СКБ201 Тур

Модуль синтеза речи или модуль искусственного интелекта

Тестовое задание на роль: ML-разработчик

Навигация

*в версии ".ipynb" через github она не работает корректно

- Модуль искусственного интелекта
  - Задание 1
- Модуль искусственного интелекта и модуль синтеза речи
  - Задание 1
  - Задание 2
  - Задание 3
  - Задание 4
  - Задание 5 (основное задание)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Модуль искусственного интелекта

1. Напишите функцию, возвращающую округленную взвешенную сумму оценок по данным оценкам и весам

In [2]:
```
def result_mark(weights: np.array, marks: np.array) -> int:
    # я не совсем уверен, если нужны все эти проверки, так что 2 версии
    if len(weights.shape)!=1 or len(marks.shape)!=1 or \
        len(weights)!=len(marks) or  sum(weights)!=1 or \
        True in (marks>10) or True in (marks<0):
            raise ValueError
    return (weights@(marks.reshape(-1,1)))[0]

def result_mark_fast(weights: np.array, marks: np.array) -> int:
    return (weights@(marks.reshape(-1,1)))[0]
```

In [3]:
```
marks = np.arange(1,11)
weights = np.arange(1,11)/sum(marks)

%timeit result_mark(weights, marks)
%timeit result_mark_fast(weights, marks)
result_mark(weights, marks), result_mark_fast(weights, marks)
```

```
13.9 µs ± 414 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
2.28 µs ± 187 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

Out[3]:
```
(7.0, 7.0)
```

Модуль искусственного интелекта и модуль синтеза речи

1. Прочитайте средствами pandas файл с данными по ссылке. Выведите базовое представление таблицы (первые 5 и последние 5 строк, размер).

In [4]:
```
url = 'https://raw.githubusercontent.com/hse-ds/iad-intro-ds/master/2022/homeworks/hw03-EDA/german_credit.csv'
df0 = pd.read_csv(url)
df0
```

Out[4]:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installn
0	no checking account	18	all credits at this bank paid back duly	car (used)	1049	unknown/no savings account	< 1 yr	
1	no checking account	9	all credits at this bank paid back duly	others	2799	unknown/no savings account	1 <= ... < 4 yrs	25 <
2	... < 0 DM	12	no credits taken/all credits paid back duly	retraining	841	... < 100 DM	4 <= ... < 7 yrs	25 <
3	no checking account	12	all credits at this bank paid back duly	others	2122	unknown/no savings account	1 <= ... < 4 yrs	20 <
4	no checking account	12	all credits at this bank paid back duly	others	2171	unknown/no savings account	1 <= ... < 4 yrs	
...	...	...	...	...	...	...	...	
995	no checking account	24	no credits taken/all credits paid back duly	furniture/equipment	1987	unknown/no savings account	1 <= ... < 4 yrs	25 <
996	no checking account	24	no credits taken/all credits paid back duly	others	2303	unknown/no savings account	>= 7 yrs	
997	... >= 200 DM / salary for at least 1 year	21	all credits at this bank paid back duly	others	12680	... >= 1000 DM	>= 7 yrs	
998	... < 0 DM	12	no credits taken/all credits paid back duly	furniture/equipment	6468	... >= 1000 DM	unemployed	25 <
999	no checking account	30	no credits taken/all credits paid back duly	car (used)	6350	... >= 1000 DM	>= 7 yrs	

1000 rows × 21 columns

2. Определите, есть ли пропуски в данных. Разделите признаки на количественные, бинарные, порядковые и категориальные.

In [5]:
```
np.nan in df0, None in df0
```

Out[5]:
```
(False, False)
```

In [6]:
```
for i in df0.columns:
    print('->', i, ':', df0[i].unique(), '\n')
```

```
-> status : ['no checking account' '... < 0 DM'
 '... >= 200 DM / salary for at least 1 year' '0<= ... < 200 DM']

-> duration : [18  9 12 10  8  6 24 11 30 48 36 15 42 21 27 33 28  4 47 14 39 60
  5 22
 54 13 16  7 20 26 45 72 40]

-> credit_history : ['all credits at this bank paid back duly'
 'no credits taken/all credits paid back duly'
 'existing credits paid back duly till now'
 'delay in paying off in the past'
 'critical account/other credits elsewhere']

-> purpose : ['car (used)' 'others' 'retraining' 'furniture/equipment' 'car (ne
w)'
 'business' 'domestic appliances' 'radio/television' 'repairs' 'vacation']

-> amount : [ 1049  2799   841  2122  2171  2241  3398  1361  1098  3758  3905
6187
  1957  7582  1936  2647  3939  3213  2337  7228  3676  3124  2384  1424
  4716  4771   652  1154  3556  4796  3017  3535  6614  1376  1721   860
  1495  1934  3378  3868   996  1755  1028  2825  1239  1216  1258  1864
  1474  1382   640  3919  1224  2331  6313   385  1655  1053  3160  3079
  1163  2679  3578 10875  1344  1237  3077  2284  1567  2032  2745  1867
  2299   929  3399  2030  3275  1940  1602  1979  2022  3342  5866  2360
  1520  3651  2346  4454   666  1965  1995  2991  4221  1364  6361  4526
  3573  4455  2136  5954  3777   806  4712  7432  1851  1393  1412  1473
  1533  2012  3959   428  2366   763  3976  6260  1919  2603   936  3062
  4795  5842  2063  1459  1213  5103   874  2978  1820  2872  1925  2515
  2116  1453  1543  1318  2325   932  3148  3835  3832  5084  2406  2394
  2476  2964  1262  1542  1743   409  8858  3512  1158  2684  1498  6416
  3617  1291  1275  3972  3343   392  2134  5771  2788  5848  1228  1297
  1552  1963  3235  4139  1804  1950 12749  1236  1055  8072  2831  1449
  5742  2390  3430  2273  2923  1901  3711  8487  2255  7253  6761  1817
  2141  3609  2333  7824  1445  7721  3763  4439  1107  1444 12169  2753
  1494  2828  2483  1299  1549  3949  2901   709 10722  1287  3656  4679
  8613  2659  1516  4380   802  1572  3566  1278   426  8588  3857   685
  1603   601  2569  1316 10366  1568   629  1750  3488  1800  4151  2631
  5248  2899  6204   804  3595  5711  2687  3643  2146  2315  3448  2708
  1313  1493  2675  2118   909  1569  7678   660  2835  2670  3447  3568
  3652  3660  1126   683  2251  4675  2353  3357   672   338  2697  2507
  1478  3565  2221  1898   960  8133  2301   983  2320   339  5152  3749
  3074   745  1469  1374   783  2606  9436   930  2751   250  1201   662
  1300  1559  3016  1360  1204  1597  2073  2142  2132  1546  1418  1343
  2662  6070  1927  2404  1554  1283   717  1747  1288  1038  2848  1413
  3632  3229  3577   682  1924   727   781  2121   701  2069  1525  7629
  3499  1346 10477  2924  1231  1961  5045  1255  1858  1221  1388  2279
  2759  1410  1403  3021  6568  2578  7758   343  1591  3416  1108  5965
  1514  6742  3650  3599 13756   276  4041   458   918  7393  1225  2812
  3029  1480  1047  1471  5511  1206  6403   707  1503  6078  2528  1037
  1352  3181  4594  5381  4657  1391  1913  7166  1409   976  2375   522
  2743  5804  1169   776  1322  1175  2133  1829 11760  1501  1200  3195
  4530  1555  2326  1887  1264   846  1532   935  2442  3590  2288  5117
 14179  1386   618  1574   700   886  4686   790   766  2212  7308  5743
  3973  7418  2629  1941  2445  6468  7374  3812  4006  7472  2028  5324
  2323  1984   999  7409  2186  4473   937  3422  3105  2748  3872  5190
  3001  3863  5801  1592  1185  3780  3612  1076  3527  2051  3331  3104
  2611  1311  2108  4042   926  1680  1249  2463  1595  2058  7814  1740
  1240  6842  5150  1203  2080  1538  3878  3186  2896  6967  1819  5943
  7127  3349 10974   518  1860  9566  2930  1505  2238  2197  1881  1880
  2389  1967  3380  1455   730  3244  1670  3979  1922  1295  1544   907
```

```
  1715  1347  1007  1402  2002  2096  1101   894  1577  2764  8358  5433
  3485  3850  7408  1377  4272  1553  9857   362  1935 10222  1330  9055
  7966  3496  6948 12204  3446   684  4281  7174  2359  3621   741  7865
  2910  5302  3620  3509  1657  1164  6229  1193  4583  5371   708   571
  2522  5179  8229  1289  2712   975  1050   609  4788  3069   836  2577
  1620  1845  6579  1893 10623  2249  3108   958  9277  6314  1526  6615
  1872  2859  1582  1238  1433  7882  4169  3249  3149  2246  1797  2957
  2348  6289  6419  6143 15857  2223  7238  2503  2622  4351   368   754
  2424  6681  2427   753  2576   590  1414  1103   585  1068   713  1092
  2329   882   866  2415  2101  1301  1113   760   625  1323  1138  1795
  2728   484  1048  1155  7057  1537  2214  1585  1521  3990  3049  1282
 10144  1168   454  3594  1768 15653  2247  4576  8335  5800  8471  3622
  2181  7685  6110  3757  3394  6304  1244  3518  2613  7476  4591  5595
  6224  1905  2993  8947  4020  2779  2782  1884 11054  9157  9283  6527
  3368  2511  5493  1338  1082  1149  1308  6148  1736  3059  2996  7596
  4811  1766  2760  5507  1199  2892  2862   654  1136  4113 14555   950
  2150  2820  3060  2600  5003  6288  2538  4933  1530  1437  1823  1422
  1217  9271  2145  1842  4297  3384  1245  4623  8386  1024 14318   433
  2149  2397   931  1512  4241  4736  1778  2327  6872   795  1908  1953
  2864  2319   915   947  1381  1285  1371  1042   900  1207  2278  6836
  3345  1198 15672  7297  1943  3190  5129  1808   759  1980 10961  6887
  1938  1835  1659  1209  3844  4843   639  5951  3804  4463  7980  4210
  4611 11560  4165  4057  6458  1977  1928  1123 11328 11938  2520 14782
  2671 12612  3031   626  3931  2302  3965  3914  4308  1534  2775  5998
  1271  9398   951  1355  3051  7855  9572  1837  4249  5234  6758  1366
  1358  2473  1337  7763  6560  3123  8065  2439  9034 14027  9629  1484
  1131  2064 12976  2580  2570  3915  1309  4817  2579  2225  4153  3114
  2124  1333  7119  4870   691  4370  2746  4110  2462  2969  4605  6331
  3552   697  1442  5293  3414  2039  3161   902 10297 14421  1056  1274
  1223  1372  2625  2235   959   884  1246  8086 10127   888   719 12389
  6850  2210  7485   797  4746   939  1188 11590  1190  2767  3441  4280
  3092  1331 15945  3234  9960  8648  1345  1647  4844  8318  2100 11816
   448 11998 18424 14896  2762  3386  2169  5096  1882  6999  2292  8978
   674  2718   750 12579  7511  3966  6199  1987  2303 12680  6350]
```

```
-> savings : ['unknown/no savings account' '... <  100 DM' '100 <= ... <  500 D
M'
 '... >= 1000 DM' '500 <= ... < 1000 DM']

-> employment_duration : ['< 1 yr' '1 <= ... < 4 yrs' '4 <= ... < 7 yrs' 'unempl
oyed' '>= 7 yrs']

-> installment_rate : ['< 20' '25 <= ... < 35' '20 <= ... < 25' '>= 35']

-> personal_status_sex : ['female : non-single or male : single' 'male : marrie
d/widowed'
 'female : single' 'male : divorced/separated']

-> other_debtors : ['none' 'guarantor' 'co-applicant']

-> present_residence : ['>= 7 yrs' '1 <= ... < 4 yrs' '4 <= ... < 7 yrs' '< 1 y
r']

-> property : ['car or other' 'unknown / no property'
 'building soc. savings agr./life insurance' 'real estate']

-> age : [21 36 23 39 38 48 40 65 24 31 44 25 37 49 33 26 51 29 56 47 34 28 41 5
8
 61 30 63 27 45 43 52 22 60 32 35 42 59 54 64 46 74 50 20 55 53 19 57 66
 68 70 67 75 62]

-> other_installment_plans : ['none' 'bank' 'stores']
```

```
-> housing : ['for free' 'rent' 'own']

-> number_credits : ['1' '2-3' '4-5' '>= 6']

-> job : ['skilled employee/official' 'unskilled - resident'
 'unemployed/unskilled - non-resident'
 'manager/self-empl./highly qualif. employee']

-> people_liable : ['0 to 2' '3 or more']

-> telephone : ['no' 'yes (under customer name)']

-> foreign_worker : ['no' 'yes']

-> credit_risk : ['good' 'bad']
```

- Пропусков данных нет
- Категории признаков (не совсем понятно что такое "порядковые", предположу что это как категориальные, только их можно упорядочить):
  - status: порядковые
  - duration: количественный
  - credit_history: категориальный
  - purpose: категориальные
  - amount: количественные
  - savings: порядковые
  - employment_duration: порядковые
  - installment_rate: порядковые
  - personal_status_sex: категориальный
  - other_debtors: категориальные
  - present_residence: порядковые
  - property: категориальные
  - age: количественный
  - other_installment_plans: категориальные
  - housing: категориальные
  - number_credits: порядковые
  - job: категориальные
  - people_liable: порядковые
  - telephone: бинарные
  - foreign_worker: бинарные
  - credit_risk: бинарные

3. Выведите среднюю сумму кредита (колонка amount) в каждой категории (колонка purpose), воспользовавшись методом groupby.

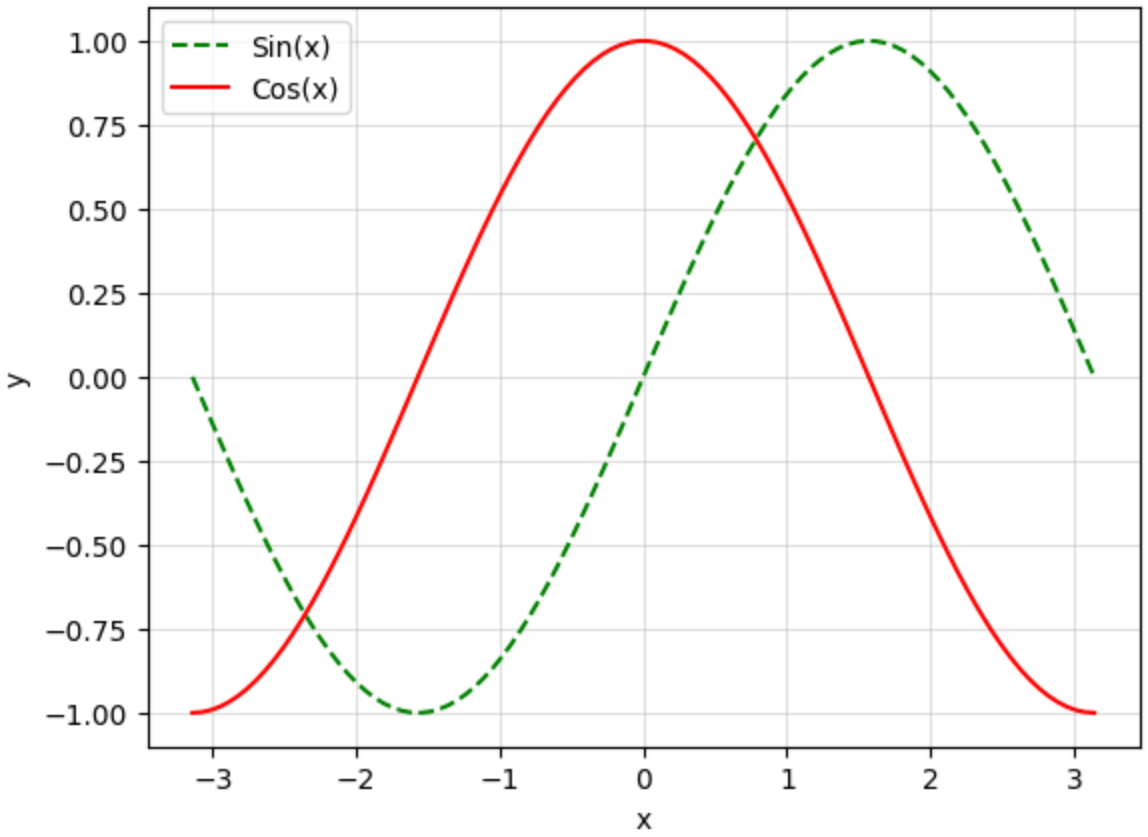
In [7]:
```
df0[['purpose', 'amount']].groupby('purpose').mean()
```

Out[7]:

purpose	amount
business	8209.333333
car (new)	5370.223301
car (used)	3066.983425
domestic appliances	2728.090909
furniture/equipment	2487.685714
others	3062.948718
radio/television	1498.000000
repairs	3180.400000
retraining	4158.041237
vacation	1205.888889

4. Используя numpy и matplotlib постройте график функций $\sin(x)$ и $\cos(x)$ от $-\pi$ до π из 100 точек с легендой и сеткой

In [8]:
```
X = np.linspace(-np.pi, np.pi, 100)
Y1 = np.sin(X)
Y2 = np.cos(X)
plt.plot(X, Y1, 'g--', label = 'Sin(x)')
plt.plot(X, Y2, 'r',   label = 'Cos(x)')
plt.grid(alpha = 0.4)
plt.xlabel('x')
plt.ylabel('y')
plt.legend();
```



5. Анализ данных, попытка предложить формулу оценки надежности

In [9]:
```
# преобразование всех категорий в числа
status_mod = ['no checking account', '... < 0 DM',
              '0<= ... < 200 DM',
              '... >= 200 DM / salary for at least 1 year']
# duration
credit_history_mod = ['critical account/other credits elsewhere',
                      'delay in paying off in the past',
                      'existing credits paid back duly till now',
                      'all credits at this bank paid back duly',
                      'no credits taken/all credits paid back duly']
purpose_mod = ['car (used)', 'others', 'retraining',
               'furniture/equipment', 'car (new)', 'business',
               'domestic appliances', 'radio/television', 'repairs',
               'vacation']
# amount
savings_mod = ['unknown/no savings account', '... <  100 DM',
               '100 <= ... <  500 DM', '500 <= ... < 1000 DM',
               '... >= 1000 DM']
employment_duration_mod = ['unemployed', '< 1 yr', '1 <= ... < 4 yrs',
                           '4 <= ... < 7 yrs', '>= 7 yrs']
```

```
installment_rate_mod = ['< 20', '20 <= ... < 25', '25 <= ... < 35',
                        '>= 35']
personal_status_sex_mod = ['female : non-single or male : single',
                           'male : married/widowed', 'female : single',
                           'male : divorced/separated']
other_debtors_mod = ['none', 'guarantor', 'co-applicant']
present_residence_mod = ['< 1 yr', '1 <= ... < 4 yrs',
                         '4 <= ... < 7 yrs', '>= 7 yrs']
property_mod = ['car or other', 'unknown / no property',
                'building soc. savings agr./life insurance', 'real estate']
# age
other_installment_plans_mod = ['none', 'bank', 'stores']
housing_mod = ['for free', 'rent', 'own']
number_credits_mod = ['1', '2-3', '4-5', '>= 6']
job_mod = ['unemployed/unskilled - non-resident', 'unskilled - resident',
           'skilled employee/official',
           'manager/self-empl./highly qualif. employee']
people_liable_mod = ['0 to 2', '3 or more']
telephone_mod = ['no', 'yes (under customer name)']
foreign_worker_mod = ['no', 'yes']
credit_risk_mod = ['bad', 'good']

mods = {'status' : lambda x: status_mod.index(x),
        'duration' : lambda x: x,
        'credit_history' : lambda x: credit_history_mod.index(x),
        'purpose' : lambda x: purpose_mod.index(x),
        'amount' : lambda x: x,
        'savings' : lambda x: savings_mod.index(x),
        'employment_duration' : lambda x: employment_duration_mod.index(x),
        'installment_rate' : lambda x: installment_rate_mod.index(x),
        'personal_status_sex' : lambda x: personal_status_sex_mod.index(x),
        'other_debtors' : lambda x: other_debtors_mod.index(x),
        'present_residence' : lambda x: present_residence_mod.index(x),
        'property' : lambda x: property_mod.index(x),
        'age' : lambda x: x,
        'other_installment_plans' : lambda x: other_installment_plans_mod.index(x
),
        'housing' : lambda x: housing_mod.index(x),
        'number_credits' : lambda x: number_credits_mod.index(x),
        'job' : lambda x: job_mod.index(x),
        'people_liable' : lambda x: people_liable_mod.index(x),
        'telephone' : lambda x: telephone_mod.index(x),
        'foreign_worker' : lambda x: foreign_worker_mod.index(x),
        'credit_risk' : lambda x: credit_risk_mod.index(x)}

df1 = df0.agg(mods)
```

In [10]:
```
df1
```

Out[10]:

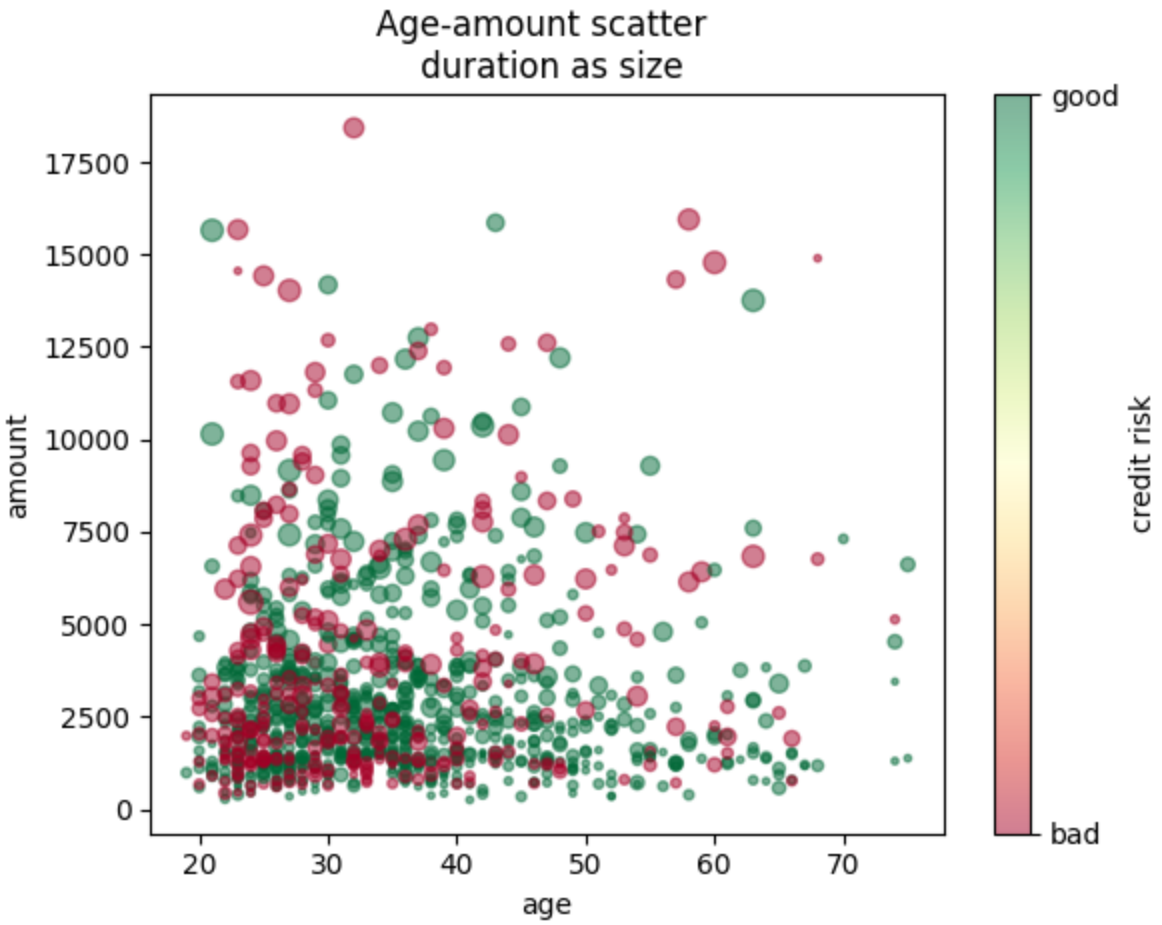
	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	pers
0	0	18	3	0	1049	0	1	0	
1	0	9	3	1	2799	0	2	2	
2	1	12	4	2	841	1	3	2	
3	0	12	3	1	2122	0	2	1	
4	0	12	3	1	2171	0	2	0	
...	...	...	...	...	...	...	...	...	
995	0	24	4	3	1987	0	2	2	
996	0	24	4	1	2303	0	4	0	
997	3	21	3	1	12680	4	4	0	
998	1	12	4	3	6468	4	0	2	
999	0	30	4	0	6350	4	4	0	

1000 rows × 21 columns

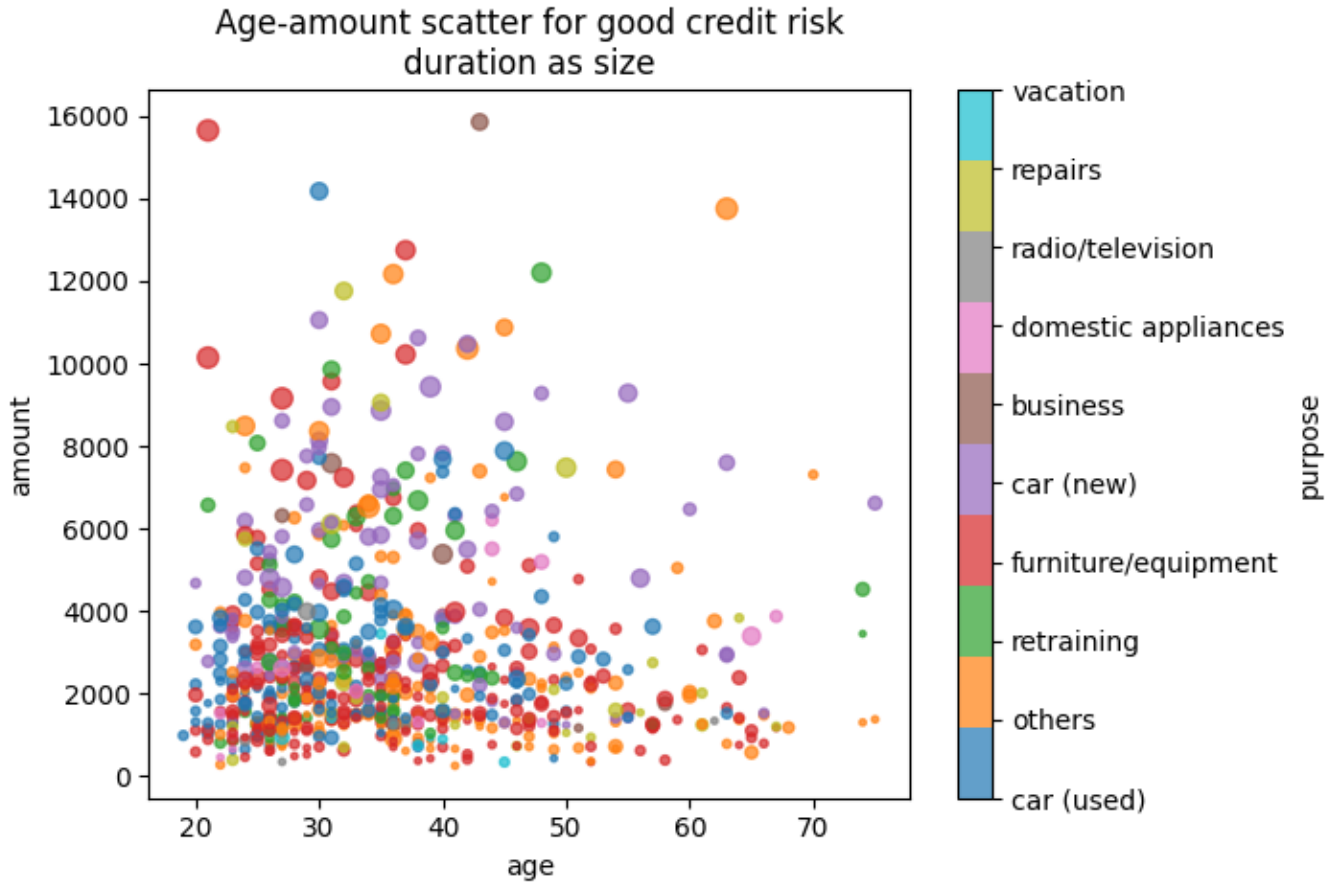
In [11]:
```
df1g = df1[df1['credit_risk']==1]
df1b = df1[df1['credit_risk']==0]
df1g.shape, df1b.shape
```

Out[11]:
```
((700, 21), (300, 21))
```

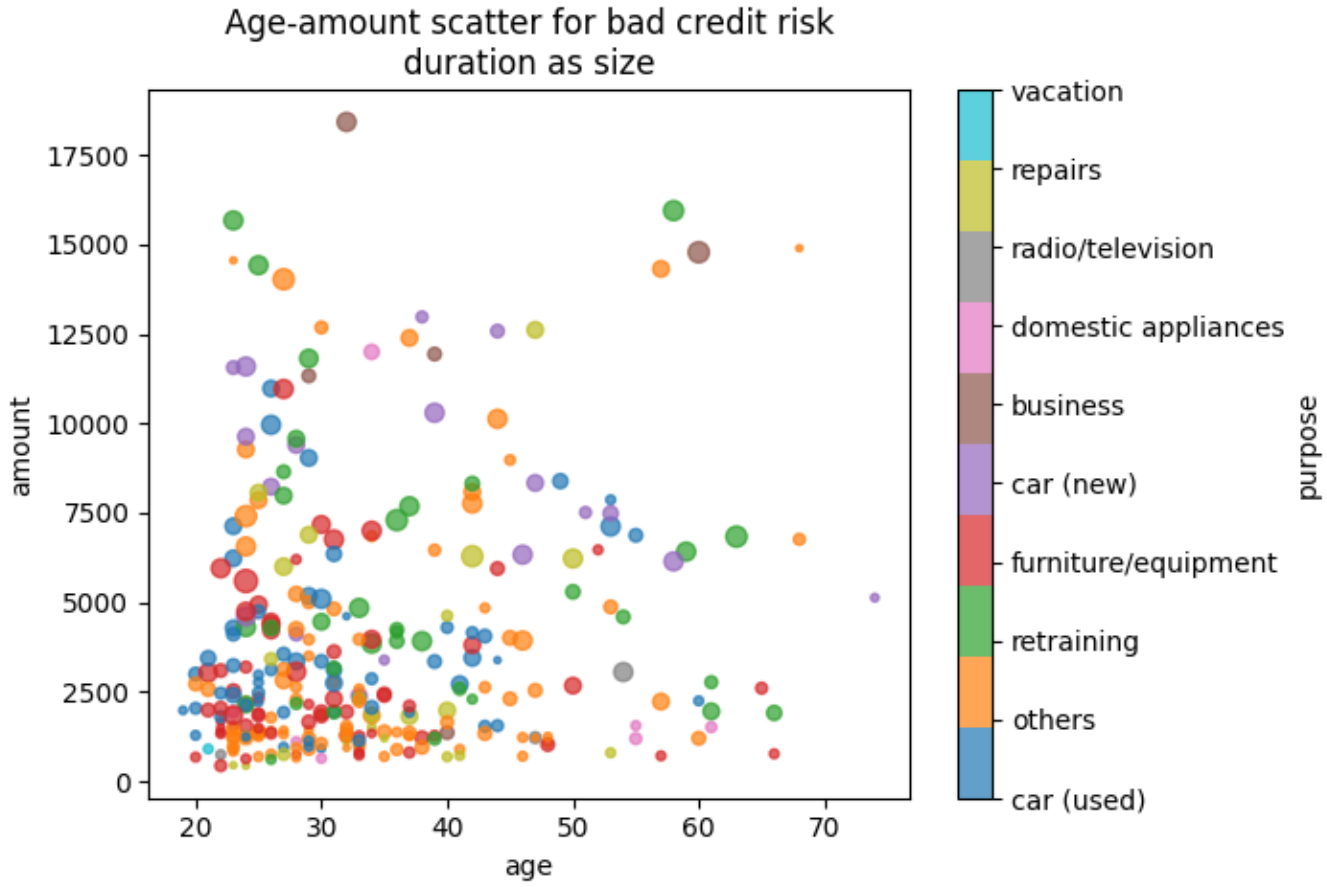
In [12]:
```
df1.plot.scatter('age','amount','duration','credit_risk', cmap='RdYlGn',
                 alpha = 0.5,
                 title = 'Age-amount scatter \n duration as size')
cbar = plt.gcf().get_axes()[1]
cbar.set(yticks=[0,1], yticklabels=['bad','good'], ylabel='credit risk');
```



In [13]:
```
df1g.plot.scatter('age','amount','duration','purpose', cmap='tab10',
                  alpha = 0.7,
                  title = 'Age-amount scatter for good credit risk\n'+
                          'duration as size')
plt.gcf().get_axes()[1].set(yticks = range(len(purpose_mod)), yticklabels=purpose_
mod);
```



In [14]:
```
df1b.plot.scatter('age','amount','duration','purpose', cmap='tab10',
                  alpha = 0.7,
                  title = 'Age-amount scatter for bad credit risk\n'+
                          'duration as size')
cbar = plt.gcf().get_axes()[1].set(yticks = range(len(purpose_mod)), yticklabels=p
urpose_mod);
```



Спустя несколько дней размышлений я подумал что вполне себе можно попробовать реализовать своего рода нейронную сеть, где под каждый параметр она берет коэффициент соотношения отказов, суммирует их и сравнивает полученное число со средним коэффициентом отказа (опишу его далее). Если оно оказалось больше, то считать кредит одобренным, иначе - не одобренным.

Да, это очень простая модель, у нее есть свои недостатки (отсутствие правильной логики работы с различными параметрами, такими как телефона и кредит, то есть бинарные, порядковые и категориальные факторы оцениваются одинаково; проблемы с анализом количественных факторов), но есть и плюсы в виде простоты реализации и возможностью проверки потом (хотя это относимо к любым нейросетям).

Это не финальное решение и возможно позже оно изменится, но этот блок менять наверное не буду, чтобы оставить последовательность развития мысли

Про средний коэффициент отказа: опишу на примере "status", разбираемым в следующей ячейке. Исходя из предположения, что 1000 это достаточная выборка для различных выводов, то применимо рассуждение: Сумма числа всех событий по всем категориям по определению будет равна 1000. Пускай в выбранную категорию попало n человек, и k из них отказали в предите. $\frac{k}{n}$ берется за коэффициент соотношения отказов. Вес отказа в этой категории будет равен $\frac{k}{n} \cdot \frac{n}{1000}$, что равно $\frac{k}{1000}$. Тогда средний отказ по "status" будет равен сумме $\frac{k}{1000}$ по всем категориям.

(и тут меня осенило...) Тогда эта сумма будет равна в точности $\frac{300}{1000}$, потому что в каждую категорию "status" попадают все элементы таблицы, потому 1000 будет числом всех событий, а 300 будет отказами. Проблема в том, что ровно такая же логика применима к абсолютно каждому столбцу таблицы, а значит что итоговый средний коэффициент отказа по всей таблице будет равен $20 \cdot 0.3 = 6$. И это проблема, ведь такое усреднение вообще не может быть правильным, потому что оно считает все столбцы абсолютно равноценными. А очень наврядли что это так.

Тем не менее других идей у меня пока что нет, так что я попробую хотя бы реализовать эту, а там может чего и придумаю

(Из реализации: сколько-то демонстрирующих графиков, потому что 20 будет много; рассчитать и сохранить все $\frac{k}{n}$ для функции; реализовать функцию через сумму правильных элементов по массиву)

*В связи с выводом, полученным выше, может показаться что количественные факторы можно расценить все также по большим массивам, но это не так, как минимум просто потому что на вход функции можно подать любое число для количественного фактора, что наткнет на проблему "а если этого числа нет". Надо подбирать функцию.

In [15]:
```
#df1['counter?'] = np.ones(df1.shape[0], dtype = np.int32)
df1 = df1.rename(columns = {'credit_risk': 'approved'})
df1['disapproved'] = 1 - df1['approved']
df1['all cases'] = 1
df1
```

Out[15]:

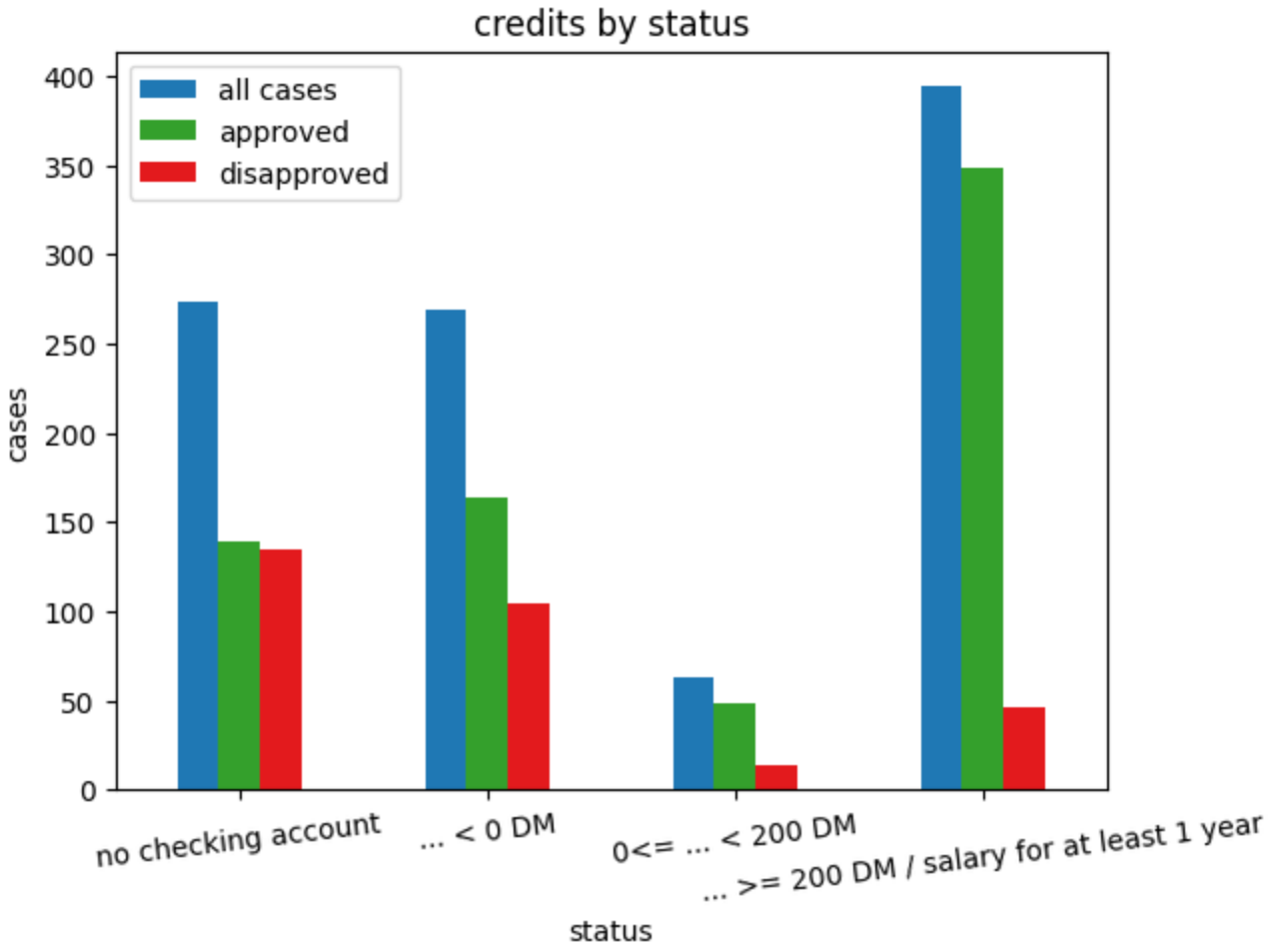
	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	pers
0	0	18	3	0	1049	0	1	0	
1	0	9	3	1	2799	0	2	2	
2	1	12	4	2	841	1	3	2	
3	0	12	3	1	2122	0	2	1	
4	0	12	3	1	2171	0	2	0	
...	...	...	...	...	...	...	...	...	
995	0	24	4	3	1987	0	2	2	
996	0	24	4	1	2303	0	4	0	
997	3	21	3	1	12680	4	4	0	
998	1	12	4	3	6468	4	0	2	
999	0	30	4	0	6350	4	4	0	

1000 rows × 23 columns

In [16]:
```
#status
df1tmp = df1[['status', 'all cases', 'approved', 'disapproved']].groupby('status')
.sum()
status_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: status_mod[i] for i in range(len(status_mod))})
df1tmp.plot.bar(rot = 7, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by status')
#попытка красиво отобразить таблицу дала мелкий текст, так что костыль..
df1tmp
```

Out[16]:

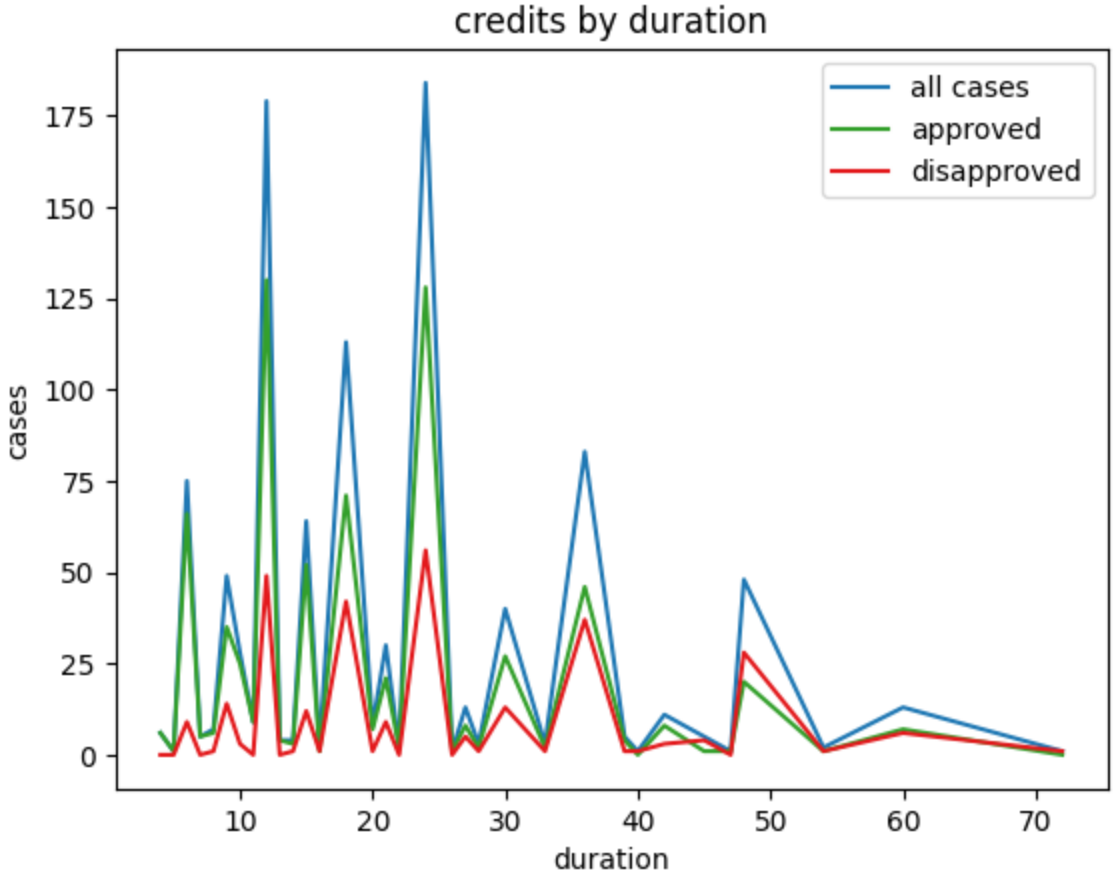
status	all cases	approved	disapproved
no checking account	274	139	135
... < 0 DM	269	164	105
0<= ... < 200 DM	63	49	14
... >= 200 DM / salary for at least 1 year	394	348	46



In [17]:
```
#duration
df1tmp = df1[['duration', 'all cases', 'approved', 'disapproved']].groupby('durati
on').sum()
#duration_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp.plot(color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved':
'#e31a1d'},
            ylabel = 'cases', title = 'credits by duration')
df1tmp
```

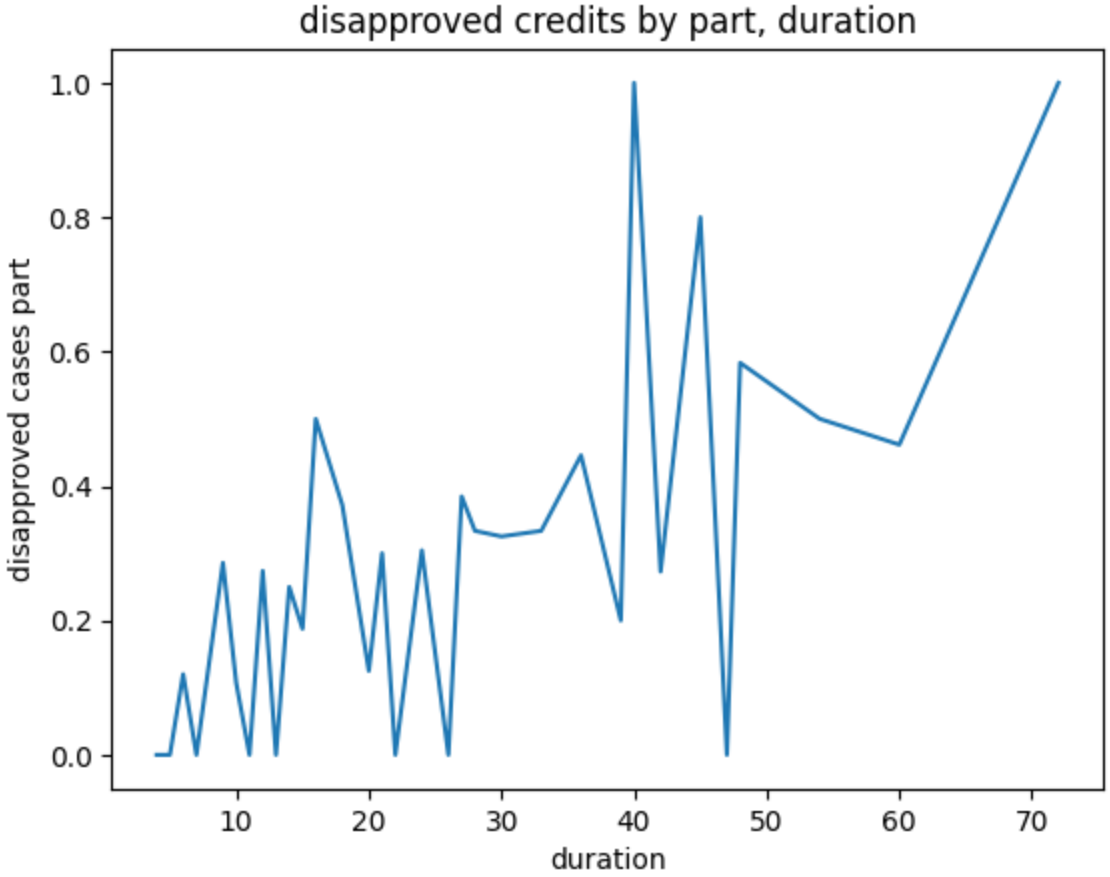
Out[17]:

duration	all cases	approved	disapproved
4	6	6	0
5	1	1	0
6	75	66	9
7	5	5	0
8	7	6	1
9	49	35	14
10	28	25	3
11	9	9	0
12	179	130	49
13	4	4	0
14	4	3	1
15	64	52	12
16	2	1	1
18	113	71	42
20	8	7	1
21	30	21	9
22	2	2	0
24	184	128	56
26	1	1	0
27	13	8	5
28	3	2	1
30	40	27	13
33	3	2	1
36	83	46	37
39	5	4	1
40	1	0	1
42	11	8	3
45	5	1	4
47	1	1	0
48	48	20	28
54	2	1	1
60	13	7	6
72	1	0	1


credits by duration
all cases
approved
disapproved
cases
duration
0
25
50
75
100
125
150
175
10
20
30
40
50
60
70

In [18]:
```
df1tmp_part = df1tmp['disapproved']/df1tmp['all cases']
df1tmp_part.plot(color='#1f78b4', ylabel = 'disapproved cases part',
                 title = 'disapproved credits by part, duration')
df1tmp_part
```

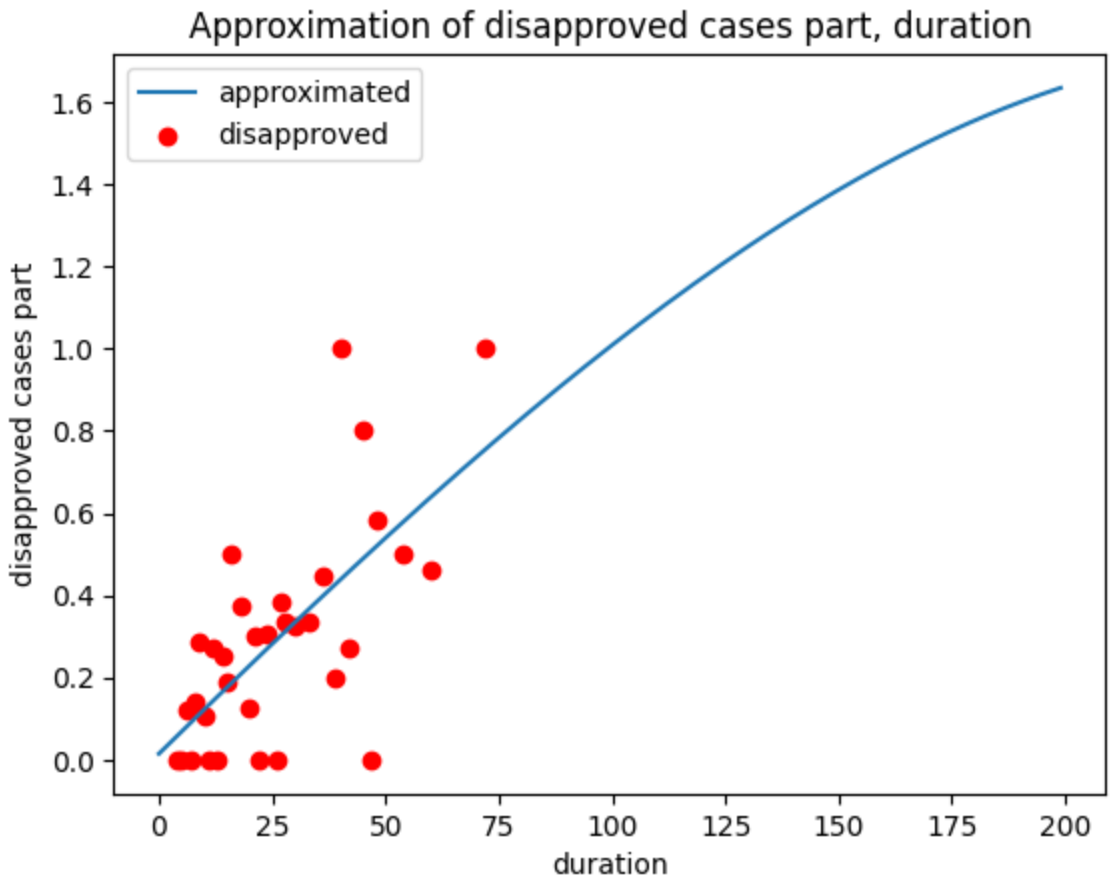
Out[18]:
```
duration
4     0.000000
5     0.000000
6     0.120000
7     0.000000
8     0.142857
9     0.285714
10    0.107143
11    0.000000
12    0.273743
13    0.000000
14    0.250000
15    0.187500
16    0.500000
18    0.371681
20    0.125000
21    0.300000
22    0.000000
24    0.304348
26    0.000000
27    0.384615
28    0.333333
30    0.325000
33    0.333333
36    0.445783
39    0.200000
40    1.000000
42    0.272727
45    0.800000
47    0.000000
48    0.583333
54    0.500000
60    0.461538
72    1.000000
dtype: float64
```


disapproved credits by part, duration
disapproved cases part
duration
1.0
0.8
0.6
0.4
0.2
0.0
10
20
30
40
50
60
70

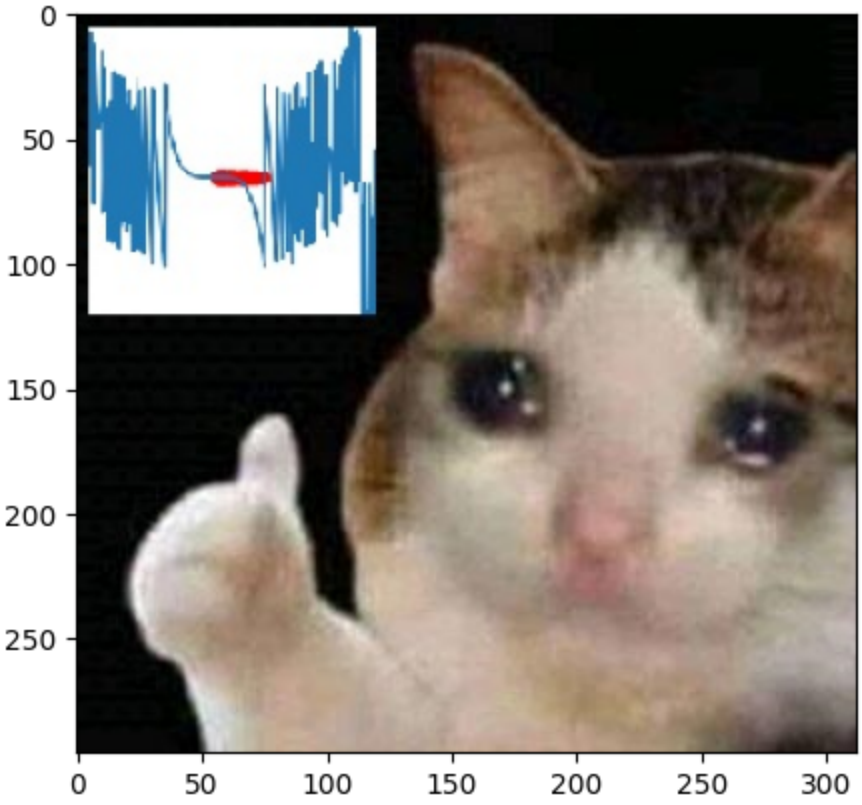
In [19]:
```
# попытка аппроксимировать функцией
# методом наименьших квадратов к x^3
A = np.array([[x**3, x**2, x, 1] for x in df1tmp_part.index])
B = df1tmp_part.to_numpy()
left = np.transpose(A)@A
right = np.transpose(A)@B
# а вы знали что numpy умеет решать уравнения?
duration_pol = np.linalg.solve(left, right)
duration_func = lambda x: duration_pol[0]*x**3+duration_pol[1]*x**2+\
                          duration_pol[2]*x   +duration_pol[3]

X = np.arange(0, 200)
Y = np.array([duration_func(x) for x in X])
plt.plot(X,Y)
plt.scatter(df1tmp_part.index, B, color = 'Red')
plt.legend(['approximated', 'disapproved'])
plt.xlabel('duration')
plt.ylabel('disapproved cases part')
plt.title('Approximation of disapproved cases part, duration')
duration_pol
```

Out[19]:
```
array([-4.71664048e-08, -4.06852033e-06,  1.08111370e-02,  1.48453990e-02])
```



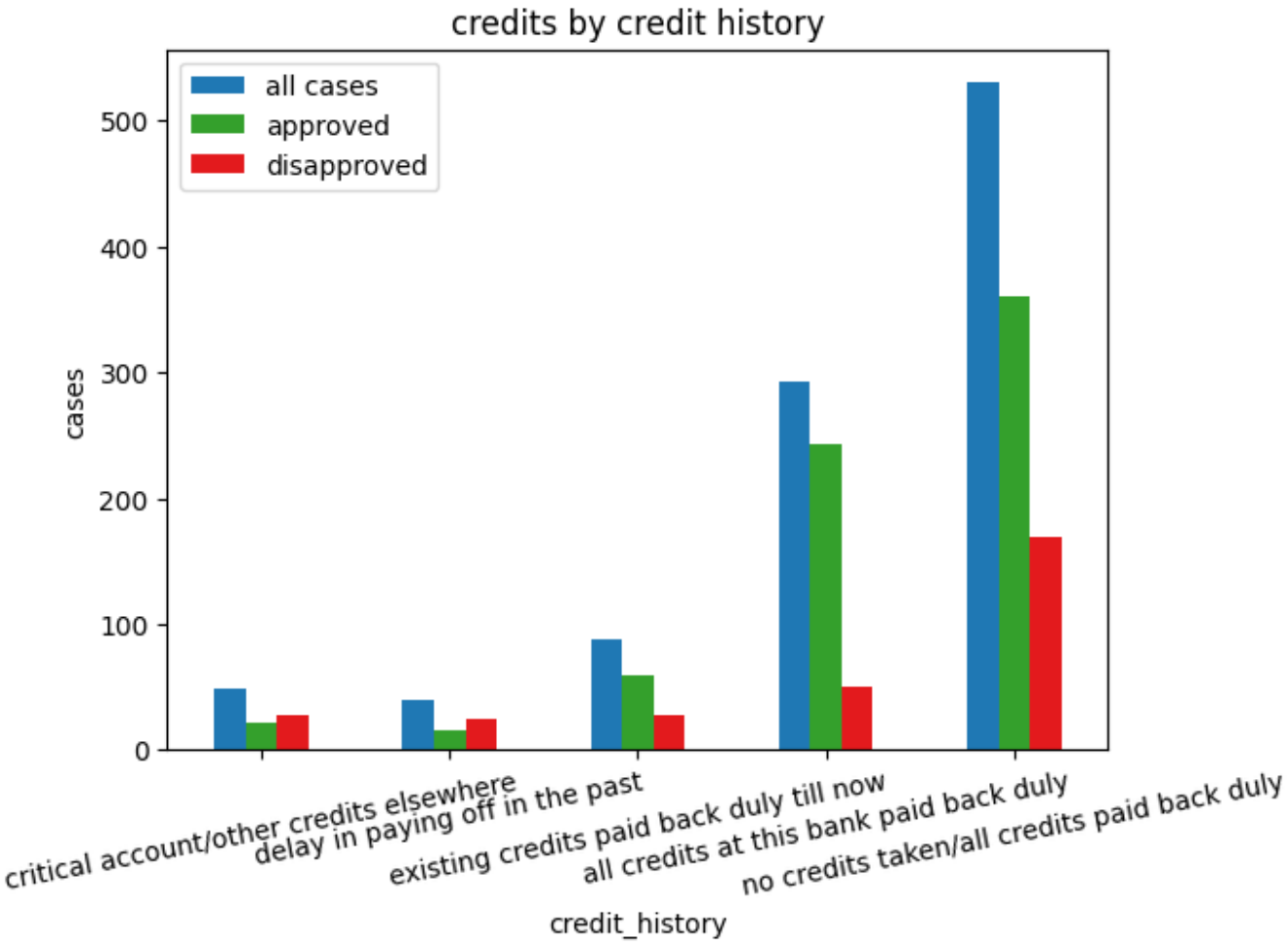
In [20]:
```
# я столько материала для ячейки выше перекопал..
plt.imshow(plt.imread('./cat.jpg'));
# оно далеко не сразу сработало
# и потому график такой странный (в +-500)
# потому что я много чего перепроверил и перепробовал
# а котика уже поставил, но решил уже не убирать
# (изначально было x^5, а он сразу от нуля в отрицательные уходил)
```



In [21]:
```
#credit_history
df1tmp = df1[['credit_history', 'all cases', 'approved', 'disapproved']].groupby(
'credit_history').sum()
credit_history_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: credit_history_mod[i] for i in range(len(credit
_history_mod))})
df1tmp.plot.bar(rot = 11, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'd
isapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by credit history')
df1tmp
```

Out[21]:

credit_history	all cases	approved	disapproved
critical account/other credits elsewhere	49	21	28
delay in paying off in the past	40	15	25
existing credits paid back duly till now	88	60	28
all credits at this bank paid back duly	293	243	50
no credits taken/all credits paid back duly	530	361	169

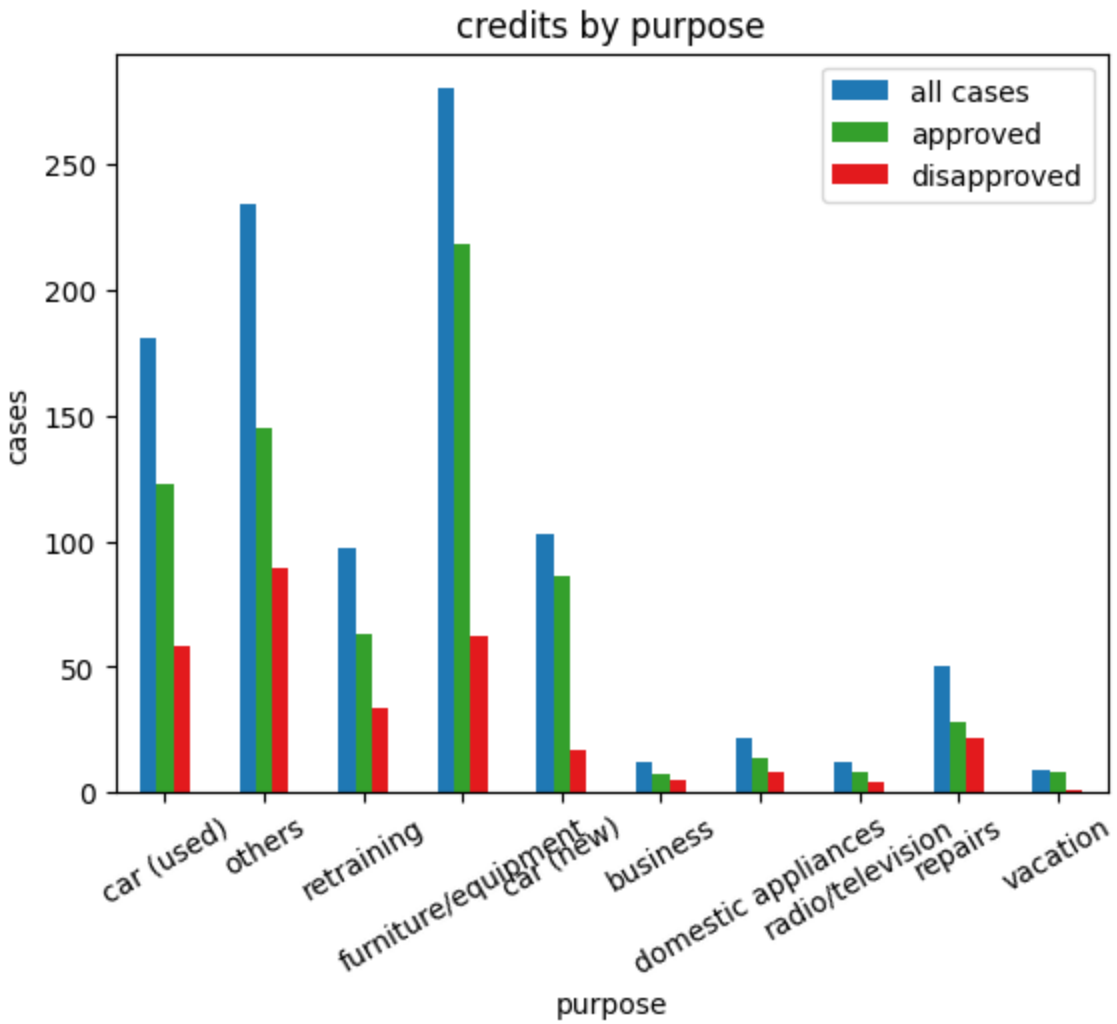


In [22]:

```
#purpose
df1tmp = df1[['purpose', 'all cases', 'approved', 'disapproved']].groupby('purpos
e').sum()
purpose_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: purpose_mod[i] for i in range(len(purpose_mod
))})
df1tmp.plot.bar(rot = 30, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'd
isapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by purpose')
df1tmp
```

Out[22]:

purpose	all cases	approved	disapproved
car (used)	181	123	58
others	234	145	89
retraining	97	63	34
furniture/equipment	280	218	62
car (new)	103	86	17
business	12	7	5
domestic appliances	22	14	8
radio/television	12	8	4
repairs	50	28	22
vacation	9	8	1

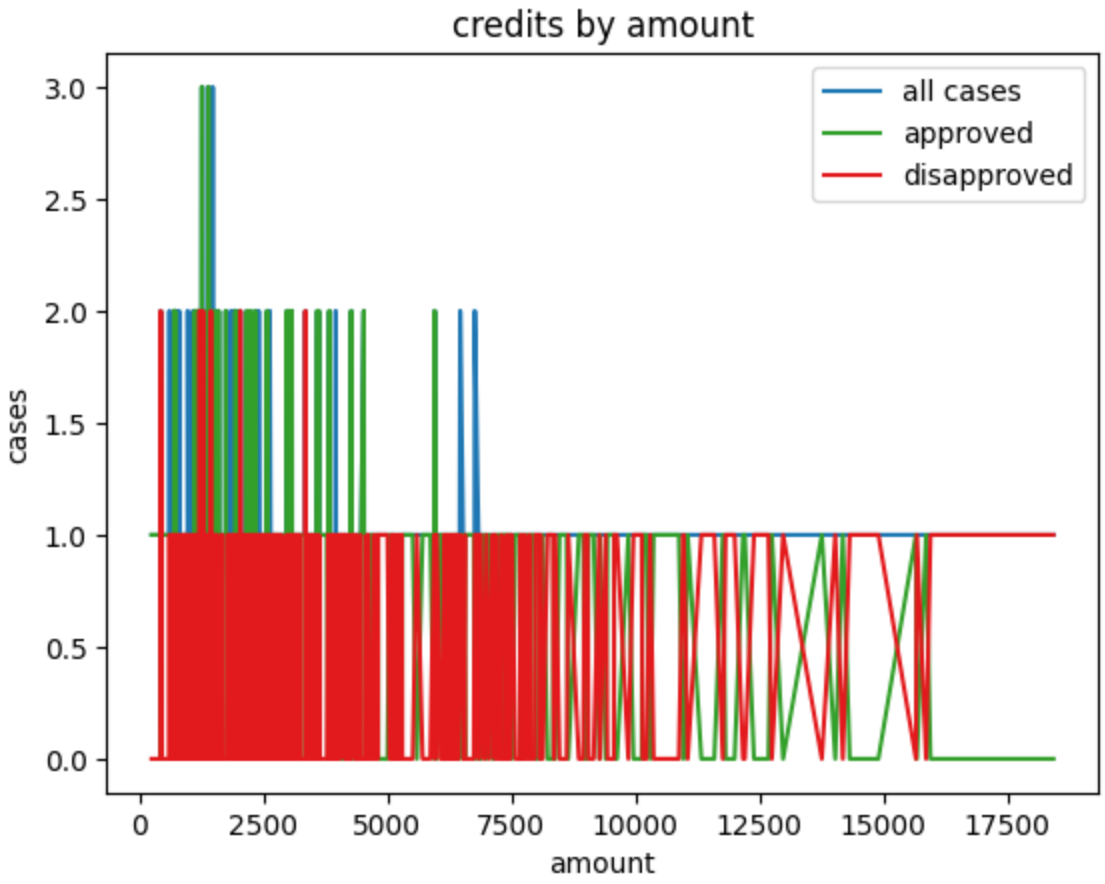


In [23]:
```
#amount
df1tmp = df1[['amount', 'all cases', 'approved', 'disapproved']].groupby('amount')
.sum()
#amount_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp.plot(color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved':
'#e31a1d'},
            ylabel = 'cases', title = 'credits by amount')
df1tmp
```

Out[23]:

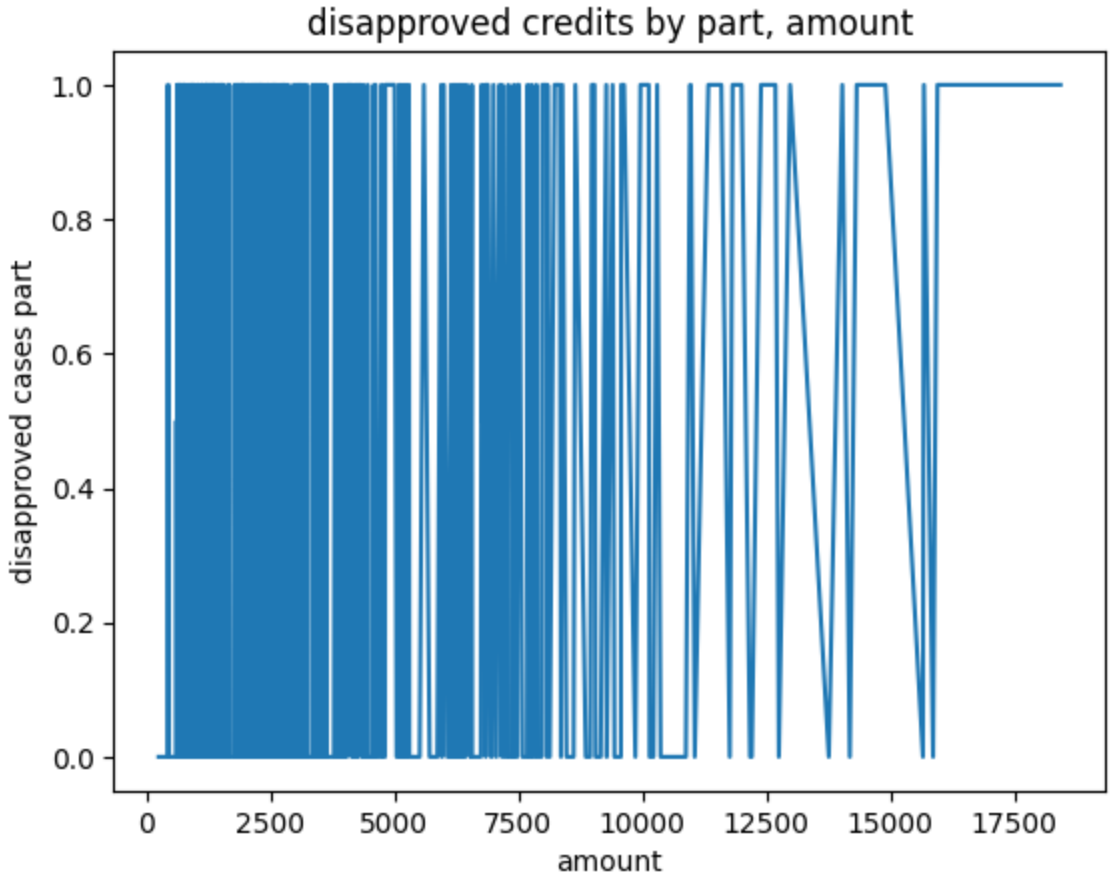
amount	all cases	approved	disapproved
250	1	1	0
276	1	1	0
338	1	1	0
339	1	1	0
343	1	1	0
...	...	...	...
15653	1	1	0
15672	1	0	1
15857	1	1	0
15945	1	0	1
18424	1	0	1

923 rows × 3 columns



In [25]:
```
df1tmp_part = df1tmp['disapproved']/df1tmp['all cases']
df1tmp_part.plot(color='#1f78b4', ylabel = 'disapproved cases part',
                 title = 'disapproved credits by part, amount')
df1tmp_part
```

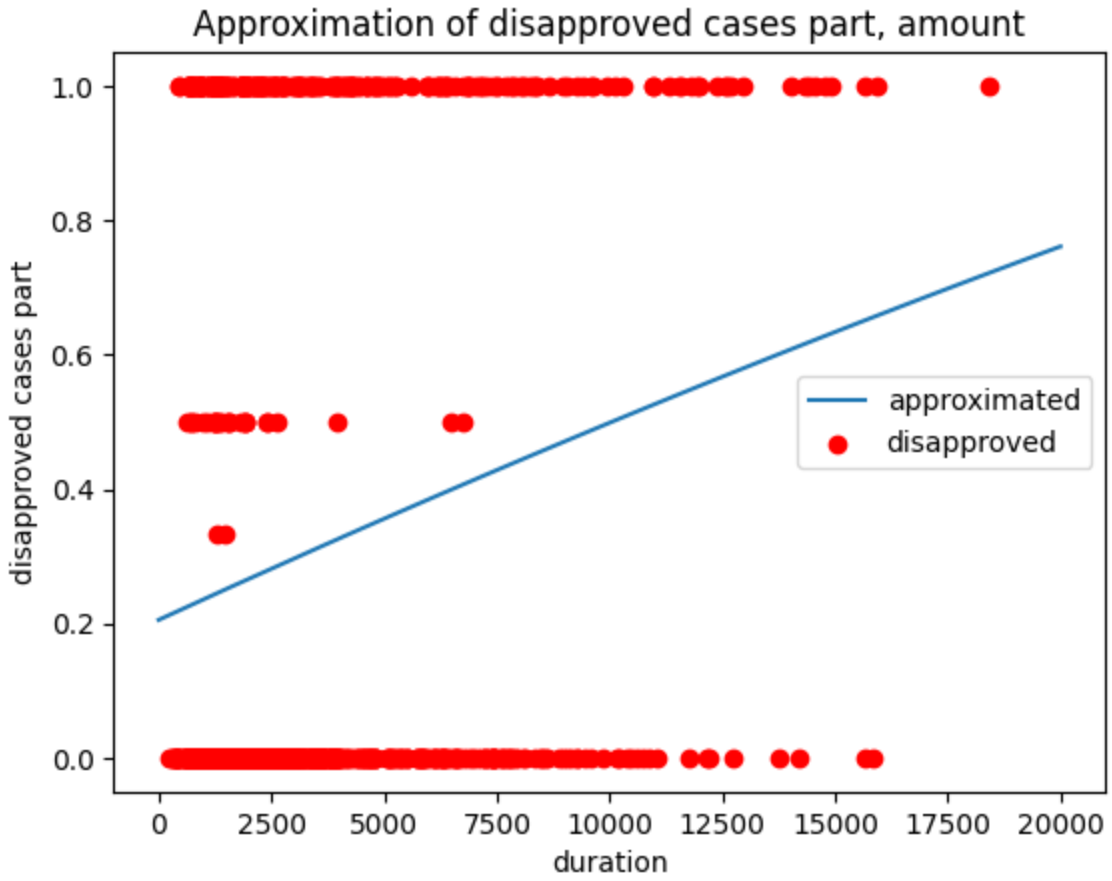
Out[25]:
```
amount
250      0.0
276      0.0
338      0.0
339      0.0
343      0.0
        ...
15653    0.0
15672    1.0
15857    0.0
15945    1.0
18424    1.0
Length: 923, dtype: float64
```



In [26]:
```
# попытка аппроксимировать функцией
# методом наименьших квадратов к x^3
A = np.array([[x**3, x**2, x, 1] for x in df1tmp_part.index])
B = df1tmp_part.to_numpy()
left = np.transpose(A)@A
right = np.transpose(A)@B
# а вы знали что numpy умеет решать уравнения?
amount_pol = np.linalg.solve(left, right)
amount_func = lambda x: amount_pol[0]*x**3+amount_pol[1]*x**2+\
                        amount_pol[2]*x   +amount_pol[3]

X = np.arange(0, 20000)
Y = np.array([amount_func(x) for x in X])
plt.plot(X,Y)
plt.scatter(df1tmp_part.index, B, color = 'Red')
plt.legend(['approximated', 'disapproved'])
plt.xlabel('duration')
plt.ylabel('disapproved cases part')
plt.title('Approximation of disapproved cases part, amount')
amount_pol
```

Out[26]:
```
array([-2.20507454e-14, -1.54975897e-10,  3.08878278e-05,  2.05259186e-01])
```



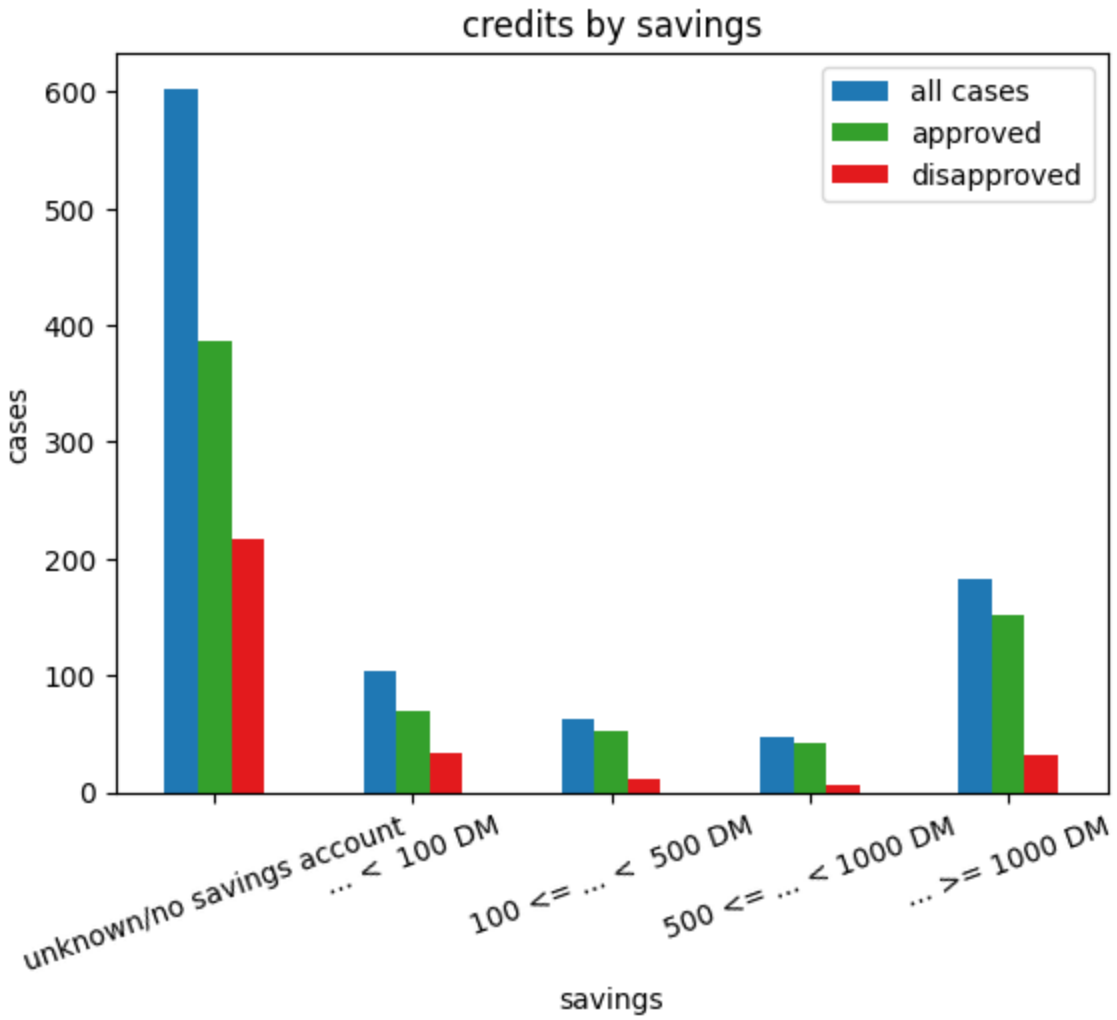
Здесь он мне прямо не дает решить для x^5, не то что раньше. И я понимаю что анализировать этот график точно также как и другие это плохая идея, но синусоиды тут рисовать тоже плохая идея. Надо что-то придумать с частотой, чтобы правильно ее трактовать (как и в остальных графиках на самом деле). *Я пытался через разбиение, но ничего не получилось.

Оставлю пока так.

In [27]:
```
#savings
df1tmp = df1[['savings', 'all cases', 'approved', 'disapproved']].groupby('saving
s').sum()
savings_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: savings_mod[i] for i in range(len(savings_mod
))})
df1tmp.plot.bar(rot = 20, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'd
isapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by savings')
df1tmp
```

Out[27]:

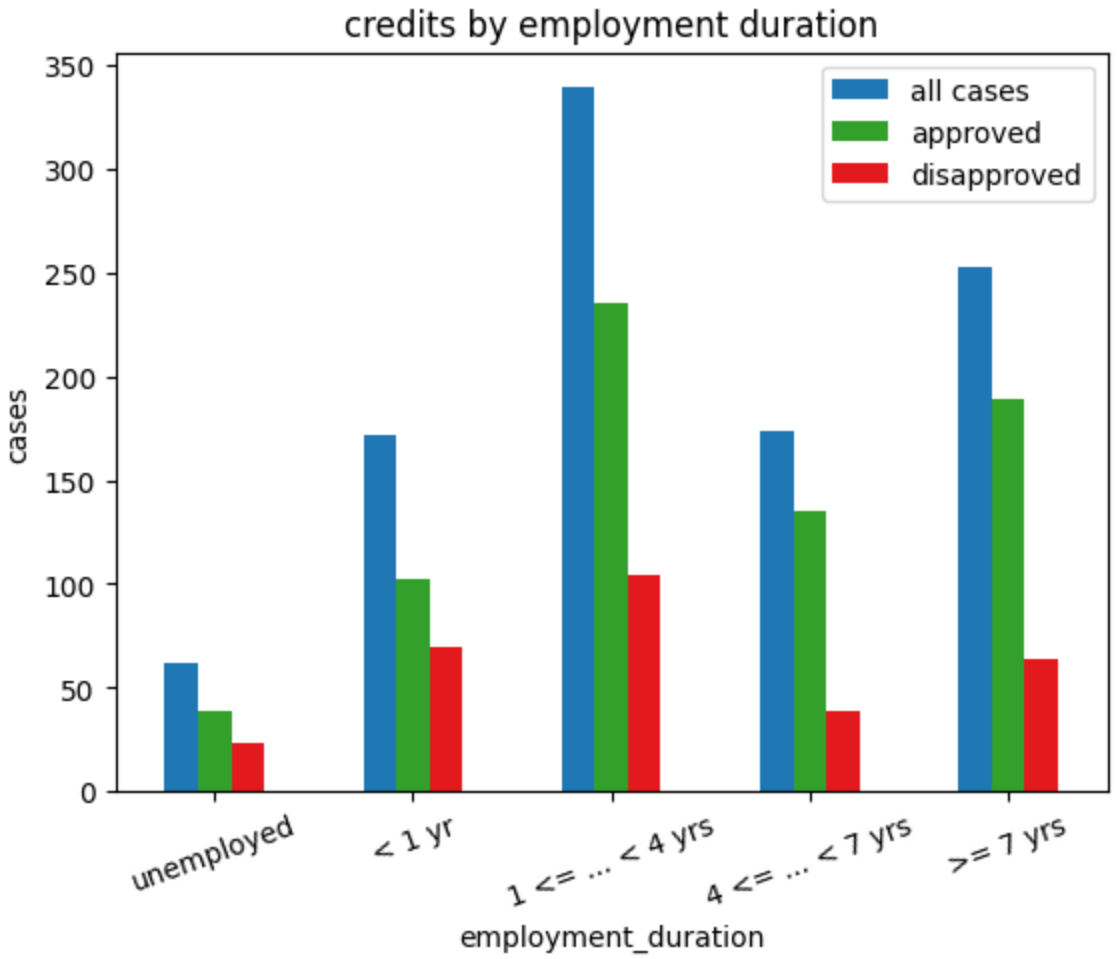
savings	all cases	approved	disapproved
unknown/no savings account	603	386	217
... < 100 DM	103	69	34
100 <= ... < 500 DM	63	52	11
500 <= ... < 1000 DM	48	42	6
... >= 1000 DM	183	151	32



In [28]:
```
#employment_duration
df1tmp = df1[['employment_duration', 'all cases', 'approved', 'disapproved']].grou
pby('employment_duration').sum()
employment_duration_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: employment_duration_mod[i] for i in range(len(e
mployment_duration_mod))})
df1tmp.plot.bar(rot = 20, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'd
isapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by employment duration')
df1tmp
```

Out[28]:

employment_duration	all cases	approved	disapproved
unemployed	62	39	23
< 1 yr	172	102	70
1 <= ... < 4 yrs	339	235	104
4 <= ... < 7 yrs	174	135	39
>= 7 yrs	253	189	64

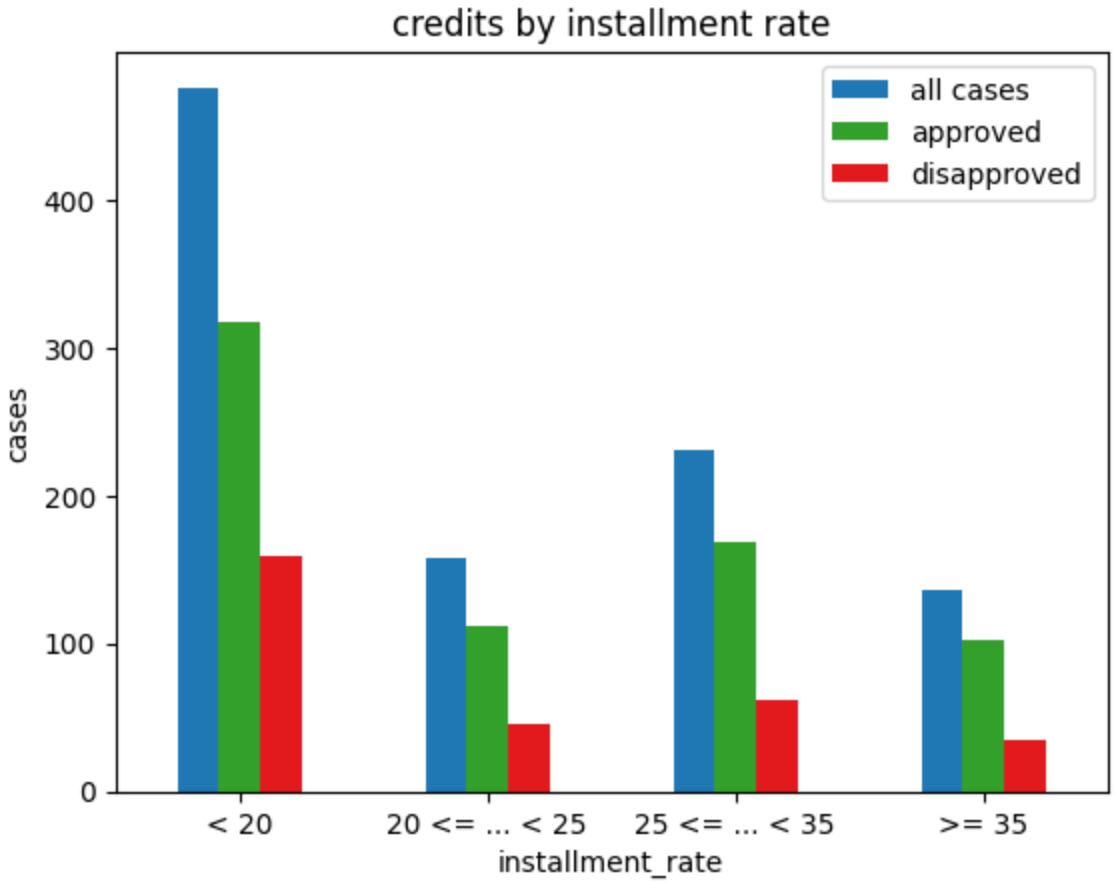


Где-то тут я сперва подумал отобразить все порядковые и бинарные, может тогда идея придет, а потом подумал что чего уж давайте и категориальные тоже. Количественные и так для граффиков надо проанализировать, так что они тоже отображаются. Так что 20 графиков. Йеееe..

In [29]:
```
#installment_rate
df1tmp = df1[['installment_rate', 'all cases', 'approved', 'disapproved']].groupby
('installment_rate').sum()
installment_rate_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: installment_rate_mod[i] for i in range(len(inst
allment_rate_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by installment rate')
df1tmp
```

Out[29]:

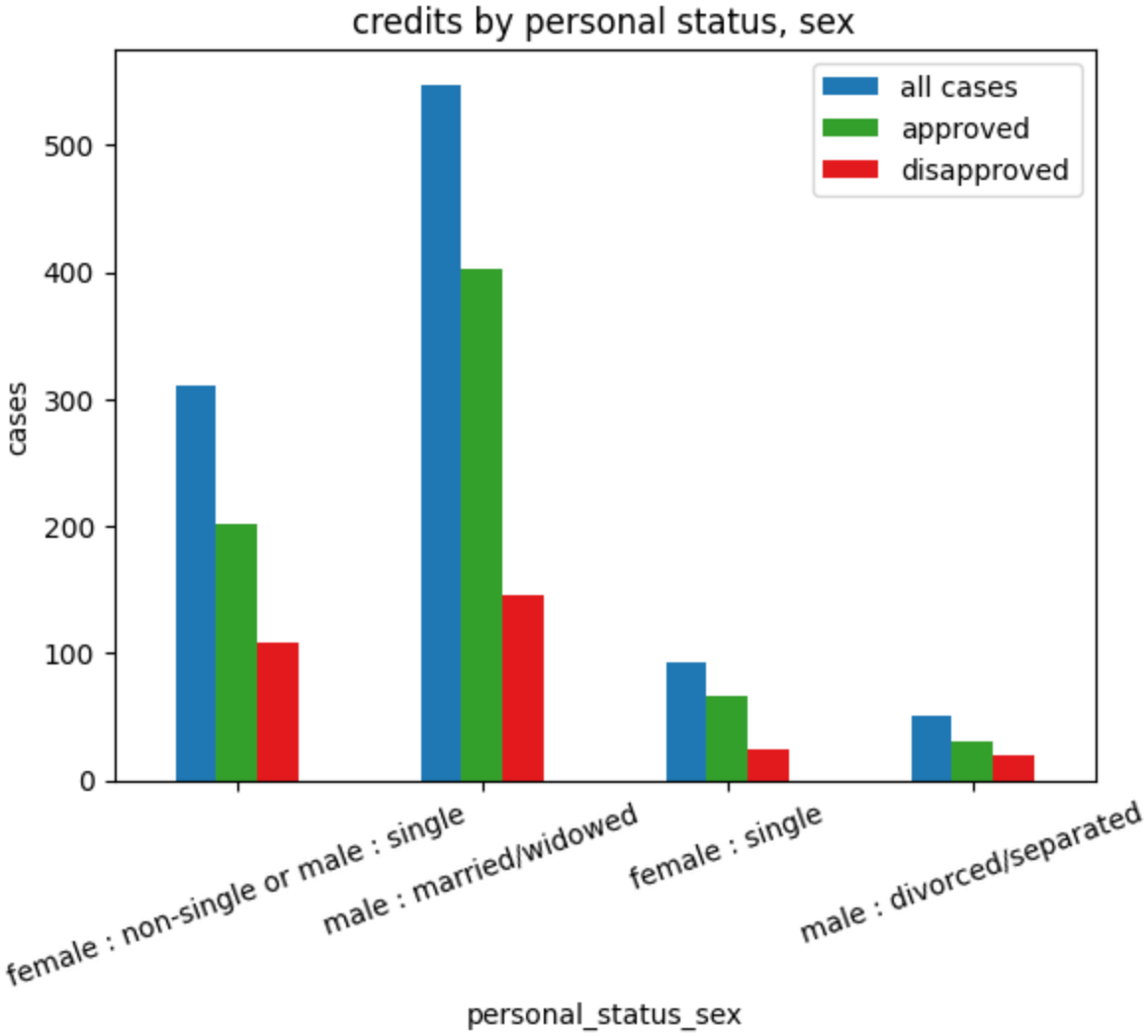
installment_rate	all cases	approved	disapproved
< 20	476	317	159
20 <= ... < 25	157	112	45
25 <= ... < 35	231	169	62
>= 35	136	102	34



In [30]:
```
#personal_status_sex
df1tmp = df1[['personal_status_sex', 'all cases', 'approved', 'disapproved']].grou
pby('personal_status_sex').sum()
personal_status_sex_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: personal_status_sex_mod[i] for i in range(len(p
ersonal_status_sex_mod))})
df1tmp.plot.bar(rot = 20, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'd
isapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by personal status, sex')
df1tmp
```

Out[30]:

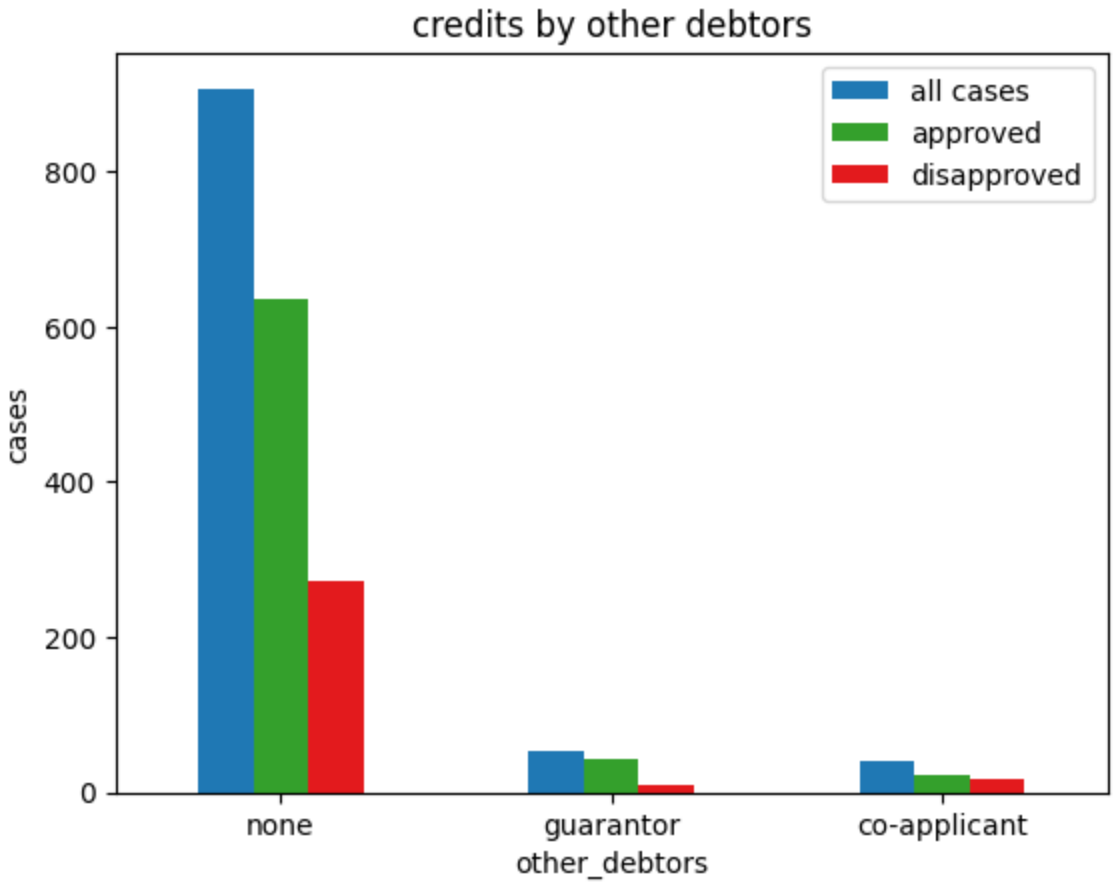
personal_status_sex	all cases	approved	disapproved
female : non-single or male : single	310	201	109
male : married/widowed	548	402	146
female : single	92	67	25
male : divorced/separated	50	30	20



In [31]:
```
#other_debtors
df1tmp = df1[['other_debtors', 'all cases', 'approved', 'disapproved']].groupby('other_debtors').sum()
other_debtors_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: other_debtors_mod[i] for i in range(len(other_debtors_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by other debtors')
df1tmp
```

Out[31]:

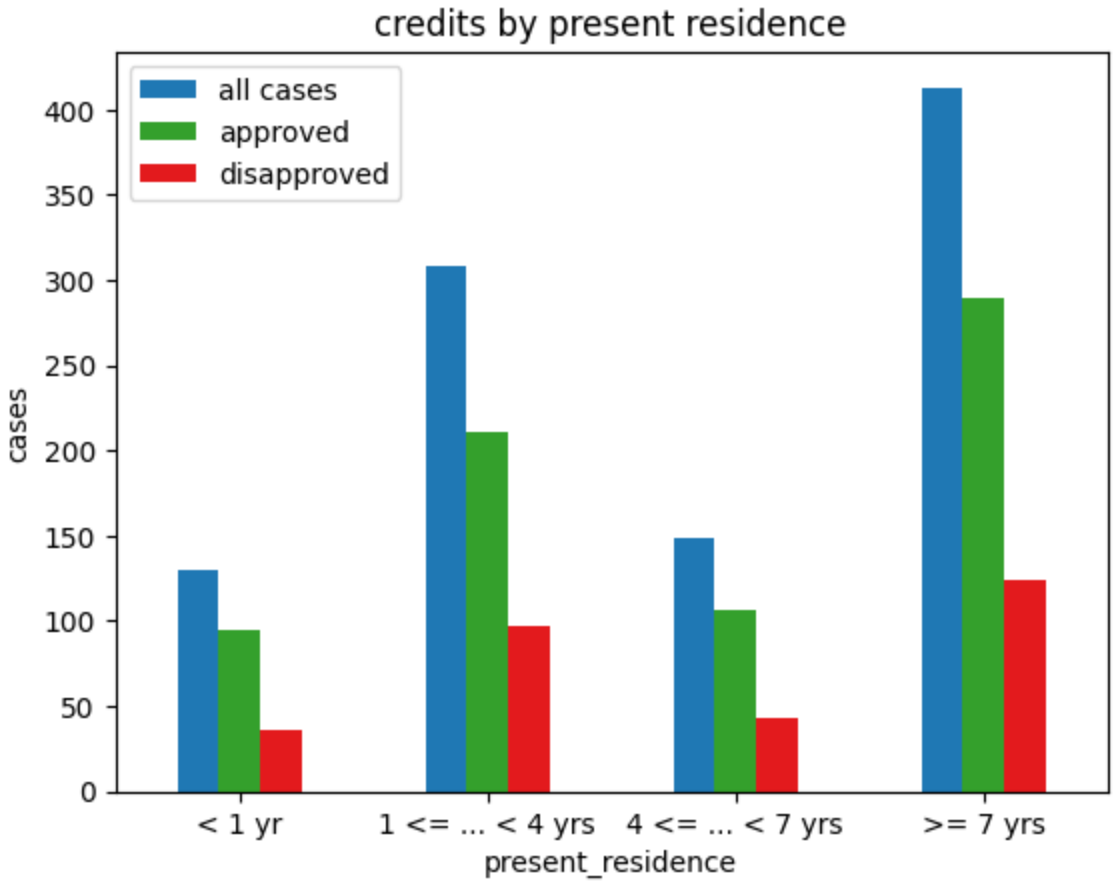
other_debtors	all cases	approved	disapproved
none	907	635	272
guarantor	52	42	10
co-applicant	41	23	18



In [32]:
```
#present_residence
df1tmp = df1[['present_residence', 'all cases', 'approved', 'disapproved']].groupb
y('present_residence').sum()
present_residence_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: present_residence_mod[i] for i in range(len(pre
sent_residence_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by present residence')
df1tmp
```

Out[32]:

present_residence	all cases	approved	disapproved
< 1 yr	130	94	36
1 <= ... < 4 yrs	308	211	97
4 <= ... < 7 yrs	149	106	43
>= 7 yrs	413	289	124

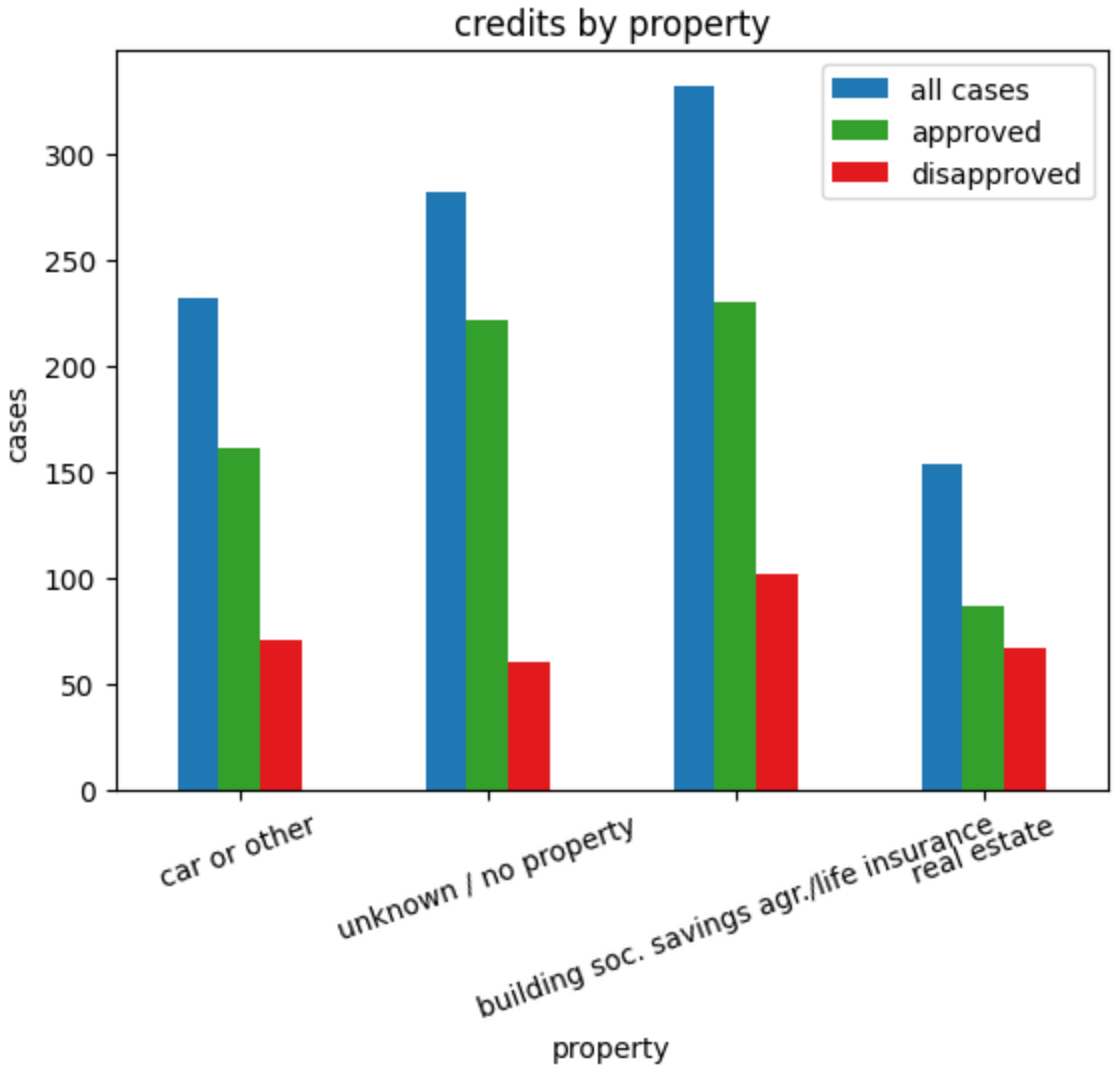


*я знаю что можно отображать несколько графиков за раз, но повторюсь, я отображаю их все в попытке что-то найти, какую-то закономерность среди всего этого, чтобы дать более правильное решение

In [33]:
```
#property
df1tmp = df1[['property', 'all cases', 'approved', 'disapproved']].groupby('property').sum()
property_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: property_mod[i] for i in range(len(property_mod))})
df1tmp.plot.bar(rot = 20, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by property')
df1tmp
```

Out[33]:

property	all cases	approved	disapproved
car or other	232	161	71
unknown / no property	282	222	60
building soc. savings agr./life insurance	332	230	102
real estate	154	87	67



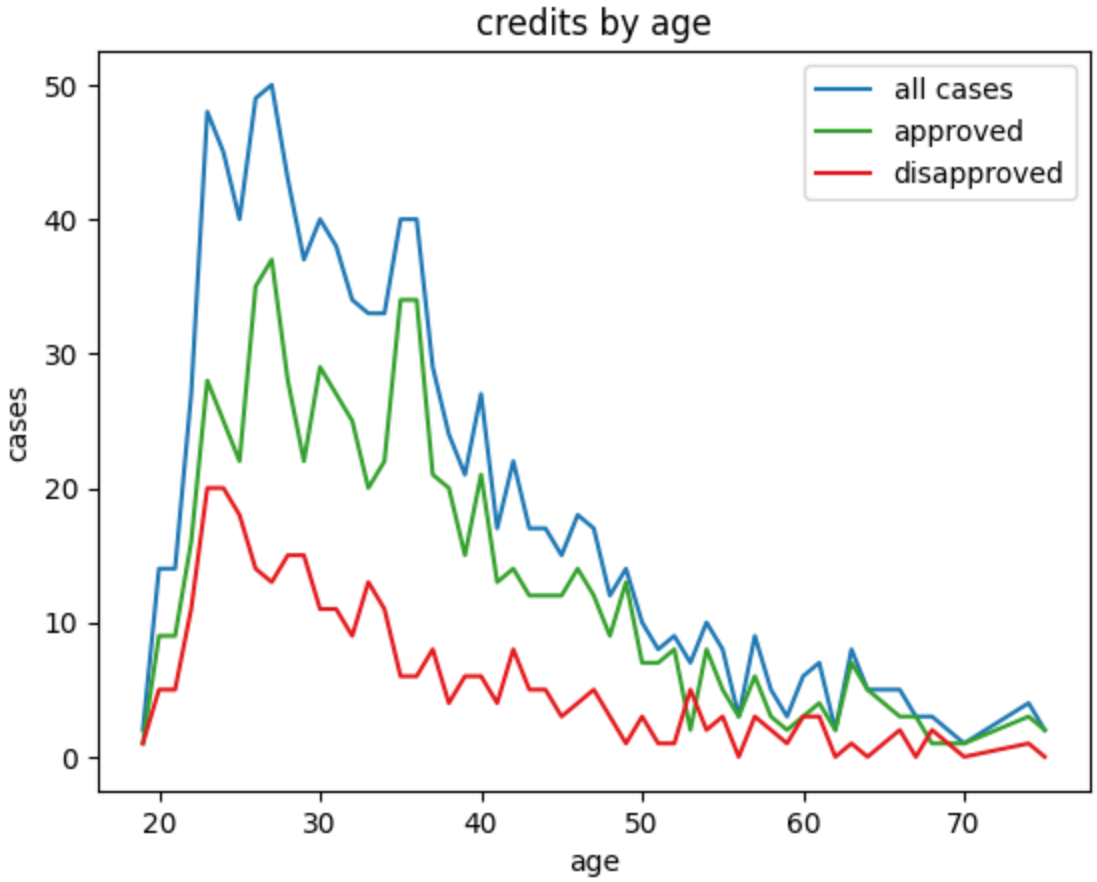
In [34]:

```
#age
df1tmp = df1[['age', 'all cases', 'approved', 'disapproved']].groupby('age').sum()
#age_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp.plot(color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved':
'#e31a1d'},
            ylabel = 'cases', title = 'credits by age')
df1tmp
```

Out[34]:

age	all cases	approved	disapproved
19	2	1	1
20	14	9	5
21	14	9	5
22	27	16	11
23	48	28	20
24	45	25	20
25	40	22	18
26	49	35	14
27	50	37	13
28	43	28	15
29	37	22	15
30	40	29	11
31	38	27	11
32	34	25	9
33	33	20	13
34	33	22	11
35	40	34	6
36	40	34	6
37	29	21	8
38	24	20	4
39	21	15	6
40	27	21	6
41	17	13	4
42	22	14	8
43	17	12	5
44	17	12	5
45	15	12	3
46	18	14	4
47	17	12	5
48	12	9	3
49	14	13	1
50	10	7	3
51	8	7	1
52	9	8	1
53	7	2	5
54	10	8	2
55	8	5	3

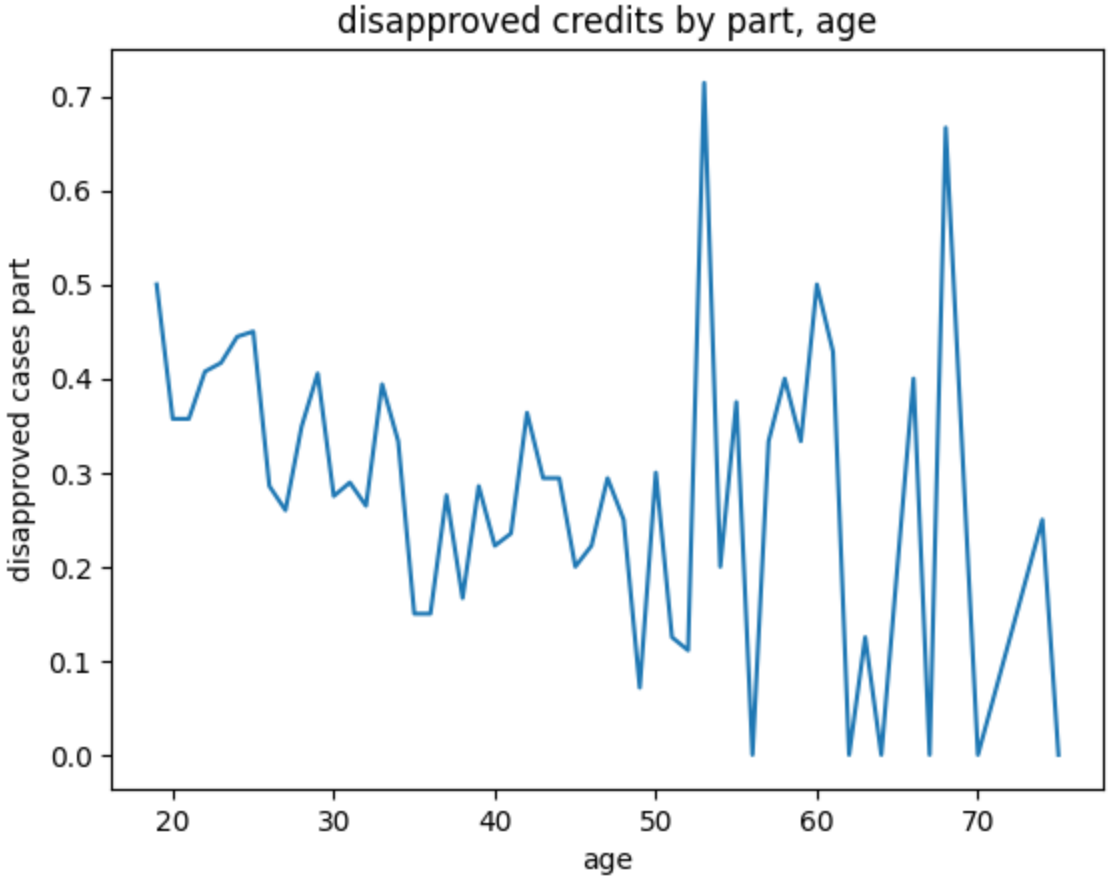
age	all cases	approved	disapproved
56	3	3	0
57	9	6	3
58	5	3	2
59	3	2	1
60	6	3	3
61	7	4	3
62	2	2	0
63	8	7	1
64	5	5	0
65	5	4	1
66	5	3	2
67	3	3	0
68	3	1	2
70	1	1	0
74	4	3	1
75	2	2	0



In [35]:
```
df1tmp_part = df1tmp['disapproved']/df1tmp['all cases']
df1tmp_part.plot(color='#1f78b4', ylabel = 'disapproved cases part',
                 title = 'disapproved credits by part, age')
df1tmp_part
```

Out[35]:
age	
19	0.500000
20	0.357143
21	0.357143
22	0.407407
23	0.416667
24	0.444444
25	0.450000
26	0.285714
27	0.260000
28	0.348837
29	0.405405
30	0.275000
31	0.289474
32	0.264706
33	0.393939
34	0.333333
35	0.150000
36	0.150000
37	0.275862
38	0.166667
39	0.285714
40	0.222222
41	0.235294
42	0.363636
43	0.294118
44	0.294118
45	0.200000
46	0.222222
47	0.294118
48	0.250000
49	0.071429
50	0.300000
51	0.125000
52	0.111111
53	0.714286
54	0.200000
55	0.375000
56	0.000000
57	0.333333
58	0.400000
59	0.333333
60	0.500000
61	0.428571
62	0.000000
63	0.125000
64	0.000000
65	0.200000
66	0.400000
67	0.000000
68	0.666667
70	0.000000
74	0.250000
75	0.000000

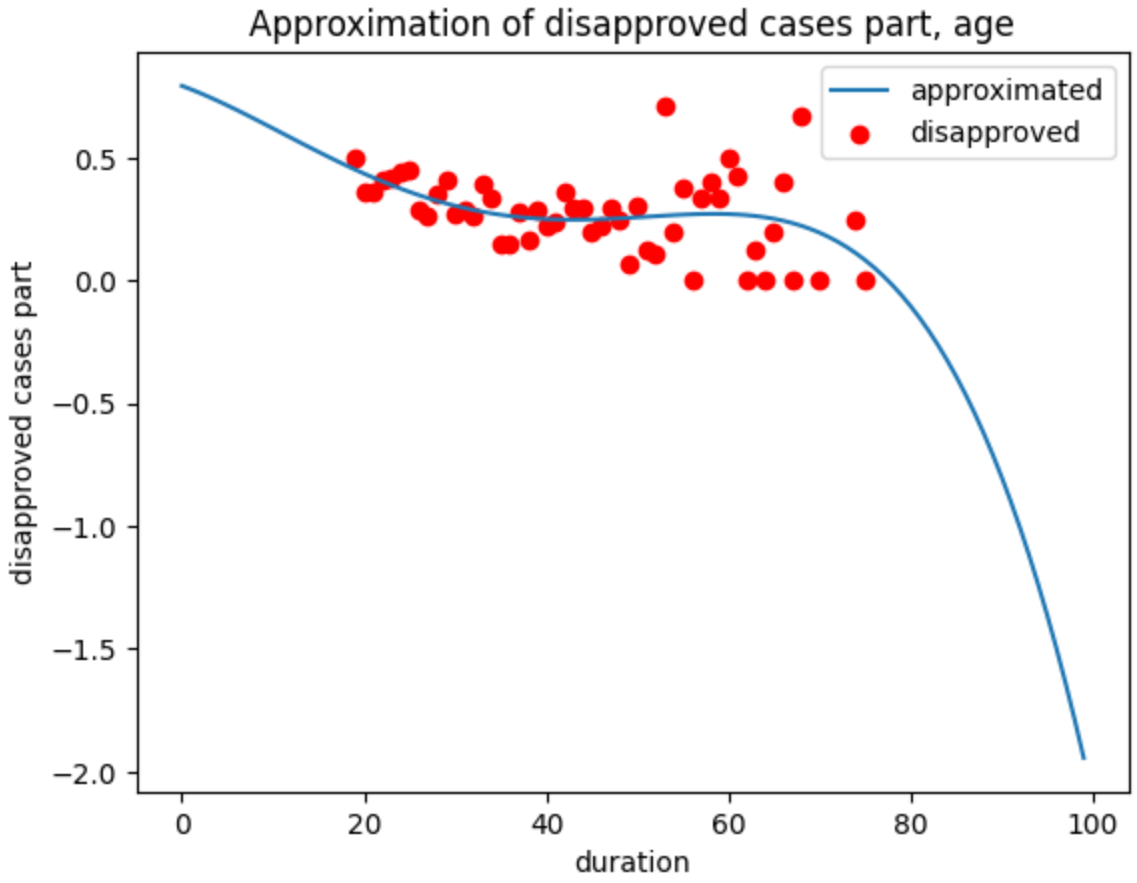
dtype: float64


disapproved credits by part, age
disapproved cases part
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
20
30
40
50
60
70
age

In [36]:
```
# попытка аппроксимировать функцией
# методом наименьших квадратов к x^5
A = np.array([[x**5, x**4, x**3, x**2, x, 1] for x in df1tmp_part.index])
B = df1tmp_part.to_numpy()
left = np.transpose(A)@A
right = np.transpose(A)@B
# а вы знали что numpy умеет решать уравнения?
age_pol = np.linalg.solve(left, right)
age_func = lambda x: age_pol[0]*x**5+age_pol[1]*x**4+\
                     age_pol[2]*x**3+age_pol[3]*x**2+\
                     age_pol[4]*x   +age_pol[5]

X = np.arange(0, 100)
Y = np.array([age_func(x) for x in X])
plt.plot(X,Y)
plt.scatter(df1tmp_part.index, B, color = 'Red')
plt.legend(['approximated', 'disapproved'])
plt.xlabel('duration')
plt.ylabel('disapproved cases part')
plt.title('Approximation of disapproved cases part, age')
age_pol
```

Out[36]:
```
array([-1.69558885e-14, -1.65094301e-07,  2.05060005e-05, -5.52865111e-04,
       -1.36906316e-02,  7.92763024e-01])
```

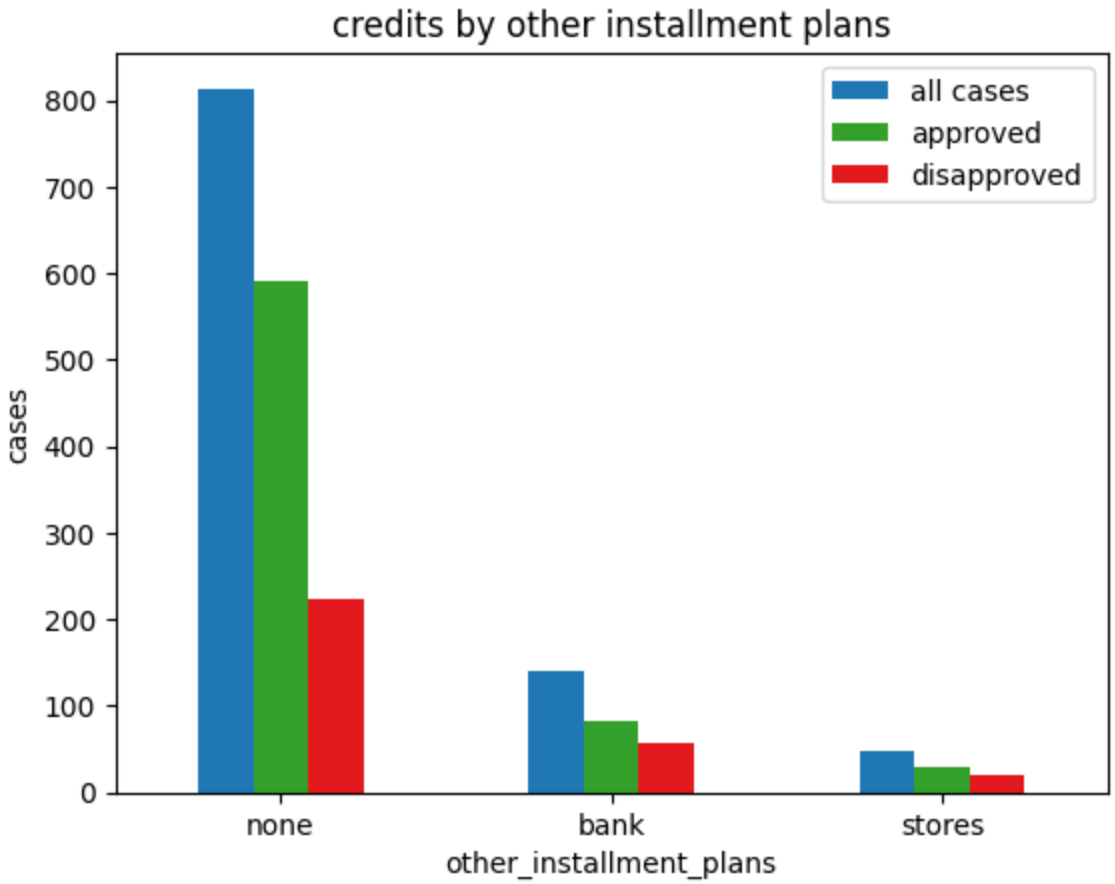


А тут даже допустимо что оно в минус уходит

In [37]:
```
#other_installment_plans
df1tmp = df1[['other_installment_plans', 'all cases', 'approved', 'disapproved']].
groupby('other_installment_plans').sum()
other_installment_plans_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy
()
df1tmp = df1tmp.rename(index = {i: other_installment_plans_mod[i] for i in range(l
en(other_installment_plans_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by other installment plans')
df1tmp
```

Out[37]:

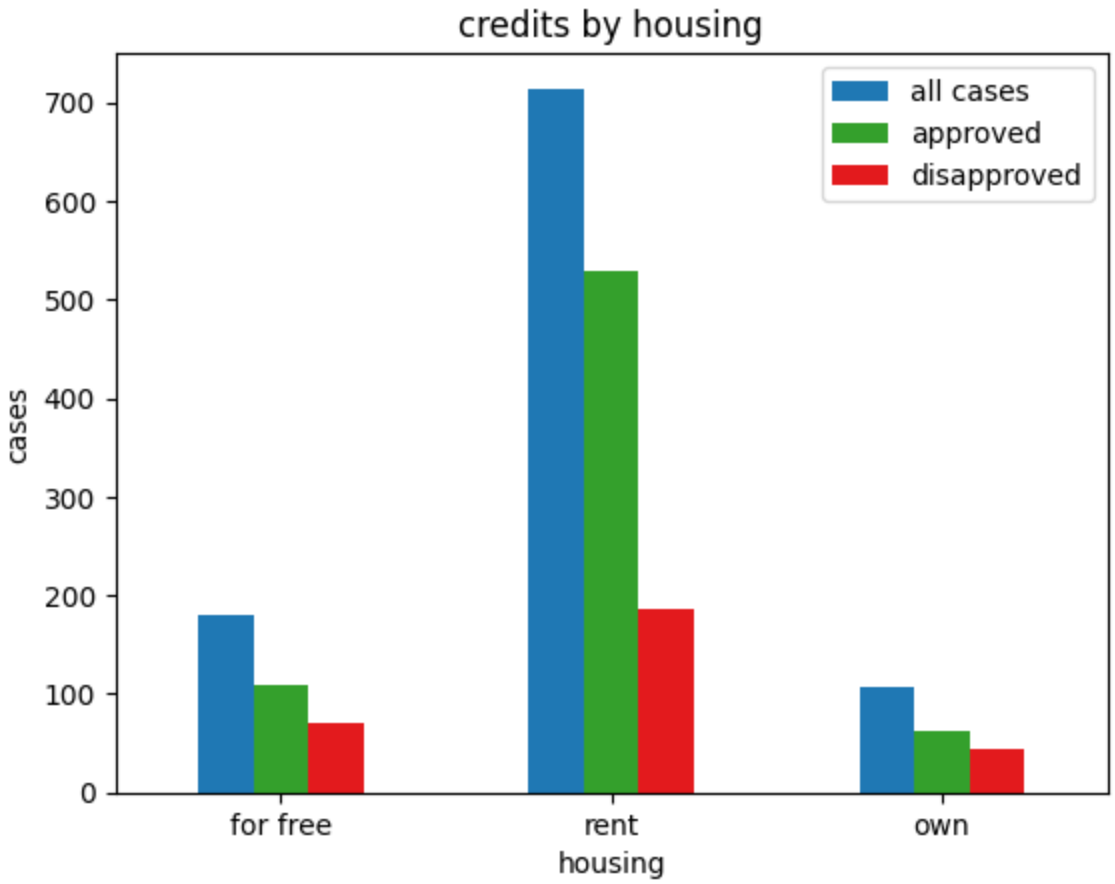
other_installment_plans	all cases	approved	disapproved
none	814	590	224
bank	139	82	57
stores	47	28	19



In [38]:
```
#housing
df1tmp = df1[['housing', 'all cases', 'approved', 'disapproved']].groupby('housing').sum()
housing_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: housing_mod[i] for i in range(len(housing_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by housing')
df1tmp
```

Out[38]:

housing	all cases	approved	disapproved
for free	179	109	70
rent	714	528	186
own	107	63	44

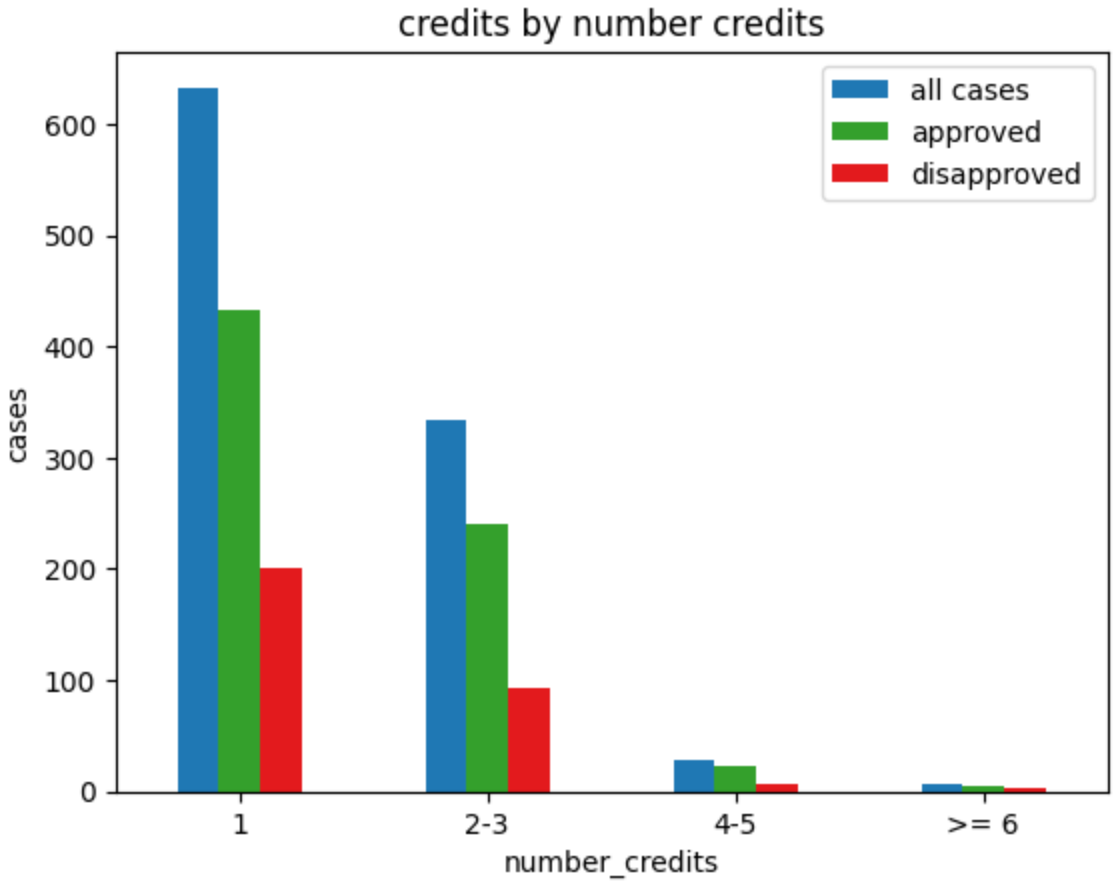


Потенциальная связь с present_residence

In [39]:
```
#number_credits
df1tmp = df1[['number_credits', 'all cases', 'approved', 'disapproved']].groupby(
'number_credits').sum()
number_credits_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: number_credits_mod[i] for i in range(len(number
_credits_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by number credits')
df1tmp
```

Out[39]:

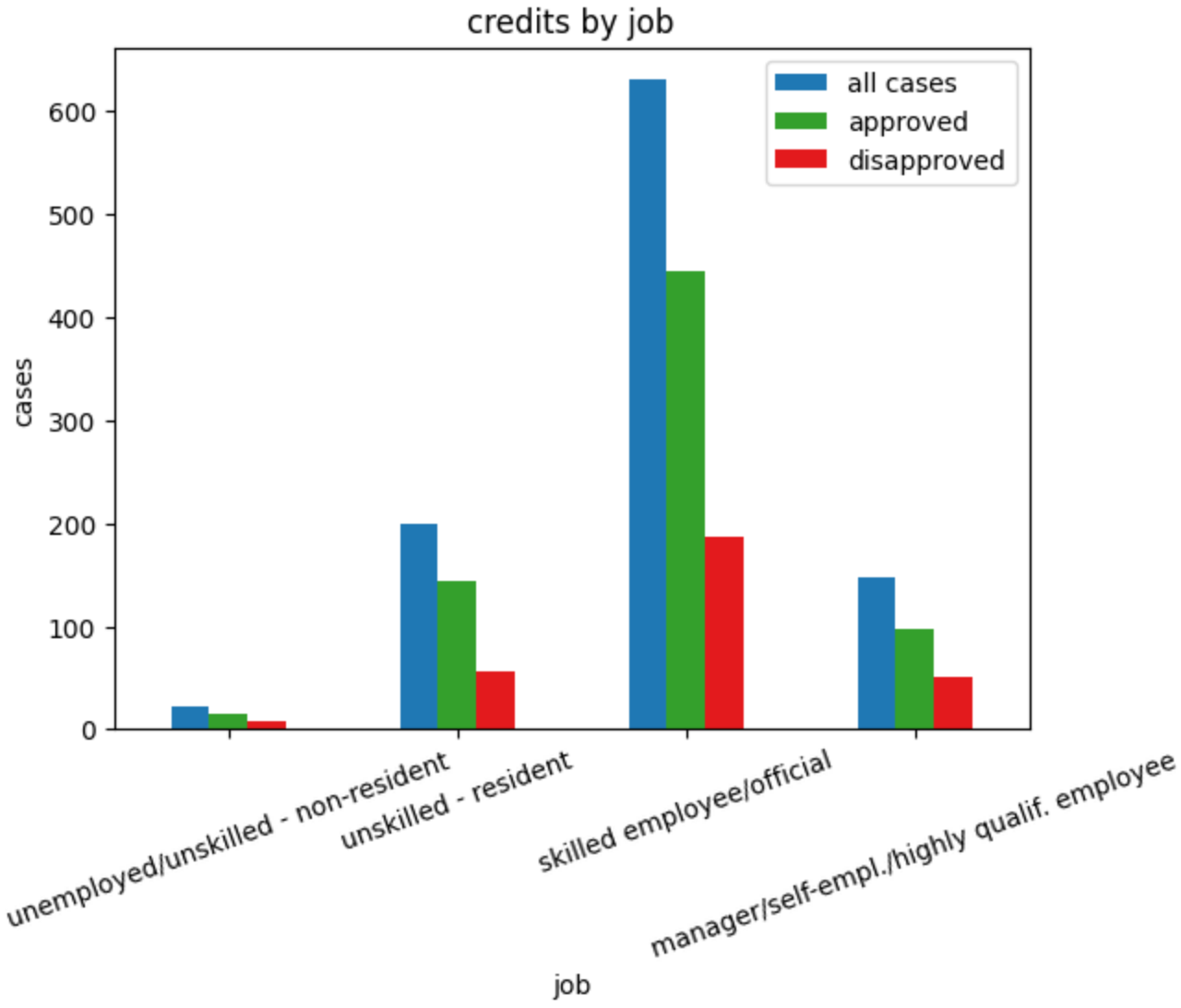
number_credits	all cases	approved	disapproved
1	633	433	200
2-3	333	241	92
4-5	28	22	6
>= 6	6	4	2



In [40]:
```
#job
df1tmp = df1[['job', 'all cases', 'approved', 'disapproved']].groupby('job').sum()
job_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: job_mod[i] for i in range(len(job_mod))})
df1tmp.plot.bar(rot = 20, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by job')
df1tmp
```

Out[40]:

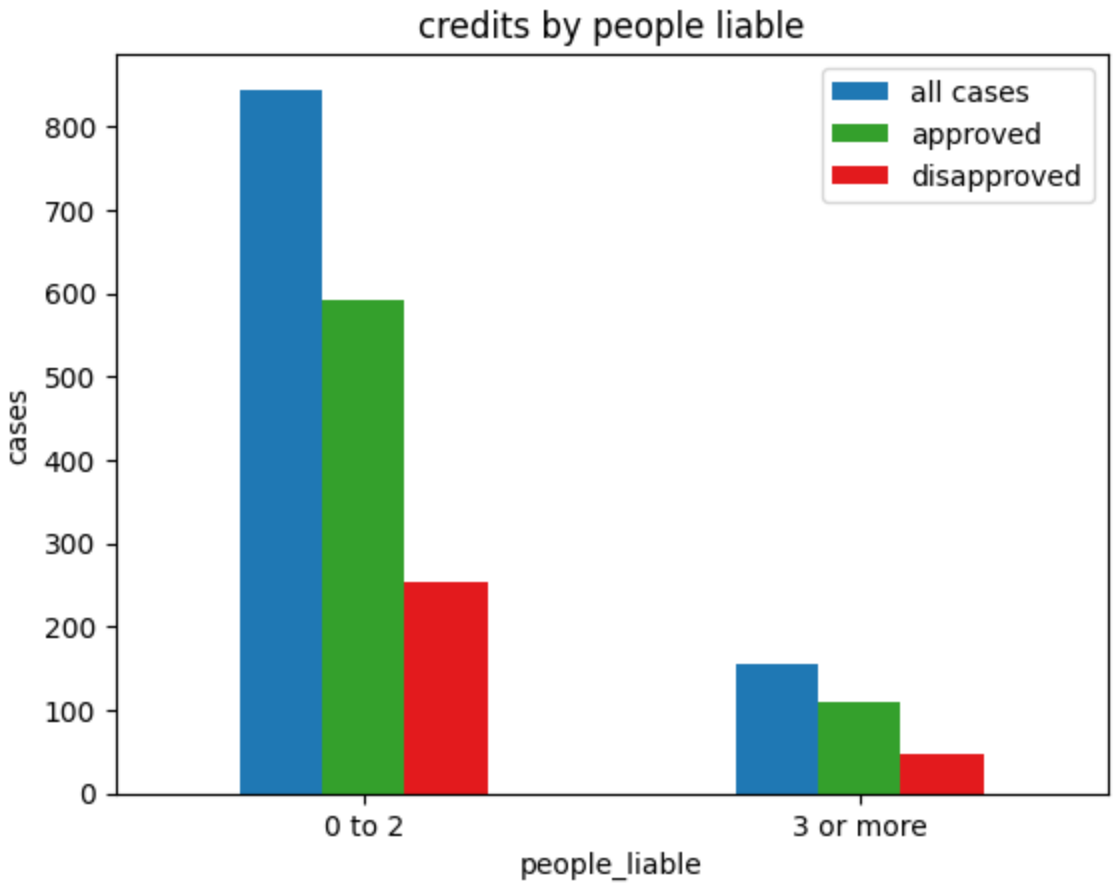
job	all cases	approved	disapproved
unemployed/unskilled - non-resident	22	15	7
unskilled - resident	200	144	56
skilled employee/official	630	444	186
manager/self-empl./highly qualif. employee	148	97	51



In [41]:
```
#people_liable
df1tmp = df1[['people_liable', 'all cases', 'approved', 'disapproved']].groupby('people_liable').sum()
people_liable_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: people_liable_mod[i] for i in range(len(people_liable_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'disapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by people liable')
df1tmp
```

Out[41]:

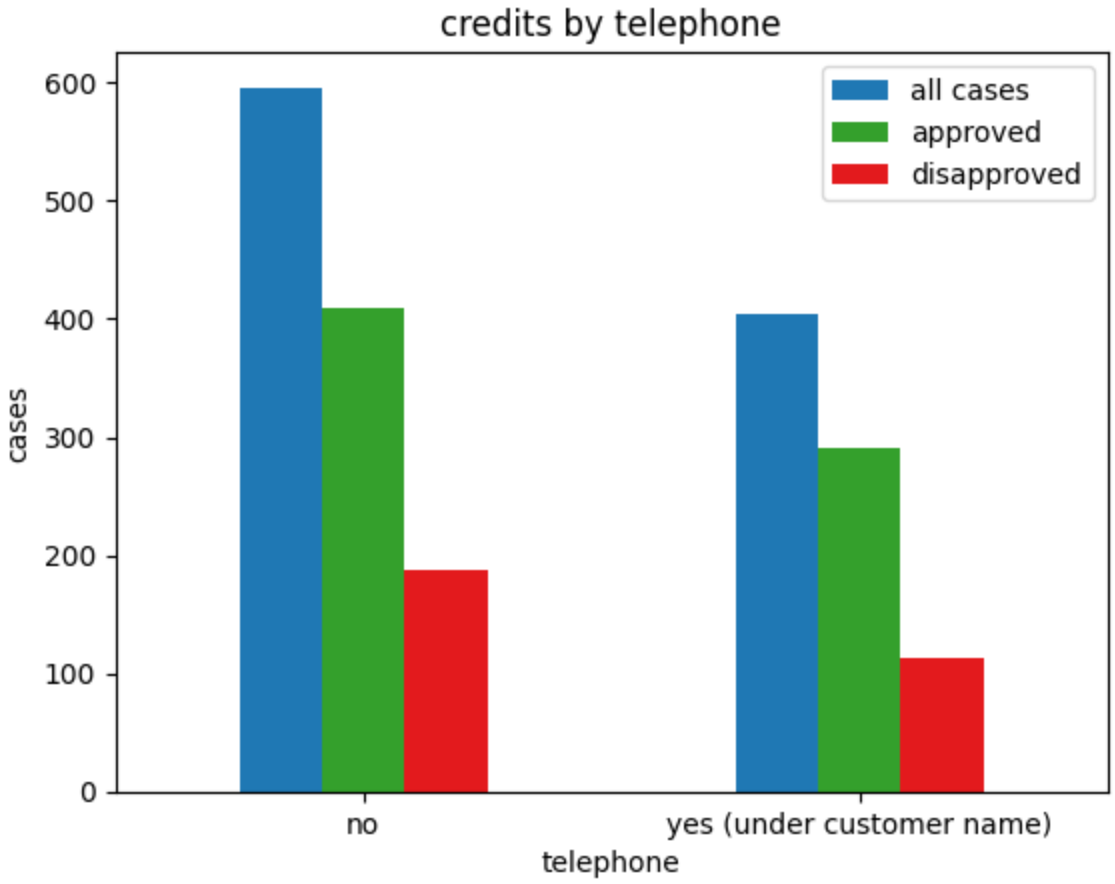
people_liable	all cases	approved	disapproved
0 to 2	845	591	254
3 or more	155	109	46



In [42]:
```
#telephone
df1tmp = df1[['telephone', 'all cases', 'approved', 'disapproved']].groupby('telep
hone').sum()
telephone_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: telephone_mod[i] for i in range(len(telephone_m
od))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by telephone')
df1tmp
```

Out[42]:

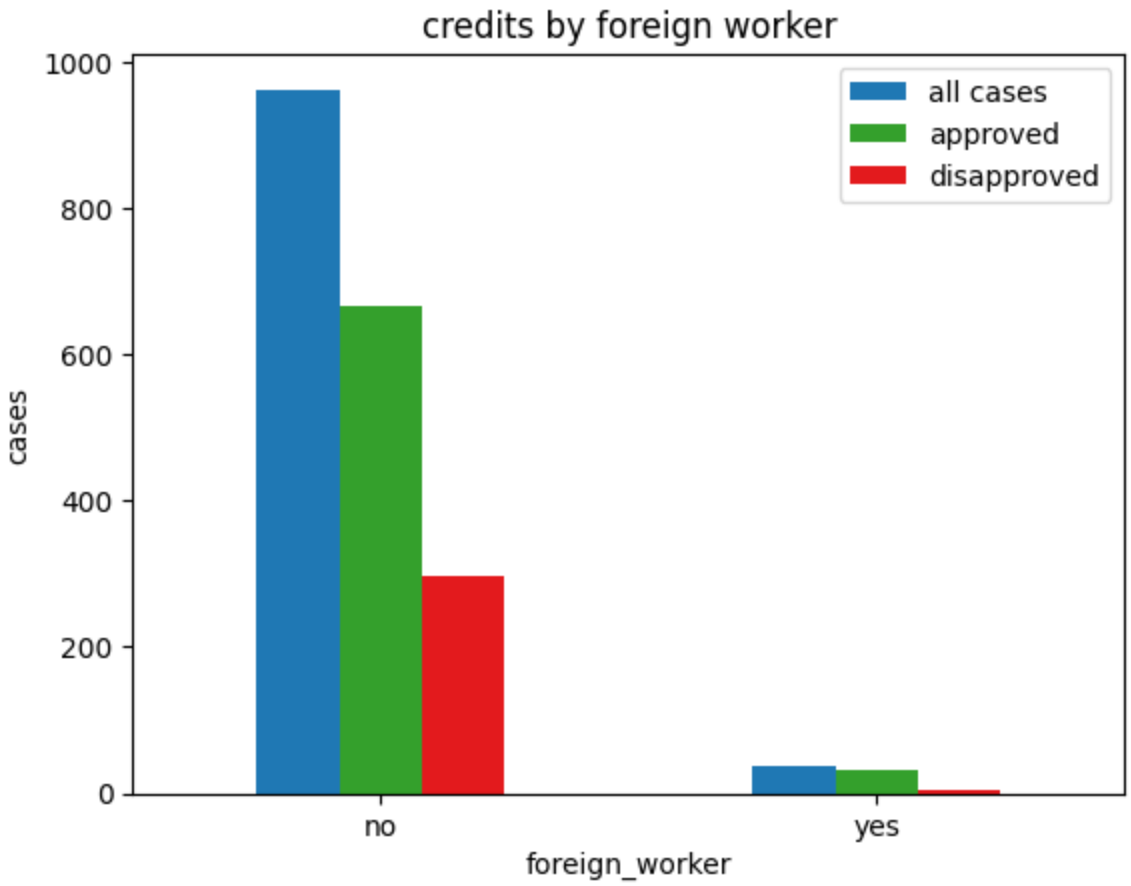
telephone	all cases	approved	disapproved
no	596	409	187
yes (under customer name)	404	291	113



In [43]:
```
#foreign_worker
df1tmp = df1[['foreign_worker', 'all cases', 'approved', 'disapproved']].groupby(
'foreign_worker').sum()
foreign_worker_fac = (df1tmp['disapproved']/df1tmp['all cases']).to_numpy()
df1tmp = df1tmp.rename(index = {i: foreign_worker_mod[i] for i in range(len(foreig
n_worker_mod))})
df1tmp.plot.bar(rot = 0, color={'all cases': '#1f78b4', 'approved': '#34a02c', 'di
sapproved': '#e31a1d'},
                ylabel = 'cases', title = 'credits by foreign worker')
df1tmp
```

Out[43]:

foreign_worker	all cases	approved	disapproved
no	963	667	296
yes	37	33	4



Сама функция

In [44]:
```
#fac = [status_fac, duration_func, credit_history_fac, purpose_fac, amount_func, s
avings_fac,
#       employment_duration_fac, installment_rate_fac, personal_status_sex_fac, ot
her_debtors_fac,
#       present_residence_fac, property_fac, age_func, other_installment_plans_fa
c, housing_fac,
#       number_credits_fac, job_fac, people_liable_fac, telephone_fac, foreign_wor
ker_fac]
def calculate_credit_risk(st, du, cr, pu, am, sa, em, ins, pe, ot, pre, pro, ag, o
th, ho, nu, jo, peo, te, fo):
    return status_fac[st]+duration_func(du)+credit_history_fac[cr]+purpose_fac[pu]
+amount_func(am)+\
           savings_fac[sa]+employment_duration_fac[em]+installment_rate_fac[ins]+p
ersonal_status_sex_fac[pe]+\
           other_debtors_fac[ot]+present_residence_fac[pre]+property_fac[pro]+age_
func(ag)+\
           other_installment_plans_fac[oth]+housing_fac[ho]+number_credits_fac[nu]
+job_fac[jo]+\
           people_liable_fac[peo]+telephone_fac[te]+foreign_worker_fac[fo]
# функция просто перебирает
```

In [45]:
```
df1['calculated'] = [calculate_credit_risk(*df1.iloc[i,:20]) for i in range(1000)]
#df1['calculated approval'] = df1['calculated']>6
df1
```

Out[45]:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	pers
0	0	18	3	0	1049	0	1	0	
1	0	9	3	1	2799	0	2	2	
2	1	12	4	2	841	1	3	2	
3	0	12	3	1	2122	0	2	1	
4	0	12	3	1	2171	0	2	0	
...	...	...	...	...	...	...	...	...	
995	0	24	4	3	1987	0	2	2	
996	0	24	4	1	2303	0	4	0	
997	3	21	3	1	12680	4	4	0	
998	1	12	4	3	6468	4	0	2	
999	0	30	4	0	6350	4	4	0	

1000 rows × 24 columns

In [49]:
```
df1['calculated approval'] = [1 if df1.loc[i, 'calculated']>6 else 0 for i in df1.
index]
df1
```

Out[49]:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	pers
0	0	18	3	0	1049	0	1	0	
1	0	9	3	1	2799	0	2	2	
2	1	12	4	2	841	1	3	2	
3	0	12	3	1	2122	0	2	1	
4	0	12	3	1	2171	0	2	0	
...	...	...	...	...	...	...	...	...	
995	0	24	4	3	1987	0	2	2	
996	0	24	4	1	2303	0	4	0	
997	3	21	3	1	12680	4	4	0	
998	1	12	4	3	6468	4	0	2	
999	0	30	4	0	6350	4	4	0	

1000 rows × 25 columns

In [50]:
```
df1[df1['calculated approval']==df1['approved']]
```

Out[50]:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	pers
0	0	18	3	0	1049	0	1	0	
9	1	24	4	3	3758	2	0	3	
11	0	30	3	4	6187	1	3	3	
13	1	48	2	5	7582	1	0	2	
26	0	12	4	0	652	0	4	0	
...	...	...	...	...	...	...	...	...	
981	3	18	2	2	2169	0	2	0	
985	3	12	3	2	2292	0	0	0	
987	0	12	4	3	674	1	3	0	
992	0	18	4	4	7511	4	4	3	
998	1	12	4	3	6468	4	0	2	

261 rows × 25 columns

Я предугадал лишь 261 строк из 1000. Скорее всего потому что разные столбцы имеют разный фактор-множитель, который в теории можно определить по выборочной дисперсии значений категорий столбца и их соотношениям. Но тут и так страниц уже больше чем в моем отчете по матстату..

In [52]:
```
plt.imshow(plt.imread('./cat2.jpg'));
```

Просто хочу сказать что для этого тестового задания я отдельно учил панду, до этого за неделю учил матплот и реализовывал его в этом самом матстате. Я очень устал на пока, а еще много дел..

In [ ]:
```
```