Лабораторная работа №4 - Модели прогнозирования

!!! Финальное задание находится в конце документа.

Упражнение по Keras API Projectț

Данные

Мы будем использовать подмножество набора данных LendingClub, полученного с Kaggle: https://www.kaggle.com/wordsforthewise/lending-club.

ВНИМАНИЕ: Не загружайте полный архив по ссылке! Мы предоставляем специальную версию этого файла, в которой вам предстоит выполнить дополнительную обработку данных. Оригинальный файл не будет соответствовать инструкциям!

LendingClub — американская платформа для однорангового кредитования (peer-to-peer lending), расположенная в Сан-Франциско, Калифорния. Это был первый сервис, который зарегистрировал свои предложения как ценные бумаги в Комиссии по ценным бумагам и биржам (SEC) и предоставил возможность вторичной торговли займами. LendingClub является крупнейшей в мире платформой однорангового кредитования.

Наша цель

Используя исторические данные о выданных кредитах и информации о том, вернул ли заемщик долг (или дефолтнулся), можем ли мы построить модель, которая предскажет вероятность возврата кредита? Это позволит нам в будущем оценивать новых клиентов и определять, насколько вероятно, что они погасят свой долг.

Колонка "loan_status" содержит метки классов.

Обзор данных

На Kaggle представлено множество наборов данных LendingClub. Здесь представлена информация о данном конкретном наборе данных.

	LoanStatNew	Description
0	loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
1	term	The number of payments on the loan. Values are in months and can be either 36 or 60.
2	int_rate	Interest Rate on the loan
3	installment	The monthly payment owed by the borrower if the loan originates.
4	grade	LC assigned loan grade
5	sub_grade	LC assigned loan subgrade
6	emp_title	The job title supplied by the Borrower when applying for the loan.*
7	emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
8	home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
9	annual_inc	The self-reported annual income provided by the borrower during registration.
10	verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
11	issue_d	The month which the loan was funded
12	loan_status	Current status of the loan
13	purpose	A category provided by the borrower for the loan request.
14	title	The loan title provided by the borrower
15	zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
16	addr_state	The state provided by the borrower in the loan application
17	dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
18	earliest_cr_line	The month the borrower's earliest reported credit line was opened

19	open_acc	The number of open credit lines in the borrower's credit file.
20	pub_rec	Number of derogatory public records
21	revol_bal	Total credit revolving balance
22	revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
23	total_acc	The total number of credit lines currently in the borrower's credit file
24	initial_list_status	The initial listing status of the loan. Possible values are – W, F
25	application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
26	mort_acc	Number of mortgage accounts.
27	pub_rec_bankruptcies	Number of public record bankruptcies

Исходный код

Примечание: Мы также предоставляем информацию о характеристиках данных в файле `.csv` для удобного просмотра в течение всего ноутбука.

In [1]:
```
import pandas as pd
```

In [2]:
```
data_info = pd.read_csv('../DATA/lending_club_info.csv',index_col='LoanStatNew')
```

In [3]:
```
print(data_info.loc['revol_util']['Description'])
```

```
Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving cre
dit.
```

In [4]:
```
def feat_info(col_name):
    print(data_info.loc[col_name]['Description'])
```

In [5]:
```
feat_info('mort_acc')
```

```
Number of mortgage accounts.
```

Загрузка данных и другие импорты

In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# might be needed depending on your version of Jupyter
%matplotlib inline
```

In [7]:
```
df = pd.read_csv('../DATA/lending_club_loan_two.csv')
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
loan_amnt               396030 non-null float64
term                    396030 non-null object
int_rate                396030 non-null float64
installment             396030 non-null float64
grade                   396030 non-null object
sub_grade               396030 non-null object
emp_title               373103 non-null object
emp_length              377729 non-null object
home_ownership          396030 non-null object
annual_inc              396030 non-null float64
verification_status     396030 non-null object
issue_d                 396030 non-null object
loan_status             396030 non-null object
purpose                 396030 non-null object
title                   394275 non-null object
dti                     396030 non-null float64
earliest_cr_line        396030 non-null object
open_acc                396030 non-null float64
pub_rec                 396030 non-null float64
revol_bal               396030 non-null float64
revol_util              395754 non-null float64
total_acc               396030 non-null float64
initial_list_status     396030 non-null object
application_type        396030 non-null object
mort_acc                358235 non-null float64
pub_rec_bankruptcies    395495 non-null float64
address                 396030 non-null object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

Задачи проекта

Выполните следующие задания! Имейте в виду, что обычно есть несколько способов выполнить задачу. Наслаждайтесь процессом!

Раздел 1: Исследовательский анализ данных

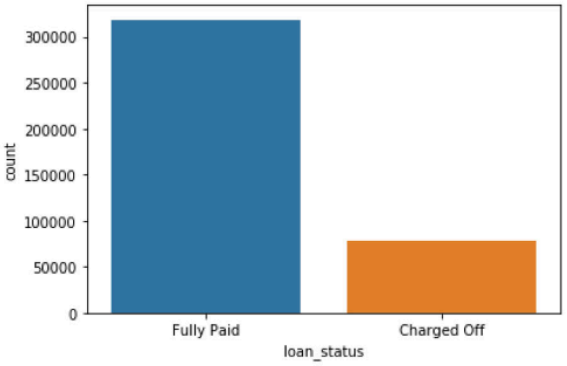
Общая цель: Понять, какие переменные важны, просмотреть сводную статистику и визуализировать данные.

Задание: Поскольку мы будем пытаться предсказать `loan_status`, создайте countplot, как показано ниже.

```
In [9]: # CODE HERE

In [10]:

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x207932022c8>
```

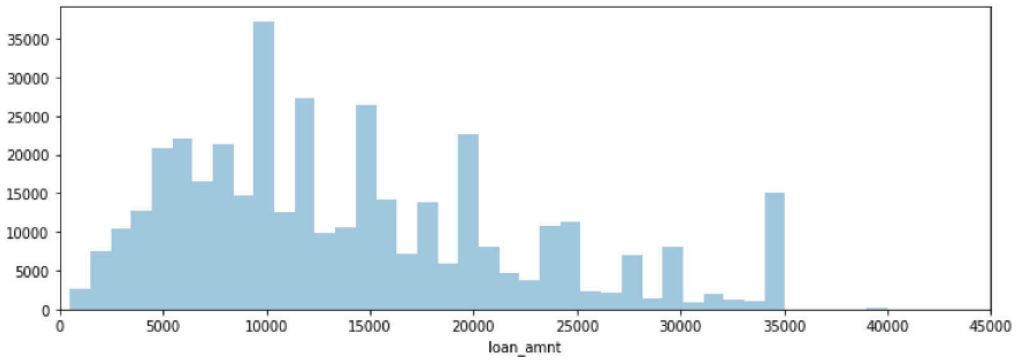


Задание: Создайте гистограмму для столбца `loan_amnt`.

```
In [11]: # CODE HERE

In [12]:

Out[12]: (0, 45000)
```



Задание: Исследуйте корреляцию между непрерывными переменными. Рассчитайте корреляцию между всеми числовыми переменными, используя метод .corr().

In [13]:
```
# CODE HERE
```

In [14]:
```

```

Out[14]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pul
loan_amnt	1.000000	0.168921	0.953929	0.336887	0.016636	0.198556	-0.077779	0.328320	0.099911	0.223886	0.222315	
int_rate	0.168921	1.000000	0.162758	-0.056771	0.079038	0.011649	0.060986	-0.011280	0.293659	-0.036404	-0.082583	
installment	0.953929	0.162758	1.000000	0.330381	0.015786	0.188973	-0.067892	0.316455	0.123915	0.202430	0.193694	
annual_inc	0.336887	-0.056771	0.330381	1.000000	-0.081685	0.136150	-0.013720	0.299773	0.027871	0.193023	0.236320	
dti	0.016636	0.079038	0.015786	-0.081685	1.000000	0.136181	-0.017639	0.063571	0.088375	0.102128	-0.025439	
open_acc	0.198556	0.011649	0.188973	0.136150	0.136181	1.000000	-0.018392	0.221192	-0.131420	0.680728	0.109205	
pub_rec	-0.077779	0.060986	-0.067892	-0.013720	-0.017639	-0.018392	1.000000	-0.101664	-0.075910	0.019723	0.011552	
revol_bal	0.328320	-0.011280	0.316455	0.299773	0.063571	0.221192	-0.101664	1.000000	0.226346	0.191616	0.194925	
revol_util	0.099911	0.293659	0.123915	0.027871	0.088375	-0.131420	-0.075910	0.226346	1.000000	-0.104273	0.007514	
total_acc	0.223886	-0.036404	0.202430	0.193023	0.102128	0.680728	0.019723	0.191616	-0.104273	1.000000	0.381072	
mort_acc	0.222315	-0.082583	0.193694	0.236320	-0.025439	0.109205	0.011552	0.194925	0.007514	0.381072	1.000000	
pub_rec_bankruptcies	-0.106539	0.057450	-0.098628	-0.050162	-0.014558	-0.027732	0.699408	-0.124532	-0.086751	0.042035	0.027239	

Задание: Визуализируйте это с помощью тепловой карты. В зависимости от вашей версии matplotlib, возможно, потребуется вручную настроить отображение.

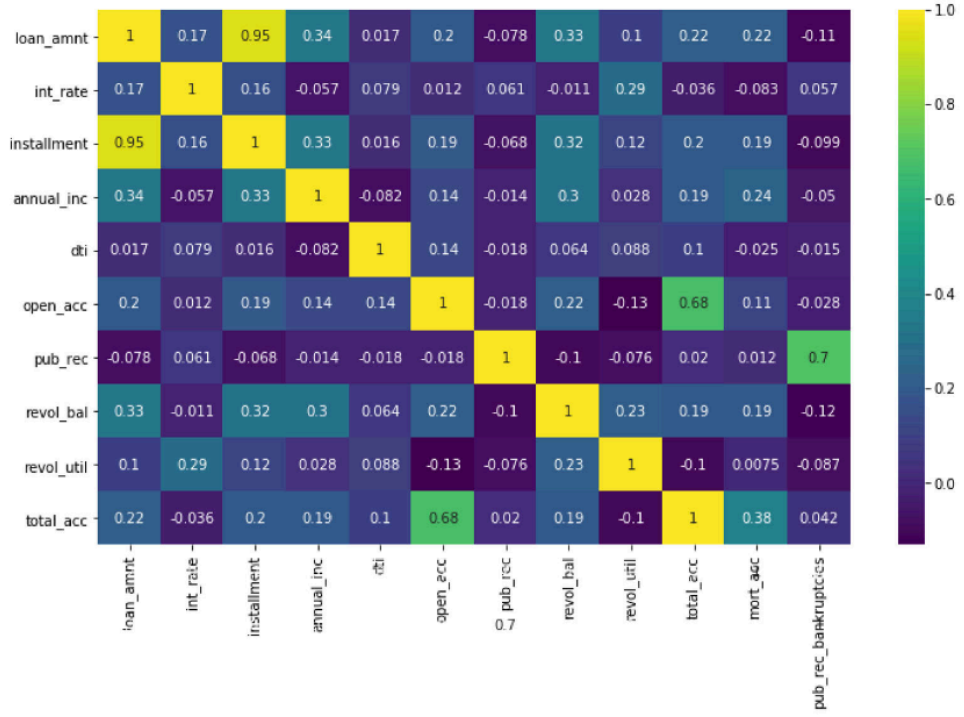
- Информация о тепловых картах
- Помощь по изменению размера: https://stackoverflow.com/questions/56942670/first-and-last-row-cut-in-half-of-heatmap-plot

In [15]:
```
# CODE HERE
```

In [16]:
```

```

Out[16]: (10, 0)



Задание: Вы могли заметить почти идеальную корреляцию с признаком "installment". Исследуйте этот признак подробнее. Выведите их описания и постройте диаграмму рассеяния. Логична ли эта взаимосвязь? Считаете ли вы, что здесь есть дублирование информации?

```
In [17]: # CODE HERE
```

```
In [18]:
```

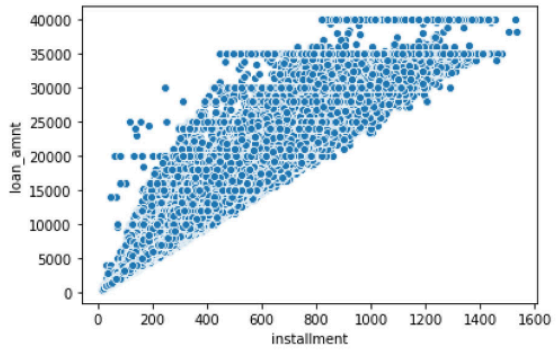
```
The monthly payment owed by the borrower if the loan originates.
```

```
In [19]:
```

```
The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the
loan amount, then it will be reflected in this value.
```

```
In [20]:
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x20798026f48>
```

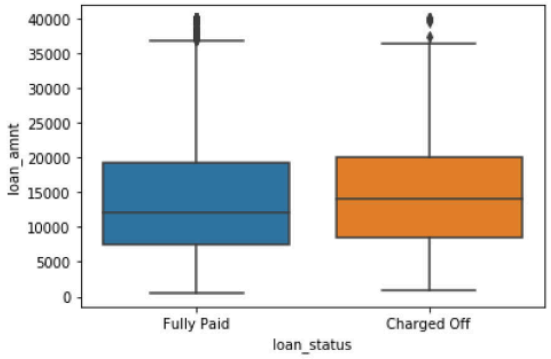


Задание: Создайте boxplot, показывающий взаимосвязь между loan_status и суммой займа.

```
In [21]: # CODE HERE
```

```
In [22]:
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x20798056c48>
```



Задание: Рассчитайте сводную статистику для суммы займа, сгруппированную по loan_status.

```
In [23]: # CODE HERE
```

```
In [24]:
```

```
Out[24]:
```

loan_status	count	mean	std	min	25%	50%	75%	max
Charged Off	77673.0	15126.300967	8505.090557	1000.0	8525.0	14000.0	20000.0	40000.0
Fully Paid	318357.0	13866.878771	8302.319699	500.0	7500.0	12000.0	19225.0	40000.0

Задание: Исследуйте столбцы Grade и SubGrade, которые LendingClub использует для оценки кредитов. Какие возможные уникальные значения существуют?

In [27]:

```
Out[27]: ['A1',
          'A2',
          'A3',
          'A4',
          'A5',
          'B1',
          'B2',
          'B3',
          'B4',
          'B5',
          'C1',
          'C2',
          'C3',
          'C4',
          'C5',
          'D1',
          'D2',
          'D3',
          'D4',
          'D5',
          'E1',
          'E2',
          'E3',
          'E4',
          'E5',
          'F1',
          'F2',
          'F3',
          'F4',
          'F5',
          'G1',
          'G2',
          'G3',
          'G4',
          'G5']
```

In [25]:

```
# CODE HERE
```

In [26]:

```
Out[26]: ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

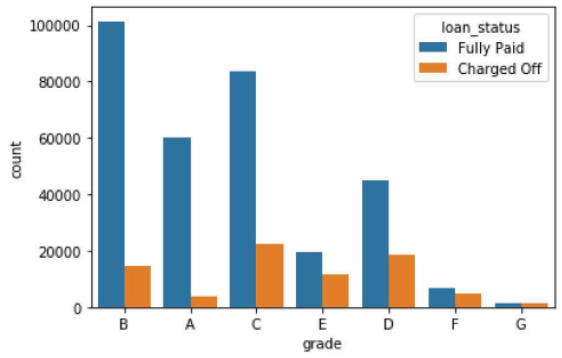
Задание: Создайте countplot для Grade. Установите параметр hue равным loan_status.

In [ ]:

```
# CODE HERE
```

In [28]:

```
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x2078f679ac8>
```



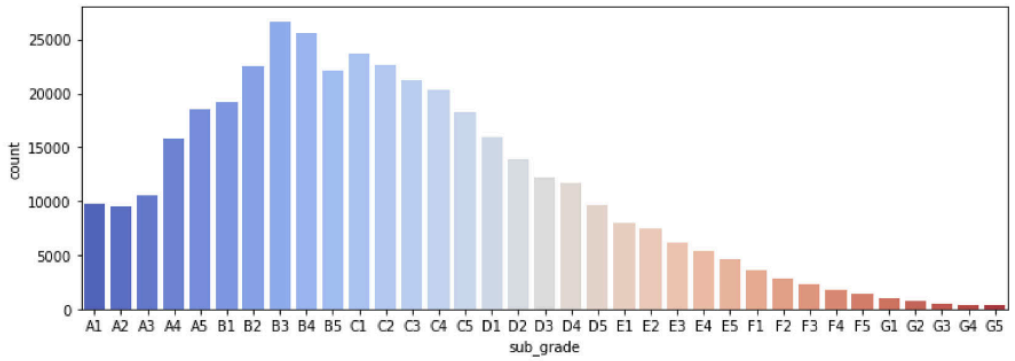
Задание: Постройте countplot для SubGrade. Возможно, потребуется изменить размер графика.

- Информация о countplot
- Изменение порядка оси X

```
In [29]: #CODE HERE
```

```
In [30]:
```

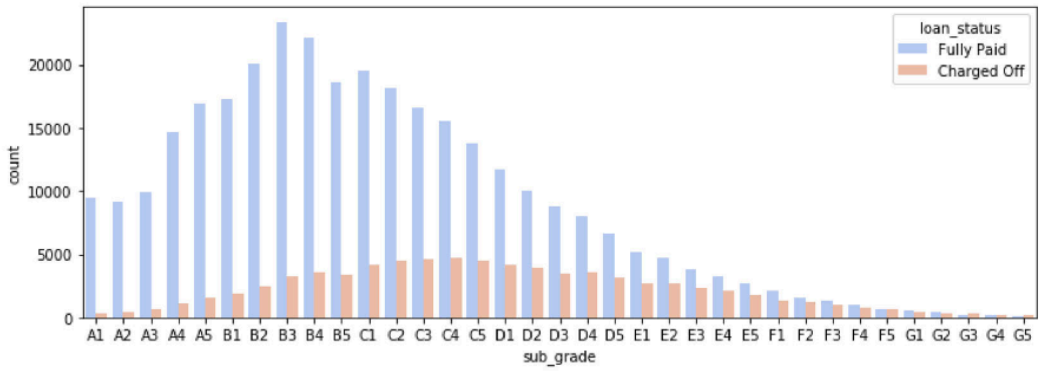
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x20798504288>
```




```
In [31]: # CODE HERE
```

```
In [32]:
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x20798359608>
```

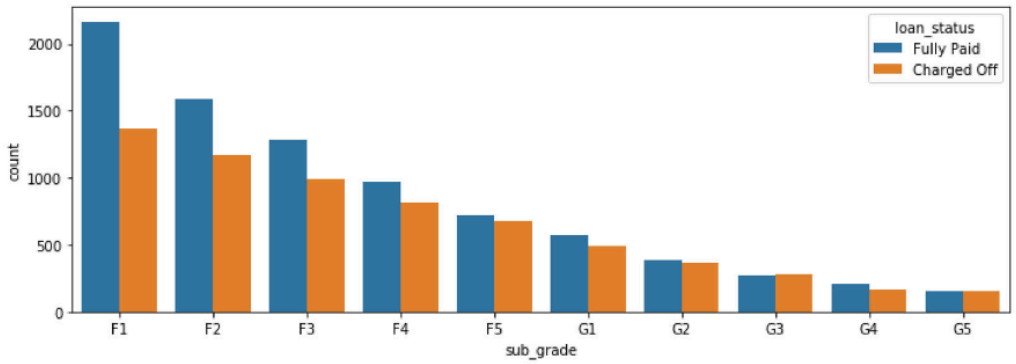


Задание: Похоже, что SubGrade F и G редко выплачиваются. Изолируйте их и создайте countplot только для этих SubGrade.

```
In [33]: # CODE HERE
```

```
In [34]:
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x20795ef7a88>
```



ЗАДАНИЕ: Создайте новый столбец с названием `'loan_repaid'`, который будет содержать `1`, если статус кредита был `"Fully Paid"`, и `0`, если `"Charged Off"`.

ЗАДАНИЕ - ВЫЗОВ: *(Обратите внимание, это сложно, но можно сделать в одной строке!)* Создайте столбчатую диаграмму, показывающую корреляцию числовых признаков с новым столбцом `loan_repaid`.

Раздел 2: Предобработка данных

Цели раздела: Удалить или заполнить отсутствующие данные. Удалить ненужные или повторяющиеся признаки. Преобразовать категориальные строковые признаки в фиктивные переменные.

Отсутствующие данные

Давайте исследуем столбцы с отсутствующими данными. Мы используем различные факторы, чтобы решить, будут ли они полезны, и определить, следует ли их оставить, удалить или заполнить.

ЗАДАНИЕ: Какова длина датафрейма?

```
In [42]: # CODE HERE

In [43]:

Out[43]: 396030
```

ЗАДАНИЕ: Создайте `Series`, отображающий общее количество пропущенных значений в каждом столбце.

```
In [44]: # CODE HERE

In [45]:

Out[45]: loan_amnt                0
         term                     0
         int_rate                 0
         installment              0
         grade                    0
         sub_grade                0
         emp_title            22927
         emp_length           18301
         home_ownership           0
         annual_inc               0
         verification_status      0
         issue_d                  0
         loan_status              0
         purpose                  0
         title                 1755
         dti                      0
         earliest_cr_line         0
         open_acc                 0
         pub_rec                  0
         revol_bal                0
         revol_util             276
         total_acc                0
         initial_list_status      0
         application_type         0
         mort_acc             37795
         pub_rec_bankruptcies   535
         address                  0
         loan_repaid              0
         dtype: int64
```

ЗАДАНИЕ: Преобразуйте этот `Series` в процентное отношение к общему `DataFrame`.

```
In [46]: # CODE HERE
```

```
In [47]:
```

```
Out[47]: loan_amnt              0.000000
         term                   0.000000
         int_rate               0.000000
         installment            0.000000
         grade                  0.000000
         sub_grade              0.000000
         emp_title              5.789208
         emp_length             4.621115
         home_ownership         0.000000
         annual_inc             0.000000
         verification_status    0.000000
         issue_d                0.000000
         loan_status            0.000000
         purpose                0.000000
         title                  0.443148
         dti                    0.000000
         earliest_cr_line       0.000000
         open_acc               0.000000
         pub_rec                0.000000
         revol_bal              0.000000
         revol_util             0.069692
         total_acc              0.000000
         initial_list_status    0.000000
         application_type       0.000000
         mort_acc               9.543469
         pub_rec_bankruptcies   0.135091
         address                0.000000
         loan_repaid            0.000000
         dtype: float64
```

ЗАДАНИЕ: Давайте исследуем `emp_title` и `emp_length`, чтобы определить, можно ли их удалить. Выведите информацию о них, используя функцию `feat_info()` из начала этого ноутбука.

```
In [48]: # CODE HERE
```

```
In [49]:
```

```
The job title supplied by the Borrower when applying for the loan.*


Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or
more years.
```

ЗАДАНИЕ: Сколько уникальных названий рабочих должностей существует?

```
In [50]: # CODE HERE
```

```
In [51]:
```

```
Out[51]: 173105
```

```
In [52]:
```

```
Out[52]: Teacher                          4389
         Manager                          4250
         Registered Nurse                 1856
         RN                               1846
         Supervisor                       1830
                                          ...
         mechanic/lead                       1
         SUPV. MECHINE SHOP                  1
         Mcccd                               1
         Dr. Dennis Norkiewicz DDS           1
         bernie little distributing llc      1
         Name: emp_title, Length: 173105, dtype: int64
```

ЗАДАНИЕ: В реальности слишком много уникальных названий должностей, чтобы пытаться преобразовать их в фиктивные переменные. Давайте удалим столбец `emp_title`.

```
In [53]: # CODE HERE
```

```
In [54]:
```

ЗАДАНИЕ: Постройте график распределения (`countplot`) для столбца `emp_length`.
Вызов: Отсортируйте значения в правильном порядке.

```
In [55]: # CODE HERE
```

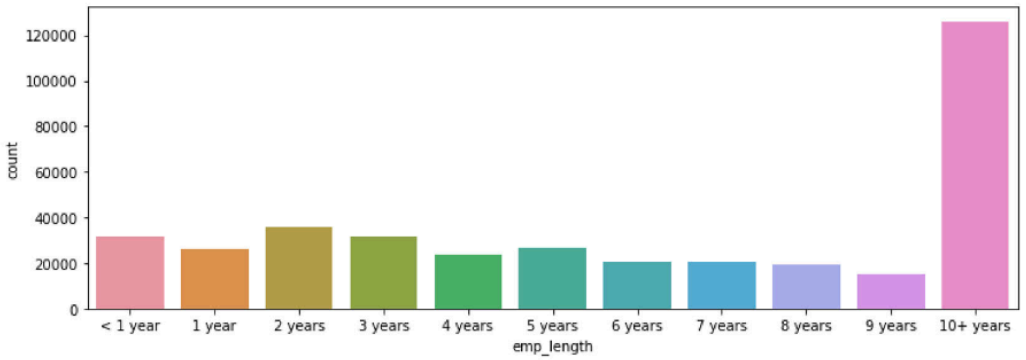
```
In [56]:
```

```
Out[56]: ['1 year',
          '10+ years',
          '2 years',
          '3 years',
          '4 years',
          '5 years',
          '6 years',
          '7 years',
          '8 years',
          '9 years',
          '< 1 year']
```

```
In [57]:
```

```
In [58]:
```

```
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x2079cee4f48>
```

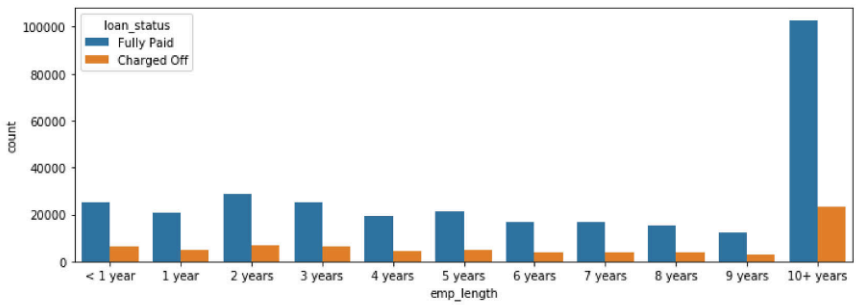


ЗАДАНИЕ: Постройте `countplot` с разделением по оттенку (`hue`) между статусами "Полностью погашен" (`Fully Paid`) и "Списан" (`Charged Off`).

```
In [59]: # CODE HERE
```

```
In [60]:
```

```
Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x20797fc6f48>
```



Challenge ЗАДАНИЕ:

Этот график не дает нам полной информации о том, существует ли сильная связь между продолжительностью занятости (`employment length`) и статусом списания (`charged off`).
Нам необходимо определить процент списанных кредитов в каждой категории, чтобы понять, какая доля людей в каждой категории занятости не выплатила заем.

Существует множество способов создания этой `Series`.
После ее создания попробуйте визуализировать ее с помощью `bar plot` (ссылка).

Это может быть непросто, поэтому обратитесь к решениям, если застрянете на создании этой `Series`.

```
In [61]: # CODE HERE
```

```
In [62]:
```

```
In [63]:
```

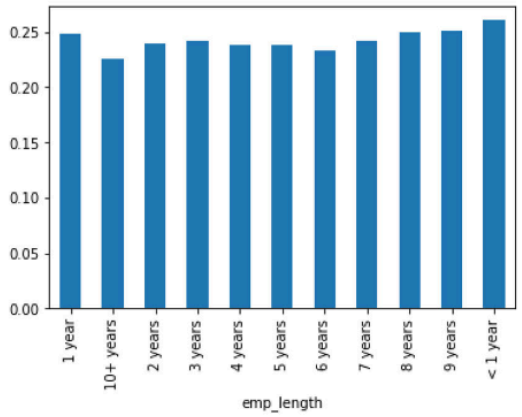
```
In [64]:
```

```
In [65]:
```

```
Out[65]: emp_length
         1 year       0.248649
         10+ years    0.225770
         2 years      0.239560
         3 years      0.242593
         4 years      0.238213
         5 years      0.237911
         6 years      0.233341
         7 years      0.241887
         8 years      0.249625
         9 years      0.250735
         < 1 year     0.260830
         Name: loan_status, dtype: float64
```

```
In [66]:
```

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x20798297d88>
```



ЗАДАНИЕ: Уровень списаний (`charge off rates`) практически одинаков во всех категориях продолжительности занятости. Удалите столбец `emp_length`.

```
In [67]: # CODE HERE
```

```
In [68]:
```

ЗАДАНИЕ: Проверьте `DataFrame`, чтобы увидеть, какие столбцы признаков (`feature columns`) все еще содержат пропущенные данные.

In [ ]:

In [69]:

Out[69]:
```
loan_amnt                 0
term                      0
int_rate                  0
installment               0
grade                     0
sub_grade                 0
home_ownership            0
annual_inc                0
verification_status       0
issue_d                   0
loan_status               0
purpose                   0
title                  1755
dti                       0
earliest_cr_line          0
open_acc                  0
pub_rec                   0
revol_bal                 0
revol_util              276
total_acc                 0
initial_list_status       0
application_type          0
mort_acc              37795
pub_rec_bankruptcies    535
address                   0
loan_repaid               0
dtype: int64
```

ЗАДАЧА: Сравните столбцы `title` и `purpose`. Это дублирующаяся информация?

In [70]:
```
# CODE HERE
```

In [71]:

Out[71]:
```
0             vacation
1    debt_consolidation
2           credit_card
3           credit_card
4           credit_card
5    debt_consolidation
6      home_improvement
7           credit_card
8    debt_consolidation
9    debt_consolidation
Name: purpose, dtype: object
```

In [72]:
```
df['title'].head(10)
```

Out[72]:
```
0                   Vacation
1         Debt consolidation
2    Credit card refinancing
3    Credit card refinancing
4      Credit Card Refinance
5         Debt consolidation
6           Home improvement
7       No More Credit Cards
8         Debt consolidation
9         Debt Consolidation
Name: title, dtype: object
```

ЗАДАЧА: Столбец `title` — это просто текстовая подкатегория/описание столбца `purpose`. Удалите `title`.

In [73]:
```
# CODE HERE
```

In [74]:
```

```

ПРИМЕЧАНИЕ: Это одна из самых сложных частей проекта! Обратитесь к видео с решениями, если вам нужна помощь, или свободно заполняйте/удаляйте пропущенные значения в `mort_acc` на своё усмотрение! Здесь мы используем очень специфический подход.

ЗАДАЧА: Узнайте, что представляет собой характеристика `mort_acc`.

In [75]:
```
# CODE HERE
```

In [76]:
```

```

```
Number of mortgage accounts.
```

ЗАДАЧА: Создайте `value_counts` для столбца `mort_acc`.

In [77]:
```
# CODE HERE
```

In [78]:
```

```

Out[78]:
```
0.0     139777
1.0      60416
2.0      49948
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
23.0         2
32.0         2
26.0         2
31.0         2
30.0         1
28.0         1
34.0         1
Name: mort_acc, dtype: int64
```

ЗАДАЧА: Существует множество способов обработки отсутствующих данных. Можно попытаться создать простую модель для их заполнения, например линейную модель, можно просто заполнить их на основе среднего значения других столбцов, а можно даже разбить столбцы на категории и затем задать NaN как отдельную категорию. Нет 100% правильного подхода! Давайте рассмотрим другие столбцы, чтобы определить, какой из них наиболее сильно коррелирует с `mort_acc`.

In [ ]:
```

```

In [79]:

```
Correlation with the mort_acc column
```

Out[79]:

```
int_rate              -0.082583
dti                   -0.025439
revol_util             0.007514
pub_rec                0.011552
pub_rec_bankruptcies   0.027239
loan_repaid            0.073111
open_acc               0.109205
installment            0.193694
revol_bal              0.194925
loan_amnt              0.222315
annual_inc             0.236320
total_acc              0.381072
mort_acc               1.000000
Name: mort_acc, dtype: float64
```

ЗАДАЧА: Похоже, что признак `total_acc` коррелирует с `mort_acc`, что логично! Давайте попробуем заполнить пропущенные значения с помощью метода `fillna()`. Мы сгруппируем DataFrame по `total_acc` и вычислим среднее значение `mort_acc` для каждого уникального `total_acc`.

In [ ]:

In [80]:

```
Mean of mort_acc column per total_acc
```

Out[80]:

```
total_acc
2.0      0.000000
3.0      0.052023
4.0      0.066743
5.0      0.103289
6.0      0.151293
           ...
124.0    1.000000
129.0    1.000000
135.0    3.000000
150.0    2.000000
151.0    0.000000
Name: mort_acc, Length: 118, dtype: float64
```

Заполним отсутствующие значения mort_acc на основе их total_acc значения. Если mort_acc отсутствует, то мы заполним это значение средним значением, соответствующим total_acc из ранее созданного нами Series. Это включает использование метода .apply() с двумя столбцами. Ознакомьтесь со ссылкой ниже для получения дополнительной информации или просмотрите обучающее видео/ноутбук.

Полезная ссылка:
https://stackoverflow.com/questions/13331698/how-to-apply-a-function-to-two-columns-of-pandas-dataframe

In [81]:

```
# CODE HERE
```

In [82]:

In [83]:

Out[83]:

```
0.0
```

In [84]:

In [85]:

In [86]:

```
Out[86]: loan_amnt               0
         term                    0
         int_rate                0
         installment             0
         grade                   0
         sub_grade               0
         home_ownership          0
         annual_inc              0
         verification_status     0
         issue_d                 0
         loan_status             0
         purpose                 0
         dti                     0
         earliest_cr_line        0
         open_acc                0
         pub_rec                 0
         revol_bal               0
         revol_util            276
         total_acc               0
         initial_list_status     0
         application_type        0
         mort_acc                0
         pub_rec_bankruptcies  535
         address                 0
         loan_repaid             0
         dtype: int64
```

ЗАДАЧА: В столбцах revol_util и pub_rec_bankruptcies имеются пропущенные значения, но их доля составляет менее 0,5% от общего объема данных. Удалите строки с пропущенными значениями в этих столбцах с помощью dropna().

In [87]:

```
# CODE HERE
```

In [88]:

In [89]:

```
Out[89]: loan_amnt              0
         term                   0
         int_rate               0
         installment            0
         grade                  0
         sub_grade              0
         home_ownership         0
         annual_inc             0
         verification_status    0
         issue_d                0
         loan_status            0
         purpose                0
         dti                    0
         earliest_cr_line       0
         open_acc               0
         pub_rec                0
         revol_bal              0
         revol_util             0
         total_acc              0
```

Категориальные переменные и фиктивные (dummy) переменные

Мы закончили работу с пропущенными данными! Теперь нам нужно обработать строковые значения в категориальных столбцах.

ЗАДАЧА: Выведите список всех столбцов, которые в настоящее время не являются числовыми. Полезная ссылка: https://stackoverflow.com/questions/22470690/get-list-of-pandas-dataframe-columns-based-on-data-type

In [90]:

```
# CODE HERE
```

In [91]:

```
Out[91]: Index(['term', 'grade', 'sub_grade', 'home_ownership', 'verification_status',
                'issue_d', 'loan_status', 'purpose', 'earliest_cr_line',
                'initial_list_status', 'application_type', 'address'],
               dtype='object')
```

Преобразование переменной `term`

ЗАДАЧА: Преобразуйте переменную `term` в целочисленный тип данных (36 или 60 месяцев) с помощью `.apply()` или `.map()`.

In [92]:

```
# CODE HERE
```

In [93]:

```
Out[93]:  36 months    301247
          60 months     93972
         Name: term, dtype: int64
```

In [94]:

Удаление переменной `grade`

ЗАДАЧА: Переменная grade уже включена в sub_grade, поэтому просто удалите grade.

In [95]:

```
# CODE HERE
```

In [96]:

ЗАДАЧА: Преобразуйте subgrade в фиктивные переменные и добавьте их в исходный DataFrame. Не забудьте удалить оригинальный столбец subgrade и установить drop_first=True в get_dummies().

In [97]:

```
# CODE HERE
```

In [98]:

In [99]:

In [100]:

```
Out[100]: Index(['loan_amnt', 'term', 'int_rate', 'installment', 'home_ownership',
                 'annual_inc', 'verification_status', 'issue_d', 'loan_status',
                 'purpose', 'dti', 'earliest_cr_line', 'open_acc', 'pub_rec',
                 'revol_bal', 'revol_util', 'total_acc', 'initial_list_status',
                 'application_type', 'mort_acc', 'pub_rec_bankruptcies', 'address',
                 'loan_repaid', 'A2', 'A3', 'A4', 'A5', 'B1', 'B2', 'B3', 'B4', 'B5',
                 'C1', 'C2', 'C3', 'C4', 'C5', 'D1', 'D2', 'D3', 'D4', 'D5', 'E1', 'E2',
                 'E3', 'E4', 'E5', 'F1', 'F2', 'F3', 'F4', 'F5', 'G1', 'G2', 'G3', 'G4',
                 'G5'],
                dtype='object')
```

In [101]:

```
Out[101]: Index(['home_ownership', 'verification_status', 'issue_d', 'loan_status',
                 'purpose', 'earliest_cr_line', 'initial_list_status',
                 'application_type', 'address'],
                dtype='object')
```

Обработка переменных verification_status, application_type, initial_list_status, purpose

ЗАДАЧА: Преобразуйте столбцы verification_status, application_type, initial_list_status, purpose в фиктивные переменные и добавьте их в исходный DataFrame. Установите drop_first=True и удалите исходные столбцы.

```
In [102]: # CODE HERE

In [103]:

In [ ]:
```

Анализ переменной home_ownership

ЗАДАЧА: Проверьте количество уникальных значений в home_ownership с помощью value_counts().

```
In [104]: #CODE HERE

In [105]:
```

```
Out[105]: MORTGAGE    198022
          RENT        159395
          OWN          37660
          OTHER          110
          NONE            29
          ANY              3
          Name: home_ownership, dtype: int64
```

ЗАДАЧА: Преобразуйте home_ownership в фиктивные переменные, но сначала замените NONE и ANY на OTHER, чтобы осталось только 4 категории: MORTGAGE, RENT, OWN, OTHER. Затем добавьте их в исходный DataFrame, установив drop_first=True и удалив исходный столбец.

```
In [106]: #CODE HERE

In [107]:
```

address

ЗАДАЧА: Давайте создадим новый столбец zip_code, который извлечет почтовый индекс из столбца address в наборе данных.

```
In [108]: #CODE HERE

In [109]:
```

ЗАДАЧА: Преобразуйте новый столбец `zip_code` в фиктивные (dummy) переменные с помощью `pandas`. Объедините результат с исходными данными и удалите оригинальные столбцы `zip_code` и `address`.

```
In [ ]:

In [110]:
```

issue_d

ЗАДАЧА: Это приведет к утечке данных, так как при обучении модели мы заранее не знаем, будет ли выдан займ. Теоретически, у нас не должно быть столбца `issue_date`, поэтому удалите этот признак.

```
In [111]: #CODE HERE

In [112]:
```

earliest_cr_line

ЗАДАЧА: Этот признак представляет собой историческую метку времени. Извлеките год из этого признака, используя функцию .apply, затем преобразуйте его в числовой признак. Запишите новое значение в столбец earliest_cr_year. После этого удалите признак earliest_cr_line.

```
In [113]: #CODE HERE

In [114]:

In [115]:
Out[115]: Index(['loan_status'], dtype='object')
```

Train Test Split

ЗАДАЧА: Импортируйте train_test_split из sklearn.

```
In [116]:
```

ЗАДАЧА: Удалите столбец load_status, который мы создали ранее, так как это дубликат столбца loan_repaid. Мы будем использовать loan_repaid, так как он уже представлен в виде 0 и 1.

```
In [1]: # CODE HERE

In [118]:
```

ЗАДАЧА: Установите переменные `X` и `y` в `.values` признаков и меток.

```
In [119]: #CODE HERE

In [120]:
```

НЕОБЯЗАТЕЛЬНО

Выборка данных для ускорения обучения

НЕОБЯЗАТЕЛЬНО: Используйте `.sample()` для выбора подвыборки из 490K+ записей, чтобы сократить время обучения.

Настоятельно рекомендуется для компьютеров с малым объемом RAM или если вы не используете GPU.

```
In [121]: # df = df.sample(frac=0.1,random_state=101)
          print(len(df))

          395219
```

ЗАДАНИЕ: Выполните разбиение данных на тренировочный и тестовый набор (`train/test split`) с `test_size=0.2` и `random_state=101`.

In [122]:
```
#CODE HERE
```

In [123]:
```

```

Нормализация данных

ЗАДАНИЕ: Используйте `MinMaxScaler` для нормализации данных `X_train` и `X_test`. Помните, что мы не должны допускать утечки данных (`data leakage`) из тестового набора, поэтому подгонку (`fit`) выполняем только на `X_train`.

In [124]:
```
# CODE HERE
```

In [125]:
```

```

In [126]:
```

```

In [127]:
```

```

In [128]:
```

```

Создание модели

ЗАДАНИЕ: Запустите ячейку ниже, чтобы импортировать необходимые функции Keras.

In [129]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout
```

ЗАДАНИЕ: Постройте последовательную модель (`Sequential`), которая будет обучаться на данных. У вас есть неограниченные возможности, но вот предложенное решение:
Модель с архитектурой 78 → 39 → 19 → 1 выходной нейрон.

Дополнительно: Исследуйте добавление слоев `Dropout`:

- Dropout layers
- Wikipedia: Dropout (нейросети) https://en.wikipedia.org/wiki/Dilution_(neural_networks)
- Подробное объяснение Dropout

In [130]:
```
# CODE HERE
model = Sequential()

# Choose whatever number of layers/neurons you want.

# https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-ne

# Remember to compile()
```

In [131]:
```

```

ЗАДАНИЕ: Обучите модель на тренировочных данных не менее 25 эпох.

Также добавьте валидационные данные для последующей визуализации.

Дополнительно: можно задать `batch_size=256`.

In [132]:
```
# CODE HERE
```

In [133]:
```
Train on 316175 samples, validate on 79044 samples
Epoch 1/25
316175/316175 [==============================] - 4s 13us/sample - loss: 0.2959 - val_loss: 0.2652
Epoch 2/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2652 - val_loss: 0.2643
Epoch 3/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2628 - val_loss: 0.2626
Epoch 4/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2613 - val_loss: 0.2621
Epoch 5/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2609 - val_loss: 0.2621
Epoch 6/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2603 - val_loss: 0.2618
Epoch 7/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2600 - val_loss: 0.2616
Epoch 8/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2595 - val_loss: 0.2616
Epoch 9/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2593 - val_loss: 0.2620
Epoch 10/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2589 - val_loss: 0.2609
Epoch 11/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2588 - val_loss: 0.2613
Epoch 12/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2584 - val_loss: 0.2607
Epoch 13/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2581 - val_loss: 0.2613
Epoch 14/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2580 - val_loss: 0.2605
Epoch 15/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2580 - val_loss: 0.2607
Epoch 16/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2574 - val_loss: 0.2609
Epoch 17/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2575 - val_loss: 0.2606
Epoch 18/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2573 - val_loss: 0.2614
Epoch 19/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2572 - val_loss: 0.2611
Epoch 20/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2567 - val_loss: 0.2606
Epoch 21/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2569 - val_loss: 0.2606
Epoch 22/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2565 - val_loss: 0.2608
Epoch 23/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2564 - val_loss: 0.2612
Epoch 24/25
316175/316175 [==============================] - 3s 10us/sample - loss: 0.2561 - val_loss: 0.2609
Epoch 25/25
316175/316175 [==============================] - 3s 11us/sample - loss: 0.2560 - val_loss: 0.2612
```

Out[133]:
```
<tensorflow.python.keras.callbacks.History at 0x20a2a8474c8>
```

ЗАДАНИЕ (НЕОБЯЗАТЕЛЬНО): Сохраните вашу модель.

In [134]:
```
# CODE HERE
```

In [135]:
```
```

In [136]:
```
```

Раздел 3: Оценка производительности модели.

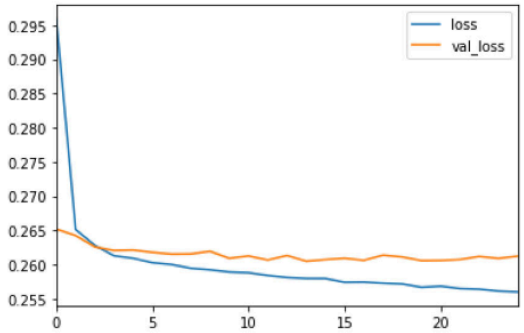
ЗАДАНИЕ: Постройте график сравнения потерь на валидационных данных и потерь на обучающей выборке.

```
In [137]: # CODE HERE
```

```
In [138]:
```

```
In [139]:
```

```
Out[139]: <matplotlib.axes._subplots.AxesSubplot at 0x20a2cf62f48>
```



ЗАДАНИЕ: Создайте предсказания для тестового набора `X_test` и отобразите отчет о классификации (`classification report`) и матрицу ошибок (`confusion matrix`) для `X_test`.

```
In [140]: # CODE HERE
```

```
In [141]:
```

```
In [142]:
```

```
In [143]:
```

```
              precision    recall  f1-score   support

           0       0.99      0.44      0.61     15658
           1       0.88      1.00      0.93     63386

    accuracy                           0.89     79044
   macro avg       0.93      0.72      0.77     79044
weighted avg       0.90      0.89      0.87     79044
```

```
In [144]:
```

```
Out[144]: array([[ 6850,  8808],
                 [  100, 63286]], dtype=int64)
```

ЗАДАНИЕ: Исходя из информации о клиенте ниже, вы бы выдали этому человеку кредит?

In [145]:
```
import random
random.seed(101)
random_ind = random.randint(0,len(df))

new_customer = df.drop('loan_repaid',axis=1).iloc[random_ind]
new_customer
```

Out[145]:
```
loan_amnt           25000.00
term                    6.00
int_rate               18.24
installment           638.11
annual_inc          61665.00
                       ...
48052                   0.00
70466                   0.00
86630                   0.00
93700                   0.00
earliest_cr_year     1996.00
Name: 305323, Length: 78, dtype: float64
```

In [146]:
```
# CODE HERE
```

In [147]:
```

```

Out[147]:
```
array([[1]])
```

ЗАДАНИЕ: Теперь проверьте, выплатил ли этот человек в итоге свой кредит?

In [148]:
```
# CODE HERE
```

In [149]:
```

```

Out[149]:
```
1.0
```

 # ЗАДАНИЕ: Теперь выберите другой набор данных и выполните бинарную или многоклассовую классификацию с использованием искусственной нейронной сети. Следуйте основным шагам, описанным выше.