Исследование надёжности заёмщиков

Заказчик — кредитный отдел банка. Нужно разобраться, влияет ли семейное положение и количество детей клиента на факт погашения кредита в срок. Входные данные от банка — статистика о платёжеспособности клиентов.

Результаты исследования будут учтены при построении модели **кредитного скоринга** — специальной системы, которая оценивает способность потенциального заёмщика вернуть кредит банку.

Шаг 1. Откройте файл с данными и изучите общую информацию

In [1]:
```
#открываем файл с данными
import pandas as pd
df = pd.read_csv('/datasets/data.csv')
df.head()
```

Out[1]:

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

In [2]:
```
#изучаем общую информацию
df.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
```

- Имеем столбцы:
  - children — количество детей в семье
  - days_employed — общий трудовой стаж в днях

- dob_years — возраст клиента в годах
- education — уровень образования клиента
- education_id — идентификатор уровня образования
- family_status — семейное положение
- family_status_id — идентификатор семейного положения
- gender — пол клиента
- income_type — тип занятости
- debt — имел ли задолженность по возврату кредитов
- total_income — ежемесячный доход
- purpose — цель получения кредита

Вывод

Есть дубликаты, отрицательный трудовой стаж,пропущенные данные,огромные числа в столбце стажа.Причем, по модулю числа в столбце стажа кажутся нормальными. Так что минус мог быть просто тире. А вот огромные вызывают много вопросов. Это могут быть часы, а могут быть и пропуски, которые при выгрузке трансформировались в аномальные для конкретного столбца числа.

Шаг 2. Предобработка данных

Обработка пропусков

In [3]:
```
print('пропуски до', df['total_income'].isna().sum())
```

```
пропуски до 2174
```

In [4]:
```
#заменяем пропущенные данные медианой,взятой по 3 характеристикам клиента
df['total_income'] = df['total_income'].fillna(df.groupby(['income_type','dob_years', 'educatio
df['total_income'] = df['total_income'].fillna(df.groupby(['children','family_status', 'purpose
df['days_employed'] = df['days_employed'].fillna(df.groupby(['income_type','dob_years', 'educat
```

In [5]:
```
print('пропуски после', df['total_income'].isna().sum())
```

```
пропуски после 0
```

Вывод

пропуски в данных "отработанные дни " и "источник дохода" являются технической ошибкой , по-видимому. Определил методом isna и заменил медианой по нескольким параметрам

Замена типа данных

In [6]:
```
#в столбце " отработанные дни" заменим отрицательные значения на положительные
df['days_employed'] = df['days_employed'].apply(abs)
df['days_employed'] = df['days_employed'].astype(int)
df['days_employed'].head()
```

Out[6]:
```
0      8437
1      4024
2      5623
3      4124
4    340266
Name: days_employed, dtype: int64
```

In [7]:
```
#в столбце " наличие детей в семье" заменим отрицательные значения на положительные
df['children'] = df['children'].apply(abs)
df['children'] = df['children'].astype(int)
df['children'].head()
```

Out[7]:
```
0    1
1    1
2    0
3    3
4    0
Name: children, dtype: int64
```

методом abs() заменил отрицательные значения на положительные , а методом astype(int) привел в целочисленный тип

Обработка дубликатов

In [8]:
```
# переводом в нижний регистр, устраняем неявные дубликаты в столбце "Образование"
df = df.drop_duplicates().reset_index(drop=True)
df['education'] = df['education'].str.lower()
df['education'].unique()
```

Out[8]:
```
array(['высшее', 'среднее', 'неоконченное высшее', 'начальное',
       'ученая степень'], dtype=object)
```

Поскольку у нас нет уникальных идентификаторов пользователей, полные дубликаты вполне могут быть разными людьми.

Воспользуемся библиотекой pymystem3

In [9]:
```
##Загружаем библиотеку и лемматизируем столбец 'цель кредита'
from pymystem3 import Mystem
m = Mystem()
##Делаем список 'purpose.unique()'
purpose_list = df['purpose'].unique()
print(purpose_list)
print()
print()
lemmas = []
##Каждую строчку в 'purpose_list' лемматизируем, получаем список лемм и добавляем значения в пу
for i in purpose_list:
    lemma = m.lemmatize(i)
    lemmas.append(lemma)
print(lemmas)
##Разбиваем на категории по леммам: "Жилье/недвижимость", "автомобиль", "образование", "свадьба
def purpose_change(purpose):
    lemmas_row = m.lemmatize(purpose)
    for i in lemmas_row:
        if 'авто' in i:
            return 'автомобиль'
        if 'недвиж' or 'жил' in i:
            return  'недвижимость'
        if 'свад' in i:
            return  'свадьба'
        if 'образов' in i:
            return 'образование'

df['purpose_def']= df['purpose'].apply(purpose_change)
```

```
['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']


[['покупка', ' ', 'жилье', '\n'], ['приобретение', ' ', 'автомобиль', '\n'], ['дополнительный',
' ', 'образование', '\n'], ['сыграть', ' ', 'свадьба', '\n'], ['операция', ' ', 'с', ' ', 'жиль
е', '\n'], ['образование', '\n'], ['на', ' ', 'проведение', ' ', 'свадьба', '\n'], ['покупка',
' ', 'жилье', ' ', 'для', ' ', 'семья', '\n'], ['покупка', ' ', 'недвижимость', '\n'], ['покупк
а', ' ', 'коммерческий', ' ', 'недвижимость', '\n'], ['покупка', ' ', 'жилой', ' ', 'недвижимос
ть', '\n'], ['строительство', ' ', 'собственный', ' ', 'недвижимость', '\n'], ['недвижимость',
'\n'], ['строительство', ' ', 'недвижимость', '\n'], ['на', ' ', 'покупка', ' ', 'подержать', '
', 'автомобиль', '\n'], ['на', ' ', 'покупка', ' ', 'свой', ' ', 'автомобиль', '\n'], ['операци
я', ' ', 'с', ' ', 'коммерческий', ' ', 'недвижимость', '\n'], ['строительство', ' ', 'жилой',
' ', 'недвижимость', '\n'], ['жилье', '\n'], ['операция', ' ', 'со', ' ', 'свой', ' ', 'недвижи
мость', '\n'], ['автомобиль', '\n'], ['заниматься', ' ', 'образование', '\n'], ['сделка', ' ',
'с', ' ', 'подержанный', ' ', 'автомобиль', '\n'], ['получение', ' ', 'образование', '\n'], ['а
втомобиль', '\n'], ['свадьба', '\n'], ['получение', ' ', 'дополнительный', ' ', 'образование',
'\n'], ['покупка', ' ', 'свой', ' ', 'жилье', '\n'], ['операция', ' ', 'с', ' ', 'недвижимост
ь', '\n'], ['получение', ' ', 'высокий', ' ', 'образование', '\n'], ['свой', ' ', 'автомобиль',
'\n'], ['сделка', ' ', 'с', ' ', 'автомобиль', '\n'], ['профильный', ' ', 'образование', '\n'],
['высокий', ' ', 'образование', '\n'], ['покупка', ' ', 'жилье', ' ', 'для', ' ', 'сдача',
'\n'], ['на', ' ', 'покупка', ' ', 'автомобиль', '\n'], ['ремонт', ' ', 'жилье', '\n'], ['заним
аться', ' ', 'высокий', ' ', 'образование', '\n']]
```

In [10]:

```
#функцией устраняю неявные дубликаты столбца "назначение"
df['purpose'] = df['purpose'].str.lower()
def replace_wrong_values(wrong_values, correct_value): # на вход функции подаются список непраб
    for wrong_value in wrong_values: # перебираем неправильные имена
        df['purpose'] = df['purpose'].replace(wrong_value, correct_value)

duplicates = ['покупка жилья','операции с жильем', 'покупка жилья для семьи', 'покупка недвижим
 'строительство собственной недвижимости', 'недвижимость','строительство недвижимости', 'строит
 'операции со своей недвижимостью', 'покупка своего жилья','операции с недвижимостью','ремонт ж
name = 'жилье' # правильное имя
replace_wrong_values(duplicates, name)
duplicates = ['приобретение автомобиля', 'на покупку подержанного автомобиля',
 'на покупку своего автомобиля', 'автомобили', 'сделка с подержанным автомобилем','автомобиль',
 'свой автомобиль', 'сделка с автомобилем', 'на покупку автомобиля'
    ]
name = 'автомобиль' # правильное имя
replace_wrong_values(duplicates, name)
duplicates = ['дополнительное образование','образование','заняться образованием',
 'получение образования', 'получение дополнительного образования',
 'получение высшего образования', 'профильное образование'
 'высшее образование','заняться высшим образованием','образование','профильное образование', 'в
name = 'образование' # правильное имя
replace_wrong_values(duplicates, name)
duplicates = ['сыграть свадьбу','на проведение свадьбы','свадьба'
    ]
name = 'свадьба' # правильное имя
replace_wrong_values(duplicates, name)
duplicates = [
 'покупка коммерческой недвижимости','операции с коммерческой недвижимостью', 'покупка жилья дл
    ]

name = 'коммерческой недвижимость' # правильное имя
replace_wrong_values(duplicates, name)

replace_wrong_values(duplicates, name)
print(df['purpose'].unique())
```

```
['жилье' 'автомобиль' 'образование' 'свадьба' 'коммерческой недвижимость']
```

In [11]:

```
# функцией устраняю неявные дубликаты столбца "тип дохода"
df['income_type'] = df['income_type'].str.lower()
def replace_wrong_values(wrong_values, correct_value): # на вход функции подаются список непраб
    for wrong_value in wrong_values: # перебираем неправильные имена
        df['income_type'] = df['income_type'].replace(wrong_value, correct_value)
duplicates = ['сотрудник', 'госслужащий','компаньон',
 'предприниматель']
```

```
name = 'работающий' # правильное имя
replace_wrong_values(duplicates, name)
replace_wrong_values(duplicates, name)
print(df['income_type'].unique())
```

```
['работающий' 'пенсионер' 'безработный' 'студент' 'в декрете']
```

In [12]:

```
  #устраняю неявные дубликаты столбца "пол"
df['gender'] = df['gender'].replace('F', 'ж')
df['gender'] = df['gender'].replace('M', 'м')
df['gender'] = df['gender'].replace('XNA', 'м')
df['gender'] = df['gender'].str.lower()
print(df['gender'].unique())
```

```
['ж' 'м']
```

методом duplicated() определил явные дубликаты , а методом drop_duplicates()их обработал. Методом unique() определил неявные дубликаты и обработал их. Дубликаты образовались в результате слияния баз данных, видимо.

Лемматизация

In [13]:

```
# Лемматизация
from collections import Counter
# подсчитываем число повторений счетчиком Counter()
lemmas = m.lemmatize(' '.join(df['purpose']))
print(Counter(lemmas))
```

```
Counter({' ': 23434, 'жилье': 8850, 'автомобиль': 4308, 'образование': 4014, 'свадьба': 2335,
'коммерческий': 1964, 'недвижимость': 1964, '\n': 1})
```

выделяем основные категории: жилье, автомобиль, образование, свадьба и недвижимость

Категоризация данных

In [14]:

```
children_group = df[['children', 'total_income']]
children_group = children_group.drop_duplicates().reset_index(drop=True)
children_group
def children_group(children):
    if children < 1:
        return '0'
    return '1'
df['children_group'] = df['children'].apply(children_group)
df['children_group'].head()
```

Out[14]:

```
0    1
1    1
2    0
3    1
4    0
Name: children_group, dtype: object
```

In [15]:

```
total_income_group = df[['debt', 'total_income']]
total_income_group = total_income_group.drop_duplicates().reset_index(drop=True)
total_income_group

def total_income_group(total_income):
    if total_income < 50000:
        return 'низкий доход'
    if 50000 <= total_income < 200000:
        return 'средний доход'
    return 'высокий доход'

df['total_income_group'] = df['total_income'].apply(total_income_group)
df.head()
```

Out[15]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_
0	1	8437	42	высшее	0	женат / замужем	0	ж	работак
1	1	4024	36	среднее	1	женат / замужем	0	ж	работак
2	0	5623	33	среднее	1	женат / замужем	0	м	работак
3	3	4124	32	среднее	1	женат / замужем	0	м	работак
4	0	340266	53	среднее	1	гражданский брак	1	ж	пенси

In [16]:

```
# посмотрим доходы
df['total_income_group'].value_counts()
```

Out[16]:

```
средний доход    15876
высокий доход     5223
низкий доход       372
Name: total_income_group, dtype: int64
```

Категоризируем данные: "Наличие детей" : есть дети, нет детей. "Семейное положение": женат/замужем, неженат/незамужем, гражданский брак, вдовец / вдова. "Уровень дохода": низкий, средний, высокий.

Шаг 3. Ответьте на вопросы

- Проверим зависимость между наличием детей и возвратом кредита в срок

In [17]:

```
#Проверим зависимость между наличием детей и возвратом кредита в срок
df.groupby('children')['debt'].agg(['count', 'sum', lambda x: '{:.2%} '.format(x.mean())])
```

Out[17]:

children	count	sum	<lambda_0>
0	14107	1063	7.54%
1	4856	445	9.16%
2	2052	194	9.45%
3	330	27	8.18%
4	41	4	9.76%
5	9	0	0.00%
20	76	8	10.53%

Минимальный процент невозврата у бездетных - 7.54%. Наличие детей влияет на возврат кредита в срок

- Проверим зависимость между семейным положением и возвратом кредита в срок

In [18]:

```
# создадим словарь
dict(zip(df['family_status_id'], df['family_status']))
```

Out[18]:
```
{0: 'женат / замужем',
 1: 'гражданский брак',
 2: 'вдовец / вдова',
 3: 'в разводе',
 4: 'Не женат / не замужем'}
```

In [19]:
```
family_dict = df[['family_status_id', 'family_status']]
family_dict = family_dict.drop_duplicates().reset_index(drop=True)
a = df.groupby('family_status_id')['debt'].agg(['count', 'sum', lambda x: '{:.2%} '.format(x.me
a.reset_index().replace({'family_status_id': family_dict.family_status.to_dict()})
```

Out[19]:

	family_status_id	count	sum	<lambda_0>
0	женат / замужем	12344	931	7.54%
1	гражданский брак	4163	388	9.32%
2	вдовец / вдова	959	63	6.57%
3	в разводе	1195	85	7.11%
4	Не женат / не замужем	2810	274	9.75%

Минимальный процент невозврата кредита в срок у вдов/вдовцов,разведенных и у семейных - 6.57, 7.11 и 7.54 соответственно. Наибольший - у живущих в гражданском браке и одиноких - 9.32 и 9.75 соответственно. Семейное положение влияет на возврат кредита в срок.

- Проверим зависимость между уровнем дохода и возвратом кредита в срок

In [20]:
```
# Проверим зависимость между уровнем дохода и возвратом кредита в срок
family_dict = df[['total_income_group', 'total_income']]
family_dict = family_dict.drop_duplicates().reset_index(drop=True)
a = df.groupby('total_income_group')['debt'].agg(['count', 'sum', lambda x: '{:.2%} '.format(x.
a.reset_index().replace({'total_income_group': family_dict.total_income.to_dict()})
```

Out[20]:

	total_income_group	count	sum	<lambda_0>
0	высокий доход	5223	366	7.01%
1	низкий доход	372	23	6.18%
2	средний доход	15876	1352	8.52%

Минимальный процент невозврата кредита в срок у заемщиков с низким уровнем дохода -6.18% , максимальный - 8.52% у заемщиков со средним доходом. Можно предположить, что более взвешанно принимают решение о кредитовании. Наибольший процент задержек по возврату кредита у заемщиков со средним доходом. Видимо, переоценивают свои возможности. Уровень дохода влияет на возврат кредита в срок.

- Проверим как разные цели кредита влияют на его возврат в срок

In [21]:
```
#Проверим как разные цели кредита влияют на его возврат в срок
report = df.pivot_table(index = ['purpose'], values = 'debt', aggfunc = ['sum', 'count', 'mean'
report.columns = ['debt', 'total', '%']
report = report.sort_values(by = ['total'], ascending = False)
report
```

Out[21]:

	debt	total	%
purpose			
жилье	631	8850	0.071299

purpose	debt	total	%
автомобиль	403	4308	0.093547
образование	370	4014	0.092177
свадьба	186	2335	0.079657
коммерческой недвижимость	151	1964	0.076884

Минимальный процент невозврата кредита в срок на приобретение жилья и недвижимости -7.13% и 7.69% соответственно. Максимальный - на приобретение автомобиля и образование - 9.35% и 9.22% соответственно. Цели кредита влияют на возврат кредита в срок.

Шаг 4. Общий вывод

- На надежность заемщиков влияют такие факторы, как наличие детей, семейное положение, уровень дохода и цели кредита.
    - Самые надежные клиенты это люди, берущие кредит на операции с недвижимостью (7.13% и 7.69%), а самые безответственные те, кто обращается в банк с целью получить деньги на покупку автомобиля (9.35)и образование (9.22%).
    - Самый низкий процент задержек по кредиту будет у бездетной семейной пары(7.54%),самый высокий - у многодетных(10.53%).
    - Самый низкий процент задержек по кредиту у вдов/вдовцов(6.57%),самый высокий - у одиноких - (9.75%)
    - Среди заемщиков по уровню дохода, самый низкий процент задержек по кредиту у заемщиков с низким уровнем дохода (6.18%),самый высокий - у заемщиков со средним доходом (8.52%)

    -Таким образом, портрет самого надежного заемщика:
    - вдова/вдовец
    - с низким уровнем дохода
    - без детей
    - цель кредита - покупка недвижимости

Чек-лист готовности проекта

Поставьте 'x' в выполненных пунктах. Далее нажмите Shift+Enter.

- [x] открыт файл;
- [x] файл изучен;
- [x] определены пропущенные значения;
- [x] заполнены пропущенные значения;
- [x] есть пояснение, какие пропущенные значения обнаружены;
- [x] описаны возможные причины появления пропусков в данных;
- [x] объяснено, по какому принципу заполнены пропуски;
- [x] заменен вещественный тип данных на целочисленный;
- [x] есть пояснение, какой метод используется для изменения типа данных и почему;
- [x] удалены дубликаты;
- [x] есть пояснение, какой метод используется для поиска и удаления дубликатов;
- [x] описаны возможные причины появления дубликатов в данных;

- [x] выделены леммы в значениях столбца с целями получения кредита;
- [x] описан процесс лемматизации;
- [x] данные категоризированы;
- [x] есть объяснение принципа категоризации данных;
- [x] есть ответ на вопрос: "Есть ли зависимость между наличием детей и возвратом кредита в срок?";
- [x] есть ответ на вопрос: "Есть ли зависимость между семейным положением и возвратом кредита в срок?";
- [x] есть ответ на вопрос: "Есть ли зависимость между уровнем дохода и возвратом кредита в срок?";
- [x] есть ответ на вопрос: "Как разные цели кредита влияют на его возврат в срок?";
- [x] в каждом этапе есть выводы;
- [x] есть общий вывод.

In [ ]:

Комментарий студента