Проект: Исследование надежности заемщиков

1 1. Откроем таблицу и изучим общую информацию о данных

In [2]:
```
# подключим библиотеку и загрузим данные
import pandas as pd

data = pd.read_csv('data.csv')
```

In [3]:
```
# выведем первые 10 строчек датафрейма
data.head(10)
```

Out[3]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875.639453	поку
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080.014102	при а
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885.952297	поку
3	3	-4124.747207	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628.550329	допол об
4	0	340266.072047	53	среднее	1	гражданский брак	1	F	пенсионер	0	158616.077870	
5	0	-926.185831	27	высшее	0	гражданский брак	1	M	компаньон	0	255763.565419	поку
6	0	-2879.202052	43	высшее	0	женат / замужем	0	F	компаньон	0	240525.971920	
7	0	-152.779569	50	СРЕДНЕЕ	1	женат / замужем	0	M	сотрудник	0	135823.934197	об
8	2	-6929.865299	35	ВЫСШЕЕ	0	гражданский брак	1	F	сотрудник	0	95856.832424	на п
9	0	-2188.756445	41	среднее	1	женат / замужем	0	M	сотрудник	0	144425.938277	поку

In [ ]:
```
# Выведем основную информацию о датафрейме с помощью метода info()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   children          21525 non-null  int64  
 1   days_employed     19351 non-null  float64
 2   dob_years         21525 non-null  int64  
 3   education         21525 non-null  object 
 4   education_id      21525 non-null  int64  
 5   family_status     21525 non-null  object 
 6   family_status_id  21525 non-null  int64  
 7   gender            21525 non-null  object 
 8   income_type       21525 non-null  object 
 9   debt              21525 non-null  int64  
 10  total_income      19351 non-null  float64
 11  purpose           21525 non-null  object 
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

2 2. Предобработка данных

2.1 2.1.Удаление пропусков

In [ ]:
```
# Выведем количество пропущенных значений для каждого столбца
data.isna().sum()
```

Out[4]:
```
children               0
days_employed       2174
dob_years              0
education              0
education_id           0
family_status          0
family_status_id       0
gender                 0
income_type            0
debt                   0
total_income        2174
purpose                0
dtype: int64
```

В двух столбцах есть пропущенные значения. Один из них — `days_employed` . Пропуски в этом столбце обработаем на следующем этапе. Другой столбец с пропущенными значениями — `total_income` — хранит данные о доходах. На сумму дохода сильнее всего влияет тип занятости, поэтому заполним пропуски в этом столбце медианным значением по каждому типу из столбца `income_type` .

In [ ]:
```
# Заполним пропуски в столбце total_income
for t in data['income_type'].unique():
    data.loc[(data['income_type'] == t) & (data['total_income'].isna()), 'total_income'] = \
    data.loc[(data['income_type'] == t), 'total_income'].median()
```

2.2 2.2. Обработка аномальных значений

В данных могут встречаться артефакты (аномалии) — значения, которые не отражают действительность и появились по какой-то ошибке.

Таким артефактом будет отрицательное количество дней трудового стажа в столбце `days_employed` . Для реальных данных это нормально.

Обработаем значения в этом столбце: заменим все отрицательные значения положительными с помощью метода `abs()` .**

In [ ]:
```
# заменим в столбце days_employed все отрицательные значения положительными
data['days_employed'] = data['days_employed'].abs()
```

In [ ]:
```
# Для каждого типа занятости выведем медианное значение трудового стажа days_employed в днях
data.groupby('income_type')['days_employed'].agg('median')
```

Out[7]:
```
income_type
безработный        366413.652744
в декрете            3296.759962
госслужащий          2689.368353
компаньон            1547.382223
пенсионер          365213.306266
предприниматель       520.848083
сотрудник            1574.202821
студент               578.751554
Name: days_employed, dtype: float64
```

У двух типов (безработные и пенсионеры) аномально большие значения. Исправить такие значения сложно, поэтому оставим их как есть. Тем более этот столбец не понадобится для исследования.

In [ ]:
```
# Выведем перечень уникальных значений столбца children
data['children'].unique()
```

Out[8]:
```
array([ 1,  0,  3,  2, -1,  4, 20,  5])
```

В столбце `children` есть два аномальных значения. Удалим строки, в которых встречаются такие аномальные значения.

In [ ]:
```
# Удалим строки, в которых встречаются аномальные значения в столбце children
data = data[(data['children'] != -1) & (data['children'] != 20)]
```

In [ ]:
```
#  выведем перечень уникальных значений столбца children, чтобы убедиться, что артефакты удалены
data['children'].unique()
```

Out[10]:
```
array([1, 0, 3, 2, 4, 5])
```

2.3 2.3. Удаление пропусков (продолжение)

In [ ]:
```
# заполним пропуски в столбце days_employed медианными значениями по каждого типа занятости income_type
for t in data['income_type'].unique():
    data.loc[(data['income_type'] == t) & (data['days_employed'].isna()), 'days_employed'] = \
    data.loc[(data['income_type'] == t), 'days_employed'].median()
```

In [ ]:
```
# проверим, что все пропуски заполнены
data.isna().sum()
```

Out[12]:
```
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
```

2.4 2.4. Изменение типов данных

In [ ]:
```
# Заменим вещественный тип данных в столбце total_income на целочисленный с помощью метода astype()
data['total_income'] = data['total_income'].astype(int)
```

2.5 2.5. Обработка дубликатов

Обработаем неявные дубликаты в столбце `education`. В этом столбце есть одни и те же значения, но записанные по-разному: с использованием заглавных и строчных букв.

In [ ]:
```
# приведем к нижнему регистру названия в столбце education
data['education'] = data['education'].str.lower()
```

In [ ]:
```
# выведем на экран количество строк-дубликатов в данных
data.duplicated().sum()
```

Out[15]:
```
71
```

In [ ]:
```
# удалим строки-дубликаты
data = data.drop_duplicates()
```

2.6 2.6. Категоризация данных

На основании диапазонов, указанных ниже, создадим в датафрейме `data` столбец `total_income_category` с категориями заемщиков по уровню дохода:

- 0–30000 — `'E'`;
- 30001–50000 — `'D'`;
- 50001–200000 — `'C'`;
- 200001–1000000 — `'B'`;
- 1000001 и выше — `'A'`.

In [ ]:
```python
# напишем функцию с именем categorize_income() для категоризации данных
def categorize_income(income):
    try:
        if 0 <= income <= 30000:
            return 'E'
        elif 30001 <= income <= 50000:
            return 'D'
        elif 50001 <= income <= 200000:
            return 'C'
        elif 200001 <= income <= 1000000:
            return 'B'
        elif income >= 1000001:
            return 'A'
    except:
        pass
```

In [ ]:
```python
# создадим столбец total_income_category
data['total_income_category'] = data['total_income'].apply(categorize_income)
```

In [ ]:
```
# выведем на экран перечень уникальных целей взятия кредита из столбца purpose
data['purpose'].unique()
```

Out[19]:
```
array(['покупка жилья', 'приобретение автомобиля',
       'дополнительное образование', 'сыграть свадьбу',
       'операции с жильем', 'образование', 'на проведение свадьбы',
       'покупка жилья для семьи', 'покупка недвижимости',
       'покупка коммерческой недвижимости', 'покупка жилой недвижимости',
       'строительство собственной недвижимости', 'недвижимость',
       'строительство недвижимости', 'на покупку подержанного автомобиля',
       'на покупку своего автомобиля',
       'операции с коммерческой недвижимостью',
       'строительство жилой недвижимости', 'жилье',
       'операции со своей недвижимостью', 'автомобили',
       'заняться образованием', 'сделка с подержанным автомобилем',
       'получение образования', 'автомобиль', 'свадьба',
       'получение дополнительного образования', 'покупка своего жилья',
       'операции с недвижимостью', 'получение высшего образования',
       'свой автомобиль', 'сделка с автомобилем',
       'профильное образование', 'высшее образование',
       'покупка жилья для сдачи', 'на покупку автомобиля', 'ремонт жилью',
       'заняться высшим образованием'], dtype=object)
```

Создадим функцию, которая на основании данных из столбца `purpose` сформирует новый столбец `purpose_category`, в который войдут следующие категории:**

- `'операции с автомобилем'`,
- `'операции с недвижимостью'`,
- `'проведение свадьбы'`,
- `'получение образования'`.

In [ ]:
```
# создание функции categorize_purpose
def categorize_purpose(row):
    try:
        if 'автом' in row:
            return 'операции с автомобилем'
        elif 'жил' in row or 'недвиж' in row:
            return 'операции с недвижимостью'
        elif 'свад' in row:
            return 'проведение свадьбы'
        elif 'образов' in row:
            return 'получение образования'
    except:
        return 'нет категории'
```

In [ ]:
```
# создадим столбец purpose_category
data['purpose_category'] = data['purpose'].apply(categorize_purpose)
```

2.7 3. Исследование данных

2.7.1 3.1 Проверим наличие зависимости между количеством детей и возвратом кредита в срок

Сгруппируем заемщиков по количеству детей в семье и рассчитаем по каждой группе:

- общее количество заемщиков в группе,
- количество заемщиков, имеющих задолженность по возврату кредита.

Даннае запишем в таблицу data_groupby_children.

Создадим в таблице data_groupby_children столбец relative_weight, в котором рассчитаем процент заемщиков, имеющих задолженность, о общему количеству заемщиков в группе.

In [ ]:
```
# создадим сводную таблицу data_groupby_children
data_groupby_children = pd.pivot_table(data, index=['children'], values = ['debt'], aggfunc = ['count', 'sum'])
data_groupby_children['relative_weight'] = data_groupby_children['sum']['debt']/data_groupby_children['count']['debt']
data_groupby_children.sort_values(by='relative_weight')
```

Out[22]:

children	count debt	sum debt	relative_weight
5	9	0	0.000000
0	14091	1063	7.543822
3	330	27	8.181818
1	4808	444	9.234609
2	2052	194	9.454191
4	41	4	9.756098

Вывод:

- Наименьший процент заемщиков, имеющих задолженность по кредиту, к общему количеству заемщиков наблюдается в группе бездетных заемщиков - 7.54%.
- Группы заемщиков, имеющих 3-5 детей в семье, не являются репрезентативными ввиду своей малочисленности.
- Заемщиков, имеющих детей в семье, существенно меньше, чем заемщиков, не имеющих детей.

Таким образом, наличие детей в семье имеет негативное влияние на возврат кредита в срок.

Можно поработать с многодетными и разделить заемщиков на категории в зависимости от количества детей в семье:

0 - бездетные 1-2 - малодетные 3-5 - многодетные

In [ ]:
```
# создадим функцию категоризации заемщиков по количеству детей
def categorize_children(children):
    try:
        if 1<=children<=2:
            return 'малодетные'
        if 3<=children:
            return 'многодетные'
        return 'бездетные'
    except:
        pass
```

In [ ]:
```
# создадим столбец children_category
data['children_category'] = data['children'].apply(categorize_children)
```

In [ ]:
```
# повторим первый шаг, используя столбец 'children_category'
data_groupby_children2 = pd.pivot_table(data, index=['children_category'], values = ['debt'], aggfunc = ['count', 'sum
data_groupby_children2['relative_weight'] = data_groupby_children2['sum']['debt']/data_groupby_children2['count']['deb
data_groupby_children2.sort_values(by='relative_weight')
```

Out[25]:

	count	sum	relative_weight
	debt	debt	
children_category			
бездетные	14091	1063	7.543822
многодетные	380	31	8.157895
малодетные	6860	638	9.300292

Вывод:

График стал компактнее, но вывод не изменился: наличие детей в семье имеет негативное влияние на возврат кредита в срок. Многодетные семьи либо более ответственные заемщики в сравнении с малодетными, либо их выборка нерепрезентативна.

2.7.2 3.2 Проверим наличие зависимости между семейным положением и возвратом кредита в срок

Сгруппируем заемщиков по семейному положению и рассчитаем по каждой группе:

- общее количество заемщиков в группе,
- количество заемщиков, имеющих задолженность по возврату кредита.

Даннае запишем в таблицу data_groupby_family_status. Создадим в таблице data_groupby_family_status столбец relative_weight, в котором рассчитаем процент заемщиков, имеющих задолженность, о общему количеству заемщиков в группе.

In [ ]:
```
# создадим сводную таблицу data_groupby_family_status
data_groupby_family_status = pd.pivot_table(data, index=['family_status'], values = ['debt'], aggfunc = ['count', 'sum
data_groupby_family_status['relative_weight'] = data_groupby_family_status['sum']['debt']/data_groupby_family_status['
data_groupby_family_status.sort_values(by='relative_weight')
```

Out[26]:

	count	sum	relative_weight
	debt	debt	
family_status			
вдовец / вдова	951	63	6.624606
в разводе	1189	84	7.064760
женат / замужем	12261	927	7.560558
гражданский брак	4134	385	9.313014
Не женат / не замужем	2796	273	9.763948

Вывод:

- Лица, состоящие в браке берут кредиты чаще, чем одинокие.
- Лица, состоящие или состоявщие ранее в официальном браке, являются более ответственными заемщиками.
- Лица, состоящие в гражданском браке либо имеющие статус "не женат / не замужем", наименее ответственные заемщики.

Таким образом, зависимость между семейным положением и возвратом кредита в срок существует: лица, состоящие или состоявшие ранее в официальном браке, чаще гасят кредит в срок, чем лица, живущие в гражданском браке или незамужние/неженатые.

2.7.3 3.3 Проверим наличие зависимости между уровнем дохода и возвратом кредита в срок

Сгруппируем заемщиков по уровню дохода и рассчитаем по каждой группе:

- общее количество заемщиков в группе,
- количество заемщиков, имеющих задолженность по возврату кредита.

Даннае запишем в таблицу data_groupby_income_category . Создадим в таблице data_groupby_income_category столбец relative_weight, в котором рассчитаем процент заемщиков, имеющих задолженность, о общему количеству заемщиков в группе.

In [ ]:
```
# создадим сводную таблицу data_groupby_income_category
data_groupby_income_category = pd.pivot_table(data, index=['total_income_category'], values = ['debt'], aggfunc = ['co
data_groupby_income_category['relative_weight'] = data_groupby_income_category['sum']['debt']/data_groupby_income_cate
data_groupby_income_category.sort_values(by='relative_weight')
```

Out[27]:

	count	sum	relative_weight
	debt	debt	
total_income_category			
D	349	21	6.017192
B	5014	354	7.060231
A	25	2	8.000000
C	15921	1353	8.498210
E	22	2	9.090909

Вывод:

- Наибольшее количество кредитов приходится на лиц с доходом от 50001 до 200000 и от 2000001 до 1000000.
- Группы заемщиков, имеющих категрию дохода D, A и E, не являются репрезентативными ввиду своей малочисленности.
- Если сравнивать заемщиков, имеющих категорию дохода B и C, то наблюдается прямая зависимость между уровнем дохода и возвратом кредита в срок.

Таким образом, если отказаться от анализа малочисленных категорий заемщиков, то уровень дохода влияет на возврат кредита в срок: процент задолженности по возврату кредита в категории B ниже, чем в категории C на 1.43%.

2.7.4 3.4 Проверим влияние цели кредита на его возврат в срок

Сгруппируем заемщиков по целям кредита и рассчитаем по каждой группе:

- общее количество заемщиков в группе,
- количество заемщиков, имеющих задолженность по возврату кредита.

Даннае запишем в таблицу data_groupby_purpose_category. Создадим в таблице data_groupby_purpose_category столбец relative_weight, в котором рассчитаем процент заемщиков, имеющих задолженность, о общему количеству заемщиков в группе.

In [ ]:
```
# создадим сводну таблицу data_groupby_purpose_category
data_groupby_purpose_category = pd.pivot_table(data, index=['purpose_category'], values = ['debt'], aggfunc = ['count'
data_groupby_purpose_category['relative_weight'] = data_groupby_purpose_category['sum']['debt']/data_groupby_purpose_c
data_groupby_purpose_category.sort_values(by='relative_weight')
```

Out[28]:

	count	sum	relative_weight
	debt	debt	
purpose_category			
операции с недвижимостью	10751	780	7.255139
проведение свадьбы	2313	183	7.911803
получение образования	3988	369	9.252758
операции с автомобилем	4279	400	9.347978

Вывод:

- Кредиты с целью "операции с недвижимостью" имеют наименьший процент задолженности - 7.25%.
- Кредиты с целью "проведение свадьбы" на втором месте по своевременности возврата кредита, процент задолженности - 7.91%.
- Кредиты с целью "получение образования" и "операции с автомобилем" чаще имеют задолженность по возврату кредита, процент задолженности - 9.25% и 9.34% соответственно.

Таким образом: цель кредита влияет на возврат кредита в срок. Наиболее надежные заемщики - лица, берущие кредиты на операции с недвижимостью, возможно, это связано с наличием залога. Кредиты на проведение свадьбы и на получение образования вроде бы подпадают под категорию потребительские кредиты, но заемщики ведут себя по разному, возможно сказывается возраст и социальный

статус(гипотеза для дальнейшего изучения). Кредиты по операциям с автомобилем имеют наибольший процент несвоевременного возврата кредита.

2.7.5 3.5 Определим возможные причины появления пропусков в исходных данных

Причины пропуска данных можно разделить на 2 группы:

- человеческий фактор;
- технический сбой.

Человеческий фактор возникает при ручном вводе данных и может быть связан с невнимательностью, ленью, отсутствием четких инструкций по заполению данных, осознанным желанием скрыть информацию или просто отсутствием необходимой информации.

Технический сбой может произойти, например, если датасет собран из нескольких источников, если с одной базой данных работают несколько пользователей, если данные фиксируются автоматически и прибор фиксации данных вышел из строя.

2.7.6 3.6 Объясним, почему заполнить пропуски медианным значением — лучшее решение для количественных переменных

Решение чем запонять пропуски зависит от целей исследования и характера данных.

Медиана устойчива к аномальным отклонениям (выбросам) и проста в расчете. При большом количестве наблюдений использование медианы для заполнения пропусков в количественных переменных, действительно, может быть лучшим решением. Когда количество наблюдений минимально, медиана непредсказуема, и может не отражать истинной картины данных.

2.8 4. Общий вывод

Проведенный анализ говорит о том, что наличие детей и семейное положение оказывают влияние на возврат кредита в срок.

Однако у меня вызывают сомнения репрезентативность выборки и статистическая значимость полученных результатов.

Полагаю, что наличие детей в семье и семейное положение - два взаимосвязанных фактора. Например, одинокие многодетные родители более рискованная категория заемщиков для банка, чем лица, состоящие в официальном браке и не имеющие детей.

Проверим данную гипотезу на имеющемся датасете.

In [ ]:
```
# создадим сводную таблицу data_groupby_children_family
data_groupby_children_family = pd.pivot_table(data, index=['family_status', 'children_category'], values = ['debt'], a
data_groupby_children_family['relative_weight'] = data_groupby_children_family['sum']['debt']/data_groupby_children_fa
data_groupby_children_family.sort_values('relative_weight')
```

Out[29]:

		count	sum	relative_weight
		debt	debt	
family_status	children_category			
вдовец / вдова	многодетные	7	0	0.000000
	бездетные	847	53	6.257379
женат / замужем	бездетные	7468	516	6.909480
в разводе	бездетные	784	55	7.015306
женат / замужем	многодетные	285	20	7.017544
в разводе	малодетные	393	28	7.124682
	многодетные	12	1	8.333333
гражданский брак	бездетные	2730	229	8.388278
женат / замужем	малодетные	4508	391	8.673469
Не женат / не замужем	бездетные	2262	210	9.283820
вдовец / вдова	малодетные	97	10	10.309278
гражданский брак	малодетные	1338	148	11.061286
Не женат / не замужем	малодетные	524	61	11.641221
гражданский брак	многодетные	66	8	12.121212
Не женат / не замужем	многодетные	10	2	20.000000

Действительно, наименьший процент задолженности у бездетных заюмщиков, состоящих в официальном браке, а наибольший - у заемщиков, имеющих детей и либо не состоящих в браке, либо разведенных.

Опять присутствует нерепрезентативная выборка - многодетные заемщики.

Для создания качественной системы скоринга имеет смысл продолжить анализ и учесть другие факторы.