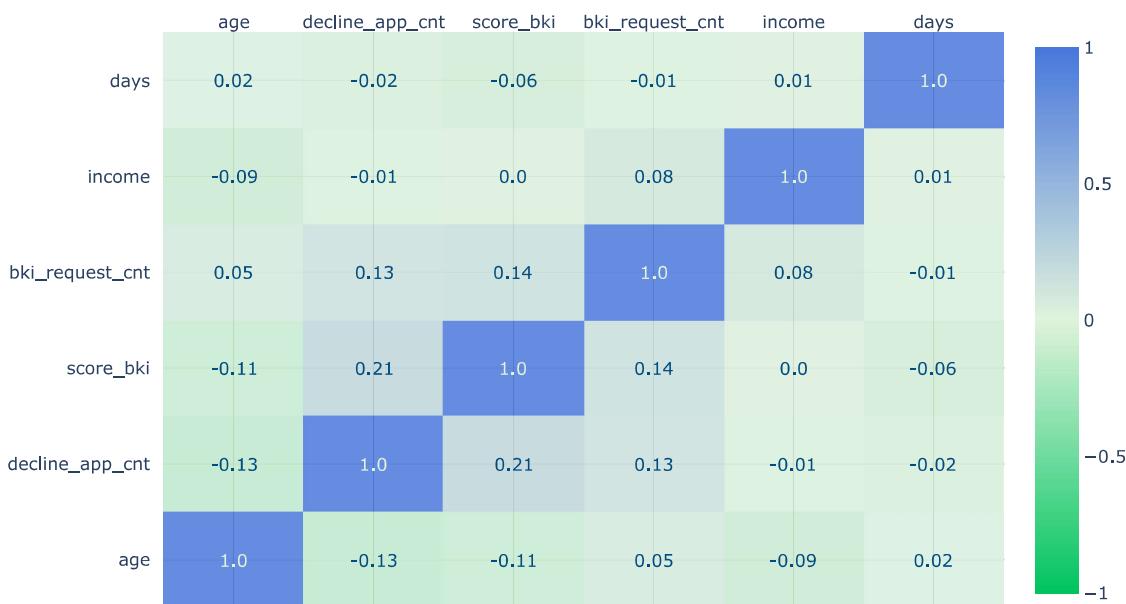
2.5. Корреляционный анализ признаков

2.5.1. Построение матрицы корреляций

In [12]:

```
mf.show_heatmap(data[data['sample'] == 1][num_cols])
```

Тепловая карта матрицы корреляций



Наибольшую корреляцию среди номинативных переменных имеют признаки **score_bki** и **decline_app_cnt**

Применим к этой паре метод главных компонент

2.5.2. Применение метода главных компонент (PCA)

In [13]:

```
list_for_pca = ['score_bki', 'decline_app_cnt']
new_sign_name = 'score_decl'
delete_leave_cols = [1,1] # Для list_for_pca: 1 - оставить; 0 - удалить
data, num_cols = mf.pca_distribution(data, list_for_pca, new_sign_name, num_cols, delete_leave_cols)
```

Признаки 'score_bki', 'decline_app_cnt' по методу главных компонент преобразованы в признак 'score_decl'.

Вектор главных компонент: [0.7071067811865475, 0.7071067811865475]

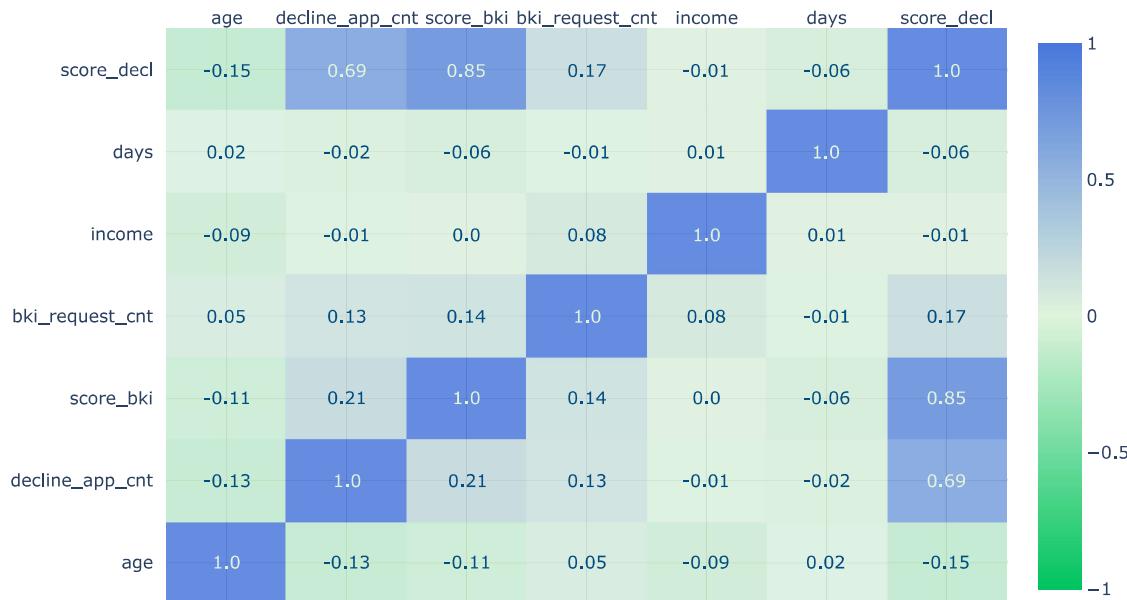
После преобразования, датасет содержит признаков - 22; строк - 110148, из них 73799 - train.

2.5.3. Контроль результатов применения PCA

In [14]:

```
mf.show_heatmap(data[data['sample'] == 1][num_cols])
```

Тепловая карта матрицы корреляций



2.5.4. Степень влияния признаков на целевую переменную

Предварительно следует преобразовать буквенные обозначения бинарных переменных в числовой формат

In [15]:

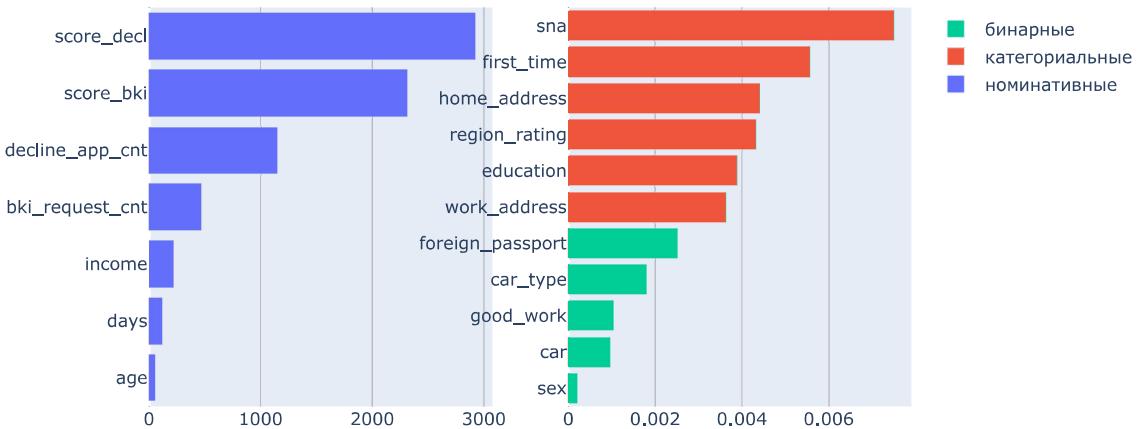
```
for column in bin_cols:
    data = mf.get_label_encoder(data, column)
```

Переменные признака sex преобразованы в соответствии со словарем {0: 'F', 1: 'M'}
 Переменные признака car преобразованы в соответствии со словарем {0: 'N', 1: 'Y'}
 Переменные признака car_type преобразованы в соответствии со словарем {0: 'N', 1: 'Y'}
 Переменные признака good_work преобразованы в соответствии со словарем {0: 0, 1: 1}
 Переменные признака foreign_passport преобразованы в соответствии со словарем {0: 'N', 1: 'Y'}

In [16]:

```
mf.view_important_sign(data, num_cols, bin_cols, cat_cols, 'default')
```

Степень влияния признаков на целевую переменную



3. Предподготовка данных (Data Preprocessing)

3.1. Преобразование категориальных признаков в dummy-переменные

In [17]:

```
dumm_cols = cat_cols
dumm_cols.remove('education')
data = pd.get_dummies(data, prefix=dumm_cols, columns=dumm_cols)
```

3.2. Интеллектуальное заполнение пропусков признака education

In [18]:

```
model, data_empty_ed = mf.train_education(data, 100, RANDOM_SEED)
```

```
predict = np.round(model.predict(data_empty_ed))
```

```
index_empty_ed = data[data['education'].isna()].index
data.loc[index_empty_ed, 'education'] = predict
```

```
[Parallel(n_jobs=-1)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done 42 tasks      | elapsed:  12.8s
[Parallel(n_jobs=-1)]: Done 100 out of 100 | elapsed:  29.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done 42 tasks      | elapsed:   0.1s
```

Метрики: mae = 0.5754, mse = 0.6898, rmse = 0.8305

Время выполнения: 0:00:29.770162

```
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   0.3s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done 42 tasks      | elapsed:   0.0s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:   0.0s finished
```

3.3. Преобразования дополненного признака и удаление нечисловых признаков

```
In [19]:  
data = data.fillna(0)  
# произведена мин-макс стандартизация (за исключением списка столбцов)  
data = pd.get_dummies(data, prefix=['education'], columns=['education'])  
  
df_preproc = mf.delete_string_sign(data)  
df_preproc = df_preproc.drop(columns = ['client_id', 'app_date'])
```

3.4. Поэлементный контроль

```
In [20]:  
display(df_preproc.sample(3))  
display(df_preproc.describe().head(1))
```

	sex	age	car	car_type	decline_app_cnt	good_work	score_bki	bki_request_cnt	income	foreign_passport	sar
1373	1	3.496508	0	0	0.693147	0	-2.885211	1.609438	10.203629	0	0
19514	0	3.713572	0	0	0.000000	0	-1.423316	0.693147	9.903538	0	0
2790	0	3.401197	0	0	0.000000	0	-0.994739	1.098612	11.695255	1	0
<hr/>											
	sex	age	car	car_type	decline_app_cnt	good_work	score_bki	bki_request_cnt	income	foreign_passport	sar
count	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0	110148.0

3.5. Выделение тестовой части датасета

```
In [21]:  
data_train = df_preproc.query('sample == 1').drop(['sample'], axis=1)  
data_test = df_preproc.query('sample == 0').drop(['sample'], axis=1)
```

4. Модель

4.1 Обучение модели

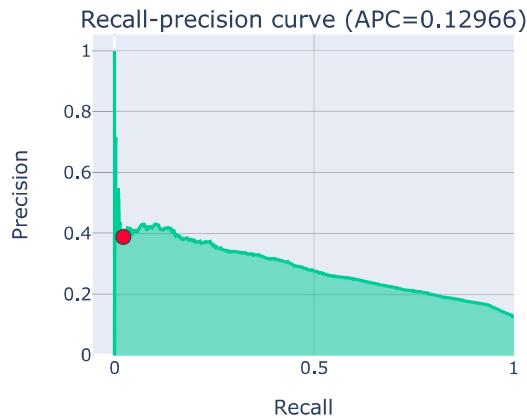
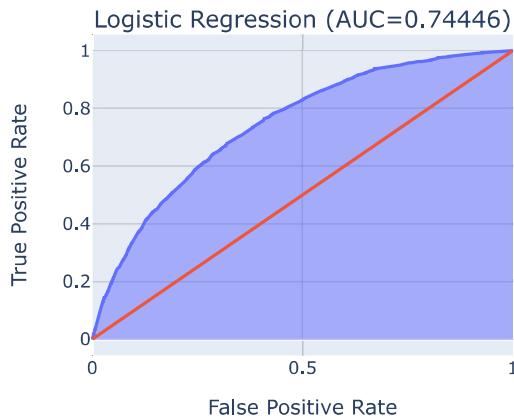
```
In [22]:  
from sklearn.linear_model import LogisticRegression  
  
model = LogisticRegression(max_iter=500, random_state=RANDOM_SEED)  
X_train, X_test, y_train, y_test, model, y_pred_prob, y_pred = mf.traning_model(data_train, data_test, model, RANDOM_SEED)  
  
Время выполнения: 0:00:03.878097
```

4.2. Оценка модели

4.2.1 Метрики качества

In [23]:

```
mf.view_metrics_LogisticRegression(model, X_train, y_train, X_test, y_test, ['Default', 'Non-default'])
```



Check metrics

	Значение	Описание
Positive, P	1827	Истинный Default
Negative, N	12933	Истинный Non-default
True Positive, TP	40	Корректная идентификация Default
True Negative, TN	12871	Корректная идентификация Non-default
False Positive, FP	62	Ошибочная идентификация Default
False Negative, FN	1787	Ошибочная идентификация Non-default
Accuracy	0.87473	Точность: Accuracy=(TP+TN)/(P+N)
Precision	0.39216	Precision = TP/(TP+FP)
Recall	0.02189	Recall = TP/P
F1-score	0.04147	Метрика, объединяющая Precision и Recall
ROC_AUC	0.74446	ROC AUC Score

Метрика ROC-AUC выглядит очень обнадеживающе. Однако 1787 случаев ошибочного прогнозирования non-default (потенциальные не плательщики) являются для банка риском образования убытков. Недостаточное качество модели подтверждается низким значением метрики Recall.

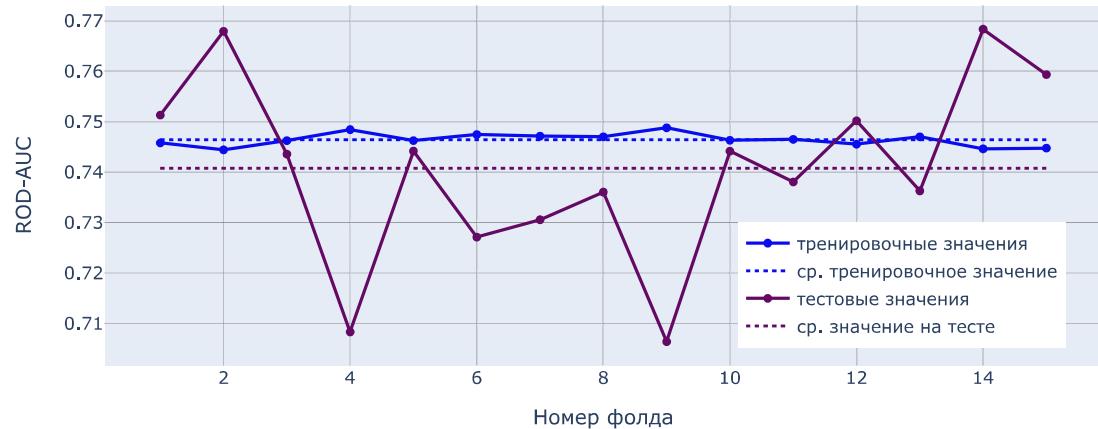
4.2.2. Кроссвалидация

In [24]:

```
mf.view_cross_validation(model, X_test, y_test, 15)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.  
[Parallel(n_jobs=1)]: Done 15 out of 15 | elapsed: 10.5s finished
```

Кросс-валидация по метрике ROC-AUC



Время выполнения: 0:00:10.493911

Результаты кросс-валидации позволяют убедиться в неоднородности данных. Поскольку перед кросс-валидацией не производилось перемешивание, можно условно связать ось X с осью времени. Таким образом, снижение предсказывающей способности модели в средней части графика, указывает на возможное ослабление требований банка к потенциальным заемщикам.

5. Повышение качества модели

5.1. Сбалансированность выборки

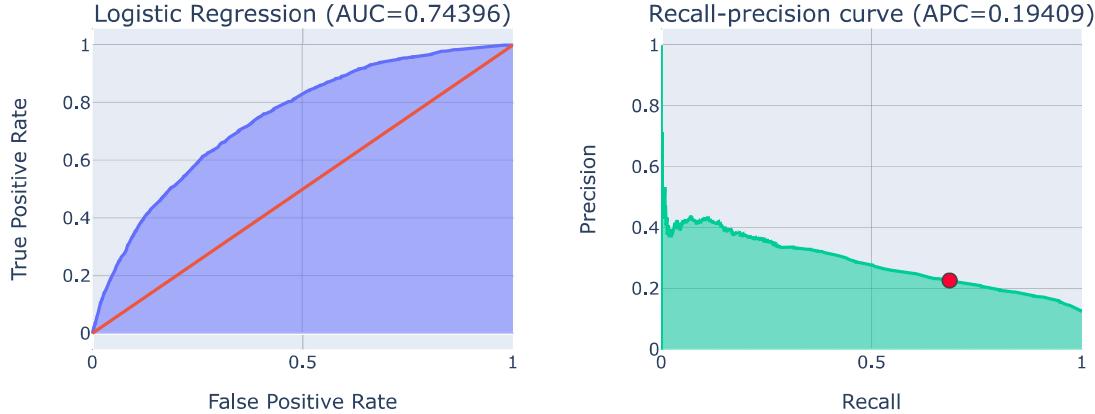
Выборка сильно разбалансирована. Для устранения негативного влияния этого факта на результат обучения модели изменен параметр `class_weight`

Проведем обучение оценим результаты

```
In [25]:
model = LogisticRegression(class_weight = 'balanced', max_iter=500, random_state=RANDOM_SEED)
X_train, X_test, y_train, y_test, model, y_pred_prob, y_pred = mf.traning_model(data_train, data_test, model, RANDOM_SEED)

mf.view_metrics_LogisticRegression(model, X_train, y_train, X_test, y_test, ['Default', 'Non-default'])
```

Время выполнения: 0:00:03.474934



Check metrics

	Значение	Описание
Positive, P	1827	Истинный Default
Negative, N	12933	Истинный Non-default
True Positive, TP	1253	Корректная идентификация Default
True Negative, TN	8649	Корректная идентификация Non-default
False Positive, FP	4284	Ошибочная идентификация Default
False Negative, FN	574	Ошибочная идентификация Non-default
Accuracy	0.67087	Точность: Accuracy=(TP+TN)/(P+N)
Precision	0.2263	Precision = TP/(TP+FP)
Recall	0.68582	Recall = TP/P
F1-score	0.3403	Метрика, объединяющая Precision и Recall
ROC_AUC	0.74396	ROC AUC Score

Учет разбалансированности ключевой переменной при обучении модели, позволил существенно увеличить долю верно определенных Default. При этом возросло и количество неверно определенных Non-default.

5.2. Регуляризация

Произведем поиск наилучших гиперпараметров. Применим их в обучении новой модели.

In [26]:

```
reg_model = LogisticRegression(random_state=RANDOM_SEED)
mf.revealing_best_parameters(reg_model, X_train, y_train, 50, 1e-3)
```

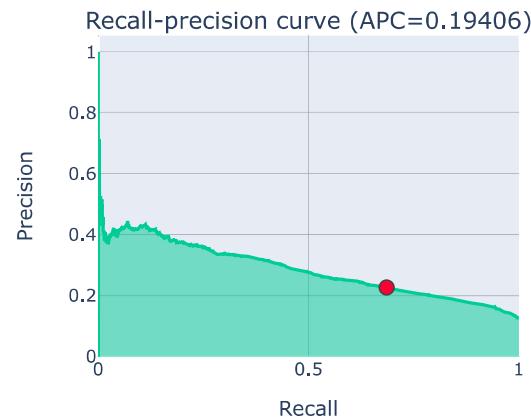
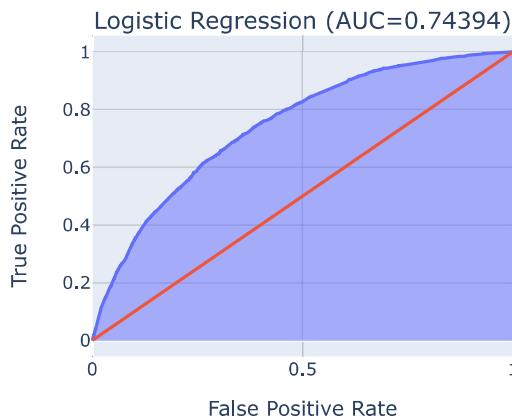
Время выполнения: 0:01:24.273897

Набор параметров для копирования:

```
C = 1.0
, class_weight = 'balanced'
, dual = False
, fit_intercept = True
, intercept_scaling = 1
, l1_ratio = None
, max_iter = 50
, multi_class = 'auto'
, n_jobs = None
, penalty = 'none'
, random_state = 42
, solver = 'newton-cg'
, tol = 0.001
, verbose = 0
, warm_start = False
```

```
In [27]:  
best_model = LogisticRegression(C = 1.0  
                                ,class_weight = 'balanced'  
                                ,dual = False  
                                ,fit_intercept = True  
                                ,intercept_scaling = 1  
                                ,l1_ratio = None  
                                ,max_iter = 50  
                                ,multi_class = 'auto'  
                                ,n_jobs = None  
                                ,penalty = 'none'  
                                ,random_state = 42  
                                ,solver = 'newton-cg'  
                                ,tol = 0.001  
                                ,verbose = 0  
                                ,warm_start = False)  
  
X_train, X_test, y_train, y_test, model, y_pred_prob, y_pred = mf.traning_model(data_train  
                                ,data_test  
                                ,best_model  
                                ,RANDOM_SEED)  
  
mf.view_metrics_LogisticRegression(best_model, X_train, y_train, X_test, y_test, ['Default', 'Non-default'])
```

Время выполнения: 0:00:01.037931



Check metrics

	Значение	Описание
Positive, P	1827	Истинный Default
Negative, N	12933	Истинный Non-default
True Positive, TP	1253	Корректная идентификация Default
True Negative, TN	8648	Корректная идентификация Non-default
False Positive, FP	4285	Ошибочная идентификация Default
False Negative, FN	574	Ошибочная идентификация Non-default
Accuracy	0.6708	Точность: Accuracy=(TP+TN)/(P+N)
Precision	0.22625	Precision = TP/(TP+FP)
Recall	0.68582	Recall = TP/P
F1-score	0.34026	Метрика, объединяющая Precision и Recall
ROC_AUC	0.74394	ROC AUC Score

Вопреки ожиданиям, регуляризация параметров не принесла заметного улучшения результатов. Наилучший отклик был получен ранее, в результате изменения параметра **class_weight**, отвечающего за настройку модели для обучения на несбалансированных выборках.

6. Резюме

В результате выполнения учебного проекта, получен вполне жизнеспособный продукт с приемлемыми параметрами качества предсказывания. В ходе работы в результате поиска лучших параметров обучения удалось существенно поднять значение метрики Recall. При этом показатель ROC_AUC, по которому оценивается работа для рейтинга kaggle незначительно снизился.

Для дальнейшего развития запланирована реализация инструмента оценки и отсева признаков, имеющих нулевое, или крайне низкое влияние на целевую переменную. В качестве тренировки, в модели заполнения пропусков заначений признака **education** можно учесть неравномерность распределения уровня образования в популяции. Однако ожидать от такой манипуляции существенного роста качества целевой модели не приходится, поскольку доля пропусков в общем объеме данных очень незначительна. Повышение качества заполнения пропусков, предположительно, не может иметь заметного влияния на качество целевой модели.

Дополнительно к оттачиванию навыков построения качественных моделей, в результате работы над проектом был расширен опыт участия в соревнованиях на kaggle. Комплект универсальных функций, предназначенных для исследования и преобразования данных был переработан с учетом накопленного опыта и существенно дополнен. Функции, обеспечивающие визуализацию были стандартизованы для обеспечения единого формата представления информации и дополнены инструментами вывода метрик качества моделей.

7. Submission

```
In [28]:
```

```
data_train = df_prepoc.query('sample == 1').drop(['sample'], axis=1)
data_test = df_prepoc.query('sample == 0').drop(['sample'], axis=1)

X_train = data_train.drop(['default'], axis=1)
y_train = data_train['default'].values
X_test = data_test.drop(['default'], axis=1)

best_model.fit(X_train, y_train)

y_pred_prob = best_model.predict_proba(X_test)[:,1]

submit = pd.DataFrame(data.query('sample == 0')['client_id'])
submit['default'] = y_pred_prob
submit.to_csv('submission.csv', index=False)
```

```
In [29]:
```

```
submit
```

	client_id	default
0	74835	0.201927
1	17527	0.793901
2	75683	0.306455
3	26883	0.408060
4	28862	0.136543
...
36344	83775	0.770901
36345	106254	0.110605
36346	81852	0.785195
36347	1971	0.628180
36348	69044	0.702282

36349 rows × 2 columns