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]:

```
import pandas as pd

df = pd.read_csv('/datasets/data.csv')

display(df.head(10))

#Вызовем метод info(), чтобы посмотреть общую информацию.
df.info()
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	2
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	1
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	1
3	3	-4124.747207	32	среднее	1	женат / замужем	0	M	сотрудник	0	2
4	0	340266.072047	53	среднее	1	гражданский брак	1	F	пенсионер	0	1
5	0	-926.185831	27	высшее	0	гражданский брак	1	M	компаньон	0	2
6	0	-2879.202052	43	высшее	0	женат / замужем	0	F	компаньон	0	2
7	0	-152.779569	50	СРЕДНЕЕ	1	женат / замужем	0	M	сотрудник	0	1
8	2	-6929.865299	35	ВЫСШЕЕ	0	гражданский брак	1	F	сотрудник	0	
9	0	-2188.756445	41	среднее	1	женат / замужем	0	M	сотрудник	0	1

```
<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
```

Вывод

На данном этапе пока видим:

- Несовпадение количества значений в двух колонках с остальными, пропуски;
- Разный регистр значений в колонке `education`;
- Типы колонок соответствуют их значениям;
- Разные типы данных для колонок с количественными значениями: `float` и `int`

Шаг 2. Предобработка данных

Обработка пропусков

In [2]:

```
#С помощью метода isna() смотрим, где есть пропуски.
print(df.isna().sum())
```

```
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
```

In [3]:

```
df['total_income'] = df['total_income'].fillna(df.groupby('income_type')['total_income']
.transform('median'))
df['days_employed'] = df['days_employed'].fillna(df.groupby('dob_years')['days_employed'
].transform('median'))

display(df.isna().sum())
```

```
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
```

Вывод

Пропусков больше не осталось. Возможно, они появились из-за технического сбоя или необязательности заполнения этих полей.

Замена типа данных

In [4]:

```
df['days_employed'] = df['days_employed'] / 365
df['total_income'] = df['total_income'] / 1000

df['days_employed'] = df['days_employed'].astype('int') # Методом astype() меняем тип да
нных с вещественных на целочисленный
df['total_income'] = df['total_income'].astype('int')

df.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 int64

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 int64

purpose             21525 non-null object

dtypes: int64(7), object(5)
```

```
memory usage: 2.0+ MB
```

Вывод

Теперь числовые значения приведены к единому целочисленному формату, благодаря которому будет удобней делать вычисления.

Обработка дубликатов

In [5]:

```
# Посчитаем количество явных дубликатов.
print(df.duplicated().sum())
```

```
55
```

In [6]:

```
# Удалим дубликаты и сбросим индексы
df = df.drop_duplicates().reset_index(drop=True)

# Проверим на наличие дубликатов
print(df.duplicated().sum())
```

```
0
```

In [7]:

```
# Обратим внимание на дубликаты из-за разных регистров, от них тоже избавляемся:
df['education'] = df['education'].str.lower()
print(df['education'].value_counts())
```

```
среднее                15188
высшее                  5250
неоконченное высшее      744
начальное                282
ученая степень             6
Name: education, dtype: int64
```

Вывод

Дубликатов не осталось, приступаем к следующему пункту.

Лемматизация

In [8]:

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

unique_purpose = ' '.join(df['purpose'].unique())     # Переводим в строку списко уникаль
ных значений колонки purpose
lemmas = m.lemmatize(unique_purpose)                  # Лемматизируем каждое слово в этой
строке

from collections import Counter
print(Counter(lemmas))                                # Подсчитываем леммы
```

```
Counter({' ': 96, 'покупка': 10, 'недвижимость': 10, 'автомобиль': 9, 'образование': 9, '
жилье': 7, 'с': 5, 'операция': 4, 'на': 4, 'свой': 4, 'свадьба': 3, 'строительство': 3, '
получение': 3, 'высокий': 3, 'дополнительный': 2, 'для': 2, 'коммерческий': 2, 'жилой': 2
, 'подержать': 2, 'заниматься': 2, 'сделка': 2, 'приобретение': 1, 'сыграть': 1, 'проведе
```

```
ние': 1, 'семья': 1, 'собственный': 1, 'со': 1, 'профильный': 1, 'сдача': 1, 'ремонт': 1,
'\n': 1})
```

Вывод

С помощью данной конструкции можно выяснить наиболее встречающиеся слова (леммы) в `purpose`, чтобы в дальнейшем все эти цели привести в единые форматы, категоризировать.

Категоризация данных

In [9]:

```
def create_category_purpose(purpose):     # Создаем функцию для лемматизации каждой строки
колонки purpose и возврата названия категории

    purpose = m.lemmatize(purpose)

    if ('недвижимость' in purpose or 'жилье' in purpose or 'жилой' in purpose):
        return 'жилье'

    if 'автомобиль' in purpose:
        return 'автомобиль'

    if 'образование' in purpose:
        return 'образование'

    if 'операция' in purpose:
        return 'лечение'

    if 'свадьба' in purpose:
        return 'свадьба'

df_status_merge = df
df_status_merge['purpose_category'] = df_status_merge['purpose'].apply(create_category_pu
rpose)
                                                                                # Добавляем но
вый столбец к нашей таблице с
                                                                                # категоризиро
ванным purpose

display(df.head(10))     # Проверяем таблицу с новой колонкой
#Боже, время час ночи, я уже не понимаю, что пишу T_T
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	t
0	1	-23	42	высшее	0	женат / замужем	0	F	сотрудник	0	
1	1	-11	36	среднее	1	женат / замужем	0	F	сотрудник	0	
2	0	-15	33	среднее	1	женат / замужем	0	M	сотрудник	0	
3	3	-11	32	среднее	1	женат / замужем	0	M	сотрудник	0	
4	0	932	53	среднее	1	гражданский брак	1	F	пенсионер	0	
5	0	-2	27	высшее	0	гражданский брак	1	M	компаньон	0	
6	0	-7	43	высшее	0	женат / замужем	0	F	компаньон	0	
7	0	0	50	среднее	1	женат / замужем	0	M	сотрудник	0	
8	2	-18	35	высшее	0	гражданский брак	1	F	сотрудник	0	

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	t
9	0	-5	41	среднее	1	женат / замужем	0	M	сотрудник	0	

✔ Комментарий ревьюера:
Основные цели кредита определены правильно!

In [10]:

```
def create_having_children(children):

    if children > 0:
        return 'Есть дети'

    else:
        return 'Нет детей'

def create_debt_status(debt):

    if debt == 1:
        return 'Должник'
    else:
        return 'Не должник'

df_status_merge['having_children'] = df_status_merge['children'].apply(create_having_chil
dren)
df_status_merge['debt_status'] = df_status_merge['debt'].apply(create_debt_status)

df_status_merge.head()
```

Out[10]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	t
0	1	-23	42	высшее	0	женат / замужем	0	F	сотрудник	0	
1	1	-11	36	среднее	1	женат / замужем	0	F	сотрудник	0	
2	0	-15	33	среднее	1	женат / замужем	0	M	сотрудник	0	
3	3	-11	32	среднее	1	женат / замужем	0	M	сотрудник	0	
4	0	932	53	среднее	1	гражданский брак	1	F	пенсионер	0	

In [11]:

```
def create_income_category(total_income):

    if total_income < 30:
        return 'бедные'

    if total_income >= 30 and total_income < 70:
        return 'выше бедности'

    if total_income >= 70 and total_income < 110:
        return 'средний класс'

    else:
        return 'высший класс'

df_status_merge['income_category'] = df_status_merge['total_income'].apply(create_income_
category)
df_status_merge.head()
```

Out[11]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	t
0	1	-23	42	высшее	0	женат / замужем	0	F	сотрудник	0	
1	1	-11	36	среднее	1	женат / замужем	0	F	сотрудник	0	
2	0	-15	33	среднее	1	женат / замужем	0	M	сотрудник	0	
3	3	-11	32	среднее	1	женат / замужем	0	M	сотрудник	0	
4	0	932	53	среднее	1	гражданский брак	1	F	пенсионер	0	

Вывод

Мы произвели категоризацию данных для дальнейшей работы и поиска взаимосвязей. Всего добавили три новых категории для более удобной работы со сводными таблицами:

- `having_children`
- `debt_status`
- `income_category`

Шаг 3. Ответьте на вопросы

- Есть ли зависимость между наличием детей и возвратом кредита в срок?

In [12]:

```
# Создадим сводную таблицу, где строки - это наличие или отсутствие долга
# В колонках будет общая сумма детей по одной из двух категорий
child_pivot = pd.pivot_table(df_status_merge,
                             index='having_children',
                             columns='debt_status',
                             values='debt', aggfunc='count',
                             margins=True,                          # Здесь добавляю коло
нку "Всего"
                             margins_name='total')

child_pivot['ratio'] = child_pivot['Должник'] / child_pivot['total'] # Нахожу долю должн
иков для каждой категории

child_pivot
```

Out[12]:

debt_status	Должник	Не должник	total	ratio
having_children				
Есть дети	677	6639	7316	0.092537
Нет детей	1064	13090	14154	0.075173
total	1741	19729	21470	0.081090

Вывод

Если смотреть на долю должников, то можно увидеть, что у клиентов с детьми будет выше шанс появления проблемы с возвратом кредита в срок.

Есть ли зависимость между семейным положением и возвратом кредита в срок?

In [13]:

```
family_status_pivot = df_status_merge.pivot_table(index='family_status',
                                                  columns='debt_status',
                                                  values='debt',
                                                  aggfunc='count',
                                                  margins=True,
                                                  margins_name='total')

family_status_pivot['ratio'] = family_status_pivot['Должник'] / family_status_pivot['tota
l']

family_status_pivot
```

Out[13]:

debt_status	Должник	Не должник	total	ratio
family_status				
Не женат / не замужем	274	2536	2810	0.097509
в разводе	85	1110	1195	0.071130
вдовец / вдова	63	896	959	0.065693
гражданский брак	388	3775	4163	0.093202
женат / замужем	931	11412	12343	0.075427
total	1741	19729	21470	0.081090

Вывод

Клиенты с семейным положением, у кого меньше проблем с возвратом кредита в срок:

- в разводе
- вдовец/вдова
- в браке

И чаще всего возникают проблемы с возвратом кредита в срок у клиентов, находящихся в таких семейных положениях:

- в гражданском браке
- не женаты/не замужем

Есть ли зависимость между уровнем дохода и возвратом кредита в срок?

In [14]:

```
total_income_pivot = df_status_merge.pivot_table(index='income_category', # В качестве ст
рок взяла значения зарплаты
                                                 columns='debt_status',    # В колонках
сгруппировала значения статуса должников
                                                 values='debt',
                                                 aggfunc='count',          # Произвела п
одсчет каждого значения
                                                 margins=True,             # Добавила с
аргументом margin колонку итог "total"
                                                 margins_name='total')

total_income_pivot['ratio'] = total_income_pivot['Должник'] / total_income_pivot['total']
total_income_pivot
```

Out[14]:

debt_status	Должник	Не должник	total	ratio
income_category				

debt_status	Должник	Не должник	total	ratio
бедные	2	20	22	0.090909
высший класс	1286	14542	15828	0.081248
income_category				
выше бедности	99	1353	1452	0.068182
средний класс	354	3814	4168	0.084933
total	**1741**	**19729**	**21470**	**0.081090**

Вывод

Зависимость от уровня дохода не наблюдается. Бедный класс (меньше 30 тыс. рублей в месяц) имеет бОльшую долю неплательщиков. Класс выше бедности (30-70 тыс.руб.) имеет наименьшую долю неплательщиков среди всех категорий, а оставшиеся категории имеют примерно одиноковую долю неплательщиков.

Как разные цели кредита влияют на его возврат в срок?

In [15]:

```
purpose_pivot = df_status_merge.pivot_table(index='purpose_category',    # Тоже самое здес
ь, только другие значения для строк.
                                            columns='debt_status',
                                            values='debt',
                                            aggfunc='count',
                                            margins=True,
                                            margins_name='total')

purpose_pivot['ratio'] = purpose_pivot['Должник'] / purpose_pivot['total']
purpose_pivot
```

Out[15]:

debt_status	Должник	Не должник	total	ratio
purpose_category				
автомобиль	**403**	**3904**	**4307**	**0.093569**
жилье	**782**	**10032**	**10814**	**0.072314**
образование	**370**	**3644**	**4014**	**0.092177**
свадьба	**186**	**2149**	**2335**	**0.079657**
total	**1741**	**19729**	**21470**	**0.081090**

Вывод

Чаще всего возникают проблемы с возвратом кредита в срок у клиентов, цель которых:

- автомобиль
- образование

Реже возникают проблемы со сроком у клиентов, которые берут кредит для:

- жилье
- свадьба

Шаг 4. Общий вывод

Подведем саммари всего проекта:

1. Обработали пропуски
2. Почистили явные и неявные дубликаты
3. Произвели замену типа данных для удобства работы с этими значениями
4. Выделили леммы, создав функцию
5. Закрепили тему категоризации данных (сильно закрепили)
6. Сделали анализ данных с помощью сводных таблиц

6. Сделали анализ данных с помощью сводных таблиц.

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.