ХАКАТОН

Fintech 2.0
задача от МТС банка

Команда REBOOT
участники: Ольга Дейкина

Задача:

построить модель, предсказывающую дефолт по займу

Цель:

внедрить результат модели в виде скоров в алгоритм NBO (Next Best Offer)

КОНЦЕПЦИЯ РЕШЕНИЯ:

1. Разведочный анализ данных (EDA).
2. Разработка и выбор Baseline на необработанных данных с применением алгоритмов машинного обучения на базе sklearn.
3. Первичная обработка данных: выбросы, Feature Selection.
4. Преобразование признаков, генерация новых признаков.
5. Построение моделей с разбивкой по срокам кредитования.
6. Новая разметка данных.
7. Выбор итоговой модели.
8. Оценка качества модели на тестовой выборке.
9. Расчет эффекта от внедрения модели.

Выбранный технический стек:

pandas, numpy, sklearn, catboost, matplotlib, seaborn.

1. EDA

В исходных данных содержится размеченный набор данных для бинарной классификации. Размер датасета: 1723 строк, 14 столбцов. Без пропусков

Целевые данные не сбалансированы:

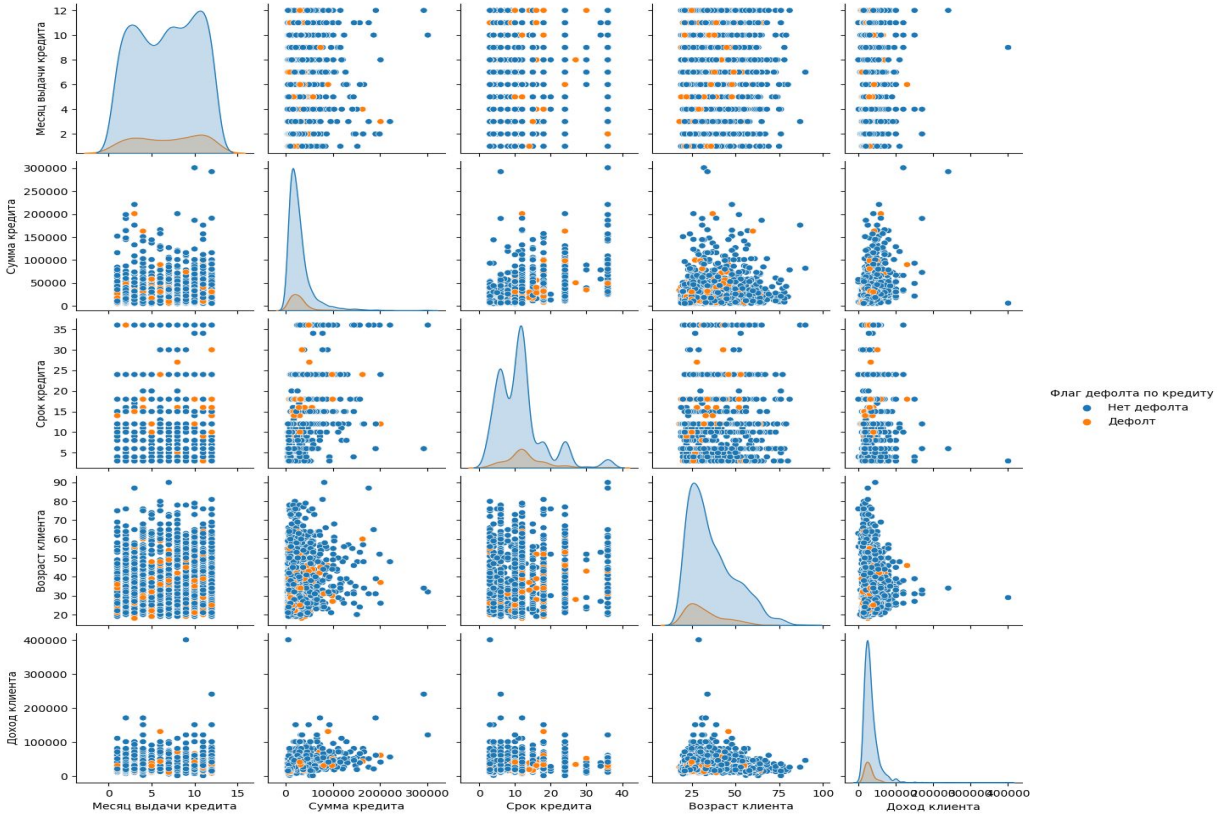
```
Нет дефолта    1527
Дефолт          196
```

Некоторые признаки можно преобразовать (сумма, срок, возраст, доход):

```
df.nunique() # смотрим количество уникальных значений

Месяц выдачи кредита          12
Сумма кредита                205
Срок кредита                  22
Возраст клиента               66
Пол клиента                    2
Образование клиента            6
Тип товара                    22
Наличие детей у клиента        2
Регион выдачи кредита          3
Доход клиента                 76
Семейное положение             3
Оператор связи                 5
Является ли клиентом банка     2
Флаг дефолта по кредиту        2
dtype: int64
```

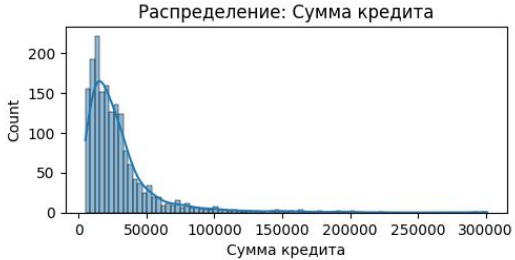
1. EDA

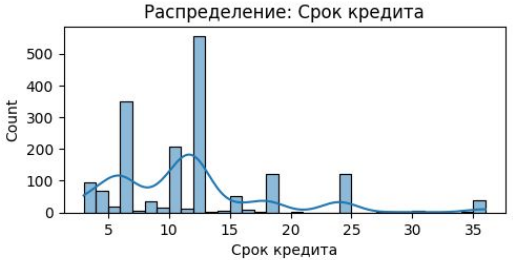
Нет признаков, которые могут самостоятельно отделить два класса.

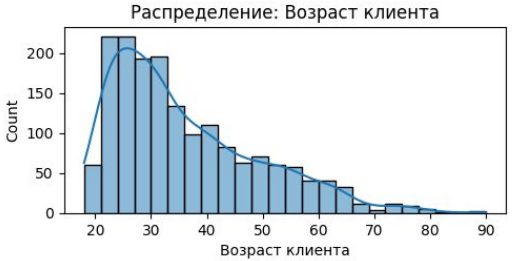


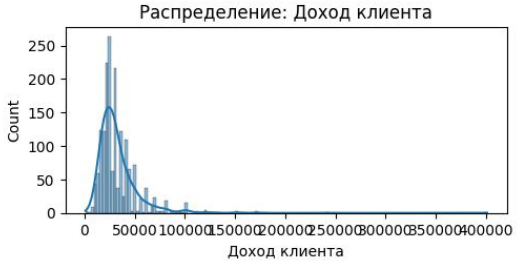
1. EDA

Нет идеального нормального распределения.
У некоторых признаков видны выбросы.









2. Baseline

Категориальные данные преобразованы с использованием LabelEncoder. Исходный датасет разделен на выборки методом train_test_split (в соотношении 60%, 20%, 20%) с сохранением пропорции целевого класса:

```
Размер исходной выборки: 1723
Размер обучающей выборки: 1033
Размер валидационной выборки: 345
Размер тестовой выборки: 345
```

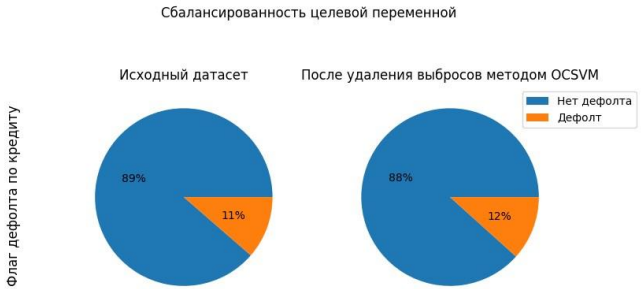
В качестве вероятной базовой модели рассматривались следующие модели:

```
classifiers = [
    LogisticRegression(random_state=42, class_weight='balanced'),
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1, max_iter=1000),
    GaussianNB(),
```

В качестве базовой модели выбрана модель с классификатором LogisticRegression.
Выбор сделан, исходя из макро-оценки F1.

3. Первичная обработка данных

Удаление выбросов методом OCSVM



	losses_OCSVM, %
Нет дефолта	7
Дефолт	5
Всего	7

Feature Selection: отбор признаков с помощью логистической регрессии, метрика f1 улучшилась на 5.5 %

Оптимизация гиперпараметров модели: не принесло положительных результатов

	f1 macro avg
Baseline	0.48
Baseline + выбросы_OCSVM	0.54
Baseline + выбросы_OCSVM + Feature Selection	0.59
Baseline + OCSVM + Feature Selection + оптимизация	0.47

4. Преобразование признаков

Признак “**Возраст клиента**” преобразован по четырем категориям:

```
df_1['Age_bin'].value_counts()
```

```
26-45     841
0-25      418
46-55     193
56-100    151
Name: Age_bin, dtype: int64
```

	Age_bin	Возраст клиента
0	0-25	23.0
1	26-45	32.0
2	46-55	50.0
3	56-100	62.0

средний возраст в категории

Признак “**Доход клиента**” преобразован по четырем категориям:

```
df_1['Income_bin'].value_counts()
```

```
20100-50000     1110
10100-20000      315
50100-100000     165
5000-10000        13
Name: Income_bin, dtype: int64
```

	Income_bin	Доход клиента
0	5000-10000	9000.0
1	10100-20000	16000.0
2	20100-50000	31000.0
3	50100-100000	56000.0

средний доход в категории

4. Генерация новых признаков

Признак **“Показатель долговой нагрузки”** получен из признаков “Сумма кредита”, “Срок кредита” и “Доход клиента” с учётом коэффициента минимальных расходов:

```
df_1['PND_bins'].value_counts()
```

```
0-33      1540
34-50       46
51-80       14
81-200       3
Name: PND_bins, dtype: int64
```

	PND_bins	Показатель долговой нагрузки
0	0-33	11.00
1	34-50	39.80
2	51-80	60.15
3	81-200	94.60

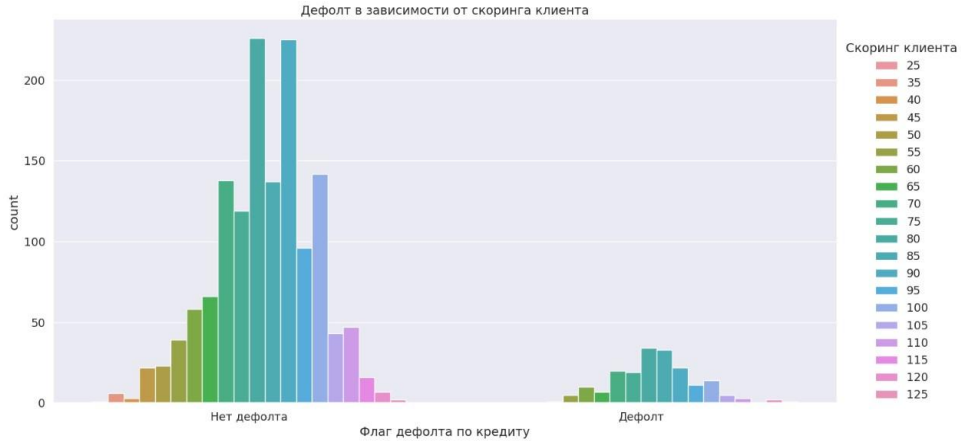
средний в категории

Признак **“Скоринг клиента”** получен по бальной методике из признаков “Возраст”, “Пол”, “Образование”, “Регион”, “Доход”, “Семейное положение”, “Дети”:

```
Уникальные значения по скорингу клиента
[ 25  35  40  45  50  55  60  65  70  75  80  85  90  95 100 105 110 115
 120 125]
----------------------------
Распределение уникальных значений по скорингу клиента
80     260
90     247
85     170
70     158
100    156
75     138
95     107
65      73
60      68
110     50
105     48
55      44
50      24
45      22
115     16
120      9
35       6
40       3
125      3
25       1
Name: Скоринг клиента, dtype: int64
```

4. Генерация новых признаков

Новые признаки также не смогли отделить два класса друг от друга:



4. Генерация новых признаков

Проверка метрики:

	f1 macro avg
Baseline	0.48
Baseline + выбросы_OCSVM	0.54
Baseline + выбросы_OCSVM + Feature Selection	0.59
Преобразование признаков	0.51
Преобразование признаков + Feature Selection	0.58
Преобразование признаков + Feature Selection + оптимизация	0.49

5. Модели по срокам кредитования

Срок кредитования - до 12 месяцев

```
print(df_short.shape)
print('---------------------------')
print(df_short['Флаг дефолта по кредиту'].value_counts())
```

```
(1292, 20)
---------------------------
Нет дефолта    1157
Дефолт          135
Name: Флаг дефолта по кредиту, dtype: int64
```

Срок кредитования - более 12 месяцев

```
print(df_long.shape)
print('---------------------------')
print(df_long['Флаг дефолта по кредиту'].value_counts())
```

```
(311, 20)
---------------------------
Нет дефолта    259
Дефолт          52
Name: Флаг дефолта по кредиту, dtype: int64
```

Не принесло положительных результатов:

	f1 macro avg
Baseline	0.48
Baseline + выбросы_OCSVM	0.54
Baseline + выбросы_OCSVM + Feature Selection	0.59
Преобразование признаков	0.51
Преобразование признаков + Feature Selection	0.58
Преобразование признаков + Feature Selection + оптимизация	0.49
Преобразование признаков + короткий срок	0.46
Преобразование признаков + длинный срок	0.40

6. Новая разметка данных

Предполагаю, что проблема низкого качества классификации состоит в том, что в исходном датасете произошло слияние двух баз -завершенные и текущие кредиты. В результате клиенты, подпадающие под дефолтные критерии имеют метку "нет дефолта".

Новая разметка данных проведена путем кластеризации с использованием модели:

```
model = KMeans(n_clusters=2,
               init='k-means++',
               n_init='auto',
               random_state=42)
```

В результате получен датасет со следующим балансом категорий:

```
df_my_concat['Флаг_new'].value_counts()
```

```
Нет дефолта    1195
Дефолт          408
Name: Флаг_new, dtype: int64
```

7. Выбор итоговой модели

	f1 macro avg
Baseline	0.48
Baseline + выбросы_OCSVM	0.54
Baseline + выбросы_OCSVM + Feature Selection	0.59
Преобразование признаков	0.51
Преобразование признаков + Feature Selection	0.58
Преобразование признаков + Feature Selection + оптимизация	0.49
Преобразование признаков + короткий срок	0.46
Преобразование признаков + длинный срок	0.40
Переразметка на преобразованных признаках	0.75
Переразметка на преобразованных признаках + оптимизация	0.81

Размер обновленного датасета:
1603 строк, 6 столбцов.
Данные не сбалансированы.

Параметры итоговой модели:

```
LogisticRegression(random_state=42, class_weight=None, max_iter = 100,
                   multi_class = 'auto', penalty = None, solver = 'newton-cg')
```

8. Оценка качества модели на тестовой выборке

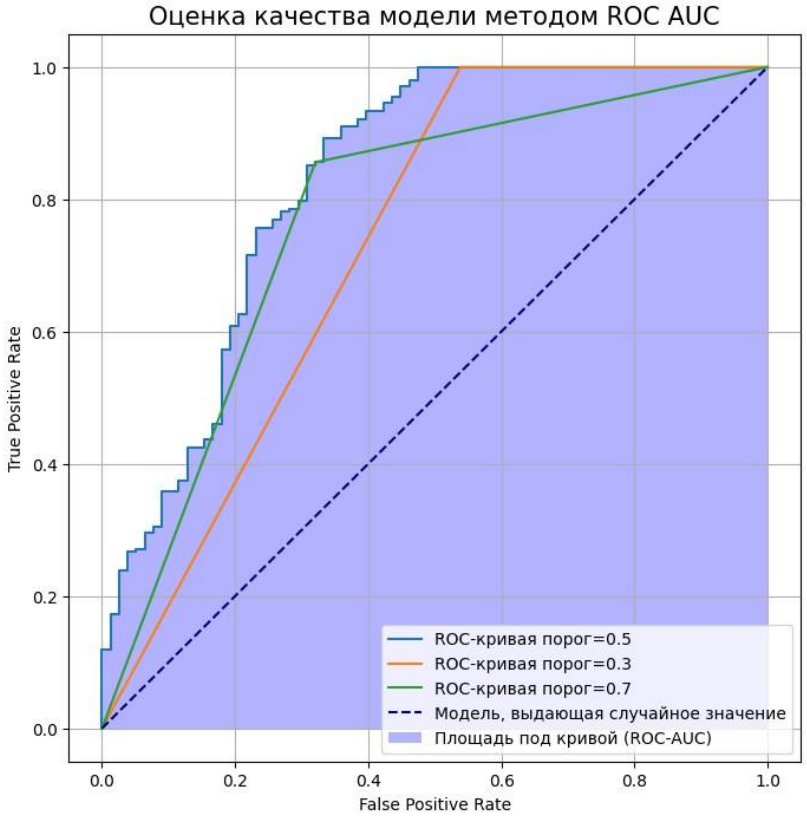
```
print(classification_report(y_test, pred_model_grid, target_names=['Дефолт', 'Нет дефолта']))
```

```
              precision    recall  f1-score   support

      Дефолт       0.80      0.55      0.65        78
 Нет дефолта       0.87      0.95      0.91       243

    accuracy                           0.86       321
   macro avg       0.83      0.75      0.78       321
weighted avg       0.85      0.86      0.85       321
```

```
f1_model_grid=f1_score(y_test, pred_model_grid, average='macro')
f1_model_grid
```

```
0.7806595365418895
```

8. Оценка качества модели на тестовой выборке



9. ЭФФЕКТ ОТ ВНЕДРЕНИЯ МОДЕЛИ

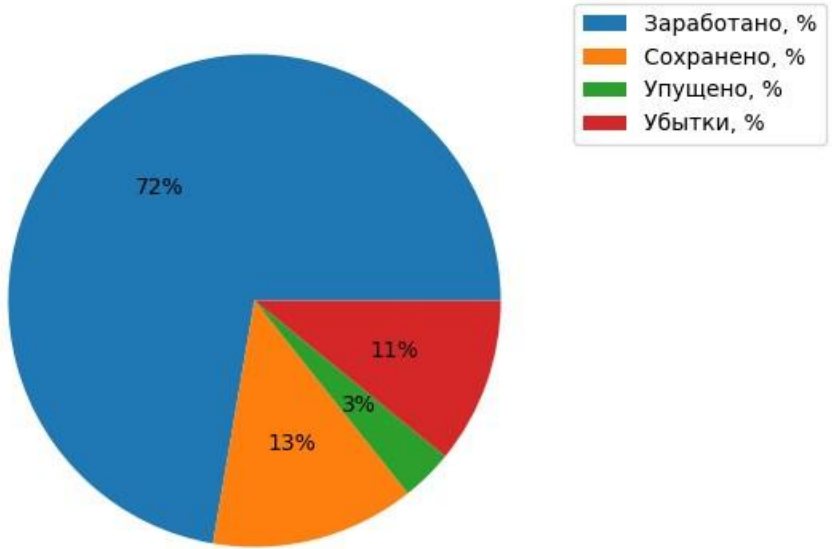
9. ЭФФЕКТ ОТ ВНЕДРЕНИЯ МОДЕЛИ

	Модель с отсечкой 0.5	Модель с отсечкой 0.3	Модель с отсечкой 0.7
Заработано, %	72,3	75,7	64,8
Сохранено, %	13,4	11,2	16,5
Упущено, %	3,4	0	10,9
Убытки, %	10,9	13,1	7,8

Сравним нашу модель по предсказанию с тремя вариантами порога классификатора:

- Если банку хочется **больше заработать**, то лучше выдавать кредиты *всем* людям, которые способны его вернуть, то есть следует понизить порог модели.
- Если банку хочется **меньше потерять**, то лучше выдавать кредиты *только* надежным людям, то есть следует повысить порог модели.

9. ЭФФЕКТ ОТ ВНЕДРЕНИЯ МОДЕЛИ



Итоговая модель со средним порогом классификатора