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.

Изучение общей информации.

In [1]:
```python
import pandas as pd
data = pd.read_csv('/datasets/data.csv')
data.info()
data.head(5)
```

```
<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
```

Out[1]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875.639453	покупка жилья
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080.014102	приобретение автомобиля
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885.952297	покупка жилья
3	3	-4124.747207	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628.550329	дополнительное образование
4	0	340266.072047	53	среднее	1	гражданский брак	1	F	пенсионер	0	158616.077870	сыграть свадьбу

Вывод

Предоставленные данные имеют пропущенные значения, дубликаты и ошибочные типы данных. Они нуждаются в предобработке.

Предобработка данных

Обработка пропусков

In [2]:
```python
for cat in data['income_type'].unique():
    median = data.loc[data.loc[:,'income_type'] == cat]['total_income'].median()
    #data.loc[data['income_type'] == cat, 'total_income'] = median
    print(median, cat)
    data['total_income'] = data['total_income'].fillna(data.groupby('income_type')['total_income'].transform('media
n'))

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

```
142594.39684740017 сотрудник
118514.48641164352 пенсионер
172357.95096577113 компаньон
150447.9352830068 госслужащий
131339.7516762103 безработный
499163.1449470857 предприниматель
98201.62531401133 студент
53829.13072905995 в декрете
Empty DataFrame
Columns: [children, days_employed, dob_years, education, education_id, family_status, family_status_id, gender, incom
e_type, debt, total_income, purpose]
Index: []
```

Вывод

Пропущенные значения есть в колонках days_employed, total_income. Для ответа на поставленные вопросы нужно определить и заполнить пропущенные значения в колонке total_income. Доля пропущенных значений в этой колонке составляет 10%. Пропущенные значения соответствуют пропущенным значениям в колонке days_employed. Можно сделать вывод, что не все клиенты предоставили данные о своих доходах. Данные других колонок при пропущенном значении total_income нужны для исследования, поэтому принято решение заполнить пропущенные значения, а не удалять полностью строку с пустым значением. Проверим наличие артефактов. Среднее значение колонки = 167422. Максимальное = 2265604, минимальное - 20667. Принимая во внимание большой разброс значений, отсутствующие значения заполним медианой. Для увеличения качества данных заполним их в разрезе категорий по типу дохода. После замены отсутствующих значений в колонке total_income нет пропусков.

Замена типа данных

In [3]:
```python
try:
    data['days_employed'] = data['days_employed'].values.astype('int64')
    data['total_income'] = data['total_income'].values.astype('int64')
    data['children'] = data['children'].values.astype('int64')
except:
    print('Не удалось привести данные к типу int64')

data['children'] = data['children'].replace(20, 2).replace(-1, 0)

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 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

Для обработки данных в разрезе предоставленных запросов нам не требуется высокая точность количественных переменных. Поэтому мы можем изменить тип данных в колонках days_employed и total_income на int64 для большей наглядности. Используем метод astype, потому что в методе to_numeric нельзя явно указать тип - int или float.

При исследовании значений колонки children были выявлены явно ошибочные значения: -1, 20. Предположим, что значение 20 является результатом ошибочного нажатия нуля после двойки. А значение -1 могло придти из системы ввода данных как булево ложное значение признака наличия детей. Заменим соответственно, значения 20 на 2, -1 на 0.

Обработка дубликатов

In [4]:
```python
data['education'] = data['education'].apply(str.lower)
data['family_status'] = data['family_status'].apply(str.lower)
data['gender'] = data['gender'].apply(str.lower)
data['income_type'] = data['income_type'].apply(str.lower)
data['purpose'] = data['purpose'].apply(str.lower)

data = data.drop_duplicates()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21454 entries, 0 to 21524
Data columns (total 12 columns):
children            21454 non-null int64
days_employed       21454 non-null int64
dob_years           21454 non-null int64
education           21454 non-null object
education_id        21454 non-null int64
family_status       21454 non-null object
family_status_id    21454 non-null int64
gender              21454 non-null object
income_type         21454 non-null object
debt                21454 non-null int64
total_income        21454 non-null int64
purpose             21454 non-null object
dtypes: int64(7), object(5)
memory usage: 2.1+ MB
```

Вывод

Для поиска дубликтов строк сначала приведем строковые данные в нижний регистр. Проведем поиск дубликатов - найден 71 дубликат. Затем удалим полные дубликаты строк при помощи метода drop_duplicates, оставив только уникальные строки. Полные дубликаты строк могли появиться в связи с ошибками при загрузке данных. Для человека такая ошибка не характерна.

Лемматизация

In [5]:
```python
from pymystem3 import Mystem
from collections import Counter
m = Mystem()

all_lemmas = []
for string in data['purpose']:
    all_lemmas += m.lemmatize(string)
#print(Counter(all_lemmas))

purpose_cats = ['недвижимость','автомобиль','образование','свадьба','ремонт']

def get_purpose_cat(purpose):
    lemmas = m.lemmatize(purpose)
    if 'ремонт' in lemmas:
        return 'ремонт'
    elif 'жилье' in lemmas:
        return 'недвижимость'
    elif 'недвижимость' in lemmas:
        return 'недвижимость'
    elif 'автомобиль' in lemmas:
        return 'автомобиль'
    elif 'образование' in lemmas:
        return 'образование'
    elif 'свадьба' in lemmas:
        return 'свадьба'

data['purpose_categories'] = data['purpose'].apply(get_purpose_cat)
data.head(10)
#data['purpose_categories'].unique()
```

Out[5]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose	p
0	1	-8437	42	высшее	0	женат / замужем	0	f	сотрудник	0	253875	покупка жилья	
1	1	-4024	36	среднее	1	женат / замужем	0	f	сотрудник	0	112080	приобретение автомобиля	
2	0	-5623	33	среднее	1	женат / замужем	0	m	сотрудник	0	145885	покупка жилья	
3	3	-4124	32	среднее	1	женат / замужем	0	m	сотрудник	0	267628	дополнительное образование	
4	0	340266	53	среднее	1	гражданский брак	1	f	пенсионер	0	158616	сыграть свадьбу	
5	0	-926	27	высшее	0	гражданский брак	1	m	компаньон	0	255763	покупка жилья	
6	0	-2879	43	высшее	0	женат / замужем	0	f	компаньон	0	240525	операции с жильем	
7	0	-152	50	среднее	1	женат / замужем	0	m	сотрудник	0	135823	образование	
8	2	-6929	35	высшее	0	гражданский брак	1	f	сотрудник	0	95856	на проведение свадьбы	
9	0	-2188	41	среднее	1	женат / замужем	0	m	сотрудник	0	144425	покупка жилья для семьи	

Вывод

Для того, чтобы выделить группы пользователей по целям кредита, нам нужно провести лемматизацию колонки purpose. Выведем уникальные найденные леммы. Цель кредита состоит из нескольких слов - прилагательные, глаголы, предлоги, существительные. Составим справочник категорий целей кредитов, отталкиваясь от лемм-существительных: у нас получилось пять категрий: недвижимость, автомобиль, образование, свадьба, ремонт. Разделим все записи по этим категориям и сохраним название категории в новую колонку purpose_cat. После этого проверим, все ли запеси были категоризированы, при помощи вывода уникальных значений колонки purpose_categories.

Категоризация данных

In [6]:
```python
def get_children_cat(count):
    if count == 0:
        return 'нет детей'
    elif count < 4:
        return 'не многодетные'
    elif count >= 4:
        return 'многодетные'

def get_income_cat(count):
    if count <= 140000:
        return 'низкий доход'
    elif (count > 140000) & (count < 150000):
        return 'средний доход'
    else:
        return 'высокий доход'

data['children_count_cat'] = data['children'].apply(get_children_cat)
data['total_income_cat'] = data['total_income'].apply(get_income_cat)

data.tail(10)
```

Out[6]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose
21515	1	-467	28	среднее	1	женат / замужем	0	f	сотрудник	1	109486	заняться образованием
21516	0	-914	42	высшее	0	женат / замужем	0	f	компаньон	0	322807	покупка своего жилья
21517	0	-404	42	высшее	0	гражданский брак	1	f	компаньон	0	178059	на покупку своего автомобиля
21518	0	373995	59	среднее	1	женат / замужем	0	f	пенсионер	0	153864	сделка с автомобилем
21519	1	-2351	37	ученая степень	4	в разводе	3	m	сотрудник	0	115949	покупка коммерческой недвижимости
21520	1	-4529	43	среднее	1	гражданский брак	1	f	компаньон	0	224791	операции с жильем
21521	0	343937	67	среднее	1	женат / замужем	0	f	пенсионер	0	155999	сделка с автомобилем
21522	1	-2113	38	среднее	1	гражданский брак	1	m	сотрудник	1	89672	недвижимость
21523	3	-3112	38	среднее	1	женат / замужем	0	m	сотрудник	1	244093	на покупку своего автомобиля
21524	2	-1984	40	среднее	1	женат / замужем	0	f	сотрудник	0	82047	на покупку автомобиля

Вывод

Для наглядности ответа на поставленные вопросы нам нужно разделить данные на категории. Категории по целям кредитов мы уже определили с помощью лемматизации. По наличию детей разделим записи на три группы: нет детей, немногоднетные, многодетные (4 ребенка и больше). По семейному положению данные уже разбиты на пять категорий - 'женат / замужем', 'гражданский брак', 'вдовец / вдова', 'в разводе', 'не женат / не замужем'. Воспользуемся существующими группами. По уровню дохода разделим данные на количественные категории категории: низкий доход, средний доход, высокий доход.

Создадми функции при помощи которых добавим колонки с необходимыми категориями.

Ответы на вопросы бизнеса

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

In [7]:
```python
pivot_table = data.pivot_table('total_income', index='children_count_cat', columns='debt', aggfunc='count')
pivot_table.set_axis(['clients_count','debtors_count'],axis='columns',inplace=True)
pivot_table['clients_count'] = pivot_table['clients_count'] + pivot_table['debtors_count']
pivot_table['debtors_percent'] = pivot_table['debtors_count'] * 100 / pivot_table['clients_count']

print(pivot_table.sort_values(by = 'debtors_percent', ascending = False))
```

```
                    clients_count  debtors_count  debtors_percent
children_count_cat
не многодетные               7266            673         9.262318
многодетные                    50              4         8.000000
нет детей                   14138           1064         7.525817
```

Вывод

Для ответа на вопрос создадим новый датафрейм, в котором будут колонки: название группы клиентов, количество клиентов в данной группе, количество должников, процент должников от общего количества клиентов данной категории.

В результате мы видим, что процент должников отличается между категориями на один-два процента. Учитывая малочисленность группы многодетных, видно, что среди бездетных немного меньше должников чем среди людей с детьми. Кроме этого, мы видим, что многодетные семьи берут кредиты намного реже, чем семьи с меньшим количеством детей и совсем без детей.

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

In [8]:
```python
pivot_table = data.pivot_table('total_income', index='family_status', columns='debt', aggfunc='count')
pivot_table.set_axis(['clients_count','debtors_count'],axis='columns',inplace=True)
pivot_table['clients_count'] = pivot_table['clients_count'] + pivot_table['debtors_count']
pivot_table['debtors_percent'] = pivot_table['debtors_count'] * 100 / pivot_table['clients_count']

print(pivot_table.sort_values(by = 'debtors_percent', ascending = False))
```

```
                       clients_count  debtors_count  debtors_percent
family_status
не женат / не замужем           2810            274         9.750890
гражданский брак                4151            388         9.347145
женат / замужем                12339            931         7.545182
в разводе                       1195             85         7.112971
вдовец / вдова                   959             63         6.569343
```

Вывод

Для поиска ответа на вопрос мы создали новый датафрейм и распредилили клиентов по категориям семейного положения. В результате мы видим, что семейное положение влияет на количество должников. Разница в количестве должников между категориями не более трех процентов, но люди, которые состоят или были в браке больше склонны возвращать долг в срок. Кроме этого, можно сказать что клиенты с семейным положением 'женат / замужем' склонны брать кредит значительно чаще остальных категорий.

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

In [9]:
```python
pivot_table = data.pivot_table('total_income', index='total_income_cat', columns='debt', aggfunc='count')
pivot_table.set_axis(['clients_count','debtors_count'],axis='columns',inplace=True)
pivot_table['clients_count'] = pivot_table['clients_count'] + pivot_table['debtors_count']
pivot_table['debtors_percent'] = pivot_table['debtors_count'] * 100 / pivot_table['clients_count']

print(pivot_table.sort_values(by = 'debtors_percent', ascending = False))
```

```
                  clients_count  debtors_count  debtors_percent
total_income_cat
средний доход              2132            197         9.240150
низкий доход               9491            781         8.228848
высокий доход              9831            763         7.761164
```

Вывод

Аналогично двум предыдущим примерм создадим сводную таблицу для анализа зависимости количества должников и уровня доходов. Видно, что среди людей с высоким доходом наименьшее количество должников, среди людей с низким доходом их чуть больше - разница 0,5%. Самый высокий процент должников среди людей со средним доходом - на 1% больше чем с низким. Можно сказать, что уровень дохода имеет влияние на возврат кредита в срок.

- Как разные цели кредита влияют на его возврат в срок?

In [10]:
```python
pivot_table = data.pivot_table('total_income', index='purpose_categories', columns='debt', aggfunc='count')
pivot_table.set_axis(['clients_count','debtors_count'],axis='columns',inplace=True)
pivot_table['clients_count'] = pivot_table['clients_count'] + pivot_table['debtors_count']
pivot_table['debtors_percent'] = pivot_table['debtors_count'] * 100 / pivot_table['clients_count']

print(pivot_table.sort_values(by = 'debtors_percent', ascending = False))
```

```
                    clients_count  debtors_count  debtors_percent
purpose_categories
автомобиль                   4306            403         9.359034
образование                  4013            370         9.220035
свадьба                      2324            186         8.003442
недвижимость                10204            747         7.320659
ремонт                        607             35         5.766063
```

Вывод

Создав и просмотрев сводную таблицу по кличеству должников в разрезе целей кредита, можно сказать что цели имеют влияние на возврат кредита в срок. Самая безопасная цель - ремонт, а реже всего возвращают в срок кредиты, предназначенные для операций с автомобилями.

Общий вывод

Мы выяснили, что должников меньше всего среди людей, которые ранее бывали в браке с низким доходом, которые берут кредит на ремонт. С другой стороны, больше всего должников среди не женатых / не замужних с детьми, обладающих средним доходом и с целью кредита - операции с автомобилем.

In [ ]: