Исследование надежности заемщиков.

Описание проекта

Заказчик: кредитный отдел банка.

Входные данные от банка: статистика о платёжеспособности клиентов.

Цель исследования: необходимо разобраться, влияет ли семейное положение и количество детей клиента на факт погашения кредита в срок. Результаты исследования будут учтены при построении модели кредитного скоринга — специальной системы, которая оценивает способность потенциального заёмщика вернуть кредит банку.

Навыки и инструменты

Предобработка данных: замена пропусков медианными значениями, модуль числа, изменение типов данных, удаление явных и неявных дубликатов, формирование дополнительных датафреймов, декомпозиция исходного.

Создание функций: категоризация дохода и целей кредита

Итоги исследования

В ходе исследования была проведена следующая преобработка исходных данных:

- в столбцах days_employed и children исправлены отрицательные значения стажа и кол-ва детей;
- найдены пропуски в столбцах days_employed и total_income и заменены на медианные значения в этих столбцах;
- исправлена найденная аномалия в столбце 'children' - выполнена замена значений с 20 на 2;
- изменины типы данных в столбцах total_income и days_employed с вещественного float64 на целочисленный int64;
- удалены неявные дубликаты в столбцах education и family_status
- и затем удалены явные дубликаты в таблице
- выполнено создание таблиц-словарей с уникальными значениями пар education_id - education и family_status_id - family_status;
- исходная таблица была декомпозирована, удалены столбцы education и family_status, по которым были созданы таблицы-словари;

Далее для выполнения целей исследования была прооведена:

- категоризация дохода - выделено 4 категории и создан столбец с категориями дохода total_income_category;
- категоризация целей кредита - выделено 4 категории и создан столбец с категориями purpose_category;

По результатам исследования выявлена зависимость семейного положения и количества детей на факт погашения кредита в срок:

минимум фактов задолженности от всего кол-ва взятых кредитов имеют:

- клиенты с семейным положением "вдовец / вдова" и "женат / замужем" (6,6% и 7,1%)
- клиенты, не имеющие детей (до 8%);

максимум фактов возникновния задолженности имеют:

- клиенты со статусом семейного положения "не женат / не замужем" (9,8%)
- клиенты, имеющих 1-2 детей (9-9,5%).

Детализация исследования

Шаг 1. Обзор данных

Для работы с данными и их обзора импортируем библиотеку pandas, прочитаем данные с помощью функции read_csv() и выведем первые 10 строк таблицы методом head().

Out[1]:

	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	
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080.014102	
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885.952297	

3	3	-4124.747207	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628.550329	до
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	
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	
8	2	-6929.865299	35	ВЫСШЕЕ	0	гражданский брак	1	F	сотрудник	0	95856.832424	
9	0	-2188.756445	41	среднее	1	женат / замужем	0	M	сотрудник	0	144425.938277	

Выведем общую информацию об исходных данных:

```
<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
```

Шаг 2.1 Заполнение пропусков

Пропуски обнаружены в столбцах: days_employed и total_income:

В обоих столбцах пропуски имеют тип NaN, которое замещает отсутствующее в ячейке число и принадлежит к типу float, поэтому с ним можно проводить математические операции.

Посчитаем кол-во пропусков и какую долю составляют пропущенные значения в каждом из столбцов с пропусками:

```
Процент пропусков в столбце days_employed составляет: 10.1
Процент пропусков в столбце total_income составляет: 10.1
```

Причинами пропусков в столбцах могут быть:

- данные были не указаны клиентами,
- пропущены при заполнении сотрудниками,
- либо вызываны технической ошибкой при записи данных в таблицу.

Значения в столбцах days_employed и total_income -являются числовыми и относятся к количественным переменным. В связи с чем заполнять пропуски будем характерными значениями, рассчитаными с помощью медианы.

Медиана является в данном случае более достоверным методом, т.к. значения в выборке как в столбце days_employed, так и total_income могут сильно колебаться и рассчитанное среднее значение может сильно сместиться из за анамальных значений и исказить результаты.

Рассчитаем медианное значение для столбцов days_employed и total_income:

```
Медианное значение для столбца days_employed -1203.4
Медианное значение для столбца total_income 145017.9
```

В результате расчёта по столбцу 'days_employed' (стаж) получили отрицательно медианное значение, что говорит о том, что столбец содержит отрицательные значения, чего в реальности быть не может. Поэтому, заменим отрицательные значения в столбце на модуль числа и пересчитаем медианное значение в столбце 'days_employed':

```
2194.2
```

Заполним пропуски в столбцах найденными медианными значениями и проверим результат- пропусков быть не должно.

Первый способ проверки:

In [8]:
```
data['days_employed'] = data['days_employed'].fillna(value=days_employed_median)
data['days_employed'].isna().sum()
```

Out[8]:
```
0
```

Второй способ убедиться, что пропусков не осталось:

In [9]:
```
data['total_income'] = data['total_income'].fillna(value=total_income_median)
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     21525 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      21525 non-null  float64
 11  purpose           21525 non-null  object 
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

Шаг 2.2 Проверка данных на аномалии и исправления.

Значения в столбцах days_employed и total_income имеют вещественный тип данных в формате float64, т.е. это числа с плавающей точкой и имеющие отрицательный значения.

Столбец days_employed:

1.Отрицательные значения в столбце days_employed - являются аномалией, т.к. общий трудовой стаж отрицательным быть не может и является ошибкой в данных, возникшей, возможно в результате некорректного переноса/записи данных в таблицу.

2.Тажке для этого столбца лишними являются значения после запятой, т.к. столбец показывает стаж в днях и достаточно будет целочисленного значения.

3.В описании столбца написано, что стаж указан в днях, но на деле это не соответсвует действительности, например в строке №5 53 летний мужчина судя по данным о стаже (если значение в днях)- проработал 932 года, чего быть не может.

Столбец total_income:

значения после запятой являются избыточными, т.к. столбец показывает ежемесячный доход и достаточно будет целочисленного значения.

Находим все уникальные значения в столбце gender:

Out[10]:
```
array([ 1,  0,  3,  2, -1,  4, 20,  5])
```

Найдём модуль числа в столбце,чтобы столбец с количеством детей перестал содержать отрицательные значения:

In [11]:
```
data['children'] = abs(data['children'])
```

Посмотрим количество клиентов имеют какое количество детей:

Out[12]:
```
0     14149
1      4865
2      2055
3       330
20       76
4        41
5         9
Name: children, dtype: int64
```

Видим аномалию в количестве 20 детей. Можно сделать предположение, что анамалия появилась в результате опечаток , когда имелось ввиду не 20, а 2 ребенка в семье. Заменим значения с "20" на "2" и выполним проверку:

Шаг 2.3. Изменение типов данных.

Изменим тип данных в столбцах total_income и days_employed с вещественного float64 на целочисленный int64 и проверим, что типы столбцов изменились на int64

```
<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     21525 non-null  int64 
 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      21525 non-null  int64 
 11  purpose           21525 non-null  object
dtypes: int64(7), object(5)
memory usage: 2.0+ MB
```

Посмотрим, как теперь выглядит таблица:

Out[16]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	
0	1	8437	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875	по
1	1	4024	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080	пр
2	0	5623	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885	по
3	3	4124	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628	допо с
4	0	340266	53	среднее	1	гражданский брак	1	F	пенсионер	0	158616	

Шаг 2.4. Удаление дубликатов.

Посчитаем количество явных строк-дубликатов в таблице:

Out[17]:
```
('Дубликатов в таблице:', 54)
```

Удалим строки-дубликаты и заменим индексацию на новую:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21471 entries, 0 to 21470
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype 
```

```
 ---  ------            --------------  -----
  0   children          21471 non-null  int64
  1   days_employed     21471 non-null  int64
  2   dob_years         21471 non-null  int64
  3   education         21471 non-null  object
  4   education_id      21471 non-null  int64
  5   family_status     21471 non-null  object
  6   family_status_id  21471 non-null  int64
  7   gender            21471 non-null  object
  8   income_type       21471 non-null  object
  9   debt              21471 non-null  int64
  10  total_income      21471 non-null  int64
  11  purpose           21471 non-null  object
dtypes: int64(7), object(5)
memory usage: 2.0+ MB
```

Для поиска неявных дубликатов применим метод unique(), чтобы найти все уникальные значения в в столбцах.

Найдём все уникальные значения в столбце ducation.

Out[19]:
```
array(['высшее', 'среднее', 'Среднее', 'СРЕДНЕЕ', 'ВЫСШЕЕ',
       'неоконченное высшее', 'начальное', 'Высшее',
       'НЕОКОНЧЕННОЕ ВЫСШЕЕ', 'Неоконченное высшее', 'НАЧАЛЬНОЕ',
       'Начальное', 'Ученая степень', 'УЧЕНАЯ СТЕПЕНЬ', 'ученая степень'],
      dtype=object)
```

Видим, что неявные дубликаты появились из-за применения разных регистров при написании одних и тех же значений, что могло быть связано с заполением этого столбца в ручную и разными пользователями.

Приведём все значения в столбце education к нижнему регистру и проверим полученный результат:

Out[20]:
```
array(['высшее', 'среднее', 'неоконченное высшее', 'начальное',
       'ученая степень'], dtype=object)
```

Выполним аналогичные действия для столбца family_status:

Out[21]:
```
array(['женат / замужем', 'гражданский брак', 'вдовец / вдова',
       'в разводе', 'Не женат / не замужем'], dtype=object)
```

Out[22]:
```
array(['женат / замужем', 'гражданский брак', 'вдовец / вдова',
       'в разводе', 'не женат / не замужем'], dtype=object)
```

Найдём все уникальные значения в столбце gender:

Out[23]:
```
array(['F', 'M', 'XNA'], dtype=object)
```

Проверим,сколько раз встречается каждое из значений в данных:

In [24]:
```
print('gender=F',len(data[data['gender']=='F']))
```

Out[24]:
```
14189
```

In [25]:
```
print('gender=M',len(data[data['gender']=='M']))
```

Out[25]:
```
7281
```

In [26]:
```
print('gender=XNA',len(data[data['gender']=='XNA']))
```

Out[26]:
```
1
```

Значение 'XNA' встречается в данных 1 раз, вероятно, может быть связано с тем, что данные не были указаны,пропуск был заменен на 'XNA'.

Найдём все уникальные значения в столбце income_type:

Out[27]:
```
array(['сотрудник', 'пенсионер', 'компаньон', 'госслужащий',
       'безработный', 'предприниматель', 'студент', 'в декрете'],
      dtype=object)
```

Найдём все уникальные значения в столбце dob_years:

Out[28]:
```
array([ 0, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
       69, 70, 71, 72, 73, 74, 75])
```

Имеем значение возраста заёмщика равное 0, чего быть не должно и, вероятно, является ошибкой в данных.

Подсчитаем количество строк, в которых возраст заемщика равен 0:

Out[29]:
```
children            101
days_employed       101
dob_years           101
education           101
education_id        101
family_status       101
family_status_id    101
gender              101
income_type         101
debt                101
total_income        101
purpose             101
dtype: int64
```

Заменим нулевые значения в столбце 'dob_years' на медианное значение в столбце:

Out[30]:
```
42.0
```

В столбцах 'gender', 'income_type', 'purpose' приводить значения книжнему регистру не требуется.

Исправление явных дубликатов в столбцах. Удалим явные найденные дубликаты методом drop_duplicates(), обновим индексацию и вновь выполним проверку на дубликаты:

Out[33]:
```
0
```

Шаг 2.5. Формирование дополнительных датафреймов словарей, декомпозиция исходного датафрейма.

Сформируем два дополнительных датафрейма-словаря из исходного датафрейма, в которых:

- каждому уникальному значению из education соответствует уникальное значение education_id — в первом;
- каждому уникальному значению из family_status соответствует уникальное значение family_status_id — во втором

Удалим дубликаты в сформированной таблице-словаре df_education и проверим полученный результат:

Out[35]:

	education	education_id
0	высшее	0
1	среднее	1
2	неоконченное высшее	2
3	начальное	3
4	ученая степень	4

Удалим дубликаты в сформированной таблице-словаре family_status и проверим полученный результат:

Out[36]:

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

Обновим исходный датафрейм, удалив из него столбцы education и family_status, которые теперь находятся в соответсвующих таблицах-словарях:

Out[37]:

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purpose
0	1	8437	42.0	0	0	F	сотрудник	0	253875	покупка жилья
1	1	4024	36.0	1	0	F	сотрудник	0	112080	приобретение автомобиля
2	0	5623	33.0	1	0	M	сотрудник	0	145885	покупка жилья
3	3	4124	32.0	1	0	M	сотрудник	0	267628	дополнительное образование
4	0	340266	53.0	1	1	F	пенсионер	0	158616	сыграть свадьбу

Шаг 2.6. Категоризация дохода.

Напишем функцию, с помощью которой создадим категории на основе диапозонов доходов:

In [38]:

```
def alert_income_category(income):
    if income <= 30000:
        return 'E'
    if income <= 50000:
        return 'D'
    if income <= 200000:
        return 'C'
    if income <= 1000000:
        return 'B'
    return 'A'
#print(alert_income_category(10000)) #выполним проверку работы функции
```

Теперь создадим столбец с категориями дохода и проверим полученный результат (столбец 'total_income_category'):

Out[39]:

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purpose	total_inc
0	1	8437	42.0	0	0	F	сотрудник	0	253875	покупка жилья	
1	1	4024	36.0	1	0	F	сотрудник	0	112080	приобретение автомобиля	
2	0	5623	33.0	1	0	M	сотрудник	0	145885	покупка жилья	
3	3	4124	32.0	1	0	M	сотрудник	0	267628	дополнительное образование	
4	0	340266	53.0	1	1	F	пенсионер	0	158616	сыграть свадьбу	
...	...	...	...	...	...	...	...	...	...	...	
21448	1	4529	43.0	1	1	F	компаньон	0	224791	операции с жильем	
21449	0	343937	67.0	1	0	F	пенсионер	0	155999	сделка с автомобилем	
21450	1	2113	38.0	1	1	M	сотрудник	1	89672	недвижимость	
21451	3	3112	38.0	1	0	M	сотрудник	1	244093	на покупку своего автомобиля	
21452	2	1984	40.0	1	0	F	сотрудник	0	82047	на покупку автомобиля	

21453 rows × 11 columns

Шаг 2.7. Категоризация целей кредита.

Создадим функцию для категоризации целей получения кредита:

In [40]:
```
def alert_purpose(purpose):
    if 'сыграть свадьбу' in purpose or 'на проведение свадьбы' in purpose or'свадьба' in purpose:
        return 'проведение свадьбы'
    if 'приобретение автомобиля'in purpose or'на покупку подержанного автомобиля'in purpose or'на покупку своего
        return 'операции с автомобилем'
    if 'дополнительное образование' in purpose or 'образование' in purpose or'заняться образованием'in purpose or
        return 'получение образования'
    return 'операции с недвижимостью'
#print(alert_purpose('жилье')) #проверим работу функции
```

Теперь создадим столбец с укрупнёнными категориями целей кредита и проверим полученный результат:

Out[41]:

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purpose	total_inc
0	1	8437	42.0	0	0	F	сотрудник	0	253875	покупка жилья	
1	1	4024	36.0	1	0	F	сотрудник	0	112080	приобретение автомобиля	
2	0	5623	33.0	1	0	M	сотрудник	0	145885	покупка жилья	
3	3	4124	32.0	1	0	M	сотрудник	0	267628	дополнительное образование	
4	0	340266	53.0	1	1	F	пенсионер	0	158616	сыграть свадьбу	
...	...	...	...	...	...	...	...	...	...	...	
21448	1	4529	43.0	1	1	F	компаньон	0	224791	операции с жильем	
21449	0	343937	67.0	1	0	F	пенсионер	0	155999	сделка с автомобилем	
21450	1	2113	38.0	1	1	M	сотрудник	1	89672	недвижимость	
21451	3	3112	38.0	1	0	M	сотрудник	1	244093	на покупку своего автомобиля	
21452	2	1984	40.0	1	0	F	сотрудник	0	82047	на покупку автомобиля	

21453 rows × 12 columns

Ответы на вопросы.

Вопрос 1: Есть ли зависимость между количеством детей и возвратом кредита в срок?

Находим все уникальные значения в столбце children:

Out[42]:
```
array([1, 0, 3, 2, 4, 5])
```

Подготовим сводную таблицу, показывающую кол-во вовремя и невовремя отданных кредитов:

Out[43]:

debt	0	1
children		
0	13027.0	1063.0
1	4410.0	445.0
2	1926.0	202.0
3	303.0	27.0
4	37.0	4.0
5	9.0	NaN

С помощью группировки и среднего наидем процент невовремя отданных кредитов в зависимости от кол-ва детеи:

Out[44]:
```
children
4    9.76
2    9.49
1    9.17
3    8.18
0    7.54
5    0.00
Name: debt, dtype: float64
```

Посмотрим ещё раз какое количество клиентов имеют какое количество детей, чтобы учёсть, насколько показательна выборка:

Out[45]:
```
0    14090
1     4855
2     2128
3      330
4       41
5        9
Name: children, dtype: int64
```

Вывод о зависимости задолженности по кредитам от количества детей:

1) наибольший % фактов задолженности от всего кол-ва взятых кредитов у семей, где 4-ро детей - 9,76%, при этом необходимо учитывать, что кол-во таких семей среди клиентов составляет менее 0,5%, в связи с чем выборка не показательна.

2) в пределах 9-9,5% у семей с 1-м, 2-мя;

3) около 8,2% - у семьей с 3-мя детьми,

4) у семей с 5-ю детьми на данный момент данных о задолженоти нет, но выборку также нельзя считать показательной, т.к. доля таких клиентов менее 0,05%.

5) наименьшая вероятность задолженности, у клиентов без детей - в пределах 8%.

Вопрос 2: есть ли зависимость между семейным положением и возвратом кредита в срок?

Посмотрим количество клиентов имеют какое семейное положение:

Out[46]:
```
не женат / не замужем    1
гражданский брак         1
женат / замужем          1
в разводе                1
вдовец / вдова           1
Name: family_status, dtype: int64
```

Подготовим сводную таблицу, показывающую кол-во вовремя и невовремя отданных кредитов в зависимости от семейного положения.

Далее соединим полученную сводную таблицу с ранее созданной таблицей df_family_status.

Далее сделаем группировку по семейному положению, применим метод mean() и сортировку,чтобы получить итоговую таблицу по каждому статусу сколько кредитов было отдано в строк (столбец '0') и сколько с задолженностью (столбец '1').

Out[47]:

family_status_id	0	1
family_status		
женат / замужем	0	11408	931
гражданский брак	1	3762	388
вдовец / вдова	2	896	63
в разводе	3	1110	85
не женат / не замужем	4	2536	274

С помощью группировки и среднего найдём процент не вовремя отданных кредитов в зависимости от семейного положения:

Out[48]:
```
family_status_id
2    6.6
```

```
3    7.1
0    7.5
1    9.3
4    9.8
Name: debt, dtype: float64
```

Зависимость наличия фактов задолженности по кредитам от семейного положения:

в пределах 7% - "вдовец / вдова",

в пределах 7,5% - "женат / замужем" и в "разводе",

в пределах 9-10% - "гражданский брак", "не женат / не замужем"

Вопрос 3: есть ли зависимость между уровнем дохода и возвратом кредита в срок?

Посмотрим как распределено количество клиентов по категориям дохода:

Out[49]:
```
C    16015
B     5041
D      350
A       25
E       22
Name: total_income_category, dtype: int64
```

Подготовим сводную таблицу, показывающую кол-во вовремя и невовремя отданных кредитов в зависимости от категории дохода:

Out[50]:

debt	0	1
total_income_category		
A	23	2
B	4685	356
C	14655	1360
D	329	21
E	20	2

С помощью группировки и среднего найдём процент невовремя отданных кредитов в зависимости от кол-ва детей:

Out[51]:
```
total_income_category
D    6.0
B    7.1
A    8.0
C    8.5
E    9.1
Name: debt, dtype: float64
```

Зависимость наличия фактов задолженности по кредитам от уровня дохода:

в пределах 6% от кол-ва всех взятых кредитов имеют задолженность клиенты из категория D(30001–50000),

в пределах 7-8 % - из категория A(1000001 и выше) и B(200001–1000000),

в пределах 8-9,1% - из категории C(50001–200000) и E (0–30000).

Вопрос 4: как разные цели кредита влияют на его возврат в срок?

Посмотрим количество клиентов имеют какие цели кредита:

Out[52]:
```
операции с недвижимостью    10811
операции с автомобилем       4306
получение образования        4013
проведение свадьбы           2323
Name: purpose_category, dtype: int64
```

Подготовим сводную таблицу, показывающую кол-во вовремя и невовремя отданных кредитов в зависимости от целей кредита:

Out[53]:

debt	0	1
purpose_category		
операции с автомобилем	3903	403
операции с недвижимостью	10029	782
получение образования	3643	370
проведение свадьбы	2137	186

С помощью группировки и среднего найдём процент невовремя отданных кредитов в зависимости от целей кредита:

In [54]:
```
data.groupby('purpose_category')['debt'].mean().sort_values().round(3)*100
```

Out[54]:
```
purpose_category
операции с недвижимостью    7.2
проведение свадьбы          8.0
получение образования       9.2
операции с автомобилем      9.4
Name: debt, dtype: float64
```

Минимальный % фактов задолжности имеют операции с недвижимостью 7,2% от всех всзятых кредитов на данную цель, максимальный % фактов задолженности у операций с автомобилем - 9,4% от всех всзятых кредитов на данную цель.