Курсовой проект по курсу Библиотеки Python для Data Science 2.

Соковнин Игорь Леонидович

Задача

Требуется, на основании имеющихся данных о клиентах банка, построить модель, используяобучающий датасет, для прогнозирования невыполнения долговых обязательств по текущемукредиту. Выполнить прогноз для примеров из тестового датасета.

Целевая переменная

Credit Default - факт невыполнения кредитных обязательств

Метрика качества

F1-score (sklearn.metrics.f1_score)

Подключение библиотек и скриптов

B [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 numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

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

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

B [2]:
```
import datetime # Для работы со временем
import gc  # Работа со сборщик мусора
import pickle  # Сохранение модели

# 1. Основные библиотеки
import numpy as np # linear algebra
import pandas as pd # Data processing, CSV file I/O (e.g. pd.read_csv)

# 2. Визуализация
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

# 3. Разделение датасета
from sklearn.model_selection import train_test_split, KFold, GridSearchCV, cross_val_score

# 4. Модели
import catboost as catb

# 5. Метрики качества
from sklearn.preprocessing import StandardScaler, MinMaxScaler
#from sklearn.model_selection import train_test_split, ShuffleSplit, cross_val_score, learning_curve
#from sklearn.model_selection import StratifiedKFold, GridSearchCV, RandomizedSearchCV
from sklearn.metrics import classification_report, f1_score#, precision_score, recall_score

# 6. Для визуализации внешних картинок в ноутбуке
from IPython.display import Image

# Magic commands
%matplotlib inline

# Вывод графики в ноутбук
%config InlineBackend.figure_format = 'svg'  # Более четкое отображение, формат файла фигуры svg
```

B [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

B [4]:
```
matplotlib.rcParams.update({'font.size':14})
```

Пути к директориям и файлам

B [5]:
```
# для kaggle
# input
TRAIN_DATASET_PATH = '/kaggle/input/654pds2courseproject/course_project_train.csv'
TEST_DATASET_PATH = '/kaggle/input/654pds2courseproject/course_project_test.csv'
# output
SUBMIT_DATASET_PATH = '/kaggle/input/realestatepriceprediction/ILSokovnin_predictions.csv'

# локально
# input
TRAIN_DATASET_PATH = './course_project/course_project_train.csv'
TEST_DATASET_PATH = './course_project/course_project_test.csv'

# output
PREP_DATASET_PATH = './training_project/training_project_data_prep.csv'
```

Шаг 1. Загрузка данных

Описание датасета

1. Home Ownership - домовладение
2. Annual Income - годовой доход
3. Years in current job - количество лет на текущем месте работы
4. Tax Liens - налоговые обременения
5. Number of Open Accounts - количество открытых счетов
6. Years of Credit History - количество лет кредитной истории
7. Maximum Open Credit - наибольший открытый кредит
8. Number of Credit Problems - количество проблем с кредитом
9. Months since last delinquent - количество месяцев с последней просрочки платежа
10. Bankruptcies - банкротства
11. Purpose - цель кредита
12. Term - срок кредита
13. Current Loan Amount - текущая сумма кредита
14. Current Credit Balance - текущий кредитный баланс
15. Monthly Debt - ежемесячный долг
16. Credit Score - Кредитный рейтинг
17. Credit Default - факт невыполнения кредитных обязательств (0 - погашен вовремя, 1 -просрочка)

B [6]:
```
# Тренировочные данные
df_train = pd.read_csv(TRAIN_DATASET_PATH)
df_train.head()
```

Out[6]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Months since last delinquent	Bankruptcies	Purpose	Term	Current Loan Amount	Current Credit Balance	Monthly Debt	Credit Score	Credit Default
0	Own Home	482087.0	NaN	0.0	11.0	26.3	685960.0	1.0	NaN	1.0	debt consolidation	Short Term	99999999.0	47386.0	7914.0	749.0	0
1	Own Home	1025487.0	10+ years	0.0	15.0	15.3	1181730.0	0.0	NaN	0.0	debt consolidation	Long Term	264968.0	394972.0	18373.0	737.0	1
2	Home Mortgage	751412.0	8 years	0.0	11.0	35.0	1182434.0	0.0	NaN	0.0	debt consolidation	Short Term	99999999.0	308389.0	13651.0	742.0	0
3	Own Home	805068.0	6 years	0.0	8.0	22.5	147400.0	1.0	NaN	1.0	debt consolidation	Short Term	121396.0	95855.0	11338.0	694.0	0
4	Rent	776264.0	8 years	0.0	13.0	13.6	385836.0	1.0	NaN	0.0	debt consolidation	Short Term	125840.0	93309.0	7180.0	719.0	0

B [7]:
```
# Тестовые данные
df_test = pd.read_csv(TEST_DATASET_PATH)
df_test.head()
```

Out[7]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Months since last delinquent	Bankruptcies	Purpose	Term	Current Loan Amount	Current Credit Balance	Monthly Debt	Credit Score
0	Rent	NaN	4 years	0.0	9.0	12.5	220968.0	0.0	70.0	0.0	debt consolidation	Short Term	162470.0	105906.0	6813.0	NaN
1	Rent	231838.0	1 year	0.0	6.0	32.7	55946.0	0.0	8.0	0.0	educational expenses	Short Term	78298.0	46037.0	2318.0	699.0
2	Home Mortgage	1152540.0	3 years	0.0	10.0	13.7	204600.0	0.0	NaN	0.0	debt consolidation	Short Term	200178.0	146490.0	18729.0	7260.0
3	Home Mortgage	1220313.0	10+ years	0.0	16.0	17.0	456302.0	0.0	70.0	0.0	debt consolidation	Short Term	217382.0	213199.0	27559.0	739.0
4	Home Mortgage	2340952.0	6 years	0.0	11.0	23.6	1207272.0	0.0	NaN	0.0	debt consolidation	Long Term	777634.0	425391.0	42605.0	706.0

B [8]:
```
print('Строк в train:',df_train.shape[0])  # gives number of row count
print('Столбцов в train:',df_train.shape[1])  # gives number of col count
print('\nСтрок test:',df_test.shape[0])
print('Столбцов в test:',df_test.shape[1])
```

```
Строк в train: 7500
Столбцов в train: 17

Строк test: 2500
Столбцов в test: 16
```

B [9]:
```
df_train.iloc[0]  # Получаем первую строку (index=0)
```

Out[9]:
```
Home Ownership                 Own Home
Annual Income                    482087
Years in current job                NaN
Tax Liens                             0
Number of Open Accounts              11
Years of Credit History            26.3
Maximum Open Credit              685960
Number of Credit Problems             1
Months since last delinquent        NaN
Bankruptcies                          1
Purpose               debt consolidation
Term                          Short Term
Current Loan Amount              1e+08
Current Credit Balance           47386
Monthly Debt                      7914
Credit Score                       749
Credit Default                       0
Name: 0, dtype: object
```

2. EDA и очистка данных

Делаем EDA для:

- Исправления выбросов
- Заполнения NaN
- Идей для генерации новых фич

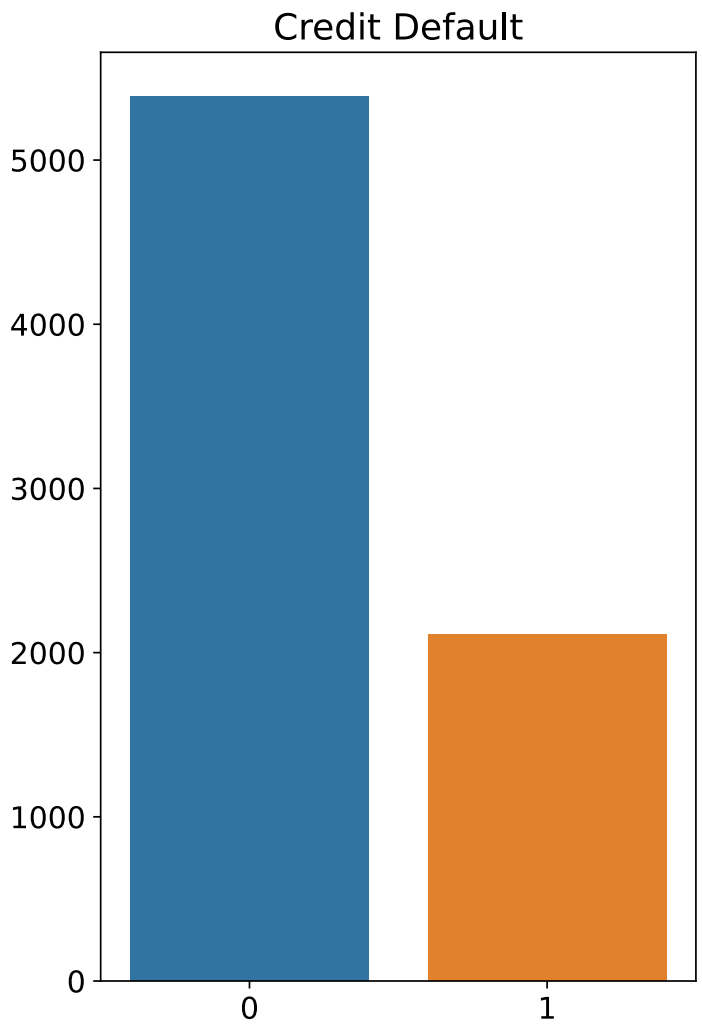
1. Обзор данных (Обзор обучающего датасета)

Обзор целевой переменной

B [10]:
```
df_train['Credit Default'].value_counts()  # Количество различных значений признака 'Credit Default'
```

Out[10]:
```
0    5387
1    2113
Name: Credit Default, dtype: int64
```

B [11]:
```
counts = df_train['Credit Default'].value_counts()

plt.figure(figsize=(5,8))
plt.title('Credit Default')
sns.barplot(counts.index, counts.values)

plt.show()
```



Приведение типов

B [12]:
```
for colname in ['Tax Liens', 'Number of Credit Problems', 'Bankruptcies']:
    df_train[colname] = df_train[colname].astype(str)
```

B [13]:
```
for colname in ['Tax Liens', 'Number of Credit Problems', 'Bankruptcies']:
    df_test[colname] = df_test[colname].astype(str)
```

B [14]:
```
df_train.dtypes
```

Out[14]:
```
Home Ownership                 object
Annual Income                 float64
Years in current job           object
Tax Liens                      object
Number of Open Accounts       float64
Years of Credit History       float64
Maximum Open Credit           float64
Number of Credit Problems      object
Months since last delinquent  float64
Bankruptcies                   object
Purpose                        object
Term                           object
Current Loan Amount           float64
Current Credit Balance        float64
Monthly Debt                  float64
Credit Score                  float64
Credit Default                  int64
dtype: object
```

Обзор количественных признаков

B [15]:
```
df_train.describe().T  # Анализ количественные признаки
```

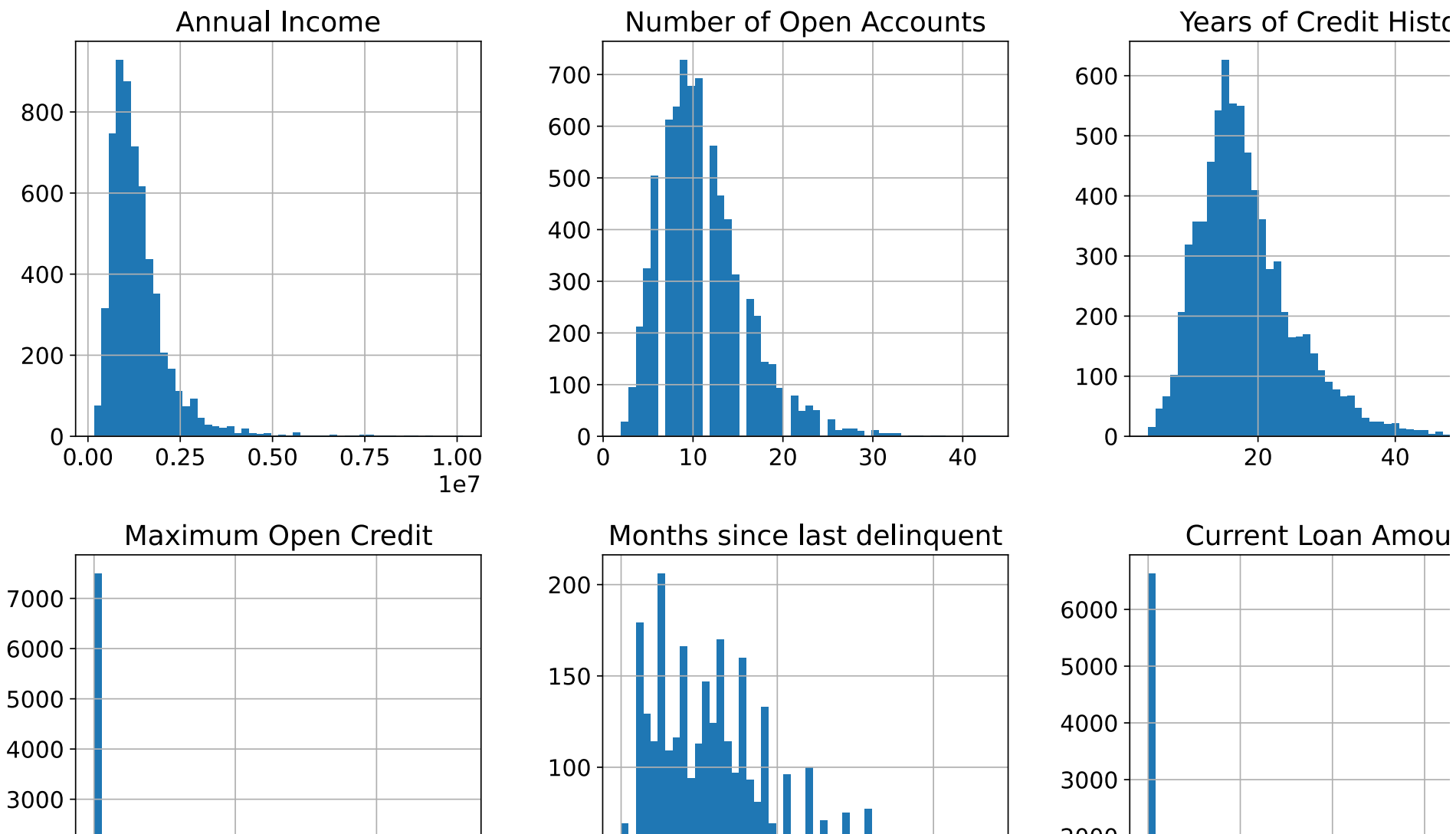
Out[15]:

	count	mean	std	min	25%	50%	75%	max
Annual Income	5943.0	1.366392e+06	8.453392e+05	164597.0	844341.0	1168386.0	1640137.00	1.014934e+07
Number of Open Accounts	7500.0	1.113093e+01	4.908924e+00	2.0	8.0	10.0	14.00	4.300000e+01
Years of Credit History	7500.0	1.831747e+01	7.041946e+00	4.0	13.5	17.0	21.80	5.770000e+01
Maximum Open Credit	7500.0	9.451537e+05	1.602622e+07	0.0	279229.5	478159.0	793501.50	1.304726e+09
Months since last delinquent	3419.0	3.469260e+01	2.168881e+01	0.0	16.0	32.0	50.00	1.180000e+02
Current Loan Amount	7500.0	1.187318e+07	3.192612e+07	11242.0	180169.0	309573.0	519882.00	1.000000e+08
Current Credit Balance	7500.0	2.898332e+05	3.178714e+05	0.0	114256.5	209323.0	360406.25	6.506797e+06
Monthly Debt	7500.0	1.831445e+04	1.192676e+04	0.0	10067.5	16076.5	23818.00	1.366790e+05
Credit Score	5943.0	1.151087e+03	1.604451e+03	585.0	711.0	731.0	743.00	7.510000e+03
Credit Default	7500.0	2.817333e-01	4.498740e-01	0.0	0.0	0.0	1.00	1.000000e+00

Выбросы

B [16]:
```
df_num_features = df_train.select_dtypes(include=['float32', 'float64', 'int8', 'int16', 'int32'])

df_num_features.hist(figsize=(16, 16), bins=50, grid=True)
```

Out[16]:
```
array([[<AxesSubplot:title={'center':'Annual Income'}>,
        <AxesSubplot:title={'center':'Number of Open Accounts'}>,
        <AxesSubplot:title={'center':'Years of Credit History'}>],
       [<AxesSubplot:title={'center':'Maximum Open Credit'}>,
        <AxesSubplot:title={'center':'Months since last delinquent'}>,
        <AxesSubplot:title={'center':'Current Loan Amount'}>],
       [<AxesSubplot:title={'center':'Current Credit Balance'}>,
        <AxesSubplot:title={'center':'Monthly Debt'}>,
        <AxesSubplot:title={'center':'Credit Score'}>]], dtype=object)
```



Наблюдаются выбросы по следующим признакам: Current Loan Amount, Maximum Open Credit, Current Credit Balance.

Ряд признаков имеют аномально высокое значение, но вполне вероятное: . Их необходимо будет ограничить.

2. Annual Income - годовой доход (http:)

В [17]:
```
print(df_train['Annual Income'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Annual Income'].sort_values())
# Считаем выбросами Annual Income > 4 000 000 (91 значения)
```

```
2083825.0    1
785954.0     1
266000.0     1
1177411.0    1
1539152.0    1
            ..
969475.0     4
1043651.0    4
1338113.0    4
1058376.0    4
1161660.0    4
Name: Annual Income, Length: 5478, dtype: int64

4240    164597.0
4485    175845.0
3946    177251.0
3310    191577.0
1114    192223.0
          ...
7482         NaN
7492         NaN
7494         NaN
7498         NaN
7499         NaN
Name: Annual Income, Length: 7500, dtype: float64
```

5. Number of Open Accounts - количество открытых счетов

В [18]:
```
print(df_train['Number of Open Accounts'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Number of Open Accounts'].sort_values())
# Считаем выбросами Number of Open Accounts > 33 (9 значений)
```

```
42.0      1
43.0      1
38.0      1
41.0      1
35.0      1
37.0      2
34.0      2
31.0      6
33.0      6
32.0      6
29.0     10
30.0     11
26.0     12
27.0     14
28.0     14
2.0      28
25.0     32
22.0     49
24.0     50
23.0     59
21.0     78
20.0     93
3.0      95
19.0    139
18.0    143
4.0     212
17.0    232
16.0    265
15.0    313
5.0     325
14.0    420
13.0    465
6.0     504
12.0    562
7.0     613
8.0     638
10.0    677
11.0    692
9.0     728
Name: Number of Open Accounts, dtype: int64

3271     2.0
3768     2.0
2321     2.0
2325     2.0
1743     2.0
        ...
6868    37.0
3475    38.0
2840    41.0
5738    42.0
1769    43.0
Name: Number of Open Accounts, Length: 7500, dtype: float64
```

6. Years of Credit History - количество лет кредитной истории

В [19]:
```
print(df_train['Years of Credit History'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Years of Credit History'].sort_values())
# Считаем выбросами Years of Credit History > 40 (83 значения)
```

```
39.8      1
41.8      1
46.3      1
6.2       1
36.3      1
        ...
17.5     83
17.0     86
16.5     91
16.0     99
15.0    104
Name: Years of Credit History, Length: 408, dtype: int64

324      4.0
5497     4.3
3784     4.5
2560     4.5
6633     4.7
        ...
3628    51.3
4716    51.5
4301    51.9
247     52.2
476     57.7
Name: Years of Credit History, Length: 7500, dtype: float64
```

7. Maximum Open Credit - наибольший открытый кредит

В [20]:
```
print(df_train['Maximum Open Credit'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Maximum Open Credit'].sort_values())
# Считаем выбросами значения 'Maximum Open Credit' > 4 000 000 (64 значений) 'Maximum Open Credit' < 50 000 (125 значений)
```

```
804958.0      1
653488.0      1
368192.0      1
3007136.0     1
243166.0      1
             ..
323312.0      3
615714.0      3
349140.0      3
319110.0      5
0.0          65
Name: Maximum Open Credit, Length: 6963, dtype: int64

2297    0.000000e+00
319     0.000000e+00
611     0.000000e+00
1427    0.000000e+00
294     0.000000e+00
            ...
2763    4.092389e+07
2023    5.756256e+07
2617    2.655129e+08
44      3.800523e+08
617     1.304726e+09
Name: Maximum Open Credit, Length: 7500, dtype: float64
```

9. Months since last delinquent - количество месяцев с последней просрочки платежа

В [21]:
```
print(df_train['Months since last delinquent'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Months since last delinquent'].sort_values())
# Считаем выбросами Months since last delinquent > 83 (5 значений)
```

```
91.0      1
86.0      1
84.0      1
118.0     1
92.0      1
         ..
13.0     65
33.0     68
8.0      68
29.0     71
14.0     76
Name: Months since last delinquent, Length: 89, dtype: int64

5705    0.0
4995    0.0
4938    0.0
3063    0.0
257     0.0
       ...
7494    NaN
7495    NaN
7497    NaN
7498    NaN
7499    NaN
Name: Months since last delinquent, Length: 7500, dtype: float64
```

13. Current Loan Amount - текущая сумма кредита

В [22]:
```
print(df_train['Current Loan Amount'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Current Loan Amount'].sort_values())
# Выбросы Current Loan Amount = 99999999.0 (870 записей) ?
# Набор данных надо разбивать на два по сумме кредита: 1 - [0, ...,2*10^7], 2 - [85*10^7, ..., 1*10^8]
```

```
264616.0        1
186846.0        1
367334.0        1
290642.0        1
200640.0        1
              ...
270226.0        5
216106.0        5
218064.0        6
89298.0         6
99999999.0    870
Name: Current Loan Amount, Length: 5386, dtype: int64

1404       11242.0
4467       21472.0
2735       21472.0
7144       21516.0
5861       21560.0
           ...
4384    99999999.0
732     99999999.0
4374    99999999.0
4555    99999999.0
0       99999999.0
Name: Current Loan Amount, Length: 7500, dtype: float64
```

14. Current Credit Balance - текущий кредитный баланс

В [23]:
```
print(df_train['Current Credit Balance'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Current Credit Balance'].sort_values())
# Считаем выбросами значения 'Current Credit Balance' > 1300000 (106 значений)
```

```
250477.0     1
474601.0     1
134900.0     1
150366.0     1
153026.0     1
            ..
198911.0     4
136401.0     4
82289.0      4
191710.0     5
0.0         53
Name: Current Credit Balance, Length: 6592, dtype: int64

4405          0.0
4274          0.0
1802          0.0
1464          0.0
2276          0.0
           ...
7278    4209659.0
1580    4249673.0
4602    4367245.0
4745    4720132.0
4769    6506797.0
Name: Current Credit Balance, Length: 7500, dtype: float64
```

15. Monthly Debt - ежемесячный долг

В [24]:
```
print(df_train['Monthly Debt'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Monthly Debt'].sort_values())
# Считаем выбросами значения 'Monthly Debt' > 55 000 (98 значений)
```

```
22292.0    1
23287.0    1
14015.0    1
21381.0    1
8390.0     1
          ..
14848.0    3
11659.0    3
19667.0    4
19222.0    4
0.0        6
Name: Monthly Debt, Length: 6716, dtype: int64

780          0.0
1643         0.0
7124         0.0
4165         0.0
3219         0.0
           ...
6253     96177.0
6946    100091.0
2535    104036.0
1615    110311.0
4745    136679.0
Name: Monthly Debt, Length: 7500, dtype: float64
```

16. Credit Score - Кредитный рейтинг

В [25]:
```
print(df_train['Credit Score'].value_counts().sort_values(), '\n')  # по значению
print(df_train['Credit Score'].sort_values())
# Набор данных надо разбивать на два по Кредитному рейтингу: 1 - [585, ...,800], 2 - [6500, ..., 7500]
# Считаем выбросами значения 'Monthly Debt' < 585 и 'Monthly Debt' > 7510
```

```
7010.0      1
6150.0      1
604.0       1
629.0       1
6600.0      1
          ...
741.0     151
745.0     152
748.0     157
747.0     168
740.0     169
Name: Credit Score, Length: 268, dtype: int64

599     585.0
6114    586.0
1455    588.0
3475    589.0
3491    590.0
        ...
7482      NaN
7492      NaN
7494      NaN
7498      NaN
7499      NaN
Name: Credit Score, Length: 7500, dtype: float64
```

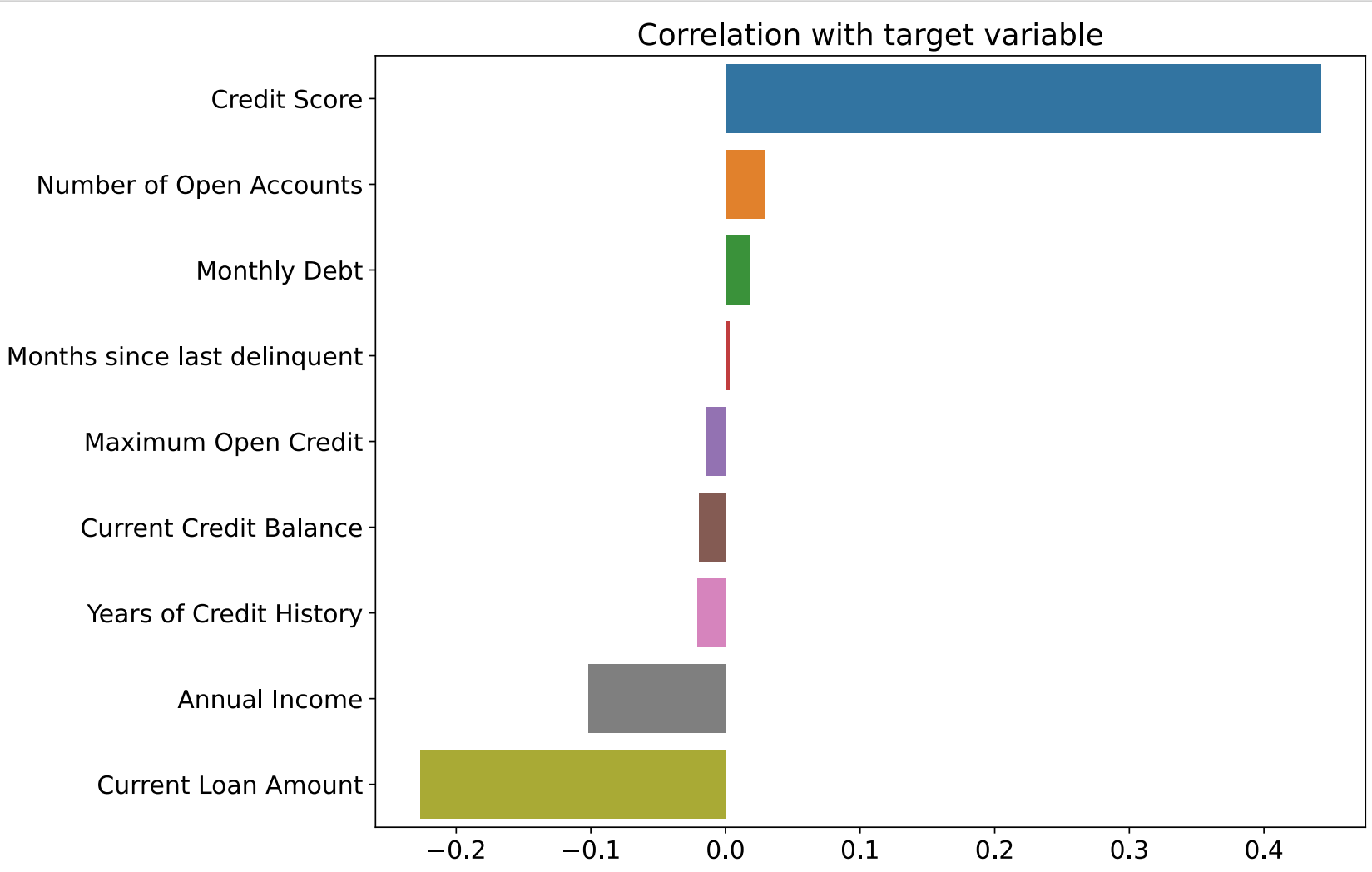
Анализ признакового пространства¶

Корреляция с базовыми признаками

В [26]:
```
TARGET_NAME = 'Credit Default'
BASE_FEATURE_NAMES = df_train.columns.drop(TARGET_NAME).tolist()
BASE_FEATURE_NAMES
```

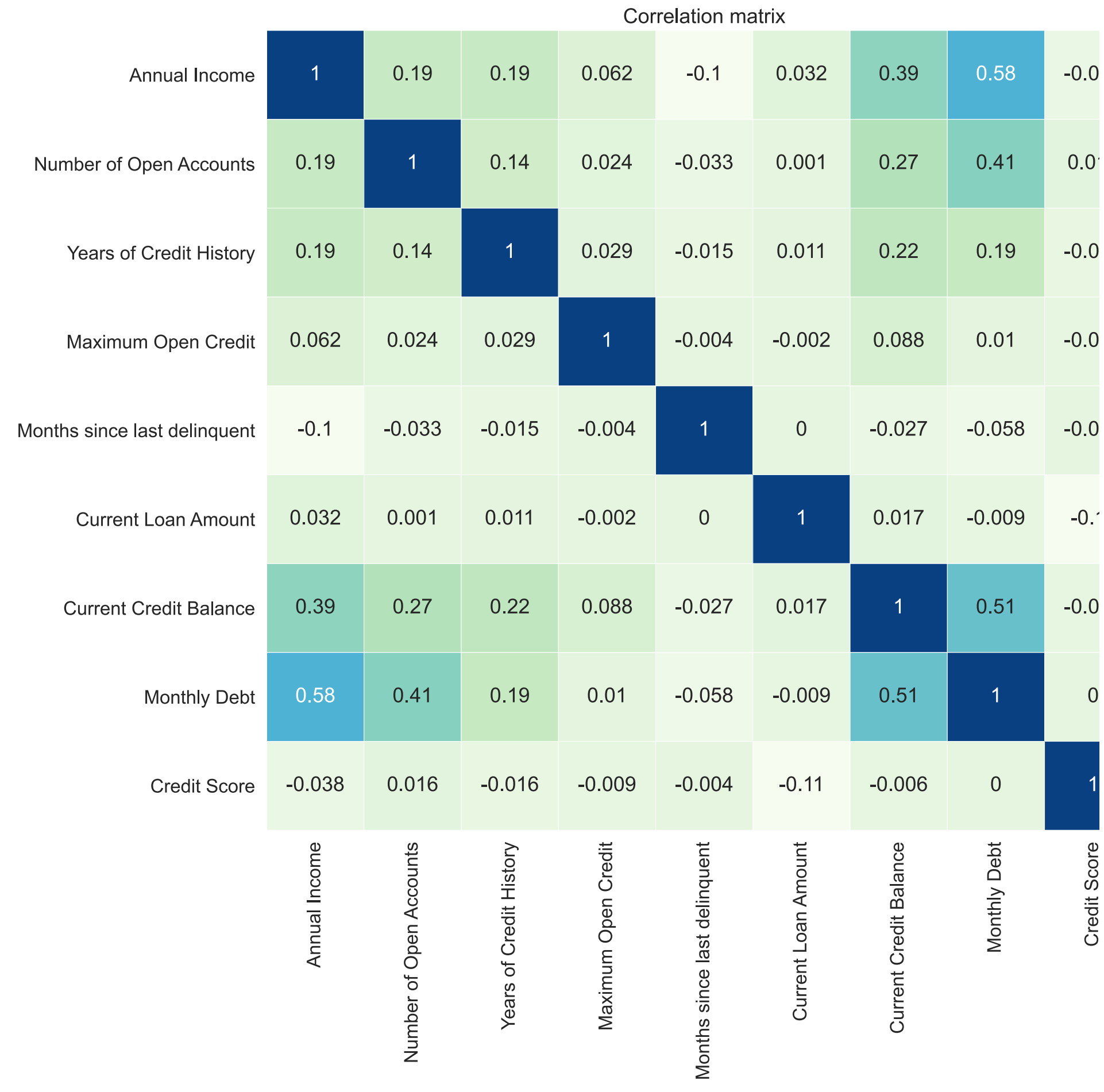
Out[26]:
```
['Home Ownership',
 'Annual Income',
 'Years in current job',
 'Tax Liens',
 'Number of Open Accounts',
 'Years of Credit History',
 'Maximum Open Credit',
 'Number of Credit Problems',
 'Months since last delinquent',
 'Bankruptcies',
 'Purpose',
 'Term',
 'Current Loan Amount',
 'Current Credit Balance',
 'Monthly Debt',
 'Credit Score']
```

В [27]:
```
corr_with_target = df_train[BASE_FEATURE_NAMES + [TARGET_NAME]].corr().iloc[:-1, -1].sort_values(ascending=False)

plt.figure(figsize=(10, 8))

sns.barplot(x=corr_with_target.values, y=corr_with_target.index)

plt.title('Correlation with target variable')
plt.show()
```



Матрица корреляций

B [28]:
```
plt.figure(figsize = (16,12))

sns.set(font_scale=1.4)
sns.heatmap(df_train[BASE_FEATURE_NAMES].corr().round(3), annot=True, linewidths=.5, cmap='GnBu')

plt.title('Correlation matrix')
plt.show()
```



1. Наблюдается сильная положительная корреляция **(0.78)** между полями **'Current Loan Amount'** и **'Maximum Open Credit'**. Поэтому исключим из рассмотрения поле **'Maximum Open Credit'**
2. Наблюдается средняя положительная корреляция **(0.39)** между полями **'Number of Open Accounts'** и **'Maximum Open Credit'**.
3. Наблюдается средняя положительная корреляция **(0.37)** между полями **'Annual Income'** и **'Current Credit Balance'**.
4. Корреляции между **'Credit Score'** и **'Current Loan Amount'** слабая, отрицательная (**-0.084**).

Обзор категориальных (номинативных, порядковых) признаков

B [29]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Home Ownership                7500 non-null   object 
 1   Annual Income                 5943 non-null   float64
 2   Years in current job          7129 non-null   object 
 3   Tax Liens                     7500 non-null   object 
 4   Number of Open Accounts       7500 non-null   float64
 5   Years of Credit History       7500 non-null   float64
 6   Maximum Open Credit           7500 non-null   float64
 7   Number of Credit Problems     7500 non-null   object 
 8   Months since last delinquent  3419 non-null   float64
 9   Bankruptcies                  7500 non-null   object 
 10  Purpose                       7500 non-null   object 
 11  Term                          7500 non-null   object 
 12  Current Loan Amount           7500 non-null   float64
 13  Current Credit Balance        7500 non-null   float64
 14  Monthly Debt                  7500 non-null   float64
 15  Credit Score                  5943 non-null   float64
 16  Credit Default                7500 non-null   int64  
dtypes: float64(9), int64(1), object(7)
memory usage: 996.2+ KB
```

B [30]:
```
df_train.select_dtypes(include='object').columns
```

Out[30]:
```
Index(['Home Ownership', 'Years in current job', 'Tax Liens',
       'Number of Credit Problems', 'Bankruptcies', 'Purpose', 'Term'],
      dtype='object')
```

Обзор значений категориальных признаков

B [31]:
```
for cat_colname in df_train.select_dtypes(include='object').columns:
    print(str(cat_colname) + '\n\n' + str(df_train[cat_colname].value_counts()) + '\n' + '*' * 100 + '\n')

# Bankruptcies имеет странное значение 'nan' (14 значений), нужно заменить на 0
```

```
Home Ownership

Home Mortgage    3637
Rent             3204
Own Home          647
Have Mortgage      12
Name: Home Ownership, dtype: int64
****************************************************************************************************

Years in current job

10+ years    2332
2 years       705
3 years       620
< 1 year      563
5 years       516
1 year        504
4 years       469
6 years       426
7 years       396
8 years       339
9 years       259
Name: Years in current job, dtype: int64
****************************************************************************************************

Tax Liens

0.0    7366
1.0      83
2.0      30
3.0      10
4.0       6
5.0       2
6.0       2
7.0       1
Name: Tax Liens, dtype: int64
****************************************************************************************************

Number of Credit Problems

0.0    6469
1.0     882
2.0      93
3.0      35
4.0       9
5.0       7
6.0       4
7.0       1
Name: Number of Credit Problems, dtype: int64
****************************************************************************************************

Bankruptcies

0.0    6660
1.0     786
2.0      31
nan      14
3.0       7
4.0       2
Name: Bankruptcies, dtype: int64
****************************************************************************************************

Purpose

debt consolidation      5944
other                    665
home improvements        412
business loan            129
buy a car                 96
medical bills             71
major purchase            40
take a trip               37
buy house                 34
small business            26
wedding                   15
moving                    11
educational expenses      10
vacation                   8
renewable energy           2
Name: Purpose, dtype: int64
****************************************************************************************************

Term

Short Term    5556
Long Term     1944
Name: Term, dtype: int64
****************************************************************************************************
```

3. Обработка пропусков¶

B [32]:
```
df_train.isnull()
```

Out[32]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Months since last delinquent	Bankruptcies	Purpose	Term	Current Loan Amount	Current Credit Balance	Monthly Debt	Credit Score	Credit Default
0	False	False	True	False	False	False	False	False	True	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7495	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
7496	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
7497	False	False	False	False	False	False	False	False	True	False	False	False	False	False	False	False	False
7498	False	True	True	False	False	False	False	False	True	False	False	False	False	False	False	True	False
7499	False	True	False	False	False	False	False	False	True	False	False	False	False	False	False	True	False

7500 rows × 17 columns

B [33]:
```
df_train.isna().sum()  # просматриваем пропуски
```

Out[33]:
```
Home Ownership                 0
Annual Income                1557
Years in current job          371
Tax Liens                       0
Number of Open Accounts         0
Years of Credit History         0
Maximum Open Credit             0
Number of Credit Problems       0
Months since last delinquent 4081
Bankruptcies                    0
Purpose                         0
Term                            0
Current Loan Amount             0
Current Credit Balance          0
Monthly Debt                    0
Credit Score                 1557
Credit Default                  0
dtype: int64
```

Нулевые значения имеются в столбцах "Annual Income", "Years in current job", "Months since last delinquent" и "Credit Score"

Years in current job - количество лет на текущем месте работы

B [34]:
```
# количество пропусков
df_train['Years in current job'].isnull().sum()
```

Out[34]:
```
371
```

B [35]:
```
cat_colname = 'Years in current job'
df_train[cat_colname] = df_train[cat_colname].replace(to_replace = np.nan, value = 'неизвестно')
```

B [36]:
```
print(str(cat_colname) + '\n\n' + str(df_train[cat_colname].value_counts()) + '\n' + '*' * 100 + '\n')
```

```
Years in current job

10+ years     2332
2 years        705
3 years        620
< 1 year       563
5 years        516
1 year         504
4 years        469
6 years        426
7 years        396
неизвестно     371
8 years        339
9 years        259
Name: Years in current job, dtype: int64
****************************************************************************************************
```

B [37]:
```
df_train.isna().sum()  # просматриваем пропуски
```

Out[37]:
```
Home Ownership                 0
Annual Income                1557
Years in current job            0
Tax Liens                       0
Number of Open Accounts         0
Years of Credit History         0
Maximum Open Credit             0
Number of Credit Problems       0
Months since last delinquent 4081
Bankruptcies                    0
Purpose                         0
Term                            0
Current Loan Amount             0
Current Credit Balance          0
Monthly Debt                    0
Credit Score                 1557
Credit Default                  0
dtype: int64
```

Очистка данных

Класс с подготовкой данных

In [38]:
```
# Считаем выбросами Годовой доход 'Annual Income' > 4 000 000 (91 значения) и Annual Income < 165000
# Считаем выбросами Количество лет кредитной истории 'Years of Credit History' > 40 (83 значения)
# Считаем выбросами Наибольший открытый кредит 'Maximum Open Credit' > 4 000 000 (64 значений)
# и 'Maximum Open Credit' < 50 000 (125 значений)
# Считаем выбросами Количество месяцев с последней просрочки платежа Months since last delinquent > 83 (5 значений)
# Считаем выбросами Текущий кредитный баланс 'Current Credit Balance' > 1300000 (106 значений)
# Считаем выбросами Ежемесячный долг 'Monthly Debt' > 55 000 (98 значений)
# Считаем выбросами Кредитный рейтинг 'Monthly Debt' < 585 и 'Monthly Debt' > 7510
```

В [39]:
```python
class DataPipeLine:
    """Подготовка исходных данных"""

    def __init__(self):
        """Параметры класса:
           Константы для обработки выбрасов"""

        self.medians = None
        self.modes = None

        self.AnnualIncome_min = 165000
        self.AnnualIncome_max = 4000000

        self.YearsofCreditHistory_max = 40

        self.MaximumOpenCredit_min = 50000
        self.MaximumOpenCredit_max = 4000000

        self.MonthsSinceLastDelinquent_max = 83
        self.CurrentLoanAmount_max = 1000000
        self.CurrentCreditBalance_max = 1300000
        self.MonthlyDebt_max = 55000

        self.MonthlyDebt_min = 585
        self.MonthlyDebt_max = 7510


    def fit(self, df):
        """Сохранение статистик"""

        # Расчёт медиан
        self.medians = df_train[['Annual Income', 'Credit Score']].median()
        df = df_train.loc[df_train['Current Loan Amount'] < self.CurrentLoanAmount_max, ['Current Loan Amount']]
        self.modes = df[['Current Loan Amount']].median()

    def transform(self, df):
        """Трансформация данных"""

        # 1. Обработка пропусков
        #df_train = df_train.fillna(median)

        df[['Annual Income', 'Credit Score']] = df[['Annual Income', 'Credit Score']].fillna(self.medians)

        # Months since last delinquent
        # 3581 пропущенное значение из 7500 - удаляем
        if 'Months since last delinquent' in df.columns:
            # df = df.drop(['Months since last delinquent'], axis=1)
            df.drop('Months since last delinquent', axis=1, inplace=True)

        # Years in current job
        cat_colname = 'Years in current job'
        df[cat_colname] = df[cat_colname].replace(to_replace = np.nan, value = 'неизвестно')

        # 2. Выбросы (outliers)

        # Annual Income - годовой доход
        df.loc[df['Annual Income'] < self.AnnualIncome_min, 'Annual Income'] = self.AnnualIncome_min
        df.loc[df['Annual Income'] >= self.AnnualIncome_max, 'Annual Income'] = self.AnnualIncome_max

        # Years of Credit History - Количество лет кредитной истории
        df.loc[df['Years of Credit History'] >= self.YearsofCreditHistory_max, 'Years of Credit History'] = self.YearsofCreditHistory_max

        # Maximum Open Credit - наибольший открытый кредит
        df.loc[df['Maximum Open Credit'] < self.MaximumOpenCredit_min, 'Maximum Open Credit'] = self.MaximumOpenCredit_min
        df.loc[df['Maximum Open Credit'] >= self.MaximumOpenCredit_max, 'Maximum Open Credit'] = self.MaximumOpenCredit_max

        # Current Loan Amount - текущая сумма кредита
        df.loc[df['Current Loan Amount'] >= self.CurrentLoanAmount_max, 'Current Loan Amount'] = self.modes['Current Loan Amount']

        # Current Credit Balance - текущий кредитный баланс
        df.loc[df['Current Credit Balance'] >= self.CurrentCreditBalance_max, 'Current Credit Balance'] = self.CurrentCreditBalance_max

        # Monthly Debt - Ежемесячный долг
        df.loc[df['Monthly Debt'] >= self.MonthlyDebt_max, 'Monthly Debt'] = self.MonthlyDebt_max

        # Monthly Debt - Кредитный рейтинг
        df.loc[df['Monthly Debt'] < self.MonthlyDebt_min, 'Monthly Debt'] = self.MonthlyDebt_min
        df.loc[df['Monthly Debt'] >= self.MonthlyDebt_max, 'Monthly Debt'] = self.MonthlyDebt_max

        # 3. Обработка категорий
        colname = 'Bankruptcies'
        df[colname] = df[colname].replace(to_replace = 'nan', value = '0.0')
        # (создание дами-переменных)
        #df = pd.concat([df, pd.get_dummies(df['Tax Liens'], prefix='Tax Liens', dtype='int8')], axis=1)
        #df = pd.concat([df, pd.get_dummies(df['Number of Credit Problems'], prefix='Number of Credit Problems', dtype='int8')], axis=1)
        #df = pd.concat([df, pd.get_dummies(df['Bankruptcies'], prefix='Bankruptcies', dtype='int8')], axis=1)

        return df


    def features(self, df):
        """4. Feature engineering
              Генерация новых фич"""

        # 1. Home Ownership - домовладение
        cat_colname = 'Home_Ownership_int'

        df[cat_colname] = df['Home Ownership']
        df.loc[df[cat_colname] == 'Have Mortgage', cat_colname] = 0
        df.loc[df[cat_colname] == 'Own Home', cat_colname] = 1
        df.loc[df[cat_colname] == 'Rent', cat_colname] = 2
        df.loc[df[cat_colname] == 'Home Mortgage', cat_colname] = 3

        # 3. 'Years in current job' (порядковые данные)
        cat_colname = 'Years_in_current_job_int'

        df[cat_colname] = df['Years in current job']
        df.loc[df[cat_colname] == '< 1 year', cat_colname] = 0
        df.loc[df[cat_colname] == '1 year', cat_colname] = 1
```

```
        df.loc[df[cat_colname] == '2 years', cat_colname] = 2
        df.loc[df[cat_colname] == '3 years', cat_colname] = 3
        df.loc[df[cat_colname] == '4 years', cat_colname] = 4
        df.loc[df[cat_colname] == '5 years', cat_colname] = 5
        df.loc[df[cat_colname] == '6 years', cat_colname] = 6
        df.loc[df[cat_colname] == '7 years', cat_colname] = 7
        df.loc[df[cat_colname] == '8 years', cat_colname] = 8
        df.loc[df[cat_colname] == '9 years', cat_colname] = 9
        df.loc[df[cat_colname] == '10+ years', cat_colname] = 10
        df.loc[df[cat_colname] == 'неизвестно', cat_colname] = 11

        # 11. Purpose - цель кредита (порядковые данные)
        cat_colname = 'Purpose_int'

        df[cat_colname] = df['Purpose']
        df.loc[df[cat_colname] == 'renewable energy', cat_colname] = 0
        df.loc[df[cat_colname] == 'vacation', cat_colname] = 1
        df.loc[df[cat_colname] == 'educational expenses', cat_colname] = 2
        df.loc[df[cat_colname] == 'moving', cat_colname] = 3
        df.loc[df[cat_colname] == 'wedding', cat_colname] = 4
        df.loc[df[cat_colname] == 'small business', cat_colname] = 5
        df.loc[df[cat_colname] == 'buy house', cat_colname] = 6
        df.loc[df[cat_colname] == 'take a trip', cat_colname] = 7
        df.loc[df[cat_colname] == 'major purchase', cat_colname] = 8
        df.loc[df[cat_colname] == 'medical bills', cat_colname] = 9
        df.loc[df[cat_colname] == 'buy a car', cat_colname] = 10
        df.loc[df[cat_colname] == 'business loan', cat_colname] = 11
        df.loc[df[cat_colname] == 'home improvements', cat_colname] = 12
        df.loc[df[cat_colname] == 'other', cat_colname] = 13
        df.loc[df[cat_colname] == 'debt consolidation', cat_colname] = 14

        # 12. Term - срок кредита (номинативные данные)
        cat_colname = 'Term_int'

        df[cat_colname] = df['Term']
        df.loc[df[cat_colname] == 'Long Term', cat_colname] = 0
        df.loc[df[cat_colname] == 'Short Term', cat_colname] = 1


        numbers = ['0.0', '1.0', '2.0', '3.0', '4.0', '5.0', '6.0', '7.0', '8.0', '9.0']
        numbers_int = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

        # Добавление признаков
        colnames_new = ['Tax_Liens_int', 'Number_of_Credit_Problems_int', 'Bankruptcies_int']
        colnames = ['Tax Liens', 'Number of Credit Problems', 'Bankruptcies']

        for i in range(len(colnames_new)):
            df[colnames_new[i]] = df[colnames[i]]
            for j in range(len(numbers)):
                df.loc[df[colnames_new[i]] == numbers[j], colnames_new[i]] = numbers_int[j]

        # Обработка категорий
        for colname in ['Home_Ownership_int', 'Years_in_current_job_int', 'Purpose_int', 'Term_int']:
             df_train[colname] = df_train[colname].astype('int8')
        for colname in colnames_new:
             df_train[colname] = df_train[colname].astype('int8')

        # 16. Credit Score - Кредитный рейтинг
        df['CreditScore_small'] = df['Credit Score']
        df['CreditScore_large'] = df['Credit Score']

        df.loc[df['Credit Score'] > 2000, 'CreditScore_small'] = 0.0
        df.loc[df['Credit Score'] < 600, 'CreditScore_small'] = 0.0

        df.loc[df['Credit Score'] < 3000, 'CreditScore_large'] =  0.0
        df.loc[df['Credit Score'] > 9000, 'CreditScore_large'] = 0.0

        return df
```

Инициализируем класс

B [40]:
```
data_pl = DataPipeLine()
data_pl.fit(df_train)

# тренировочные данные
df = data_pl.transform(df_train)
df = data_pl.features(df_train)

# валидационные данные
df_tst = data_pl.transform(df_test)
df_tst = data_pl.features(df_test)
```

B [41]:
```
df.columns
```

Out[41]:
```
Index(['Home Ownership', 'Annual Income', 'Years in current job', 'Tax Liens',
       'Number of Open Accounts', 'Years of Credit History',
       'Maximum Open Credit', 'Number of Credit Problems', 'Bankruptcies',
       'Purpose', 'Term', 'Current Loan Amount', 'Current Credit Balance',
       'Monthly Debt', 'Credit Score', 'Credit Default', 'Home_Ownership_int',
       'Years_in_current_job_int', 'Purpose_int', 'Term_int', 'Tax_Liens_int',
       'Number_of_Credit_Problems_int', 'Bankruptcies_int',
       'CreditScore_small', 'CreditScore_large'],
      dtype='object')
```

B [42]:
```
df.describe()
```

Out[42]:

	Annual Income	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Current Loan Amount	Current Credit Balance	Monthly Debt	Credit Score	Credit Default	Home_Ownership_int	Years_in_current_job_int	Purpose_ir
count	7.500000e+03	7500.000000	7500.000000	7.500000e+03	7500.000000	7.500000e+03	7500.000000	7500.000000	7500.000000	7500.000000	7500.000000	7500.00000
mean	1.306657e+06	11.130933	18.270907	6.393339e+05	304013.377067	2.791043e+05	7069.953467	1063.877333	0.281733	2.395467	6.133600	13.44760
std	6.459892e+05	4.908924	6.872207	5.878506e+05	171951.057747	2.458047e+05	1286.141495	1438.335832	0.449874	0.649047	3.699907	1.55129
min	1.650000e+05	2.000000	4.000000	5.000000e+04	11242.000000	0.000000e+00	585.000000	585.000000	0.000000	0.000000	0.000000	0.00000
25%	9.311330e+05	8.000000	13.500000	2.792295e+05	180169.000000	1.142565e+05	7510.000000	718.000000	0.000000	2.000000	3.000000	14.00000
50%	1.168386e+06	10.000000	17.000000	4.781590e+05	265826.000000	2.093230e+05	7510.000000	731.000000	0.000000	2.000000	6.000000	14.00000
75%	1.499974e+06	14.000000	21.800000	7.935015e+05	396929.500000	3.604062e+05	7510.000000	740.000000	1.000000	3.000000	10.000000	14.00000
max	4.000000e+06	43.000000	40.000000	4.000000e+06	789030.000000	1.300000e+06	7510.000000	7510.000000	1.000000	3.000000	11.000000	14.00000

B [43]:
```
df.info()  # Рассмотрим типы признаков
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7500 entries, 0 to 7499
Data columns (total 25 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Home Ownership                7500 non-null   object
 1   Annual Income                 7500 non-null   float64
 2   Years in current job          7500 non-null   object
 3   Tax Liens                     7500 non-null   object
 4   Number of Open Accounts       7500 non-null   float64
 5   Years of Credit History       7500 non-null   float64
 6   Maximum Open Credit           7500 non-null   float64
 7   Number of Credit Problems     7500 non-null   object
 8   Bankruptcies                  7500 non-null   object
 9   Purpose                       7500 non-null   object
 10  Term                          7500 non-null   object
 11  Current Loan Amount           7500 non-null   float64
 12  Current Credit Balance        7500 non-null   float64
 13  Monthly Debt                  7500 non-null   float64
 14  Credit Score                  7500 non-null   float64
 15  Credit Default                7500 non-null   int64
 16  Home_Ownership_int            7500 non-null   int8
 17  Years_in_current_job_int      7500 non-null   int8
 18  Purpose_int                   7500 non-null   int8
 19  Term_int                      7500 non-null   int8
 20  Tax_Liens_int                 7500 non-null   int8
 21  Number_of_Credit_Problems_int  7500 non-null   int8
 22  Bankruptcies_int              7500 non-null   int8
 23  CreditScore_small             7500 non-null   float64
 24  CreditScore_large             7500 non-null   float64
dtypes: float64(10), int64(1), int8(7), object(7)
memory usage: 1.1+ MB
```

B [44]:
```
for cat_colname in df.select_dtypes(include='int8').columns:
    print(str(cat_colname) + '\n\n' + str(df[cat_colname].value_counts()) + '\n' + '*' * 100 + '\n')
```

```
Home_Ownership_int

3    3637
2    3204
1     647
0      12
Name: Home_Ownership_int, dtype: int64
****************************************************************************************************

Years_in_current_job_int

10    2332
2      705
3      620
0      563
5      516
1      504
4      469
6      426
7      396
11     371
8      339
9      259
Name: Years_in_current_job_int, dtype: int64
****************************************************************************************************

Purpose_int

14    5944
13     665
12     412
11     129
10      96
9       71
8       40
7       37
6       34
5       26
4       15
3       11
2       10
1        8
0        2
Name: Purpose_int, dtype: int64
****************************************************************************************************

Term_int

1    5556
0    1944
Name: Term_int, dtype: int64
****************************************************************************************************

Tax_Liens_int

0    7366
1      83
2      30
3      10
4       6
6       2
5       2
7       1
Name: Tax_Liens_int, dtype: int64
****************************************************************************************************

Number_of_Credit_Problems_int

0    6469
1     882
2      93
3      35
4       9
5       7
6       4
7       1
Name: Number_of_Credit_Problems_int, dtype: int64
****************************************************************************************************

Bankruptcies_int

0    6674
1     786
2      31
3       7
4       2
Name: Bankruptcies_int, dtype: int64
****************************************************************************************************
```

5. Отбор признаков

B [45]:
```
df.columns.tolist()
```

Out[45]:
```
['Home Ownership',
 'Annual Income',
 'Years in current job',
 'Tax Liens',
 'Number of Open Accounts',
 'Years of Credit History',
 'Maximum Open Credit',
 'Number of Credit Problems',
 'Bankruptcies',
 'Purpose',
 'Term',
 'Current Loan Amount',
 'Current Credit Balance',
 'Monthly Debt',
 'Credit Score',
 'Credit Default',
 'Home_Ownership_int',
 'Years_in_current_job_int',
 'Purpose_int',
 'Term_int',
 'Tax_Liens_int',
 'Number_of_Credit_Problems_int',
 'Bankruptcies_int',
 'CreditScore_small',
 'CreditScore_large']
```

B [46]:
```
df.head(2)
```

Out[46]:

	Home Ownership	Annual Income	Years in current job	Tax Liens	Number of Open Accounts	Years of Credit History	Maximum Open Credit	Number of Credit Problems	Bankruptcies	Purpose	...	Credit Default	Home_Ownership_int	Years_in_current_job_int	Purpose_int	Ter
0	Own Home	482087.0	неизвестно	0.0	11.0	26.3	685960.0	1.0	1.0	debt consolidation	...	0	1	11	14	
1	Own Home	1025487.0	10+ years	0.0	15.0	15.3	1181730.0	0.0	0.0	debt consolidation	...	1	1	10	14	

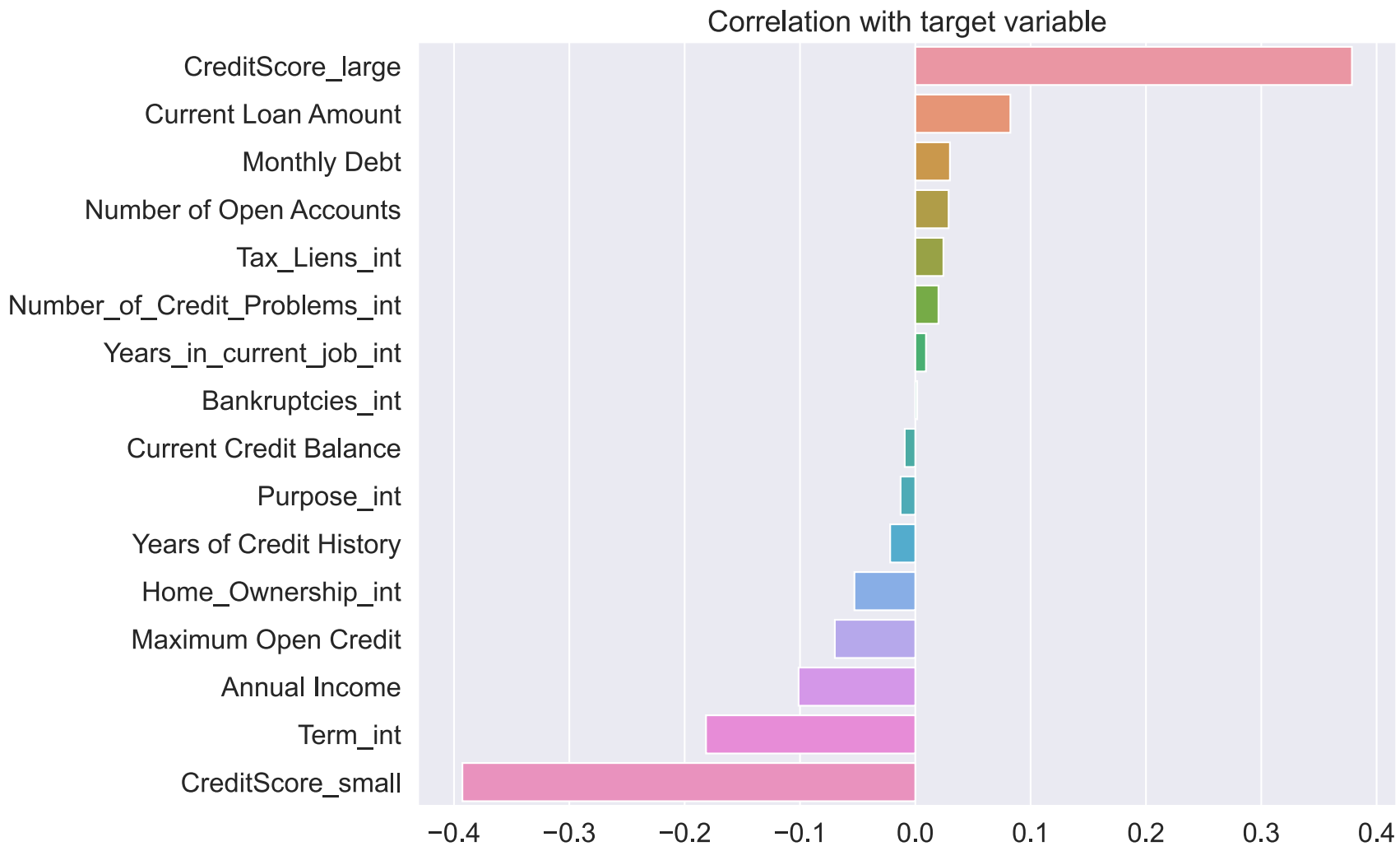
2 rows × 25 columns

B [47]:
```
feature_names = [#'Home Ownership',
                 'Annual Income',
                 #'Years in current job',
                 #'Tax Liens',
                 'Number of Open Accounts',
                 'Years of Credit History',
                 'Maximum Open Credit',
                 #'Number of Credit Problems',
                 #'Bankruptcies',
                 #'Purpose',
                 #'Term',
                 'Current Loan Amount',
                 'Current Credit Balance',
                 'Monthly Debt',
                 #'Credit Score',
                 #'Credit Default',
                 'Home_Ownership_int',
                 'Years_in_current_job_int',
                 'Purpose_int',
                 'Term_int',
                 'Tax_Liens_int',
                 'Number_of_Credit_Problems_int',
                 'Bankruptcies_int',
                 'CreditScore_small',
                 'CreditScore_large']

target_name = 'Credit Default'
```

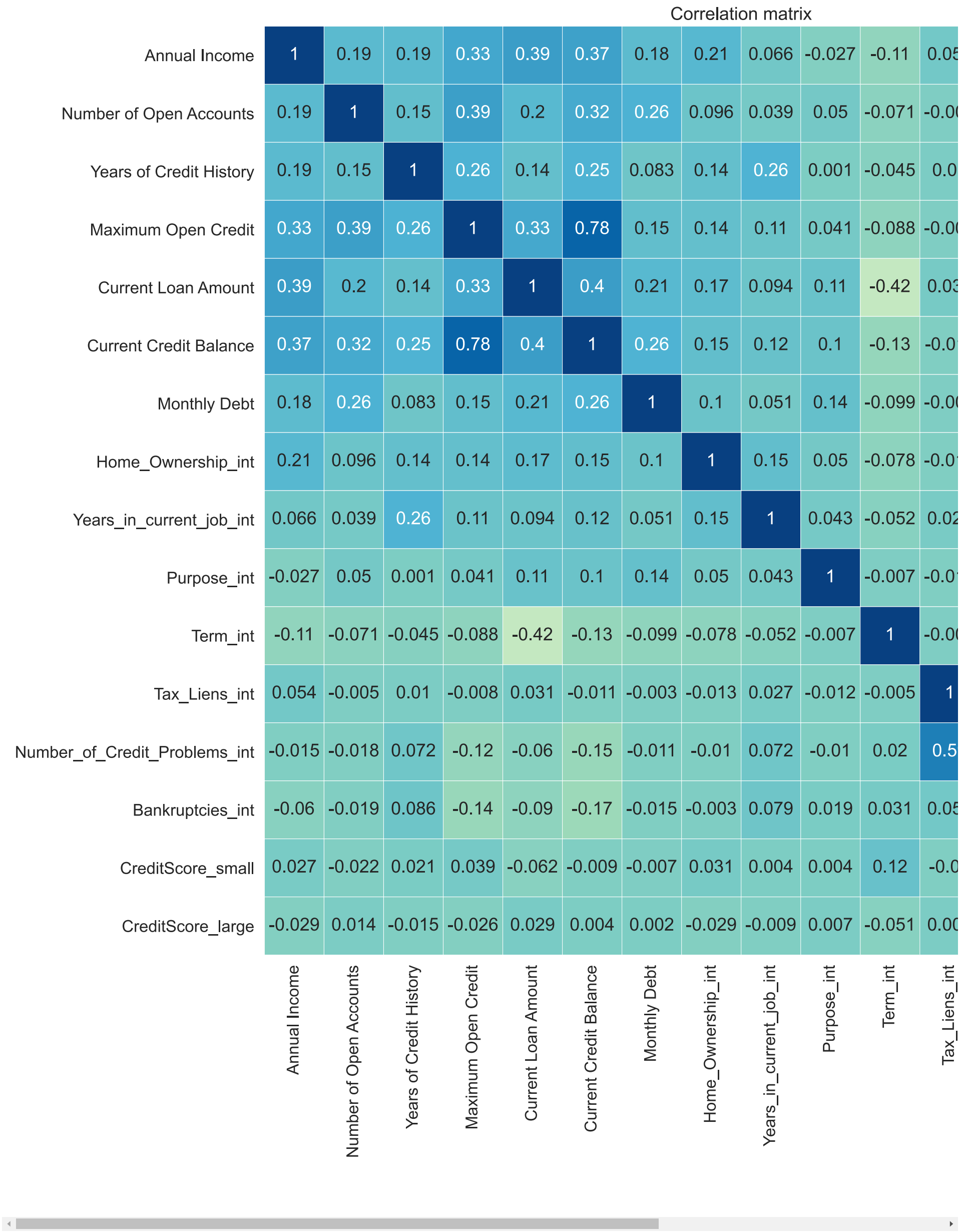
B [48]:
```
TARGET_NAME = 'Credit Default'
BASE_FEATURE_NAMES = feature_names
BASE_FEATURE_NAMES
```

Out[48]:
```
['Annual Income',
 'Number of Open Accounts',
 'Years of Credit History',
 'Maximum Open Credit',
 'Current Loan Amount',
 'Current Credit Balance',
 'Monthly Debt',
 'Home_Ownership_int',
 'Years_in_current_job_int',
 'Purpose_int',
 'Term_int',
 'Tax_Liens_int',
 'Number_of_Credit_Problems_int',
 'Bankruptcies_int',
 'CreditScore_small',
 'CreditScore_large']
```

B [49]:
```
corr_with_target = df[BASE_FEATURE_NAMES + [TARGET_NAME]].corr().iloc[:-1, -1].sort_values(ascending=False)

plt.figure(figsize=(10, 8))

sns.barplot(x=corr_with_target.values, y=corr_with_target.index)

plt.title('Correlation with target variable')
plt.show()
```



B [50]:
```
plt.figure(figsize = (20,16))

sns.set(font_scale=1.4)
sns.heatmap(df[BASE_FEATURE_NAMES].corr().round(3), annot=True, linewidths=.5, cmap='GnBu')

plt.title('Correlation matrix')
plt.show()
```



1. Наблюдается сильная положительная корреляция (**0.78**) между признаками **'Current Loan Amount'** и **'Maximum Open Credit'**. Оба признака сильно влияют на целевой показатель. Оставляем оба признака.
2. Наблюдается сильная положительная корреляция (**0.73**) между признаками **'Bankruptcies_int'** и **'Number_of_Credit_Problems_int'**. При этом **'Bankruptcies_int'** слабо влияет на целевой показатель, данный признак можно исключить из анализа.
3. Наблюдается средняя положительная корреляция (**0.59**) между признаками **'Number_of_Credit_Problems_int'** и **'Tax_Liens_int'**. При этом **'Number_of_Credit_Problems_int'** слабо влияет на целевой показатель. Но **'Number_of_Credit_Problems_int'** сильно связан с признаком **'Bankruptcies_int'**, который мы исключили. Поэтому **'Number_of_Credit_Problems_int'** оставляем.
4. Наблюдается сильная отрицательная корреляция (**-0.97**) между признаками **'CreditScore_small'** и **'CreditScore_large'**. При этом оба признака сильно влияют на целевой показатель. Оставляем оба признака.

Отбор признаков

B [51]:
```
NUM_FEATURE_NAMES = [
                  'Annual Income',
                  'Number of Open Accounts',
                  'Years of Credit History',
                  'Maximum Open Credit',
                  'Current Loan Amount',
                  'Current Credit Balance',
                  'Monthly Debt',
                  #'Credit Score',
                  #'Credit Default',
                  'CreditScore_small',
                  'CreditScore_large']
CAT_FEATURE_NAMES = [
                  'Home Ownership',
                  'Years in current job',
                  'Tax Liens',
                  'Number of Credit Problems',
                  #'Bankruptcies',
                  'Purpose',
                  'Term']

NEW_FEATURE_NAMES = [
                  'Home_Ownership_int',
                  'Years_in_current_job_int',
                  'Purpose_int',
                  'Term_int',
                  'Tax_Liens_int',
                  'Number_of_Credit_Problems_int']
                  #'Bankruptcies_int']

TARGET_NAME = 'Credit Default'

# SELECTED_FEATURE_NAMES = NUM_FEATURE_NAMES + CAT_FEATURE_NAMES + NEW_FEATURE_NAMES
SELECTED_FEATURE_NAMES = NUM_FEATURE_NAMES + NEW_FEATURE_NAMES
```

Масштабрование данных

B [52]:
```
scaler = StandardScaler()

df_norm = df.copy()
df_norm[NUM_FEATURE_NAMES] = scaler.fit_transform(df_norm[NUM_FEATURE_NAMES])

#df = df_norm.copy()
```

4. Рабиение на train/test

B [53]:
```
X = df[SELECTED_FEATURE_NAMES]
y = df[TARGET_NAME]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    shuffle=True,
                                                    test_size=0.3,
                                                    random_state=21,
                                                    stratify=y)

display(y_train.value_counts(normalize=True), y_test.value_counts(normalize=True))
```

```
0    0.718286
1    0.281714
Name: Credit Default, dtype: float64

0    0.718222
1    0.281778
Name: Credit Default, dtype: float64
```

5. Построение модели¶

B [54]:
```
def get_classification_report(y_train_true, y_train_pred, y_test_true, y_test_pred):
    print('TRAIN\n\n' + classification_report(y_train_true, y_train_pred))
    print('CONFUSION MATRIX\n')
    print(pd.crosstab(y_train_true, y_train_pred))
    print('TEST\n\n' + classification_report(y_test_true, y_test_pred))
    print('CONFUSION MATRIX\n')
    print(pd.crosstab(y_test_true, y_test_pred))
```

B [55]:
```
def evaluate_preds(model, X_train, X_test, y_train, y_test):
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    get_classification_report(y_train, y_train_pred, y_test, y_test_pred)
```

B [56]:
```
disbalance = y_train.value_counts()[0] / y_train.value_counts()[1]
print(y_train.value_counts()[0])
print(y_train.value_counts()[1])
disbalance
```

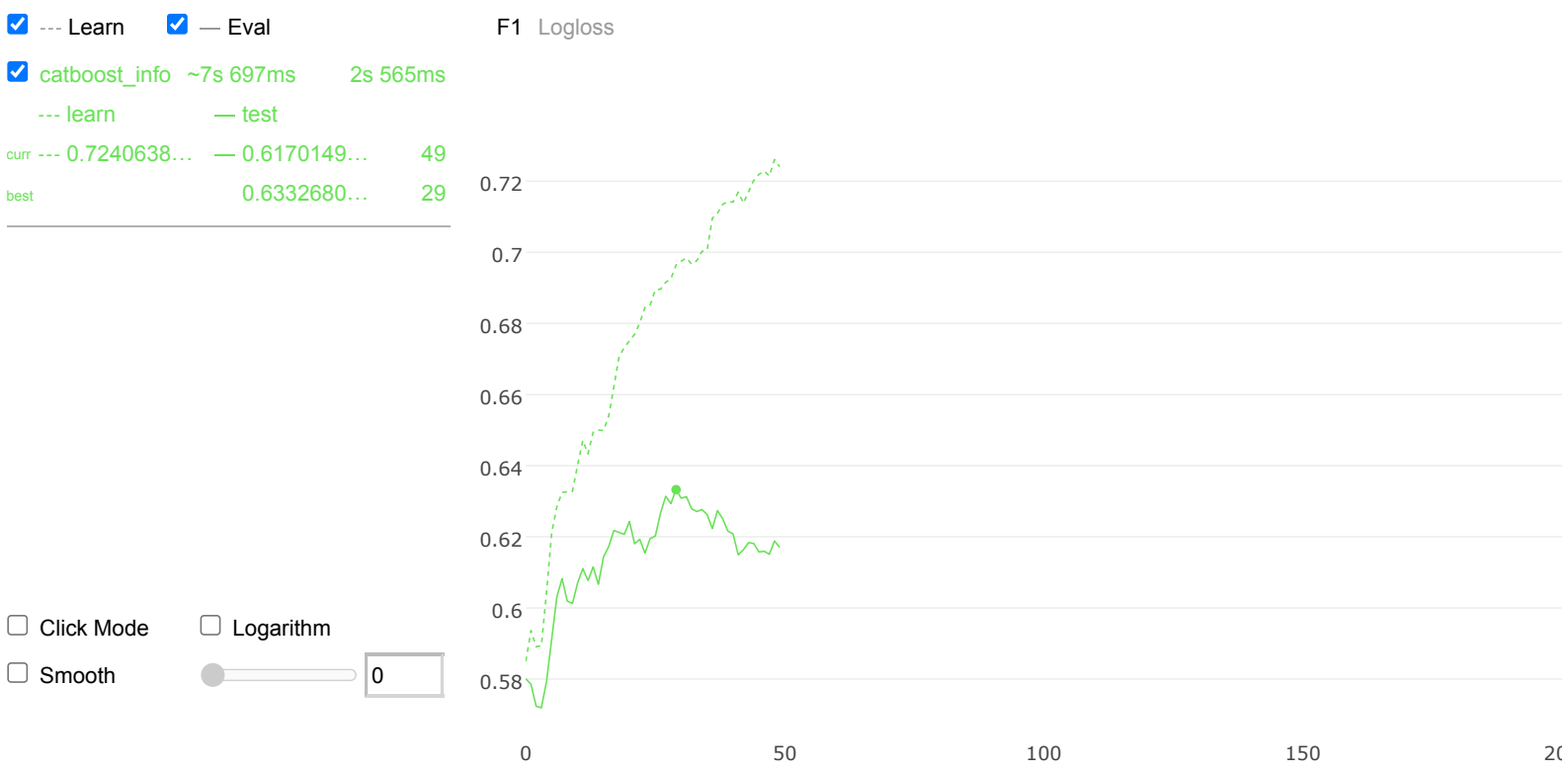
```
3771
1479
```

Out[56]:
```
2.5496957403651117
```

B [57]:
```
frozen_params = {
     'class_weights':[1, disbalance],
     'silent':True,
     'random_state':21,
     'cat_features':NEW_FEATURE_NAMES,
     'eval_metric':'F1',
     'early_stopping_rounds':20
}

model_catb = catb.CatBoostClassifier(**frozen_params)
```

Обучение и оценка модели

B [58]:
```
%%time

final_model = catb.CatBoostClassifier(**frozen_params, iterations=200, max_depth=7)
final_model.fit(X_train, y_train, plot=True, eval_set=(X_test, y_test))

evaluate_preds(final_model, X_train, X_test, y_train, y_test)
```




```
TRAIN

              precision    recall  f1-score   support

           0       0.84      0.80      0.82      3771
           1       0.55      0.63      0.59      1479

    accuracy                           0.75      5250
   macro avg       0.70      0.71      0.70      5250
weighted avg       0.76      0.75      0.75      5250

CONFUSION MATRIX

col_0              0    1
Credit Default
0               3014  757
1                554  925
TEST

              precision    recall  f1-score   support

           0       0.82      0.78      0.80      1616
           1       0.50      0.57      0.53       634

    accuracy                           0.72      2250
   macro avg       0.66      0.67      0.66      2250
weighted avg       0.73      0.72      0.72      2250

CONFUSION MATRIX

col_0              0    1
Credit Default
0               1257  359
1                275  359
Wall time: 3.42 s
```

8. Прогнозирование на тестовом датасете

B [59]:
```
df_tst.shape
```

Out[59]:
```
(2500, 24)
```

B [60]:
```
X_test = df_tst[SELECTED_FEATURE_NAMES]
```

B [61]:
```
y_test_preds = final_model.predict(X_test)
```

B [62]:
```
#evaluate_preds(final_model, X_train, X_test, y_train, y_test_preds)
```

B [63]:
```
predictions = pd.DataFrame()
predictions['Credit Default'] = y_test_preds
predictions['id'] = np.arange(len(predictions))
```

B [64]:
```
# Выгружаем предсказания в файл
predictions.to_csv('ILSokovnin_predictions.csv', index=False, encoding='utf-8', sep=',')

predictions.head()
```

Out[64]:

	Credit Default	id
0	0	0
1	0	1
2	1	2
3	0	3
4	0	4

B [65]:
```
predictions.shape
```

Out[65]:
```
(2500, 2)
```

B [66]:
```
counts = predictions['Credit Default'].value_counts()

plt.figure(figsize=(5,8))
plt.title('Credit Default')
sns.barplot(counts.index, counts.values)

plt.show()
```