Credit Score Reliability Analysis

Project description:

- Client: bank credit department
- Target of the analysis: find out whether the demographics affect the fact of repayment of the loan on time
- Data: bank statistics
- The study results may be taken into account when building the model of "credit scoring" - a special system that evaluates the ability of a potential borrower to repay a loan to a bank

Data

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv('credit_score_reliability_eda.csv')
```

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
children            21525 non-null int64
days_employed       19351 non-null float64
dob_years           21525 non-null int64
education           21525 non-null object
education_id        21525 non-null int64
family_status       21525 non-null object
family_status_id    21525 non-null int64
gender              21525 non-null object
income_type         21525 non-null object
debt                21525 non-null int64
total_income        19351 non-null float64
purpose             21525 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

In [4]:
```
data.head()
```

Out[4]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ger
0	1	-8437.673028	42	высшее	0	женат / замужем	0	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	
3	3	-4124.747207	32	среднее	1	женат / замужем	0	
4	0	340266.072047	53	среднее	1	гражданский брак	1	

Comment

- There are 21,525 rows and 12 columns in the data table
- The days_employed and total_income columns each have 19351 values, which is less than the length of the table, there is a relationship, you need to check further. Both columns are numeric
- Column data types are generally intended for their intended use; accurate verification is required
- Categorical variables: education, family_status, gender, income_type
- Logical variables: education_id, family_status_id, debt
- The remaining variables are quantitative (purpose - it may be necessary to lead to categorical)

Data preprocessing

Empty cells processing

- Work with the days_employed column. The amount of data allocated in the column = 19351 is allocated, this is less than the length of the table
- It is also seen that the data of two types - about the minus sign and without it. Logic suggests that the correct numbers are with a minus sign, since if we divide the number without a minus by the number of days in a year, we get very long periods for one person’s work experience
- We need to understand how many such values and what they affect

In [5]:

```
# check different conditions for understanding groups and quantity

days_employed_retired = data[(data['days_employed']>0) & (data['income_type']=
='пенсионер')].count()

print(days_employed_retired)
print()

days_employed_jobless = data[(data['days_employed']>0) & (data['income_type']=
='безработный')].count()

print(days_employed_jobless)
```

```
children            3443
days_employed       3443
dob_years           3443
education           3443
education_id        3443
family_status       3443
family_status_id    3443
gender              3443
income_type         3443
debt                3443
total_income        3443
purpose             3443
dtype: int64

children            2
days_employed       2
dob_years           2
education           2
education_id        2
family_status       2
family_status_id    2
gender              2
income_type         2
debt                2
total_income        2
purpose             2
dtype: int64
```

- The check showed that the numbers with a minuses refer to the data from the "income_type" == "senior citizen" and "unemployed" columns -- The first 3443, the second 2 -- We do not change the values
- Check the Null next

Check 'days_employed': NaN = 2174 rows.

In [6]:
```
# check for Null

data[data['days_employed'].isnull()].head()
```

Out[6]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ge
12	0	NaN	65	среднее	1	гражданский брак	1	
26	0	NaN	41	среднее	1	женат / замужем	0	
29	0	NaN	63	среднее	1	Не женат / не замужем	4	
41	0	NaN	50	среднее	1	женат / замужем	0	
55	0	NaN	54	среднее	1	гражданский брак	1	

In [7]:
```
# check different conditions for understanding groups and quantity

days_employed_nan = data[(data['days_employed']==0) & (data['total_income']==0
)].count()

days_employed_nan
```

Out[7]:
```
children            0
days_employed       0
dob_years           0
education           0
education_id        0
family_status       0
family_status_id    0
gender              0
income_type         0
debt                0
total_income        0
purpose             0
dtype: int64
```

The values in the days_employed column are directly related to the values of the total_income column. There are no values in one, so there are no values in the other column.

'days_employed': NaN = 2174 rows. fillna to 0.

In [8]:
```
data['days_employed'] = data['days_employed'].fillna(0)
```

In [9]:
```
# check the column for Null

data[data['days_employed'].isnull()]
```

Out[9]:

children	days_employed	dob_years	education	education_id	family_status	family_status_id	genc

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
children            21525 non-null int64
days_employed       21525 non-null float64
dob_years           21525 non-null int64
education           21525 non-null object
education_id        21525 non-null int64
family_status       21525 non-null object
family_status_id    21525 non-null int64
gender              21525 non-null object
income_type         21525 non-null object
debt                21525 non-null int64
total_income        19351 non-null float64
purpose             21525 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

The days_employed column is now 21525 values.

Check the column "total_income" with its 19351 value.

In [11]:
```
# check for Null

data[data['total_income'].isnull()].head()
```

Out[11]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ge
12	0	0.0	65	среднее	1	гражданский брак	1	
26	0	0.0	41	среднее	1	женат / замужем	0	
29	0	0.0	63	среднее	1	Не женат / не замужем	4	
41	0	0.0	50	среднее	1	женат / замужем	0	
55	0	0.0	54	среднее	1	гражданский брак	1	

The NaN values in this column are associated with the days_employed column, where null values affect the filling of this column.

Verification confirmed the relationship. Fill NaN in "total_income" as 0.

In [12]:
```
data['total_income'] = data['total_income'].fillna(0)

# check the column for Null

data[data['total_income'].isnull()]
```

Out[12]:

children	days_employed	dob_years	education	education_id	family_status	family_status_id	genc

In [13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
children            21525 non-null int64
days_employed       21525 non-null float64
dob_years           21525 non-null int64
education           21525 non-null object
education_id        21525 non-null int64
family_status       21525 non-null object
family_status_id    21525 non-null int64
gender              21525 non-null object
income_type         21525 non-null object
debt                21525 non-null int64
total_income        21525 non-null float64
purpose             21525 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

All gaps in the columns are filled.

Comment

- Missing values were found in the "days_employed" and "total_income" columns in the amount of 2174 pieces. These values are numeric.
- It can be assumed that one value is calculated based on the presence of another according to the formula
- Gaps are filled with the fillna method to 0

Data types replacement

In [14]:

```
# check first 20 rows

data.head(20)
```

Out[14]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id
0	1	-8437.673028	42	высшее	0	женат / замужем	0
1	1	-4024.803754	36	среднее	1	женат / замужем	0
2	0	-5623.422610	33	Среднее	1	женат / замужем	0
3	3	-4124.747207	32	среднее	1	женат / замужем	0
4	0	340266.072047	53	среднее	1	гражданский брак	1
5	0	-926.185831	27	высшее	0	гражданский брак	1
6	0	-2879.202052	43	высшее	0	женат / замужем	0
7	0	-152.779569	50	СРЕДНЕЕ	1	женат / замужем	0
8	2	-6929.865299	35	ВЫСШЕЕ	0	гражданский брак	1
9	0	-2188.756445	41	среднее	1	женат / замужем	0
10	2	-4171.483647	36	высшее	0	женат / замужем	0
11	0	-792.701887	40	среднее	1	женат / замужем	0
12	0	0.000000	65	среднее	1	гражданский брак	1
13	0	-1846.641941	54	неоконченное высшее	2	женат / замужем	0
14	0	-1844.956182	56	высшее	0	гражданский брак	1
15	1	-972.364419	26	среднее	1	женат / замужем	0
16	0	-1719.934226	35	среднее	1	женат / замужем	0
17	0	-2369.999720	33	высшее	0	гражданский брак	1
18	0	400281.136913	53	среднее	1	вдовец / вдова	2
19	0	-10038.818549	48	СРЕДНЕЕ	1	в разводе	3

During the study identified artifacts in the following columns: days_employed - values with a minus sign and too large values to be a term; education - a different font, you may need to bring it to one view for analysis;

Check the "children" column by counting unique values.

In [15]:
```
# check the number of unique values

print(data['children'].value_counts())

# check for numbers, exclude str

print(data['children'].sum())
```

```
 0     14149
 1      4818
 2      2055
 3       330
 20       76
-1        47
 4        41
 5         9
Name: children, dtype: int64
11600
```

When studying the first column of 'children', there values that are not similar to real ones: 20 and -1 children per borrower. The data from this column is important for answering one of the questions posed, let's try to understand in more detail.

In the 'children' column, all values are numeric. However, we see unusual amounts in 20 and -1 children. Considering that after 5 children there is a further gap of up to 20, then these 20 children per borrower, it is either a printing error instead of 2, or the system works - rounding off all values from 6 and above to the number 20.

In [16]:
```
# children ==20, to find connections

data[data['children'] == 20]
```

Out[16]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id
606	20	-880.221113	21	среднее	1	женат / замужем	0
720	20	-855.595512	44	среднее	1	женат / замужем	0
1074	20	-3310.411598	56	среднее	1	женат / замужем	0
2510	20	-2714.161249	59	высшее	0	вдовец / вдова	2
2941	20	-2161.591519	0	среднее	1	женат / замужем	0
...	...	...	...	...	...	...	...
21008	20	-1240.257910	40	среднее	1	женат / замужем	0
21325	20	-601.174883	37	среднее	1	женат / замужем	0
21390	20	0.000000	53	среднее	1	женат / замужем	0
21404	20	-494.788448	52	среднее	1	женат / замужем	0
21491	20	-173.954460	27	среднее	1	женат / замужем	0

76 rows × 12 columns

Explicit correlations with other variables in other columns are not visible at first glance. There is a decision to check through the critical value for us "the presence of the fact of loan debt". We will find out the ratio in the group and decide how to correct the value in “20 children” in order to minimize the impact on further calculations.

In [17]:
```
children_debt = data[(data['children']==20) & (data['debt']==1)].count()

# find the number of borrowers with the number of children 20 and the presence of the fact of loan debt

children_debt
```

Out[17]:
```
children            8
days_employed       8
dob_years           8
education           8
education_id        8
family_status       8
family_status_id    8
gender              8
income_type         8
debt                8
total_income        8
purpose             8
dtype: int64
```

It is found that debtors are 8 to 76 = 10.5%

In [18]:
```
# calculate the total value of the debt column

count_debt = data['debt'].value_counts()
print(count_debt)
print()

# correlation

print(count_debt/len(data)*100) #соотношение фактов
```

```
0    19784
1     1741
Name: debt, dtype: int64

0    91.911731
1     8.088269
Name: debt, dtype: float64
```

The ratio of debtors to the total number of borrowers in a group with 20 children = 10.5%, when, as in the whole database, this ratio is 8.1%. It seems logical to find a ratio for all groups in order to rename the number 20 into the correct group (with a similar characteristic in relation).

In [19]:
```
# group by the number of children and display on these groups the amount and n
umber of debtors

debt_grouped = data.groupby('children').agg({'debt': ['sum', 'count']})

# we find the ratio of the amount to the number of debtors in the group of # c
hildren

debt_grouped['ratio'] = debt_grouped['debt']['sum'] / debt_grouped['debt']['co
unt']

debt_grouped
```

Out[19]:

	debt		ratio
	sum	count	
children			
-1	1	47	0.021277
0	1063	14149	0.075129
1	444	4818	0.092154
2	194	2055	0.094404
3	27	330	0.081818
4	4	41	0.097561
5	0	9	0.000000
20	8	76	0.105263

The “2 children” group has a close level of the ratio of the presence of the fact of debt to the total number to the group of “20 children”. Suppose there was a typo here and rename 20 to 2. This replacement will not greatly affect further research.

In [20]:
```
# change 20 to 2

data['children'] = data['children'].replace(20, 2)

data['children'].value_counts()
```

Out[20]:
```
 0    14149
 1     4818
 2     2131
 3      330
-1       47
 4       41
 5        9
Name: children, dtype: int64
```

By analogy, we will proceed with the indicator -1 in the column "number of children": check through the list and if there are no explicit groups for some reason - we assign to the group of values with a similar ratio the presence of the fact of debt to the number of borrowers.

In [21]:
```
# display a data table with the number of children == - 1, find the relationsh
ip

data[data['children'] == -1].head()
```

Out[21]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id
291	-1	-4417.703588	46	среднее	1	гражданский брак	1
705	-1	-902.084528	50	среднее	1	женат / замужем	0
742	-1	-3174.456205	57	среднее	1	женат / замужем	0
800	-1	349987.852217	54	среднее	1	Не женат / не замужем	4
941	-1	0.000000	57	Среднее	1	женат / замужем	0

Correlations with other variables in other columns are not visible. We assign this group to the group of “0 children”, since this is the closest block in terms of ratio.

In [22]:
```
# change -1 for 0

data['children'] = data['children'].replace(-1, 0)

data['children'].value_counts()
```

Out[22]:
```
0    14196
1     4818
2     2131
3      330
4       41
5        9
Name: children, dtype: int64
```

As a result, we made a clear 6 groups in the column 'children' from 0 to 5 according to the number of children.

Checked all the columns through the value_counts and sum methods: Column "dob_years" - 101 value "0" years. The quantity is insignificant, does not affect the calculations - we do nothing. The "gender" column is one XNA value, you need to check.

In [23]:
```
# check the number of unique values

#print(data['purpose'].value_counts())

# check for numbers, exclude str

#print(data['purpose'].sum())
```

In [24]:
```
data['gender'].value_counts()
```

Out[24]:
```
F      14236
M       7288
XNA        1
Name: gender, dtype: int64
```

In [25]:
```
# picked up XNA in the gender column

data[data['gender'] == 'XNA']
```

Out[25]:

	children	days_employed	dob_years	education	education_id	family_status	family_statu
10701	0	-2358.600502	24	неоконченное высшее	2	гражданский брак	

Verification showed that the value of 'gender' == 'XNA' is not critical due to the lack of influence on further calculations. Leave as is.

In [26]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
children            21525 non-null int64
days_employed       21525 non-null float64
dob_years           21525 non-null int64
education           21525 non-null object
education_id        21525 non-null int64
family_status       21525 non-null object
family_status_id    21525 non-null int64
gender              21525 non-null object
income_type         21525 non-null object
debt                21525 non-null int64
total_income        21525 non-null float64
purpose             21525 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

Replace the data in the days_employed column with the integers in the new column.

In [27]:
```
# create a new column for 'days_employed_int' integers

data['days_employed_int'] = data['days_employed'].astype('int')

data.head()
```

Out[27]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ger
0	1	-8437.673028	42	высшее	0	женат / замужем	0	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	
3	3	-4124.747207	32	среднее	1	женат / замужем	0	
4	0	340266.072047	53	среднее	1	гражданский брак	1	

In [28]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 13 columns):
children             21525 non-null int64
days_employed        21525 non-null float64
dob_years            21525 non-null int64
education            21525 non-null object
education_id         21525 non-null int64
family_status        21525 non-null object
family_status_id     21525 non-null int64
gender               21525 non-null object
income_type          21525 non-null object
debt                 21525 non-null int64
total_income         21525 non-null float64
purpose              21525 non-null object
days_employed_int    21525 non-null int64
dtypes: float64(2), int64(6), object(5)
memory usage: 2.1+ MB
```

Comment

In addition to replacing several non-ordinary values in the "children" column. Replaced the value of 'days in employment' ('days_employed_int') with float in int. There is no need to translate negative values.

Duplicates processing

Check the columns for duplicates using the value_count and duplicated () methods

In [29]:
```
# check the number of unique

data['education'].value_counts()
```

Out[29]:
```
среднее                13750
высшее                  4718
СРЕДНЕЕ                  772
Среднее                  711
неоконченное высшее      668
ВЫСШЕЕ                   274
Высшее                   268
начальное                250
Неоконченное высшее       47
НЕОКОНЧЕННОЕ ВЫСШЕЕ       29
НАЧАЛЬНОЕ                 17
Начальное                 15
ученая степень             4
Ученая степень             1
УЧЕНАЯ СТЕПЕНЬ             1
Name: education, dtype: int64
```

In [30]:
```
# doubles check

data.duplicated('education').sum()
```

Out[30]:
```
21510
```

Duplicates are identified in the columns: 'education' We bring to lower case in the same column.

In [31]:
```
data['education'] = data['education'].str.lower()
```

In [32]:
```
data['education'].value_counts()
```

Out[32]:
```
среднее                15233
высшее                  5260
неоконченное высшее      744
начальное                282
ученая степень             6
Name: education, dtype: int64
```

Comment

Duplicates detected in the "education" column. Eliminated by changing the case of values. The data may have arisen due to the fact that there is no check on the register of the field at the data input, you need to apply the default lower case rule. It is also possible that data comes from different sources. In favor of this assumption is the unified names of values.

Lemmatization

In [33]:
```
# load library

from pymystem3 import Mystem
m = Mystem()
```

Our database contains one column that is important for the calculation, but contains information that the borrower transmitted in free form: "purpose" - the purpose of the loan. There is a need to create an additional column for further categorization, which will contain the keyword from the "loan purpose" column. We use lemmatization (transform words to its lemmas).

In [34]:
```
# add the column 'purpose_lemma' and lemmatize the row from 'purpose' into it

data['purpose_lemma'] = data['purpose'].apply(m.lemmatize)
```

In [35]:
```
data.head()
```

Out[35]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ger
0	1	-8437.673028	42	высшее	0	женат / замужем	0	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	
2	0	-5623.422610	33	среднее	1	женат / замужем	0	
3	3	-4124.747207	32	среднее	1	женат / замужем	0	
4	0	340266.072047	53	среднее	1	гражданский брак	1	

Now you need to transfer purpose_lemma to the list of categorical values, according to which to further study the categories.

In [36]:
```
# function through which we select by values and add to a new column

def get_purpose(purpose_lemma):
    if 'свадьба' in purpose_lemma:
        return 'свадьба'
    if 'жильё' in purpose_lemma:
        return 'недвижимость'
    if 'жилье' in purpose_lemma:
        return 'недвижимость'
    if 'образование' in purpose_lemma:
        return 'образование'
    if 'автомобиль' in purpose_lemma:
        return 'автомобиль'
    if 'недвижимость' in purpose_lemma:
        return 'недвижимость'

    return 'other'

# add a new column 'purpose_id' with data from the function

data['purpose_id'] = data['purpose_lemma'].apply(get_purpose)

# through a boolean function I look which lemmas have not yet been verified

#print(data[data['purpose_id'] == 'other'].head())
```

We check how the categories for the purpose of the loan developed:

In [37]:
```
data['purpose_id'].value_counts()
```

Out[37]:
```
недвижимость    10840
автомобиль       4315
образование      4022
свадьба          2348
Name: purpose_id, dtype: int64
```

In [38]:
```
# the function through which we select by values and add to the new column

def get_purpose_business(purpose_lemma):
    if 'коммерческий' in purpose_lemma:
        return 'Yes'
    if 'сдача' in purpose_lemma:
        return 'Yes'
    if 'бизнес' in purpose_lemma:
        return 'Yes'

    return 'No'

# add a new column 'purpose_id_business' with data from the function

data['purpose_id_business'] = data['purpose_lemma'].apply(get_purpose_business
)

# I look through the logical function which lemmas have not yet been verified

#print(data[data['purpose_id'] == 'other'].head())
```

In [39]:
```
data['purpose_id_business'].value_counts()
```

Out[39]:
```
No     19557
Yes     1968
Name: purpose_id_business, dtype: int64
```

All the loan objectives are in 4 categories:

- Real estate (incl. Commercial)
- Car
- Education
- Wedding

Out of these, 1968 requests were for commercial purposes

Comment

All loan requests can be divided into 4 blocks, where "real estate" is the largest (10,840 values), the "car" block is the second most important, but whiter than half as much (4,315 values), slightly inferior to the previous block " education "- 4022 values, and the wedding block closes the chain - 2348 values.

Data categorization

To answer the questions we need categorize: "children" - Yes / No; and "total_income" - define a step in the execution process. We apply dictionaries where necessary.

'children' - Yes/No

In [40]:
```
def children_group(number):
        """
        Returns Yes or No:
        - 'Yes' where number !=0;
        - 'No' where number ==0;
        """

        if number != 0:
                return 'Yes'
        if number == 0:
                return 'No'

# Test the function for each rule:

#print(children_group(5))
```

In [41]:
```
# apply the function and add a column

data['children_group'] = data['children'].apply(children_group)

data.head()
```

Out[41]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ger
0	1	-8437.673028	42	высшее	0	женат / замужем	0	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	
2	0	-5623.422610	33	среднее	1	женат / замужем	0	
3	3	-4124.747207	32	среднее	1	женат / замужем	0	
4	0	340266.072047	53	среднее	1	гражданский брак	1	

Categorization by children - Yes / No applied.

In [42]:
```
data['children_group'].value_counts()
```

Out[42]:
```
No      14196
Yes      7329
Name: children_group, dtype: int64
```

We break "total_income" into categories.

In [43]:
```
# checking for negative values

data[data['total_income']<0]
```

Out[43]:

children	days_employed	dob_years	education	education_id	family_status	family_status_id	genc

In [44]:
```
total_income_max = data['total_income'].max()
print(total_income_max)
```

```
2265604.028722744
```

Categorization "total_income" - 0-120K, 121K-180K, 181K-240K, 241K-360K, 361K-up.

In [45]:
```
def income_group(number):

        if number <0:
            return 'check number'
        if number >=0:
            if number <=120000:
                return '0-120K'
        if number >121000:
            if number <=180000:
                return '121K-180K'
        if number >181000:
            if number <=240000:
                return '181K-240K'
        if number >241000:
            if number <=360000:
                return '241K-360K'
        return '361K-up'

# Test the function for each rule:

income_group(180000)
```

Out[45]: '121K-180K'

We assign to the new column 'income_group' the values obtained from the results of the function.

In [46]:
```
data['income_group'] = data['total_income'].apply(income_group)
data.head()
```

Out[46]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	ger
0	1	-8437.673028	42	высшее	0	женат / замужем	0	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	
2	0	-5623.422610	33	среднее	1	женат / замужем	0	
3	3	-4124.747207	32	среднее	1	женат / замужем	0	
4	0	340266.072047	53	среднее	1	гражданский брак	1	

In [47]:
```
# group by the new income_group column to check the values

data['income_group'].value_counts()
```

Out[47]:
```
0-120K       9019
121K-180K    5982
181K-240K    3228
241K-360K    2310
361K-up       986
Name: income_group, dtype: int64
```

Revenue categorization looks like this: 0-120K 9019 121K-180K 5982 181K-240K 3228 241K-360K 2310 361K-up 986

We categorize the marital status of the borrower.

In [48]:
```
# categorization of id status and number of children based on status

family_status_log = data[['family_status_id', 'children']]
family_status_log.head()
```

Out[48]:

	family_status_id	children
0	0	1
1	0	1
2	0	0
3	0	3
4	1	0

In [49]:
```
# dictionary via family status id and status name

family_status_dict = data[['family_status_id', 'family_status']]
family_status_dict.head(10)
```

Out[49]:

	family_status_id	family_status
0	0	женат / замужем
1	0	женат / замужем
2	0	женат / замужем
3	0	женат / замужем
4	1	гражданский брак
5	1	гражданский брак
6	0	женат / замужем
7	0	женат / замужем
8	1	гражданский брак
9	0	женат / замужем

In [50]:
```
# delete duplicates in the dictionary

family_status_dict = family_status_dict.drop_duplicates().reset_index(drop=True)
family_status_dict.head()
```

Out[50]:

	family_status_id	family_status
0	0	женат / замужем
1	1	гражданский брак
2	2	вдовец / вдова
3	3	в разводе
4	4	Не женат / не замужем

In [51]:
```
# group by status and display the average number of children per borrower

family_status_log.groupby('family_status_id').mean().sort_values('children',as
cending=False).head(10)
```

Out[51]:

	children
family_status_id	
0	0.569305
1	0.459660
3	0.430962
4	0.230715
2	0.153125

Comment

- Categorized the "children" columns, where the values are No = 14196 and Yes = 7329; and "total_income" (monthly income), where 0-120K - 9019 observations; 121K-180K - 5982; 181K-240K - 3228; 241K-360K - 2310; 361K-up - 986;
- We also categorized through the id of family status and deduced the average number of children in each category.

Hypotheses

- Is there a relationship between having children and repaying a loan on time?

In [52]:
```
# pivot where to show the average debt based on the availability of parameter
 "children" Yes / No

data.pivot_table(values='debt',index='children_group',aggfunc=['count','mean'
])
```

Out[52]:

	count	mean
	debt	debt
children_group		
No	14196	0.074951
Yes	7329	0.092373

In [53]:
```
# let's check the average values using grouping by number of children

data.pivot_table(values='debt',index='children',aggfunc=['count','mean'])
```

Out[53]:

	count	mean
	debt	debt
children		
0	14196	0.074951
1	4818	0.092154
2	2131	0.094791
3	330	0.081818
4	41	0.097561
5	9	0.000000

Comment

- There is a correlation between having children and repaying a loan on time
- A group of borrowers with children of any number has a value of 1.74 pp higher than the one without children (9.2% and 7.5% average values, respectively)
- For the groups of borrowers with children, the upper two values are among borrowers with 4 and 2 children (9.76% and 9.48% average values, respectively)
- The minimum average value of the group of borrowers with children for those who indicated the presence of 3 children (8.2%)
- For credit scoring, you can leave the dependence on the division into two groups with and without children, since dividing by the number of children does not give a big difference in average values

- Is there a relationship between marital status and repayment of the loan on time?

In [54]:
```
# pivot where to show the average debt based on data on family status

data.pivot_table(values='debt',index='family_status',aggfunc=['count','mean'])
```

Out[54]:

	count	mean
	debt	debt
family_status		
Не женат / не замужем	2813	0.097405
в разводе	1195	0.071130
вдовец / вдова	960	0.065625
гражданский брак	4177	0.092890
женат / замужем	12380	0.075202

In [55]:
```
# add gender values

data.pivot_table(values='debt',index='family_status',columns='gender',aggfunc=
['count','mean'])
```

Out[55]:

	count			mean		
gender	F	M	XNA	F	M	XNA
family_status						
Не женат / не замужем	1732.0	1081.0	NaN	0.068129	0.144311	NaN
в разводе	936.0	259.0	NaN	0.065171	0.092664	NaN
вдовец / вдова	905.0	55.0	NaN	0.057459	0.200000	NaN
гражданский брак	2868.0	1308.0	1.0	0.081241	0.118502	0.0
женат / замужем	7795.0	4585.0	NaN	0.067992	0.087459	NaN

Comment

- There is a correlation between data on marital status and repayment of a loan on time
- The lowest discipline is for borrowers with the status of “not married” - an average of 9.7% of cases of violation of the terms of payment of the loan. Moreover, it should be noted that such a high percentage of violations is provided by males - 14.4% with violations, when the average % of violations among female borrowers is significantly lower than in the group as a whole - 6.8% (9.7% group)
- The second significant group in terms of the number of violations in loan repayment is “civil marriage” - 9.3%. A similar picture, disaggregated by gender - the percentage of violations by female borrowers is below the average values of the group
- The best situation with overdue loans in the widower / widow group is 6.6%. But, you need to take into account the general trend of low discipline among male borrowers - 20%
- Based on the foregoing, in credit scoring it is necessary to take into account the gender component, and not just data on a particular family status

- Is there a relationship between income and repayment of the loan on time?

In [56]:

```python
print(data.head())
print()
print(data['total_income'].mean())
print()
print(data['total_income'].median())
```

```
   children  days_employed  dob_years education  education_id  \
0         1   -8437.673028         42    высшее             0
1         1   -4024.803754         36   среднее             1
2         0   -5623.422610         33   среднее             1
3         3   -4124.747207         32   среднее             1
4         0  340266.072047         53   среднее             1

      family_status  family_status_id gender income_type  debt   total_income  \
0   женат / замужем                 0      F   сотрудник     0  253875.639453
1   женат / замужем                 0      F   сотрудник     0  112080.014102
2   женат / замужем                 0      M   сотрудник     0  145885.952297
3   женат / замужем                 0      M   сотрудник     0  267628.550329
4  гражданский брак                 1      F   пенсионер     0  158616.077870

                    purpose  days_employed_int  \
0             покупка жилья              -8437
1   приобретение автомобиля              -4024
2             покупка жилья              -5623
3  дополнительное образование            -4124
4           сыграть свадьбу             340266

                          purpose_lemma    purpose_id purpose_id_business  \
0              [покупка,   , жилье, \n]  недвижимость                  No
1     [приобретение,   , автомобиль, \n]    автомобиль                  No
2              [покупка,   , жилье, \n]  недвижимость                  No
3  [дополнительный,   , образование, \n]   образование                  No
4            [сыграть,   , свадьба, \n]       свадьба                  No

  children_group income_group
0            Yes    241K-360K
1            Yes       0-120K
2             No    121K-180K
3            Yes    241K-360K
4             No    121K-180K

150512.84413613725

135514.71128002706
```

In [57]:
```python
# pivot where to show the average debt based on data on income of the borrower

data.pivot_table(values='debt',index='income_group',aggfunc=['count','mean'])
```

Out[57]:

	count	mean
	debt	debt
income_group		
0-120K	9019	0.079942
121K-180K	5982	0.088599
181K-240K	3228	0.079926
241K-360K	2310	0.071861
361K-up	986	0.066937

In [58]:
```python
# group by income groups and study the most risky groups by type of employment
'income type'

data.groupby(['income_type','income_group'])['debt'].agg(['count','mean']).sort_values('mean',ascending=False)
```

Out[58]:

income_type	income_group	count	mean
безработный	0-120K	1	1.000000
в декрете	0-120K	1	1.000000
сотрудник	121K-180K	3260	0.106135
	0-120K	4726	0.094160
	181K-240K	1638	0.089744
	241K-360K	1089	0.083563
компаньон	121K-180K	1380	0.081159
сотрудник	361K-up	406	0.078818
компаньон	0-120K	1534	0.075619
	181K-240K	957	0.073145
	361K-up	396	0.070707
госслужащий	241K-360K	164	0.067073
пенсионер	181K-240K	411	0.065693
госслужащий	0-120K	598	0.065217
	181K-240K	221	0.063348
компаньон	241K-360K	818	0.061125
пенсионер	241K-360K	239	0.058577
	0-120K	2157	0.055169
	121K-180K	945	0.053968
госслужащий	121K-180K	397	0.052897
пенсионер	361K-up	104	0.048077
госслужащий	361K-up	79	0.012658
предприниматель	0-120K	1	0.000000
	361K-up	1	0.000000
безработный	181K-240K	1	0.000000
студент	0-120K	1	0.000000

In [59]:
```
# categorization by income and education

data.groupby(['income_group','education'])['debt'].agg(['count','mean']).sort_
values('mean',ascending=False)
```

Out[59]:

income_group	education	count	mean
361K-up	неоконченное высшее	38	0.184211
121K-180K	начальное	70	0.142857
181K-240K	начальное	36	0.138889
241K-360K	неоконченное высшее	115	0.113043
181K-240K	неоконченное высшее	126	0.111111
121K-180K	среднее	4318	0.096341
0-120K	начальное	157	0.095541
361K-up	среднее	451	0.088692
181K-240K	среднее	2123	0.087141
0-120K	среднее	7019	0.086907
241K-360K	среднее	1322	0.085477
0-120K	неоконченное высшее	259	0.077220
241K-360K	начальное	14	0.071429
121K-180K	неоконченное высшее	206	0.067961
	высшее	1388	0.064841
181K-240K	высшее	942	0.057325
0-120K	высшее	1581	0.048071
241K-360K	высшее	857	0.045508
361K-up	высшее	492	0.038618
0-120K	ученая степень	3	0.000000
181K-240K	ученая степень	1	0.000000
241K-360K	ученая степень	2	0.000000
361K-up	начальное	5	0.000000

Comment

- There is a correlation between data on the borrower's income level and loan repayment on time. However, the differences between the income groups by the average level of violation of credit discipline are small: from 6.7% (income over 351,000 per month) to 8.86% (from 121,000 to 180,000 rubles per month)
- If we apply the grouping by type of employment, we see that there are large groups of borrowers that have a higher value than income groups: 9624 borrowers (45% of the base) from the employee group with income from a minimum of up to 240,000 rubles a month from 8.97% to 10.6% of cases of violation (maximum by income groups 8.86%). The remaining groups by type of employment show better results
- Going deep into details, you can find statistics on the level of violations of credit discipline based on the level of education of the borrower: 7 groups out of 23 (30%) have an excess in average of up to 18.4%
- Based on the foregoing, it is recommended to apply scoring not so much to categories by income separately, but rather in conjunction with the level of education and type of employment

- How do different loan objectives affect its repayment on time?

In [60]:
```
# categorization by purpose of loan and fact of debt

data.groupby(['purpose_id'])['debt'].agg(['count','mean']).sort_values('mean',
ascending=False)
```

Out[60]:

purpose_id	count	mean
автомобиль	4315	0.093395
образование	4022	0.091994
свадьба	2348	0.079216
недвижимость	10840	0.072140

In [61]:
```
# categorization by purpose of loan and fact of debt

data.groupby(['purpose_id_business','purpose_id'])['debt'].agg(['count','mean'
]).sort_values('mean',ascending=False)
```

Out[61]:

purpose_id_business	purpose_id	count	mean
	автомобиль	4315	0.093395
No	образование	4022	0.091994
	свадьба	2348	0.079216
Yes	недвижимость	1968	0.076728
No	недвижимость	8872	0.071123

Comment

- By loan objectives, it can be said that the most risky type of goal is a “car” (9.3% of the facts of an overdue loan); the second is "education" with 9.2%. By a large margin, there is a "wedding" - 7.9% and closes the "real estate" with 7.2% of violations
- The division into commercial and non-commercial real estate does not give a big difference in values - there is no reason to separate it into a separate category because of the quantity

RESUME

- A study of the data showed that there are such groups of borrowers for which the number of facts of overdue loans reaches high values (up to 19%). We are talking about such characteristics as gender, level of education
- If scoring is applied to groups divided according to the basic principle, as in the questions posed, then this does not lead to a significant difference in values: variations of 2-3 pp