Business requirement

Dataset: `loan_status_2007_2014`

- Sử dụng các phương pháp data mining để tiền xử lý dữ liệu
- Đưa ra insights từ các performance metrics và visualize dữ liệu
- Sử dụng bộ dữ liệu về khoản vay để dự đoán khả năng vỡ nợ của khách hàng

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

pd.options.display.max_rows = 4000
pd.options.display.max_columns = None
pd.options.display.float_format = '{:.3f}'.format
np.set_printoptions(suppress=True, precision=5)

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv(r'E:\learn st new\Credit score\Scorecard\loan_data_2007_2014.cs
df.shape
```

Out[2]:
```
(466285, 74)
```

Xử lý biến Target

- Những giao dịch `Current` và giao dịch `In Grace Period` sẽ loại bỏ

In [3]:
```
df =  df[(df.loan_status != 'In Grace Period') & (df.loan_status != 'Current')]
```

In [4]:
```
#kiểm tra lại
df.loan_status.value_counts()
```

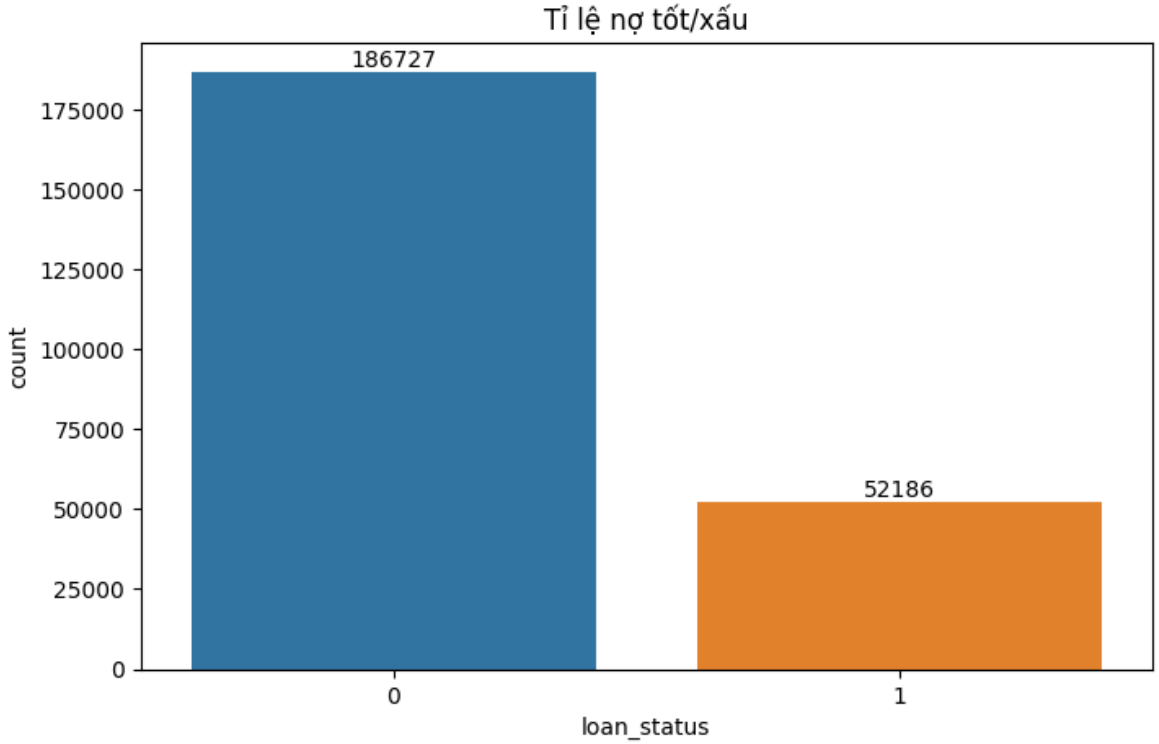
Out[4]:
```
Fully Paid                                             184739
Charged Off                                             42475
Late (31-120 days)                                       6900
Does not meet the credit policy. Status:Fully Paid       1988
Late (16-30 days)                                        1218
Default                                                   832
Does not meet the credit policy. Status:Charged Off       761
Name: loan_status, dtype: int64
```

In [5]:
```
non_default = ['Fully Paid','Does not meet the credit policy. Status:Fully Paid'
df['loan_status'] = np.where(df['loan_status'].isin(non_default), 0, 1)
100*df.loan_status.value_counts()/df.shape[0]
```

Out[5]:
```
0   78.157
1   21.843
Name: loan_status, dtype: float64
```

In [6]:
```
# Biểu đồ so sánh số lượng nợ
plt.subplots(figsize=(8,5))
plt.title('Tỉ lệ nợ tốt/xấu')
a = sns.countplot(x = df.loan_status)
a.bar_label(a.containers[0])
```

Out[6]:
```
[Text(0, 0, '186727'), Text(0, 0, '52186')]
```



Xử lý dữ liệu khuyết thiếu

- Đối với những dữ liệu `khuyết thiếu trên 70%`, loại bỏ ra khỏi bộ dữ liệu
- Đối với những dữ liệu category có `số lượng lớn và chứa khuyết thiếu`, nên loại ra khỏi mô hình
- Đối với những dữ liệu có `khuyết thiếu thấp`, nên xử lí nhanh bằng cách loại bỏ các dòng
- Đối với những dữ liệu `chỉ có 1 loại`, đưa ra khỏi mô hình
- Có 3 cột có dữ liệu khuyết thiếu giống nhau, do dữ liệu `tập trung` về 1 phía nên loại

In [7]:
```
# Dữ liệu memberid và id trùng nhau và không có ý nghĩa trong mô hình, nên bỏ kh
df = df.drop(columns = ['member_id', 'id'])
```

In [8]:
```
# Kiểm tra khuyết thiếu
pct_missing_value = 100* df.isnull().sum()/df.shape[0]
pct_missing_value.head(10)
```

Out[8]:
```
loan_amnt          0.000
funded_amnt        0.000
funded_amnt_inv    0.000
term               0.000
int_rate           0.000
installment        0.000
grade              0.000
sub_grade          0.000
emp_title          5.641
emp_length         3.861
dtype: float64
```

In [9]:
```
df.emp_title.value_counts().shape[0],df.title.value_counts().shape[0],df.sub_gra
```

Out[9]:
```
(129889, 49930, 35)
```

In [9]:
```
# Xử lý các biến định tính
cat_col = []
for x in df.dtypes.index:
    if df.dtypes[x] == 'object':
        cat_col.append(x)

for col in cat_col:
    print(col)
    print(df[col].unique())
```

```
term
[' 36 months' ' 60 months']
grade
['B' 'C' 'A' 'E' 'F' 'D' 'G']
sub_grade
['B2' 'C4' 'C5' 'C1' 'A4' 'E1' 'F2' 'B5' 'C3' 'B1' 'D1' 'A1' 'B3' 'B4'
 'C2' 'D2' 'A3' 'A5' 'D5' 'A2' 'E4' 'D3' 'D4' 'F3' 'E3' 'F1' 'E5' 'G4'
 'E2' 'G2' 'G1' 'F5' 'F4' 'G5' 'G3']
emp_title
[nan 'Ryder' 'AIR RESOURCES BOARD' ... 'MecÃ¡nica'
 'Chief of Interpretation (Park Ranger)' 'Server Engineer Lead']
emp_length
['10+ years' '< 1 year' '3 years' '9 years' '4 years' '5 years' '1 year'
 '6 years' '2 years' '7 years' '8 years' nan]
home_ownership
['RENT' 'OWN' 'MORTGAGE' 'OTHER' 'NONE' 'ANY']
verification_status
['Verified' 'Source Verified' 'Not Verified']
issue_d
['Dec-11' 'Nov-11' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11' 'Jun-11' 'May-11'
 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10' 'Oct-10' 'Sep-10'
 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Mar-10' 'Feb-10' 'Jan-10'
 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Aug-09' 'Jul-09' 'Jun-09' 'May-09'
 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08' 'Nov-08' 'Oct-08' 'Sep-08'
 'Aug-08' 'Jul-08' 'Jun-08' 'May-08' 'Apr-08' 'Mar-08' 'Feb-08' 'Jan-08'
 'Dec-07' 'Nov-07' 'Oct-07' 'Sep-07' 'Aug-07' 'Jul-07' 'Jun-07' 'Dec-13'
 'Nov-13' 'Oct-13' 'Sep-13' 'Aug-13' 'Jul-13' 'Jun-13' 'May-13' 'Apr-13'
 'Mar-13' 'Feb-13' 'Jan-13' 'Dec-12' 'Nov-12' 'Oct-12' 'Sep-12' 'Aug-12'
 'Jul-12' 'Jun-12' 'May-12' 'Apr-12' 'Mar-12' 'Feb-12' 'Jan-12' 'Dec-14'
 'Nov-14' 'Oct-14' 'Sep-14' 'Aug-14' 'Jul-14' 'Jun-14' 'May-14' 'Apr-14'
 'Mar-14' 'Feb-14' 'Jan-14']
pymnt_plan
['n' 'y']
url
['https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077501'
 'https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077430'
 'https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077175'
 ...
 'https://www.lendingclub.com/browse/loanDetail.action?loan_id=9745590'
 'https://www.lendingclub.com/browse/loanDetail.action?loan_id=9684700'
 'https://www.lendingclub.com/browse/loanDetail.action?loan_id=9604874']
desc
['  Borrower added on 12/22/11 > I need to upgrade my business technologies.<br>'
 '  Borrower added on 12/22/11 > I plan to use this money to finance the motorcyc
le i am looking at. I plan to have it paid off as soon as possible/when i sell my
old bike. I only need this money because the deal im looking at is to good to pas
s up.<br><br>  Borrower added on 12/22/11 > I plan to use this money to finance t
he motorcycle i am looking at. I plan to have it paid off as soon as possible/whe
n i sell my old bike.I only need this money because the deal im looking at is to
good to pass up. I have finished college with an associates degree in business an
d its takingmeplaces<br>'
 nan ...
 '  Borrower added on 12/11/13 > A Diamond for an engagement and wedding expense
s.<br>'
 '  Borrower added on 12/10/13 > all my loans in one auto payment<br>'
 '  Borrower added on 12/09/13 > consolidate all credit accounts<br>']
purpose
['credit_card' 'car' 'small_business' 'other' 'wedding'
 'debt_consolidation' 'home_improvement' 'major_purchase' 'medical'
 'moving' 'vacation' 'house' 'renewable_energy' 'educational']
```

```
title
['Computer' 'bike' 'real estate business' ... 'Stop the bleeding'
 'LoanGetter' 'Consolidation 01']
zip_code
['860xx' '309xx' '606xx' '917xx' '852xx' '900xx' '958xx' '774xx' '853xx'
 '913xx' '245xx' '951xx' '641xx' '921xx' '067xx' '890xx' '770xx' '335xx'
 '799xx' '605xx' '103xx' '150xx' '326xx' '564xx' '141xx' '080xx' '974xx'
 '934xx' '405xx' '946xx' '445xx' '850xx' '292xx' '088xx' '180xx' '029xx'
 '700xx' '010xx' '441xx' '104xx' '061xx' '616xx' '947xx' '914xx' '765xx'
 '980xx' '017xx' '972xx' '752xx' '787xx' '077xx' '540xx' '225xx' '440xx'
 '437xx' '559xx' '912xx' '325xx' '300xx' '923xx' '352xx' '013xx' '146xx'
 '074xx' '786xx' '937xx' '331xx' '115xx' '191xx' '114xx' '908xx' '902xx'
 '750xx' '950xx' '329xx' '226xx' '992xx' '614xx' '083xx' '100xx' '926xx'
 '931xx' '712xx' '060xx' '707xx' '342xx' '604xx' '895xx' '430xx' '919xx'
 '996xx' '891xx' '935xx' '801xx' '928xx' '233xx' '927xx' '970xx' '211xx'
 '303xx' '070xx' '194xx' '263xx' '403xx' '301xx' '553xx' '993xx' '312xx'
 '432xx' '602xx' '216xx' '151xx' '971xx' '305xx' '334xx' '050xx' '129xx'
 '925xx' '483xx' '760xx' '961xx' '200xx' '085xx' '981xx' '330xx' '601xx'
 '117xx' '063xx' '920xx' '543xx' '775xx' '570xx' '221xx' '985xx' '275xx'
 '236xx' '148xx' '028xx' '450xx' '532xx' '729xx' '321xx' '959xx' '941xx'
 '955xx' '217xx' '880xx' '660xx' '062xx' '193xx' '857xx' '306xx' '271xx'
 '142xx' '956xx' '983xx' '945xx' '672xx' '112xx' '802xx' '187xx' '630xx'
 '435xx' '488xx' '287xx' '705xx' '318xx' '549xx' '212xx' '347xx' '274xx'
 '265xx' '785xx' '027xx' '089xx' '813xx' '260xx' '201xx' '349xx' '322xx'
 '075xx' '124xx' '940xx' '967xx' '111xx' '773xx' '997xx' '076xx' '538xx'
 '021xx' '304xx' '113xx' '234xx' '308xx' '809xx' '071xx' '363xx' '296xx'
 '240xx' '011xx' '207xx' '140xx' '336xx' '619xx' '208xx' '618xx' '014xx'
 '644xx' '276xx' '109xx' '631xx' '243xx' '960xx' '181xx' '922xx' '975xx'
 '105xx' '986xx' '218xx' '652xx' '782xx' '410xx' '328xx' '719xx' '982xx'
 '065xx' '081xx' '954xx' '346xx' '480xx' '442xx' '025xx' '122xx' '282xx'
 '120xx' '082xx' '766xx' '229xx' '840xx' '744xx' '933xx' '451xx' '907xx'
 '159xx' '333xx' '293xx' '701xx' '984xx' '811xx' '597xx' '957xx' '165xx'
 '720xx' '119xx' '359xx' '084xx' '969xx' '924xx' '531xx' '716xx' '337xx'
 '841xx' '323xx' '740xx' '179xx' '805xx' '285xx' '551xx' '658xx' '944xx'
 '232xx' '905xx' '600xx' '327xx' '711xx' '906xx' '444xx' '856xx' '777xx'
 '072xx' '554xx' '280xx' '145xx' '537xx' '847xx' '295xx' '829xx' '320xx'
 '131xx' '939xx' '281xx' '064xx' '550xx' '078xx' '452xx' '778xx' '313xx'
 '851xx' '784xx' '804xx' '571xx' '210xx' '988xx' '400xx' '995xx' '023xx'
 '158xx' '657xx' '016xx' '283xx' '019xx' '290xx' '366xx' '066xx' '991xx'
 '968xx' '069xx' '721xx' '439xx' '640xx' '546xx' '751xx' '741xx' '904xx'
 '156xx' '299xx' '087xx' '949xx' '261xx' '222xx' '244xx' '617xx' '018xx'
 '286xx' '759xx' '952xx' '930xx' '911xx' '220xx' '731xx' '730xx' '262xx'
 '338xx' '160xx' '031xx' '054xx' '223xx' '272xx' '152xx' '882xx' '557xx'
 '797xx' '725xx' '130xx' '030xx' '206xx' '324xx' '170xx' '291xx' '161xx'
 '073xx' '647xx' '916xx' '665xx' '209xx' '915xx' '173xx' '761xx' '110xx'
 '086xx' '484xx' '844xx' '020xx' '354xx' '978xx' '757xx' '953xx' '577xx'
 '315xx' '664xx' '186xx' '182xx' '574xx' '800xx' '197xx' '137xx' '314xx'
 '755xx' '973xx' '603xx' '481xx' '780xx' '894xx' '341xx' '178xx' '068xx'
 '565xx' '622xx' '611xx' '288xx' '560xx' '535xx' '499xx' '162xx' '756xx'
 '168xx' '827xx' '541xx' '615xx' '989xx' '037xx' '863xx' '339xx' '367xx'
 '273xx' '052xx' '623xx' '648xx' '918xx' '436xx' '898xx' '674xx' '496xx'
 '294xx' '762xx' '128xx' '903xx' '932xx' '195xx' '650xx' '246xx' '633xx'
 '666xx' '228xx' '015xx' '302xx' '573xx' '998xx' '767xx' '490xx' '350xx'
 '591xx' '254xx' '566xx' '224xx' '637xx' '763xx' '871xx' '494xx' '431xx'
 '402xx' '545xx' '190xx' '184xx' '239xx' '977xx' '297xx' '284xx' '144xx'
 '748xx' '038xx' '310xx' '147xx' '153xx' '544xx' '024xx' '948xx' '576xx'
 '107xx' '846xx' '344xx' '351xx' '754xx' '910xx' '656xx' '357xx' '791xx'
 '493xx' '278xx' '175xx' '530xx' '171xx' '703xx' '620xx' '438xx' '572xx'
 '626xx' '307xx' '319xx' '708xx' '816xx' '625xx' '316xx' '133xx' '612xx'
 '238xx' '166xx' '231xx' '241xx' '826xx' '793xx' '646xx' '188xx' '108xx'
```

```
 '032xx' '653xx' '796xx' '990xx' '219xx' '662xx' '724xx' '456xx' '214xx'
 '237xx' '125xx' '783xx' '737xx' '121xx' '199xx' '548xx' '453xx' '704xx'
 '636xx' '368xx' '828xx' '598xx' '136xx' '610xx' '433xx' '722xx' '743xx'
 '810xx' '706xx' '235xx' '139xx' '361xx' '613xx' '454xx' '746xx' '486xx'
 '033xx' '279xx' '407xx' '448xx' '803xx' '794xx' '457xx' '189xx' '196xx'
 '539xx' '424xx' '492xx' '482xx' '667xx' '845xx' '608xx' '401xx' '362xx'
 '443xx' '627xx' '717xx' '607xx' '963xx' '198xx' '645xx' '713xx' '227xx'
 '883xx' '563xx' '893xx' '079xx' '360xx' '172xx' '422xx' '768xx' '034xx'
 '594xx' '215xx' '628xx' '356xx' '749xx' '806xx' '101xx' '814xx' '255xx'
 '745xx' '495xx' '132xx' '183xx' '864xx' '106xx' '663xx' '943xx' '057xx'
 '094xx' '177xx' '365xx' '897xx' '776xx' '843xx' '116xx' '421xx' '253xx'
 '727xx' '528xx' '808xx' '317xx' '735xx' '447xx' '358xx' '815xx' '250xx'
 '230xx' '790xx' '884xx' '242xx' '012xx' '534xx' '458xx' '404xx' '397xx'
 '870xx' '936xx' '434xx' '655xx' '277xx' '675xx' '053xx' '859xx' '126xx'
 '102xx' '256xx' '673xx' '446xx' '489xx' '258xx' '423xx' '788xx' '270xx'
 '127xx' '176xx' '380xx' '058xx' '635xx' '498xx' '820xx' '599xx' '822xx'
 '830xx' '638xx' '723xx' '449xx' '420xx' '157xx' '726xx' '185xx' '527xx'
 '298xx' '769xx' '257xx' '881xx' '575xx' '624xx' '134xx' '877xx' '781xx'
 '976xx' '718xx' '670xx' '138xx' '026xx' '678xx' '398xx' '497xx' '149xx'
 '875xx' '838xx' '651xx' '364xx' '203xx' '795xx' '427xx' '629xx' '355xx'
 '174xx' '547xx' '567xx' '558xx' '035xx' '999xx' '634xx' '455xx' '143xx'
 '562xx' '779xx' '561xx' '789xx' '812xx' '268xx' '051xx' '406xx' '661xx'
 '758xx' '676xx' '491xx' '734xx' '728xx' '135xx' '411xx' '267xx' '596xx'
 '595xx' '259xx' '163xx' '264xx' '409xx' '118xx' '376xx' '471xx' '154xx'
 '375xx' '747xx' '123xx' '714xx' '590xx' '247xx' '639xx' '416xx' '412xx'
 '425xx' '022xx' '855xx' '874xx' '369xx' '825xx' '266xx' '096xx' '251xx'
 '593xx' '487xx' '609xx' '169xx' '413xx' '155xx' '764xx' '710xx' '408xx'
 '668xx' '056xx' '671xx' '669xx' '167xx' '542xx' '679xx' '792xx' '824xx'
 '249xx' '798xx' '370xx' '485xx' '654xx' '865xx' '289xx' '807xx' '164xx'
 '252xx' '556xx' '353xx' '677xx' '090xx' '371xx' '831xx' '736xx' '007xx'
 '332xx' '468xx' '461xx' '093xx' '248xx' '463xx' '391xx' '381xx' '415xx'
 '462xx' '592xx' '378xx' '414xx' '396xx' '836xx' '044xx' '392xx' '772xx'
 '374xx' '823xx' '395xx' '394xx' '965xx' '390xx' '388xx' '386xx' '040xx'
 '385xx' '379xx' '681xx' '837xx' '373xx' '753xx' '834xx' '383xx' '384xx'
 '372xx' '833xx' '522xx' '523xx' '474xx' '465xx' '689xx' '473xx' '041xx'
 '685xx' '479xx' '469xx' '738xx' '739xx' '418xx' '204xx' '059xx' '878xx'
 '460xx' '426xx' '514xx' '500xx' '503xx' '832xx' '691xx' '470xx' '036xx'
 '466xx' '476xx' '377xx' '477xx' '472xx' '979xx' '464xx' '467xx' '475xx'
 '478xx' '382xx' '680xx' '873xx' '049xx' '994xx' '879xx' '502xx' '942xx'
 '417xx' '091xx' '962xx' '643xx' '821xx' '340xx' '393xx' '682xx' '311xx'
 '387xx' '929xx' '389xx' '938xx' '524xx' '510xx' '909xx' '516xx' '587xx'
 '043xx' '098xx']
addr_state
['AZ' 'GA' 'IL' 'CA' 'TX' 'VA' 'MO' 'CT' 'UT' 'FL' 'NY' 'PA' 'MN' 'NJ'
 'OR' 'KY' 'OH' 'SC' 'RI' 'LA' 'MA' 'WA' 'WI' 'AL' 'NV' 'AK' 'CO' 'MD'
 'WV' 'VT' 'MI' 'DC' 'SD' 'NC' 'AR' 'NM' 'KS' 'HI' 'OK' 'MT' 'WY' 'NH'
 'DE' 'MS' 'TN' 'IA' 'NE' 'ID' 'IN' 'ME']
earliest_cr_line
['Jan-85' 'Apr-99' 'Nov-01' 'Feb-96' 'Nov-04' 'Jan-07' 'Apr-04' 'Sep-04'
 'Jan-98' 'Oct-89' 'Jul-03' 'May-91' 'Sep-07' 'Oct-98' 'Aug-93' 'Oct-03'
 'Jan-01' 'Nov-97' 'Feb-83' 'Jul-85' 'Apr-03' 'Jun-01' 'Feb-02' 'Aug-84'
 'Nov-06' 'Dec-87' 'Nov-81' 'Apr-05' 'Oct-07' 'Jul-05' 'Dec-00' 'Apr-07'
 'Jan-03' 'Mar-94' 'Sep-98' 'Jun-04' 'Nov-95' 'Jul-99' 'Jun-95' 'Sep-92'
 'Jan-02' 'Apr-92' 'Oct-06' 'May-00' 'Dec-98' 'Dec-04' 'Oct-00' 'May-02'
 'Jul-02' 'Jul-06' 'May-97' 'Oct-05' 'Apr-95' 'Oct-02' 'Jan-00' 'Apr-00'
 'Dec-94' 'Sep-05' 'Dec-84' 'Dec-99' 'Nov-03' 'Jun-89' 'Jun-03' 'Oct-96'
 'May-03' 'Jun-02' 'Jun-07' 'Dec-96' 'Sep-02' 'Jan-86' 'May-98' 'Jan-97'
 'Jun-05' 'Feb-90' 'Mar-04' 'Jul-95' 'Aug-94' 'Jun-92' 'May-06' 'Mar-97'
 'Apr-06' 'Apr-90' 'Aug-99' 'Sep-00' 'Feb-01' 'Dec-88' 'Feb-99' 'Dec-91'
 'Aug-00' 'Oct-04' 'Aug-04' 'Feb-05' 'Nov-05' 'Nov-00' 'May-07' 'Jan-91'
```

'Jun-00' 'Aug-06' 'Dec-02' 'Jun-93' 'Jun-06' 'Feb-04' 'Dec-90' 'Mar-00'
'Feb-95' 'Jul-01' 'Apr-02' 'Dec-01' 'Sep-06' 'May-99' 'Aug-98' 'Dec-05'
'May-04' 'Oct-01' 'Jun-83' 'Mar-86' 'Apr-80' 'Jul-04' 'Jul-08' 'May-96'
'Jan-04' 'Nov-02' 'Aug-02' 'Aug-01' 'Mar-91' 'Sep-94' 'Sep-99' 'Aug-05'
'Dec-86' 'Nov-98' 'Feb-06' 'May-94' 'Nov-07' 'Feb-93' 'Nov-91' 'May-05'
'Mar-90' 'Mar-96' 'Oct-79' 'Jun-81' 'Mar-01' 'Apr-01' 'Jun-99' 'Nov-93'
'Jan-06' 'Dec-97' 'Nov-94' 'Jul-97' 'Oct-91' 'Jun-94' 'Mar-06' 'Sep-96'
'Apr-91' 'Jul-93' 'Jan-95' 'Sep-87' 'Mar-03' 'Oct-99' 'Jul-96' 'Dec-03'
'Aug-88' 'Sep-03' 'Mar-98' 'Feb-07' 'Dec-92' 'Jul-98' 'Jul-89' 'May-90'
'Jul-94' 'Sep-01' 'Mar-84' 'Nov-99' 'Mar-07' 'Mar-08' 'Apr-94' 'Jan-05'
'Jul-86' 'Aug-90' 'May-92' 'Jul-00' 'May-83' 'Apr-93' 'Jul-78' 'Mar-95'
'Feb-00' 'Dec-81' 'Mar-92' 'Jan-81' 'Sep-90' 'Jun-98' 'May-93' 'May-01'
'Nov-96' 'Feb-97' 'Jan-92' 'Mar-02' 'Jan-88' 'Aug-97' 'Aug-87' 'Aug-08'
'Oct-94' 'Feb-94' 'Jun-96' 'Feb-98' 'Nov-08' 'Apr-98' 'Jul-79' 'Jan-93'
'May-87' 'Jul-71' 'Aug-07' 'Jun-97' 'Mar-80' 'Dec-06' 'Jul-07' 'Oct-95'
'Jan-96' 'Jul-91' 'Jul-92' 'Dec-72' 'Dec-93' 'Jan-99' 'Feb-03' 'Apr-97'
'Dec-95' 'Jul-90' 'Mar-70' 'Nov-84' 'Apr-84' 'Jul-84' 'Aug-95' 'Mar-99'
'Sep-88' 'Mar-89' 'Mar-87' 'Oct-97' 'Dec-80' 'Jan-94' 'Aug-03' 'Mar-05'
'Jan-89' 'Apr-96' 'Oct-86' 'Feb-92' 'Jan-90' 'Nov-90' 'Mar-69' 'Jun-75'
'Mar-85' 'Dec-07' 'Sep-95' 'Oct-93' 'Dec-89' 'Sep-80' 'Jun-88' 'May-78'
'Aug-89' 'Oct-90' 'Sep-91' 'Feb-82' 'Feb-87' 'Nov-85' 'Jul-88' 'May-08'
'Oct-85' 'Mar-83' 'Aug-91' 'Sep-86' 'Jun-90' 'Feb-86' 'Jun-84' 'Sep-81'
'Apr-86' 'Aug-79' 'Nov-92' 'Sep-93' 'Jun-87' 'Feb-84' 'Aug-92' 'Aug-85'
'Jul-83' 'Dec-83' 'Jan-87' 'Nov-78' 'Aug-96' 'Nov-89' 'Sep-76' 'Nov-86'
'Oct-87' 'Sep-08' 'May-77' 'May-86' 'Mar-81' 'Jan-83' 'Sep-89' 'Sep-79'
'Oct-83' 'Sep-62' 'Jun-85' 'May-82' 'Feb-88' 'Oct-92' 'Aug-83' 'Sep-97'
'Jun-73' 'Apr-85' 'Oct-88' 'Oct-81' 'Sep-68' 'Jul-74' 'Nov-87' 'May-95'
'Mar-93' 'Jun-08' 'Jul-80' 'Dec-82' 'Mar-75' 'Oct-84' 'Mar-88' 'Feb-80'
'Nov-88' 'Apr-88' 'Sep-85' 'Sep-71' 'Mar-78' 'Feb-08' 'Aug-78' 'Nov-70'
'Jun-79' 'Jun-80' 'Apr-89' 'Sep-83' 'Feb-89' 'Oct-82' 'Aug-86' 'May-88'
'Dec-85' 'Jan-82' 'Sep-77' 'Dec-76' 'Apr-82' 'May-84' 'Apr-08' 'Feb-79'
'Jan-08' 'Sep-64' 'Jul-87' 'Jan-78' 'May-89' 'Oct-77' 'Dec-75' 'Jan-84'
'Oct-08' 'Feb-85' 'Nov-82' 'May-75' 'May-85' 'Feb-71' 'Jun-77' 'Apr-81'
'May-79' 'Jan-72' 'Jun-86' 'Sep-67' 'Apr-78' 'Feb-65' 'Nov-75' 'Jun-67'
'Feb-91' 'Dec-79' 'Aug-67' 'Apr-71' 'Sep-84' 'Aug-82' 'May-81' 'Dec-70'
'Oct-73' 'Jan-71' 'Dec-63' 'Apr-74' 'Jan-80' 'Apr-75' 'Jul-77' 'Mar-77'
'Nov-69' 'Jan-76' 'Nov-83' 'Mar-82' 'Apr-87' 'Dec-69' 'May-74' 'Aug-74'
'Jun-91' 'Jun-72' 'Mar-63' 'Aug-69' 'Oct-80' 'Jul-72' 'Aug-75' 'Sep-82'
'Sep-74' 'Aug-81' 'Nov-76' 'May-73' 'Dec-73' 'Sep-73' 'Mar-73' 'Dec-77'
'Oct-76' 'Jan-74' 'Jan-70' 'Aug-68' 'Apr-83' 'Jan-75' 'Dec-74' 'Feb-73'
'Nov-65' 'Jun-82' 'Jun-74' 'May-65' 'Oct-70' 'Apr-76' 'Oct-71' 'Apr-77'
'Aug-80' 'Sep-78' 'Oct-78' 'Oct-54' 'Feb-81' 'Jan-77' 'Aug-77' 'Dec-78'
'Aug-76' 'Jun-68' 'Jun-78' 'Oct-72' 'Jun-69' 'May-80' 'Jan-79' 'Oct-65'
'Nov-74' 'Apr-66' 'Jun-76' 'Feb-72' 'May-76' 'Mar-76' 'Jul-70' 'Mar-79'
'Apr-73' 'Jul-76' 'Jul-82' 'Sep-65' 'Apr-67' 'Oct-63' 'Feb-70' 'Jul-73'
'Feb-78' 'Nov-71' 'Aug-72' 'Jul-75' 'Sep-70' 'Jul-81' 'Sep-72' 'May-70'
'May-63' 'Feb-69' 'Nov-80' 'Jul-67' 'Apr-70' 'Nov-77' 'Nov-66' 'May-71'
'Mar-68' 'Apr-79' 'May-72' 'Feb-68' 'Nov-67' 'Apr-64' 'Feb-75' 'Mar-74'
'Jun-59' 'Sep-56' 'Jun-66' 'Jan-46' 'Mar-66' 'Jan-63' 'Dec-50' 'Jan-68'
'Jul-69' 'Nov-73' 'Jun-70' 'Feb-74' 'Jan-73' 'Feb-66' 'Dec-61' 'Aug-73'
'Feb-77' 'Aug-70' 'Sep-69' 'Sep-75' 'Dec-68' 'Feb-76' 'Nov-54' 'Mar-72'
'Nov-79' 'Oct-69' 'Dec-65' 'Apr-72' 'Nov-72' 'Sep-63' 'Apr-69' 'Nov-62'
'Oct-67' 'Jun-71' 'May-67' 'Nov-61' 'Feb-67' 'Nov-68' 'Oct-75' 'Mar-71'
'Aug-71' 'Dec-66' 'Oct-68' 'Oct-74' 'Nov-63' 'Apr-68' 'May-69' 'Nov-59'
nan 'Jan-10' 'Sep-09' 'Nov-10' 'Jan-09' 'Oct-10' 'May-10' 'Apr-09'
'Dec-09' 'Jul-10' 'Dec-08' 'Oct-09' 'Aug-09' 'Jun-10' 'Nov-09' 'Jul-09'
'Jun-09' 'Mar-10' 'Sep-10' 'Apr-10' 'Feb-09' 'Oct-62' 'Jun-64' 'Mar-09'
'Apr-62' 'Aug-10' 'Sep-66' 'Jan-61' 'Dec-56' 'May-09' 'Feb-10' 'Jan-64'
'May-68' 'Jan-62' 'Jul-65' 'Oct-64' 'Dec-67' 'Oct-60' 'Jun-65' 'May-60'
'Oct-59' 'Jun-63' 'Jan-69' 'Jul-63' 'Dec-71' 'Nov-60' 'Mar-64' 'Jul-68'

```
 'Jan-66' 'May-64' 'Mar-60' 'Apr-55' 'Aug-66' 'Aug-65' 'Jul-66' 'Oct-66'
 'Jan-65' 'Oct-61' 'Jan-67' 'Nov-55' 'Feb-57' 'Nov-64' 'May-66' 'Sep-60'
 'Mar-67' 'Nov-58' 'Aug-60' 'Aug-62' 'Oct-58' 'Dec-60' 'Feb-64' 'May-62'
 'Mar-65' 'Mar-11' 'Dec-10' 'Jun-11' 'Feb-11' 'Sep-11' 'Jan-11' 'Oct-11'
 'Aug-11' 'Nov-11' 'May-11' 'Jul-11' 'Apr-11' 'Apr-63' 'Aug-63' 'Sep-59'
 'Jan-55' 'Apr-65' 'Jul-58' 'Jul-64' 'May-59' 'Dec-62' 'Aug-58' 'Jan-59'
 'Jan-56' 'Jan-54' 'Dec-64' 'Jan-48' 'Jan-60' 'Jul-61' 'Jun-60' 'Dec-58'
 'Aug-64' 'Mar-61' 'Nov-56']
initial_list_status
['f' 'w']
last_pymnt_d
['Jan-15' 'Apr-13' 'Jun-14' 'Apr-12' 'Nov-12' 'Jun-13' 'Sep-13' 'Jul-12'
 'Oct-13' 'May-13' 'Feb-15' 'Aug-15' 'Oct-12' 'Sep-12' nan 'Dec-12'
 'Dec-14' 'Aug-13' 'Nov-13' 'Jan-14' 'Apr-14' 'Aug-14' 'Oct-14' 'Aug-12'
 'Jul-14' 'Jul-13' 'Jan-16' 'Apr-15' 'Feb-14' 'Sep-14' 'Jun-12' 'Feb-13'
 'Mar-13' 'May-14' 'Mar-15' 'Jan-13' 'Dec-13' 'Feb-12' 'Mar-14' 'Sep-15'
 'Nov-15' 'Jan-12' 'Oct-15' 'Nov-14' 'Mar-12' 'May-12' 'Dec-15' 'Jun-15'
 'May-15' 'Jul-15' 'Dec-11' 'Nov-11' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11'
 'Jun-11' 'May-11' 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10'
 'Oct-10' 'Sep-10' 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Mar-10'
 'Feb-10' 'Jan-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Aug-09' 'Jul-09'
 'Jun-09' 'May-09' 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08' 'Oct-08'
 'Aug-08' 'Jul-08' 'Sep-08' 'Jun-08' 'May-08' 'Nov-08' 'Apr-08' 'Mar-08'
 'Feb-08' 'Jan-08' 'Dec-07']
next_pymnt_d
[nan 'Feb-16' 'Jan-16' 'Sep-13' 'Feb-14' 'May-14' 'Jun-13' 'Mar-12'
 'Apr-12' 'May-13' 'Aug-12' 'Aug-13' 'Jun-12' 'Nov-13' 'Feb-12' 'Oct-11'
 'Jan-13' 'Jan-14' 'Jul-13' 'Jul-15' 'Jan-12' 'Dec-12' 'Jun-11' 'Feb-13'
 'Nov-11' 'Nov-12' 'Dec-11' 'Aug-11' 'Sep-11' 'Apr-11' 'Mar-14' 'Apr-13'
 'Mar-11' 'Jul-12' 'Aug-14' 'Oct-13' 'Sep-12' 'May-12' 'Apr-15' 'Jul-11'
 'Dec-15' 'Dec-13' 'Jan-11' 'Oct-12' 'Nov-14' 'Mar-13' 'Aug-15' 'Feb-15'
 'May-15' 'Jul-14' 'Nov-15' 'Sep-14' 'Oct-15' 'May-11' 'Feb-11' 'Dec-14'
 'Jun-15' 'Apr-14' 'Jan-15' 'Sep-15' 'Jun-14' 'Nov-10' 'Oct-10' 'Dec-10'
 'Mar-15' 'Oct-14' 'Jul-10' 'Sep-10' 'May-10' 'Aug-10' 'Mar-10' 'Jun-10'
 'Apr-10' 'Feb-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Jan-10' 'Sep-09' 'Jun-09'
 'Aug-09' 'Jul-09' 'May-09' 'Apr-09' 'Jan-09' 'Oct-08' 'Feb-09' 'Nov-08'
 'Sep-08' 'Mar-09' 'Dec-08' 'Aug-08' 'Jun-08' 'Jul-08' 'Apr-08' 'May-08'
 'Feb-08' 'Jan-08' 'Mar-08' 'Dec-07']
last_credit_pull_d
['Jan-16' 'Sep-13' 'Jan-15' 'Sep-15' 'Dec-14' 'Aug-12' 'Mar-13' 'Dec-15'
 'Aug-13' 'Nov-12' 'Mar-14' 'Apr-15' 'May-14' 'Jul-15' 'Jul-12' 'Sep-12'
 'May-13' 'Oct-15' 'Jun-12' 'Mar-15' 'Dec-12' 'Jul-14' 'Sep-14' 'Feb-14'
 'Jun-15' 'Oct-13' 'Apr-14' 'Oct-14' 'Feb-13' 'Nov-15' 'Oct-12' 'Nov-13'
 'Nov-14' 'Feb-12' 'Apr-12' 'Aug-15' 'Jun-14' 'Jan-12' 'Aug-14' 'Jun-13'
 'Dec-13' 'May-12' 'Jan-14' 'Jul-13' 'Apr-13' 'May-15' 'Feb-15' 'Mar-12'
 'Nov-11' 'Dec-11' 'Jan-13' 'Oct-11' 'Sep-11' 'Aug-11' 'Jul-11' 'Jun-11'
 'May-11' 'Apr-11' 'Mar-11' 'Feb-11' 'Jan-11' 'Dec-10' 'Nov-10' 'Oct-10'
 nan 'Sep-10' 'Aug-10' 'Jul-10' 'Jun-10' 'May-10' 'Apr-10' 'Feb-10'
 'Mar-10' 'Aug-07' 'Jan-10' 'Dec-09' 'Nov-09' 'Oct-09' 'Sep-09' 'Jul-09'
 'Aug-09' 'Jun-09' 'May-09' 'Apr-09' 'Mar-09' 'Feb-09' 'Jan-09' 'Dec-08'
 'Jun-08' 'Sep-08' 'May-08' 'Aug-08' 'Mar-08' 'Oct-08' 'Feb-08' 'Jan-08'
 'Dec-07' 'Jul-08' 'Oct-07' 'Sep-07' 'Jun-07' 'May-07' 'Jul-07' 'Nov-07']
application_type
['INDIVIDUAL']
```

In [10]:
```
df = df.drop(columns = [x for x in pct_missing_value[pct_missing_value >= 70].ir
```

In [11]:
```
df = df.drop(columns = ['emp_title','sub_grade','title','addr_state','desc'])
```

In [12]:
```
df = df.dropna(subset = ['annual_inc','delinq_2yrs','earliest_cr_line','inq_last
                         'last_pymnt_d','collections_12_mths_ex_med','acc_now_de
```

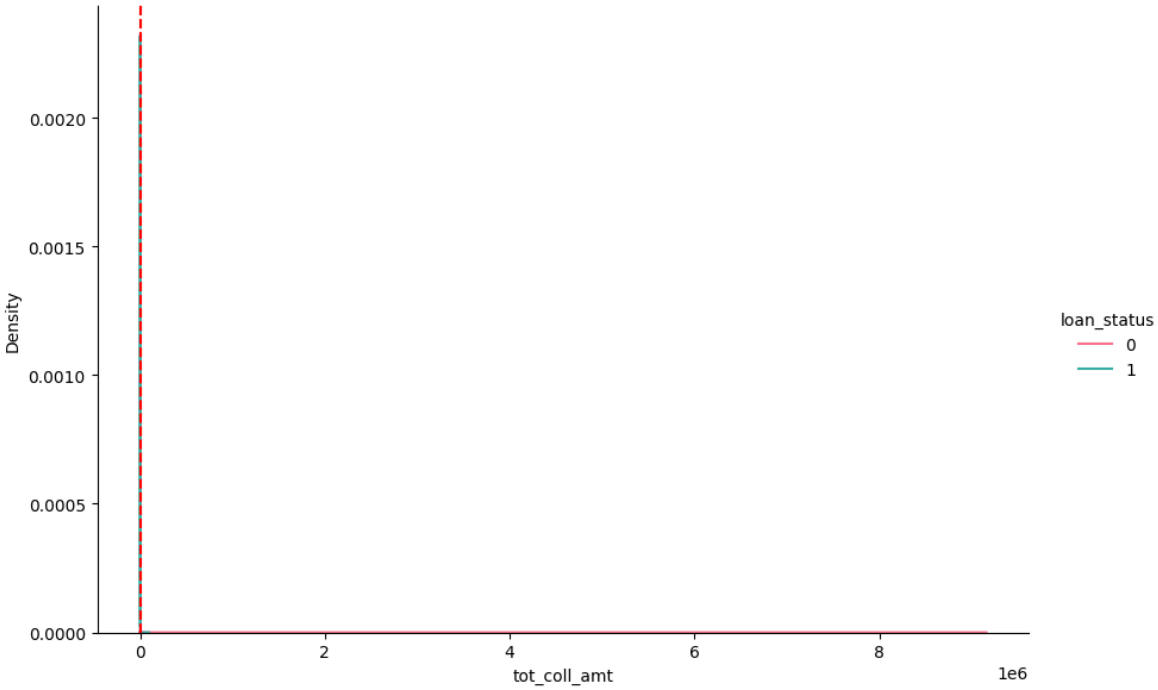
In [13]:
```
df = df.drop(columns = ['policy_code','url', 'zip_code','application_type'])
```

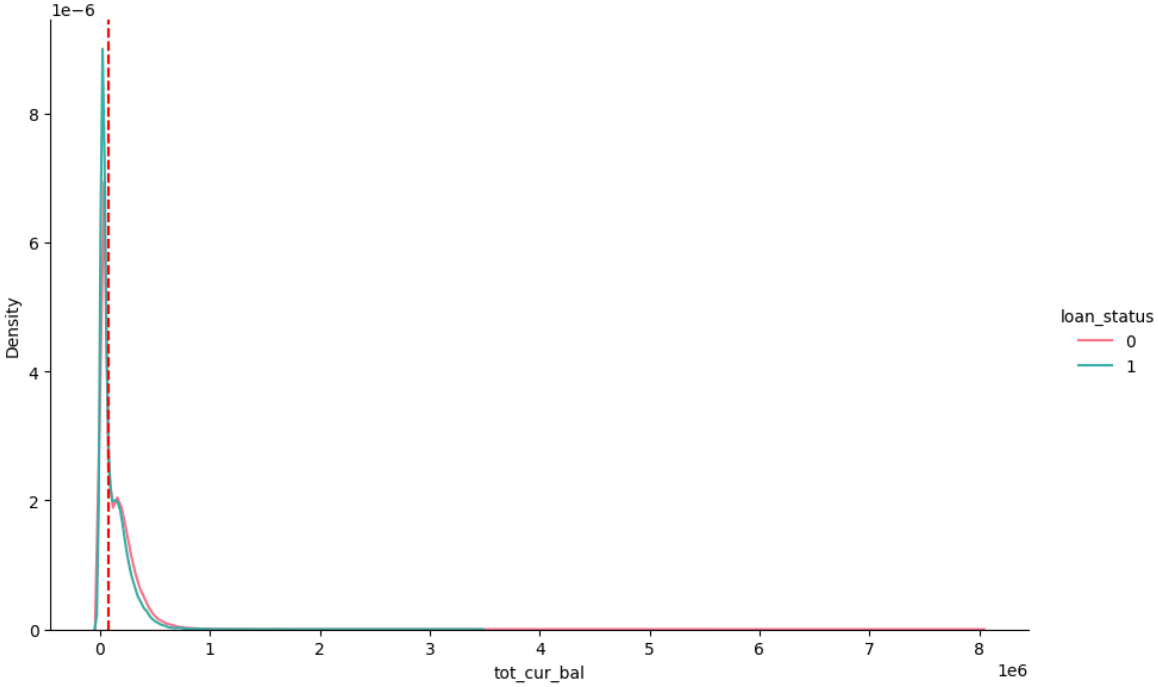
In [15]:
```
# số lượng null khá lớn, chiếm trên khoảng 28%
df[['tot_coll_amt','tot_cur_bal','total_rev_hi_lim']].describe()
```

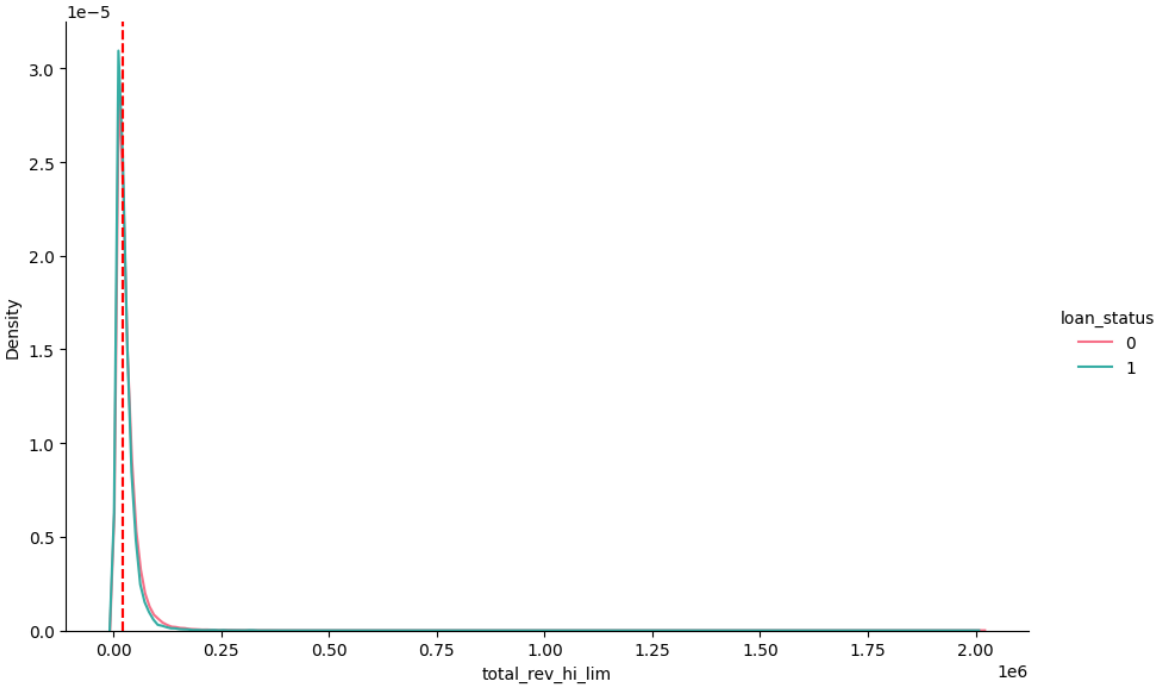
Out[15]:

	tot_coll_amt	tot_cur_bal	total_rev_hi_lim
count	171899.000	171899.000	171899.000
mean	200.820	136581.484	29118.968
std	22134.785	150119.007	28537.528
min	0.000	0.000	100.000
25%	0.000	27976.500	13200.000
50%	0.000	79293.000	22000.000
75%	0.000	206441.500	36200.000
max	9152545.000	8000078.000	2013133.000

In [16]:
```
# Đồ thị minh họa 3 cột dữ liệu
g= sns.FacetGrid(df, hue='loan_status',height = 6,aspect = 1.5, palette = 'husl'
g.map(sns.kdeplot, 'tot_coll_amt', shade = False).add_legend()
g.refline(x = df.tot_coll_amt.median(), color = 'r')

g= sns.FacetGrid(df, hue='loan_status',height = 6,aspect = 1.5,palette = 'husl')
g.map(sns.kdeplot, 'tot_cur_bal', shade = False).add_legend()
g.refline(x = df.tot_cur_bal.median(), color = 'r')

g= sns.FacetGrid(df, hue='loan_status',height = 6,aspect = 1.5,palette = 'husl')
g.map(sns.kdeplot, 'total_rev_hi_lim', shade = False).add_legend()
g.refline(x = df.total_rev_hi_lim.median(), color = 'r')

pass
```


Density
0.0020
0.0015
0.0010
0.0005
0.0000
0
2
4
6
8
1e6
tot_coll_amt
loan_status
0
1


1e−6
Density
8
6
4
2
0
0
1
2
3
4
5
6
7
8
1e6
tot_cur_bal
loan_status
0
1



In [14]:
```
df = df.drop(columns = ['tot_coll_amt','tot_cur_bal','total_rev_hi_lim'])
```

In [18]:
```
df[['mths_since_last_delinq','last_pymnt_d','delinq_2yrs']].head()
```

Out[18]:

	mths_since_last_delinq	last_pymnt_d	delinq_2yrs
0	NaN	Jan-15	0.000
1	NaN	Apr-13	0.000
2	NaN	Jun-14	0.000
3	35.000	Jan-15	0.000
5	NaN	Jan-15	0.000

In [19]:
```
df[['mths_since_last_delinq','last_pymnt_d','delinq_2yrs']][df['mths_since_last_
```

Out[19]:

	mths_since_last_delinq	delinq_2yrs
count	0.000	133155.000
mean	NaN	0.004
std	NaN	0.133
min	NaN	0.000
25%	NaN	0.000
50%	NaN	0.000
75%	NaN	0.000
max	NaN	14.000

In [15]:
```
df['mths_since_last_delinq'] = df['mths_since_last_delinq'].replace(np.nan, 0)
```

Biến đổi các dữ liệu đầu vào

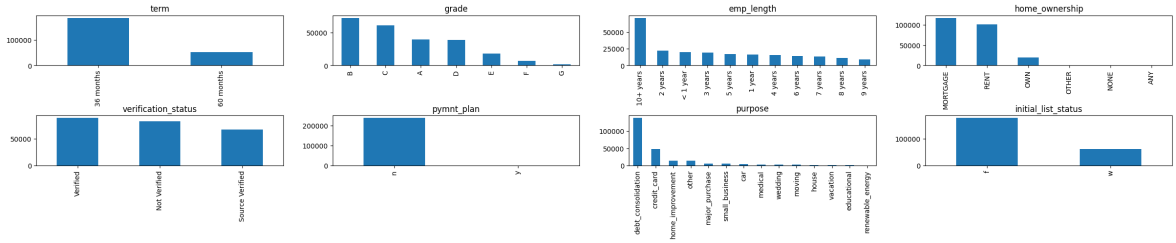
- Format lại các dữ liệu `ngày tháng`

In [18]:
```
df.issue_d = pd.to_datetime(df.issue_d, format ='%b-%y')
df.earliest_cr_line = pd.to_datetime(df.earliest_cr_line,format ='%b-%y')
df.last_pymnt_d = pd.to_datetime(df.last_pymnt_d,format ='%b-%y')
df.last_credit_pull_d = pd.to_datetime(df.last_credit_pull_d, format = '%b-%y')
```

In [19]:
```
cat_col = []
for x in df.dtypes.index:
    if df.dtypes[x] == 'object':
        cat_col.append(x)

num_col = []
for x in df.dtypes.index:
    if df.dtypes[x] != 'object':
        num_col.append(x)
```

In [24]:
```
def plot_bar_classes(df, cols):
    df[cols].value_counts().plot.bar()

def distribution_cate(df, cat_col, row = 8, col = 4, figsize = (30, 20)):
  print('Số biến demographic: ', len(cat_col))
  plt.figure(figsize = figsize)
  plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.2,
  for i in range(1, len(cat_col)+1):
    try:
      plt.subplot(row, col, i)
      plot_bar_classes(df, cat_col[i-1])
      plt.title(cat_col[i-1])
    except:
      break

distribution_cate(df, cat_col)
```

```
Số biến demographic:  8
```



- `pymnt_plan` có số lượng y rất ít, nên bỏ cột này ra khỏi mô hình
- `home_ownership` có 3 nhóm có số lượng thấp, nên nhóm cùng loại `Rent`

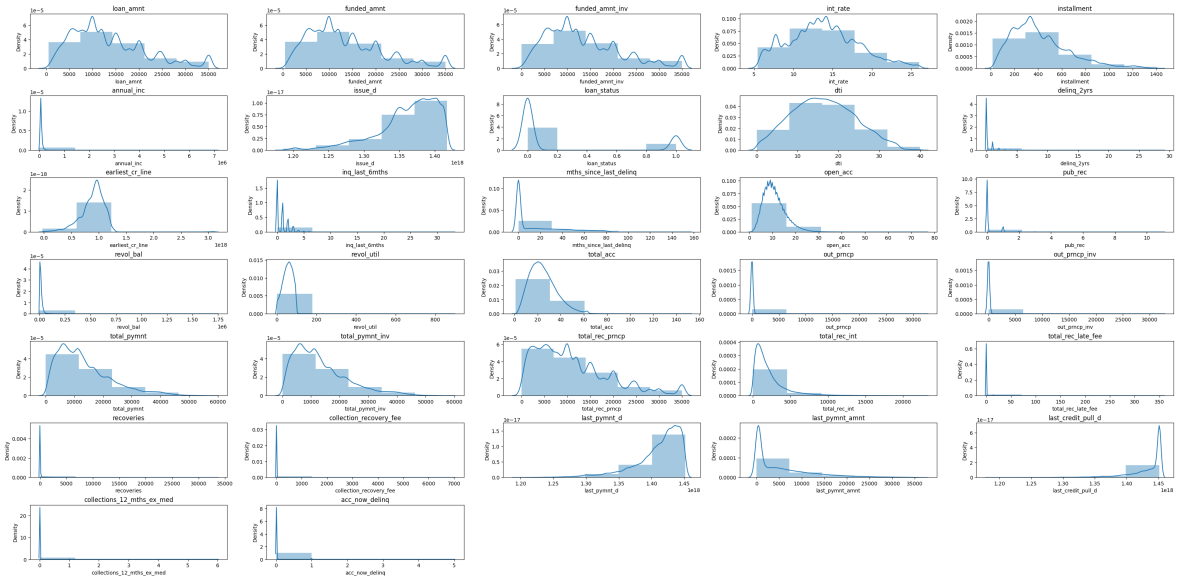
In [20]:
```
df = df[df['pymnt_plan'] != 'y' ]
df = df.drop(columns = 'pymnt_plan')
```

In [21]:
```
df.home_ownership = df.home_ownership.replace('OTHER', 'RENT')
df.home_ownership = df.home_ownership.replace('NONE', 'RENT')
df.home_ownership = df.home_ownership.replace('ANY', 'RENT' )
```

In [22]:
```
df.emp_length = df.emp_length.str.replace('\+ years', '')
df.emp_length = df.emp_length.str.replace('\< 1 year', str(0))
df.emp_length = df.emp_length.replace(np.nan, str(0))
df.emp_length = df.emp_length.str.replace(' year', '')
df.emp_length = df.emp_length.str.replace(' years', '')
df.emp_length = df.emp_length.str.replace('s', '')
# Chuyển dữ liệu sang dạng số
df.emp_length = pd.to_numeric(df.emp_length)
```

In [31]:
```
# Mô tả biến định lượng

def _plot_numeric_classes(df, col, bins=10, hist=True, kde=True):
    sns.distplot(df[col],
                 bins = bins,
                 hist = hist,
                 kde = kde)

def distribution_numeric(df, numeric_cols, row= 9, col=5, figsize=(40, 25), bins
    print('Số biến định lượng: ', len(numeric_cols))
    #assert row*(col-1) < len(numeric_cols)
    plt.figure(figsize = figsize)
    plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.2
    for i in range(1, len(numeric_cols)+1):
      try:
        plt.subplot(row, col, i)
        _plot_numeric_classes(df, numeric_cols[i-1], bins = bins)
        plt.title(numeric_cols[i-1])
      except:
        print('Error {}'.format(numeric_cols[i-1]))
        break

distribution_numeric(df, num_col)
```

```
Số biến định lượng:  32
```



- Các biến có `Q3 =0` nên loại ra khỏi mô hình
- Tỉ lệ xoay vòng vốn có giá trị lớn `bất thường`, những giá trị >100 sẽ thay thế =100
- 3 biến đầu tiên về khoản vay có phân phối `giống nhau` và giá trị khá `tương đồng`, tức là gần như người đi vay sẽ nhận được khoản vay mà họ đã apply, nên loại ra khỏi mô hình 2 trong 3 biến

In [26]:
```
df[['revol_bal', 'mths_since_last_delinq', 'out_prncp', 'out_prncp_inv', 'recove
```

Out[26]:

	revol_bal	mths_since_last_delinq	out_prncp	out_prncp_inv	recoveries	total
count	238170.000	238170.000	238170.000	238170.000	238170.000	
mean	15235.198	15.410	375.655	375.527	165.314	
std	19168.970	22.609	2331.120	2330.415	759.954	
min	0.000	0.000	0.000	0.000	0.000	
25%	5931.000	0.000	0.000	0.000	0.000	
50%	11004.000	0.000	0.000	0.000	0.000	
75%	19084.000	28.000	0.000	0.000	0.000	
max	1746716.000	152.000	32160.380	32160.380	33520.270	

In [23]:
```
df = df.drop(columns =['out_prncp_inv', 'recoveries', 'total_rec_late_fee','coll
```

In [24]:
```
df = df.drop(columns =['funded_amnt','funded_amnt_inv'])
```

In [29]:
```
df.revol_util.describe()
```

Out[29]:
```
count   238170.000
mean        55.006
std         24.665
min          0.000
25%         37.300
50%         56.700
75%         74.500
max        892.300
Name: revol_util, dtype: float64
```

In [25]:
```
df = df.reset_index()
```

In [26]:
```
for i in range(0, len(df)):
    if df['revol_util'][i] >= 100 :
        df['revol_util'] = df['revol_util'].replace(df['revol_util'][i], 100)
```

Thêm các biến để phục vụ EDA

In [27]:
```
# %delinquency trong vòng 2 năm (Behaviourial Risk)
df['delinq_pct'] = 100* df['delinq_2yrs']/df['mths_since_last_delinq']
df['delinq_pct']= df['delinq_pct'].replace(np.nan, 0)
df['delinq_pct']= df['delinq_pct'].replace(np.inf, 100)
```

In [28]:
```
# Thời gian tính từ lần đầu hạn mức tín dụng được đặt và tg tính từ lần đầu khoả
df['loan_issue_m'] = pd.to_datetime('2015-12-31') - df['issue_d']
df['loan_issue_m'] = df['loan_issue_m'].astype('timedelta64[M]')

df['cr_line_issue_m'] = pd.to_datetime('2015-12-31') - df['earliest_cr_line']
df['cr_line_issue_m'] =  df['cr_line_issue_m'].astype('timedelta64[M]')
```

In [190…
```
#những tín dụng có số ngày được mở âm
df.loc[:, ['earliest_cr_line','cr_line_issue_m']][df.cr_line_issue_m< 0].head()
```

Out[190]:

	earliest_cr_line	cr_line_issue_m
1420	2062-09-01	-561.000
1592	2068-09-01	-633.000
2515	2064-09-01	-585.000
2942	2067-09-01	-621.000
3010	2065-02-01	-590.000

In [29]:
```
# Loại những tín dụng có thời gian giao dịch sau 2015
df['earliest_cr_year'] = df['earliest_cr_line'].dt.year
df = df[df['earliest_cr_year'] < 2016]
```

In [30]:
```
df['term'] = df['term'].str.replace(' months', '')
```

In [37]:
```
df.head()
```

Out[37]:

	index	loan_amnt	term	int_rate	installment	grade	emp_length	home_ownership
0	0	5000	36	10.650	162.870	B	10	RENT
1	1	2500	60	15.270	59.830	C	0	RENT
2	2	2400	36	15.960	84.330	C	10	RENT
3	3	10000	36	13.490	339.310	C	10	RENT
4	5	5000	36	7.900	156.460	A	3	RENT

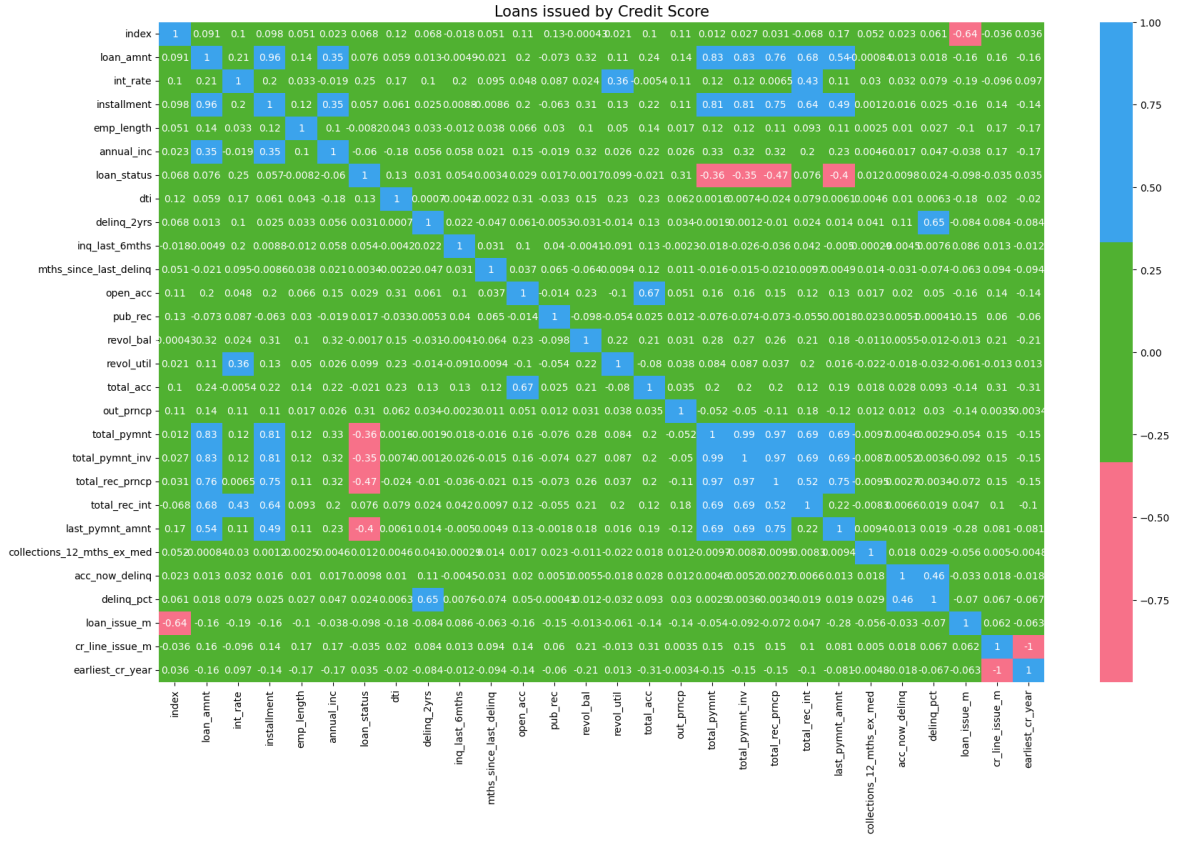
Phân tích khoản vay

Correlation

- Loại những biến có `tương quan cao`

In [194…
```
num_col = []
for i in df.dtypes.index:
    if df.dtypes[i] != 'object':
        num_col.append(i)
```

In [195…
```
# Biểu đồ ma trận tương quan
fig = plt.figure(figsize = (20,12))
ax = fig.add_subplot()
sns.heatmap(data = df[num_col].corr(method = 'pearson'), annot = True, cmap=sns.
ax.set_title('Loans issued by Credit Score', fontsize= 15)
pass
```



In [31]:
```
df = df.drop(columns = ['installment','total_pymnt_inv', 'total_pymnt', 'total_r
```

EDA

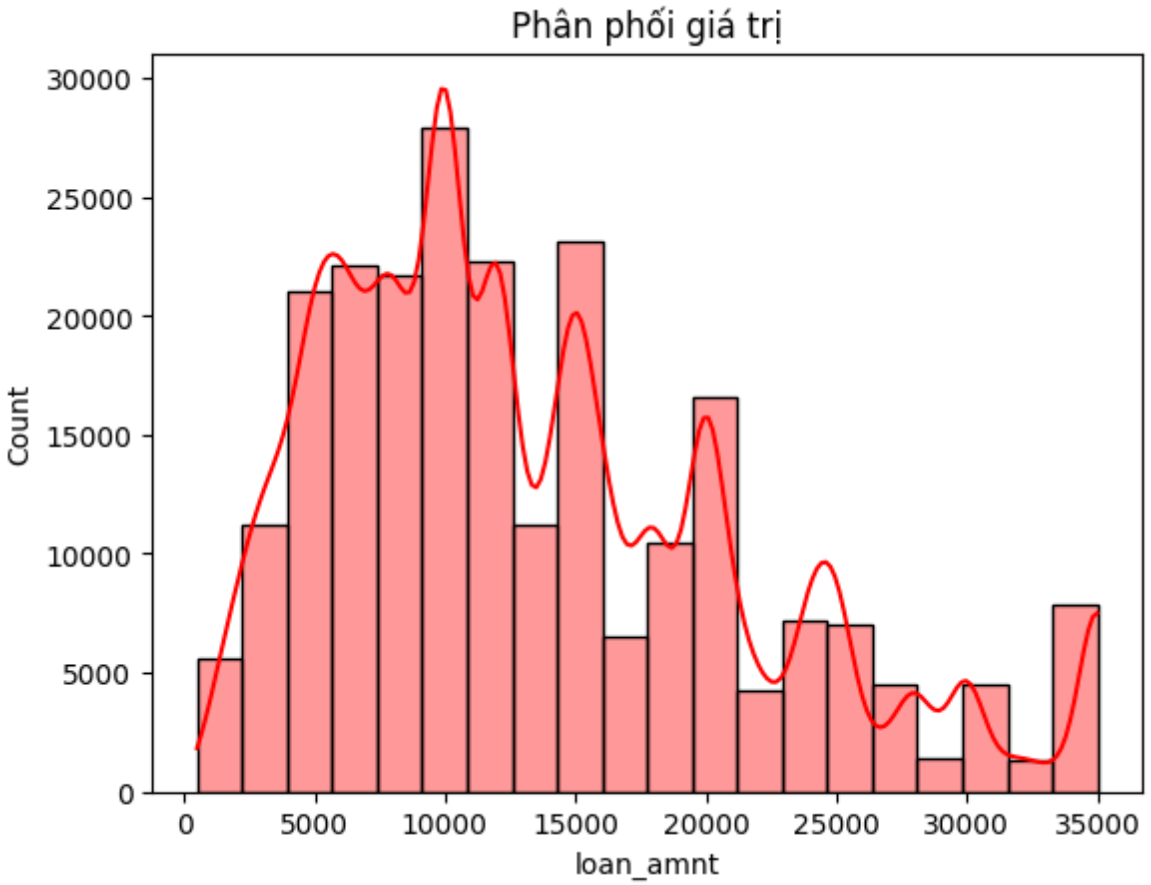
Phân tích về khoản vay và thu nhập

In [41]:
```
df.loan_amnt.describe()
```

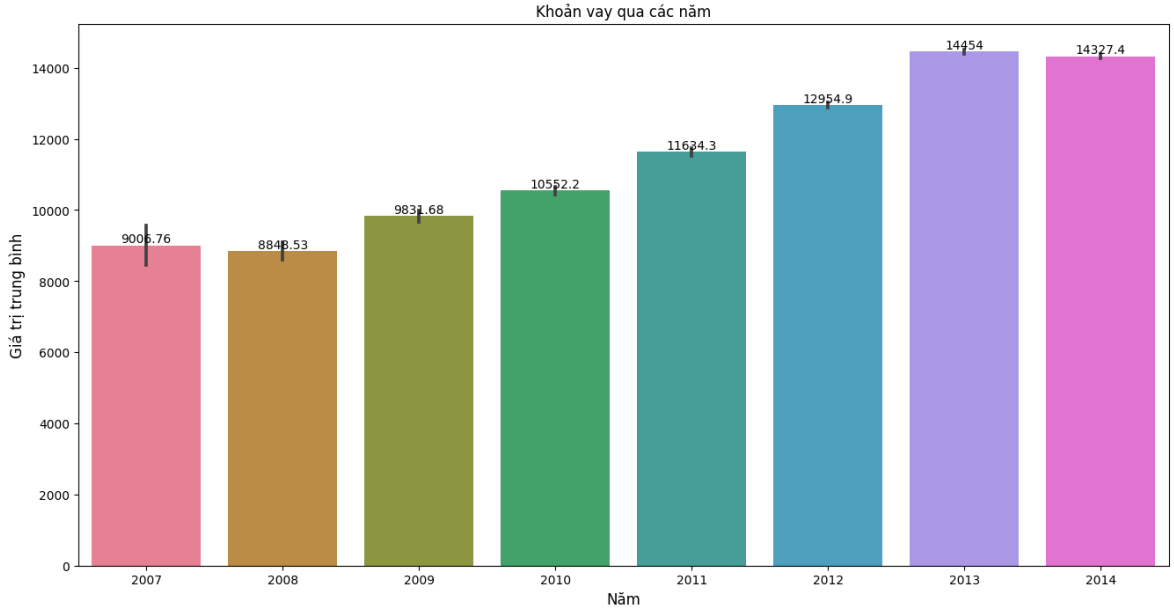
Out[41]:
```
count   237653.000
mean     13495.096
std       8061.988
min        500.000
25%       7200.000
50%      12000.000
75%      18000.000
max      35000.000
Name: loan_amnt, dtype: float64
```

In [42]:
```
sns.histplot(data = df , x = 'loan_amnt',bins = 20, kde = True, alpha = 0.4, col
plt.title('Phân phối giá trị', fontsize= 12)
```

Out[42]:
```
Text(0.5, 1.0, 'Phân phối giá trị')
```

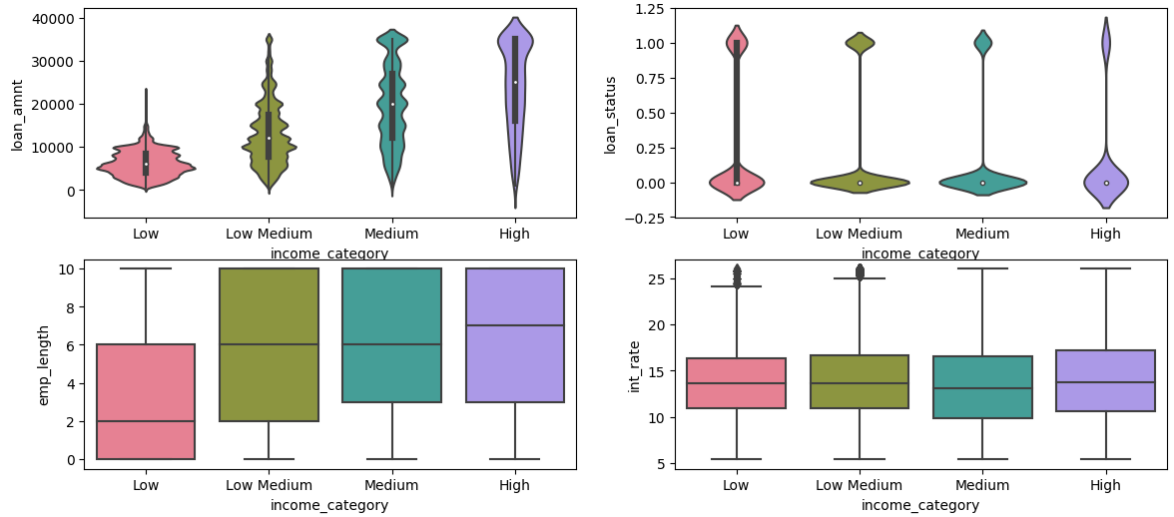


In [32]:
```
df['year_start'] = df['issue_d'].dt.year

plt.figure(figsize=(16,8))
a = sns.barplot(x= 'year_start', y = 'loan_amnt', data = df, palette='husl')
plt.title('Khoản vay qua các năm', fontsize= 12)
plt.xlabel('Năm', fontsize = 12)
plt.ylabel('Giá trị trung bình', fontsize= 12)
a.bar_label(a.containers[0])
pass
```



In [33]:
```
df['income_category'] = np.nan
lst = [df]
df.head()

for col in lst:
    col.loc[col['annual_inc'] <= 30000, 'income_category'] = 'Low'
    col.loc[(col['annual_inc'] > 30000) & (col['annual_inc'] <= 100000), 'income
    col.loc[(col['annual_inc'] > 100000) & (col['annual_inc'] <= 3000000), 'incc
    col.loc[(col['annual_inc'] > 300000) & (col['annual_inc'] <= 7500000), 'incc


fig, ((ax1, ax2), (ax3, ax4))= plt.subplots(nrows=2, ncols=2, figsize=(14,6))
sns.violinplot(x="income_category", y="loan_amnt", data=df, palette="husl", ax=a
sns.violinplot(x="income_category", y="loan_status", data=df, palette="husl", ax
sns.boxplot(x="income_category", y="emp_length", data=df,palette="husl",ax=ax3)
sns.boxplot(x="income_category", y="int_rate", data=df, palette="husl", ax=ax4)
```

Out[33]: <Axes: xlabel='income_category', ylabel='int_rate'>



Nhận xét:

Khoản vay

- Giá trị khoản vay trong thời kỳ xem xét tập trung trong khoảng từ `5000 đến 20000`
- Giá trị khoản vay trung bình `lớn nhất` vào năm 2013 và sau đó là 2014

Thu nhập

- Người ở nhóm thu nhập cao `vay mượn nhiều` hơn nhóm thu nhập thấp và trung bình thấp
- Người ở nhóm thu nhập cao và thấp dễ có `xu hướng trở thành nợ xấu` so với 2 nhóm còn lại
- Cũng dễ hiểu khi người đi vay có thu nhập thấp đồng nghĩa với việc thời gian lao động thấp và ngược lại
- Người ở `thu nhập thấp` có `lãi suất lớn` hơn nhưng cũng thiếu ổn định hơn so với những người ở nhóm thu nhập cao

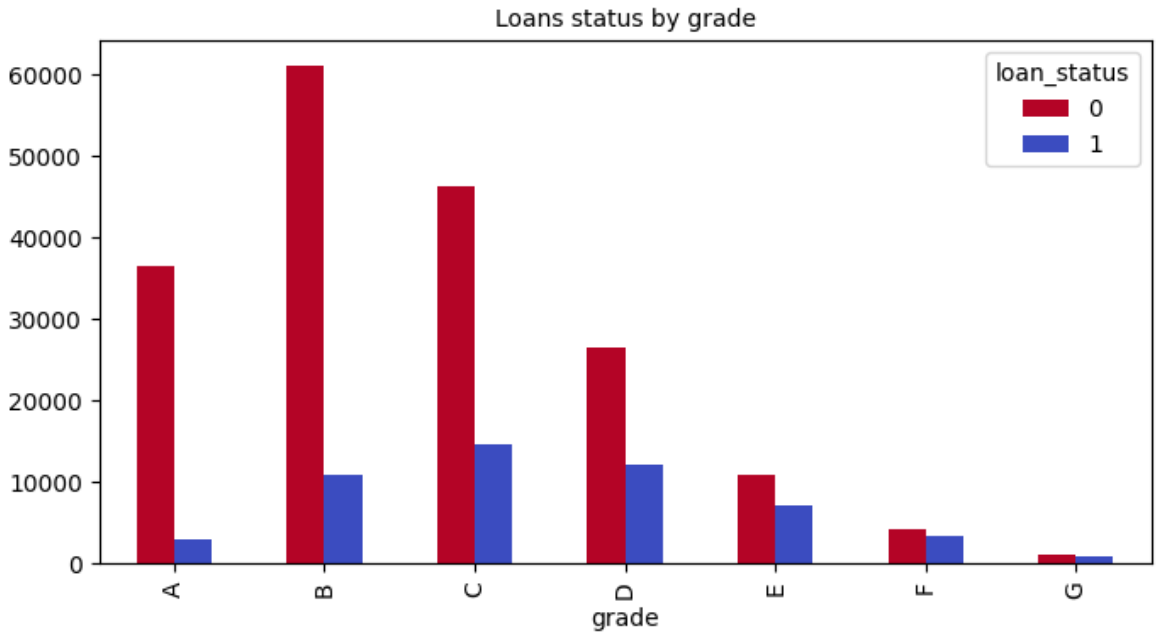
Rủi ro về hạng tín dụng và thời gian lao động

In [45]:
```
fig, ((ax1, ax2)) = plt.subplots(1, 2)
cmap = plt.cm.coolwarm

by_credit_score = df.groupby(['year_start', 'grade']).loan_amnt.mean()
by_credit_score.unstack().plot(legend=False, ax=ax1, figsize=(14, 4), colormap=c
ax1.set_title('Loans issued by Credit Score', fontsize=10)

by_inc = df.groupby(['year_start', 'grade']).int_rate.mean()
by_inc.unstack().plot(ax=ax2, figsize=(14, 4), colormap=cmap)
ax2.set_title('Interest Rates by Credit Score', fontsize=10)

ax2.legend(bbox_to_anchor=(-1.0, -0.3, 1.7, 0.1), loc=5, prop={'size':12},
           ncol=7, mode="expand", borderaxespad=0.)
```

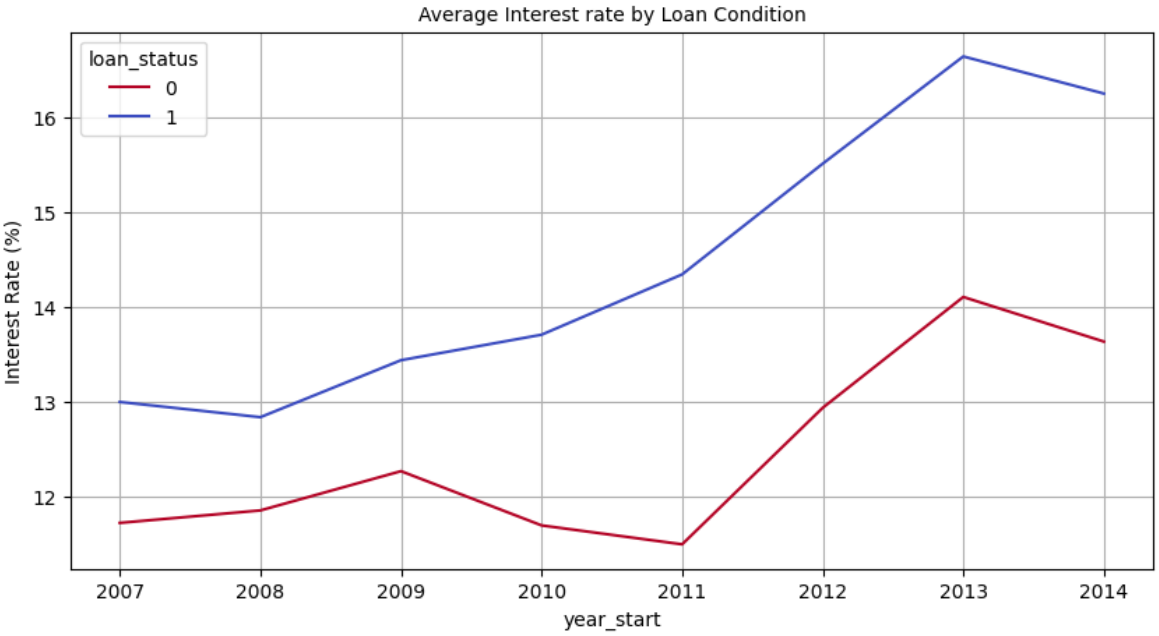
Out[45]:
```
<matplotlib.legend.Legend at 0x2286708f3a0>
```

In [46]:
```
fig = plt.figure(figsize=(8,4))
ax = fig.add_subplot()
cmap = plt.cm.coolwarm_r

loans_by_grade = df.groupby(['grade', 'loan_status']).size()
loans_by_grade.unstack().plot(kind='bar', stacked= False, colormap=cmap, ax=ax)
ax.set_title('Loans status by grade', fontsize=10)
```

Out[46]: Text(0.5, 1.0, 'Loans status by grade')

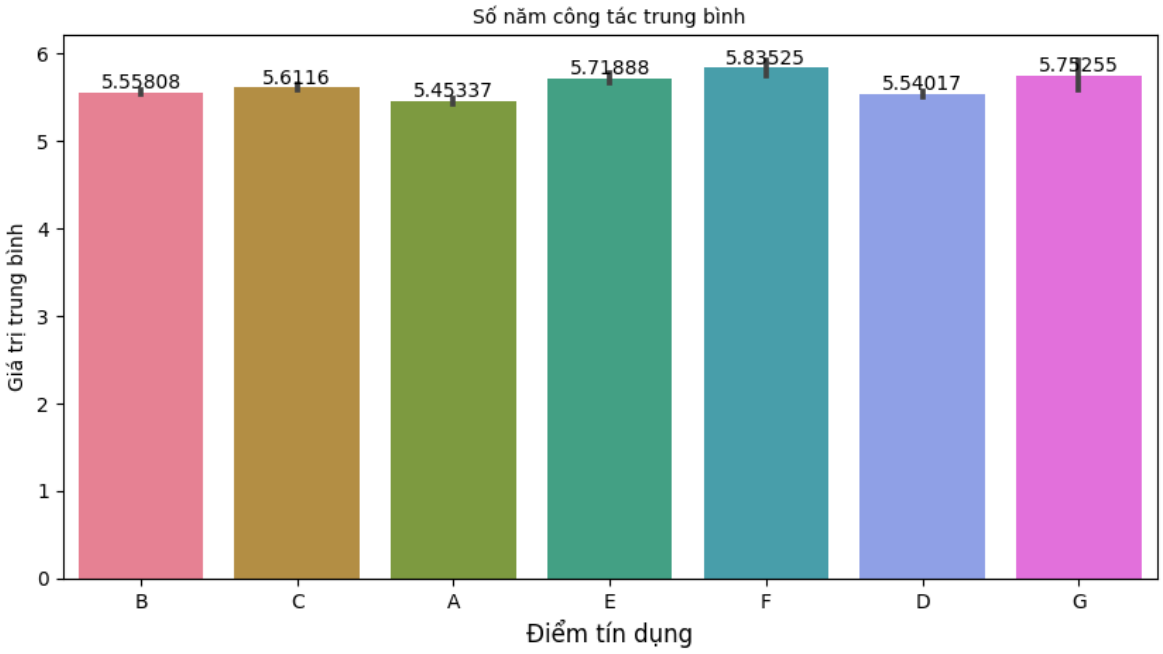


In [47]:
```
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot()
cmap = plt.cm.coolwarm_r

by_int_rate = df.groupby(['year_start', 'loan_status']).int_rate.mean()
by_int_rate.unstack().plot(ax=ax, colormap=cmap)
ax.set_title('Average Interest rate by Loan Condition', fontsize=10)
ax.set_ylabel('Interest Rate (%)', fontsize=10)

plt.grid()
```



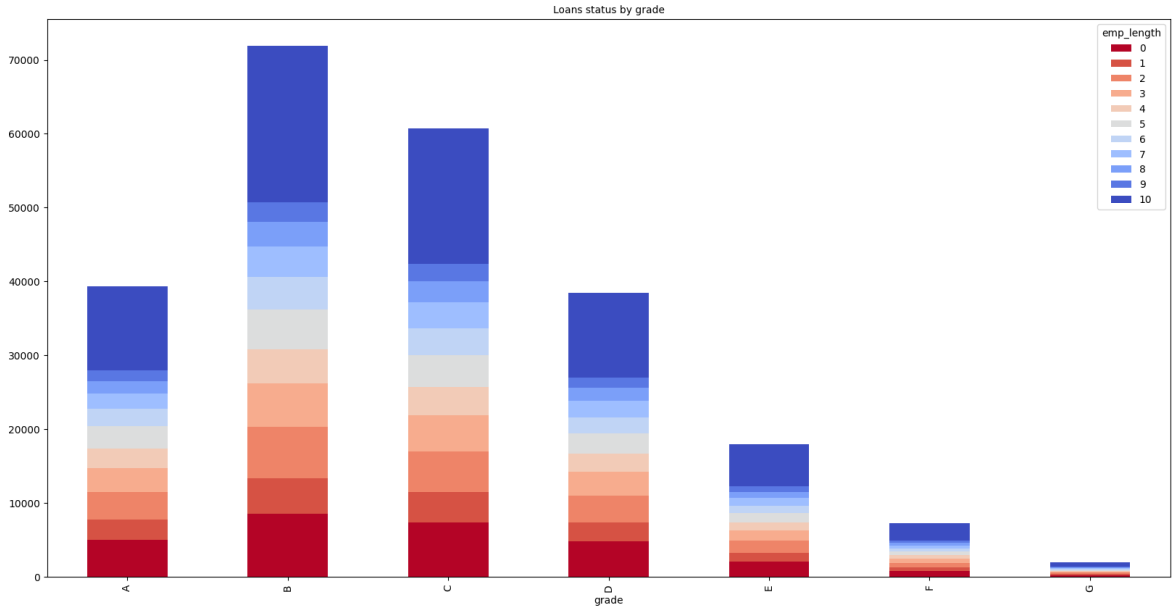
In [34]:
```
plt.figure(figsize=(10,5))
a = sns.barplot(x= 'grade', y = 'emp_length', data = df, palette='husl')
plt.title('Số năm công tác trung bình', fontsize= 10)
plt.xlabel('Điểm tín dụng', fontsize = 12)
plt.ylabel('Giá trị trung bình', fontsize= 10)
```

```
a.bar_label(a.containers[0])
pass
```



In [79]:
```
fig = plt.figure(figsize=(20,10))
ax = fig.add_subplot()
cmap = plt.cm.coolwarm_r

grade_per_emp_length =df.groupby(['grade', 'emp_length']).size()
grade_per_emp_length.unstack().plot(kind='bar', stacked= True, colormap=cmap, ax
ax.set_title('Loans status by grade', fontsize=10)
```

Out[79]:
```
Text(0.5, 1.0, 'Loans status by grade')
```



Nhận xét:

- Người có `grade thấp thì khoản vay và lãi suất cũng cao`
- Người có `grade cao, lãi suất thấp`
- Lãi suất ổn định trong giai đoạn 2007-2011 ở nhóm tín dụng tốt và đối với nhóm nợ xấu, lãi suất tăng trong giai đoạn dài hơn 2008 -2013

- Nhóm `B,C,D` có khả năng `nợ xấu` là cao nhất nhưng nhóm B cũng là nhóm có số lượng tín dụng đánh giá là tốt cao nhất

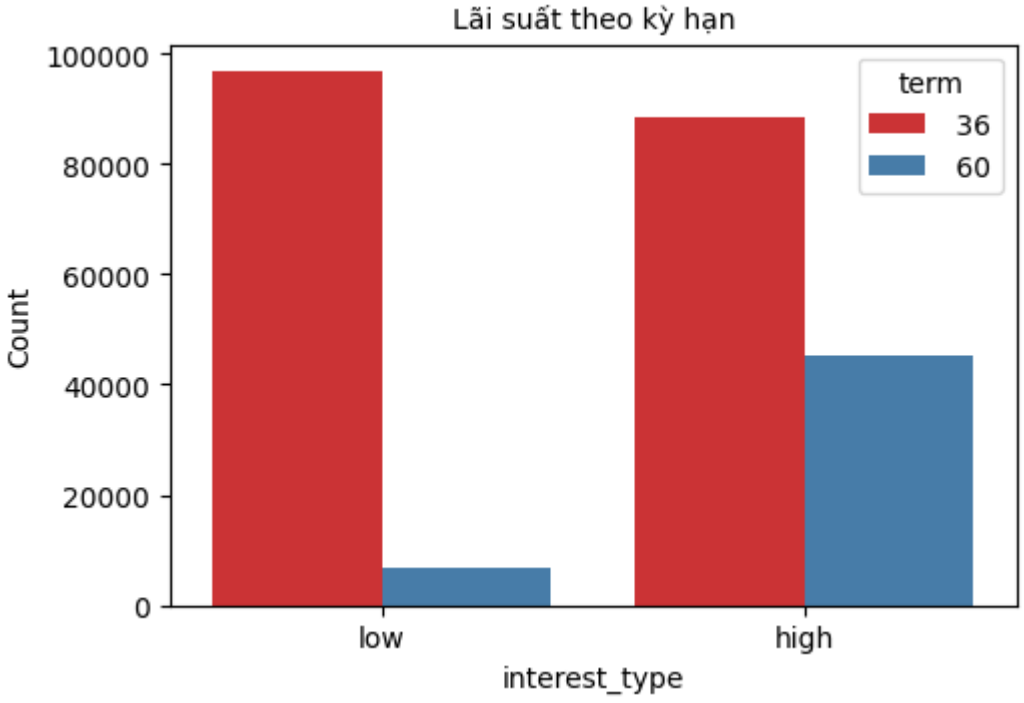
Điểm tín dụng

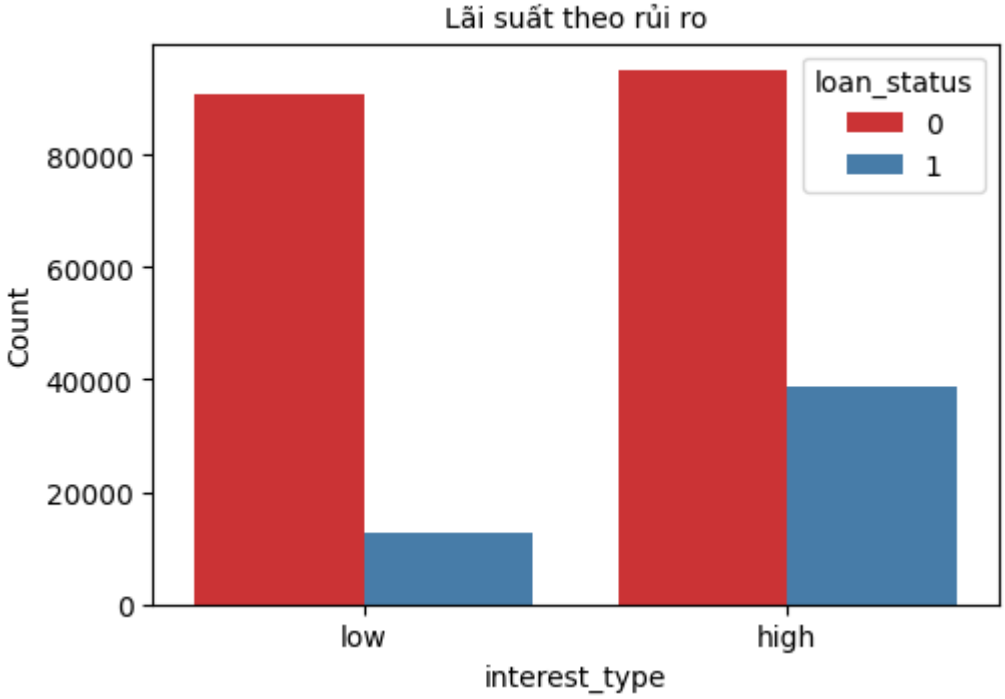
- Số năm công tác công tác trung bình giữa các nhóm là `gần giống nhau`, nằm trong khoảng 5.5 năm thâm niên.
- Nhóm B có số lượng người có thâm niên cao nhất và thấp nhất.
- Điểm được đánh giá chủ yếu rơi vào nhóm `B,C`

Đặc điểm của nhóm có bị coi là nợ xấu

In [35]:
```
# Giả sử khi lãi suất từ 13% đổ lên thì được coi là nợ xấu
df['interest_type'] = np.nan
lst = [df]
for col in lst:
    col.loc[col['int_rate']< 13, 'interest_type'] = 'low'
    col.loc[col['int_rate']>=13, 'interest_type'] = 'high'

plt.figure(figsize = (12,8))
plt.subplot(221)
ax = sns.countplot(x = 'interest_type', data = df, palette = 'Set1', hue = 'term
ax.set_title('Lãi suất theo kỳ hạn', fontsize = 10)
ax.set_ylabel('Count', fontsize= 10)


plt.figure(figsize = (12,8))
plt.subplot(222)
ax1 = sns.countplot(x = 'interest_type', data = df,palette = 'Set1', hue = 'loar
ax1.set_title('Lãi suất theo rủi ro', fontsize = 10)
ax1.set_ylabel('Count', fontsize= 10)
```

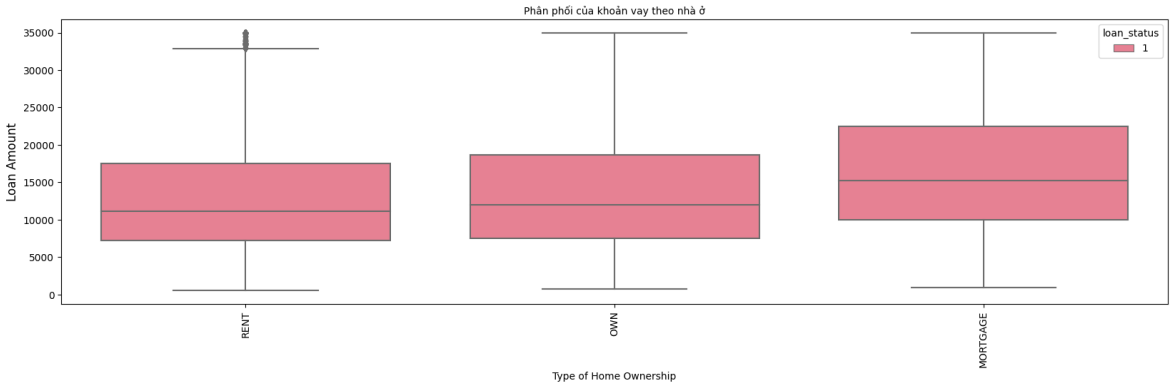
Out[35]:
```
Text(0, 0.5, 'Count')
```

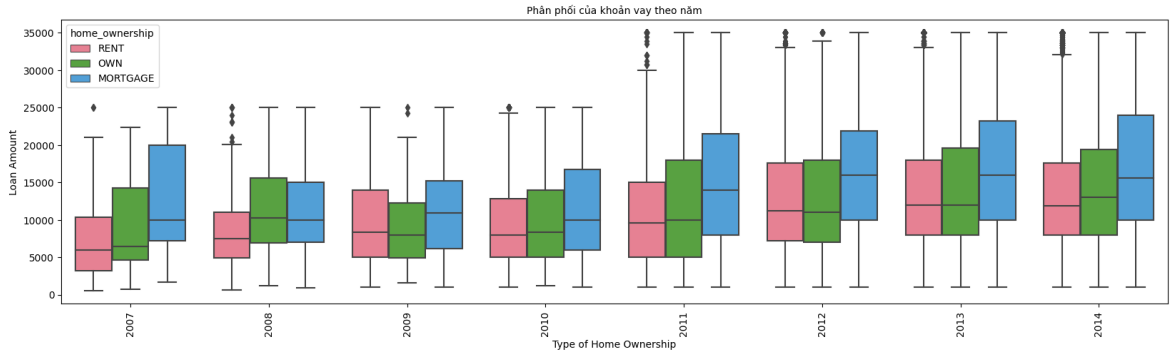




In [50]:

```
plt.figure(figsize=(20,15))
bad_df = df.loc[df['loan_status'] == 1]

plt.subplot(211)
g = sns.boxplot(x='home_ownership', y='loan_amnt', hue='loan_status',
                data = bad_df, palette = 'husl')

g.set_xticklabels(g.get_xticklabels(),rotation= 90)
g.set_xlabel("Type of Home Ownership", fontsize=10)
g.set_ylabel("Loan Amount", fontsize=12)
g.set_title("Phân phối của khoản vay theo nhà ở", fontsize=10)

plt.subplot(212)
g1 = sns.boxplot(x='year_start', y='loan_amnt', hue='home_ownership',
                data=bad_df, palette="husl")
g1.set_xticklabels(g1.get_xticklabels(),rotation=90)
g1.set_xlabel("Type of Home Ownership", fontsize=10)
g1.set_ylabel("Loan Amount", fontsize=10)
g1.set_title("Phân phối của khoản vay theo năm", fontsize= 10)


plt.subplots_adjust(hspace = 0.6, top = 0.8)

plt.show()
```





Nhận xét:

- Nhóm có lãi suất cao dễ bị rủi ro hơn nhóm lãi suất thấp
- Nhóm `kỳ hạn 60 có rủi ro cao hơn`
- Người có tài sản thế chấp dễ đi vay và vay càng cao khả năng là nợ xấu
- Khoản `vay thế chấp` có trung vị ổn định hơn qua các năm và có giá trị vay tăng - được xem xét như dễ bị coi là `nợ xấu`
- Nhóm `thuê nhà` là nhóm vay ít so với các nhóm còn lại và là nhóm nếu vay dễ vay những `khoản tiền lớn (outliers)`

Binning, Weight of Evidence (WoE), Information Value (IV)

WoE

- `WOE (weight of evidence)` Phương pháp này sẽ xếp hạng các biến thành mạnh, trung bình, yếu, không tác động,... dựa trên khả năng, sức mạnh dự báo nợ xấu. Tiêu chuẩn xếp hạng sẽ là chỉ số giá trị thông tin IV (information value) được tính toán từ phương pháp WOE. Đồng thời mô hình cũng tạo ra các giá trị features cho mỗi biến. Giá trị này sẽ đo lường sự khác biệt trong phân phối giữa good và bad. Cụ thể như sau:

Phương pháp WOE sẽ có các kĩ thuật xử lý khác biệt đối với biến liên tục và biến phân loại:

- Trường hợp `biến liên tục`, WOE sẽ gán nhãn cho mỗi một quan sát theo nhãn giá trị bins mà nó thuộc về. Các `bins` sẽ là các khoảng liên tiếp được xác định từ biến liên tục sao cho số lượng quan sát ở mỗi bin là bằng nhau. Để xác định các bins

thì ta cần xác định số lượng bins. Chúng ta có thể hình dung đầu mút của các khoảng bins chính là các quantile.

- Trường hợp `biến phân loại`, WOE có thể cân nhắc mỗi một class là một bin hoặc có thể nhóm vài nhóm có số lượng quan sát ít vào một bin. Ngoài ra mức độ chênh lệch giữa phân phối good/bad được đo lường thông qua chỉ số WOE cũng có thể được sử dụng để nhận diện các nhóm có cùng tính chất phân loại. Nếu giá trị `WOE của chúng càng gần nhau thì có thể chúng sẽ được nhóm vào một nhóm`. Ngoài ra, trường hợp Null cũng có thể được coi là một nhóm riêng biệt nếu số lượng của nó là đáng kể hoặc nhóm vào các nhóm khác nếu nó là thiểu số.

`Tính chất của WOE`: Giá trị WOE tại một bin càng lớn là dấu hiệu chứng tỏ đặc trưng rất tốt trong việc nhận diện hồ sơ Good và trái lại nếu giá trị WOE càng nhỏ thì đặc trưng bin sẽ rất tốt trong việc nhận diện hồ sơ Bad. WOE > 1 thì phân phối của hồ sơ Good đang chiếm ưu thế hơn Bad và trái lại.

Information Value

- <= 0.02: Biến không có tác dụng trong việc phân loại hồ sơ Good/Bad
- 0.02 - 0.1: yếu
- 0.1 - 0.3: trung bình
- 0.3 - 0.5: mạnh
- => 0.5: Biến rất mạnh

Ref : https://phamdinhkhanh.github.io/2020/01/17/ScoreCard.html

In [ ]:
```
df = df.drop(columns = 'index')
```

In [37]:
```
df2 = df.drop(columns = ['issue_d','earliest_cr_line', 'last_pymnt_d', 'last_cre
                         'year_start', 'income_category','interest_type'])
```

Scale lại dữ liệu

In [38]:
```
from scipy import stats
def analyze_skewness(x):
    fig, ax = plt.subplots(2, 2, figsize=(5,5))
    sns.distplot(input[x], ax=ax[0,0])
    sns.distplot(np.log(input[x]), ax=ax[0,1])
    sns.distplot(np.sqrt(input[x]), ax=ax[1,0])
    sns.distplot(stats.boxcox(input[x])[0], ax=ax[1,1])
    plt.tight_layout()
    plt.show()

    print(input[x].skew().round(2))
    print(np.log(input[x]).skew().round(2))
    print(np.sqrt(input[x]).skew().round(2))
    print(pd.Series(stats.boxcox(input[x])[0]).skew().round(2))
```

In [39]:
```
# Scale
df2['loan_amnt'] = stats.boxcox(df2['loan_amnt'])[0]
df2['int_rate'] = stats.boxcox(df2['int_rate'])[0]
df2['annual_inc'] = stats.boxcox(df2['annual_inc'])[0]
df2['open_acc'] = stats.boxcox(df2['open_acc'])[0]
df2['total_acc'] = stats.boxcox(df2['total_acc'])[0]
```

```
df2['loan_issue_m'] = stats.boxcox(df2['loan_issue_m'])[0]
df2['cr_line_issue_m'] = stats.boxcox(df2['cr_line_issue_m'])[0]

df2['mths_since_last_delinq'] = np.sqrt(df2['mths_since_last_delinq'])[0]
df2['open_acc'] = np.sqrt(df2['open_acc'])[0]
df2['revol_bal'] = np.sqrt(df2['revol_bal'])[0]
df2['out_prncp'] = np.sqrt(df2['out_prncp'])[0]
df2['collections_12_mths_ex_med'] = np.sqrt(df2['collections_12_mths_ex_med'])[0
df2['delinq_pct'] = np.sqrt(df2['delinq_pct'])[0]
```

In [75]:
```
target = pd.DataFrame(df2['loan_status'])
input = df2.drop(columns = 'loan_status')
```

In [76]:
```
input.head()
```

Out[76]:

	loan_amnt	term	int_rate	grade	emp_length	home_ownership	annual_inc	verifica
0	51.526	36	4.732	B	10	RENT	7.168	
1	39.978	60	6.128	C	0	RENT	7.275	Sou
2	39.380	36	6.318	C	10	RENT	6.835	
3	66.182	36	5.617	C	10	RENT	7.506	Sou
4	51.526	36	3.761	A	3	RENT	7.361	Sou

In [77]:
```
def iv_woe(data, target, bins=10, show_woe=False):

    newDF,woeDF = pd.DataFrame(), pd.DataFrame()
    cols = data.columns

    #Run WoE and IV on all the independent variables
    for ivars in cols[~cols.isin([target])]:
        if (data[ivars].dtype.kind in 'bifc') and (len(np.unique(data[ivars]))>1
            binned_x = pd.qcut(data[ivars], bins,  duplicates='drop')
            d0 = pd.DataFrame({'x': binned_x, 'y': data[target]})
        else:
            d0 = pd.DataFrame({'x': data[ivars], 'y': data[target]})
        d = d0.groupby("x", as_index=False).agg({"y": ["count", "sum"]})
        d.columns = ['Cutoff', 'N', 'Events']
        d['% of Events'] = np.maximum(d['Events'], 0.5) / d['Events'].sum()
        d['Non-Events'] = d['N'] - d['Events']
        d['% of Non-Events'] = np.maximum(d['Non-Events'], 0.5) / d['Non-Events'
        d['WoE'] = np.log(d['% of Events']/d['% of Non-Events'])
        d['IV'] = d['WoE'] * (d['% of Events'] - d['% of Non-Events'])
        d.insert(loc=0, column='Variable', value=ivars)
        print("Information value of " + ivars + " is " + str(round(d['IV'].sum()
        temp =pd.DataFrame({"Variable" : [ivars], "IV" : [d['IV'].sum()]}, colum
        newDF=pd.concat([newDF,temp], axis=0)
        woeDF=pd.concat([woeDF,d], axis=0)

        if show_woe == True:
            print(d)

    return newDF, woeDF

iv, woe = iv_woe(df2, target ='loan_status', bins=10)
```

```
Information value of loan_amnt is 0.036372
Information value of term is 0.162496
Information value of int_rate is 0.406067
Information value of grade is 0.395751
Information value of emp_length is 0.001736
Information value of home_ownership is 0.013449
Information value of annual_inc is 0.047286
Information value of verification_status is 0.043506
Information value of purpose is 0.025225
Information value of dti is 0.093996
Information value of delinq_2yrs is 0.003856
Information value of inq_last_6mths is 0.012058
Information value of mths_since_last_delinq is 0.0
Information value of open_acc is 0.0
Information value of pub_rec is 0.000904
Information value of revol_bal is 0.0
Information value of revol_util is 0.06352
Information value of total_acc is 0.002891
Information value of initial_list_status is 0.008995
Information value of out_prncp is 0.0
Information value of last_pymnt_amnt is 4.712845
Information value of collections_12_mths_ex_med is 0.0
Information value of acc_now_delinq is 0.00056
Information value of delinq_pct is 0.0
Information value of loan_issue_m is 0.077281
Information value of cr_line_issue_m is 0.010763
```

In [78]:
```
# Nhóm lại các đầu vào dữ liệu
selected_iv = iv[(iv['IV']>0.02) & (iv['IV'] <0.5)]
selected_var = list(selected_iv['Variable'].unique())
not_selected_var = [x for x in list(iv['Variable']) if x not in selected_var]
```

In [79]:
```
input = input[selected_var]
input_score = input.copy()
```

In [80]:
```
woe = woe[(woe['Variable'] == 'loan_amnt') |(woe['Variable'] == 'term')| (woe['V
          (woe['Variable'] == 'verification_status') |(woe['Variable'] == 'annua
          (woe['Variable'] == 'purpose') | (woe['Variable'] == 'dti') | (woe['Va
```

In [81]:
```
woe = woe.reset_index(drop = True)
woe.head(10)
```

Out[81]:

	Variable	Cutoff	N	Events	% of Events	Non-Events	% of Non-Events	WoE	IV
0	loan_amnt	(21.744999999999997, 50.477]	23813	4199	0.081	19614	0.105	-0.258	0.006
1	loan_amnt	(50.477, 55.868]	24104	4391	0.085	19713	0.106	-0.218	0.005
2	loan_amnt	(55.868, 61.078]	24135	4433	0.086	19702	0.106	-0.208	0.004
3	loan_amnt	(61.078, 66.182]	32750	6679	0.130	26071	0.140	-0.079	0.001
4	loan_amnt	(66.182, 70.654]	24770	5181	0.100	19589	0.105	-0.047	0.000
5	loan_amnt	(70.654, 74.898]	13247	2883	0.056	10364	0.056	0.004	0.000
6	loan_amnt	(74.898, 78.654]	23623	5344	0.104	18279	0.098	0.053	0.000
7	loan_amnt	(78.654, 84.785]	28405	6938	0.135	21467	0.115	0.154	0.003
8	loan_amnt	(84.785, 91.782]	21875	5526	0.107	16349	0.088	0.199	0.004
9	loan_amnt	(91.782, 103.401]	20931	5997	0.116	14934	0.080	0.371	0.013

In [82]:
```
woe['Cutoff'] = woe['Cutoff'].replace(woe['Cutoff'].values[0],pd.Interval(21.745
woe['Cutoff'] = woe['Cutoff'].replace(woe['Cutoff'].values[29],pd.Interval(5.82,
woe['Cutoff'] = woe['Cutoff'].replace(woe['Cutoff'].values[85], pd.Interval(3.27
```

In [84]:
```
# Bining lại dữ liệu
pd.options.display.float_format = '{:.3f}'.format
input['loan_amnt'] = pd.cut(input['loan_amnt'], bins = (21.745, 50.477,  55.868,
input['int_rate'] = pd.cut(input['int_rate'], bins = (2.737, 3.761, 4.511, 5.004
input['annual_inc'] = pd.cut(input['annual_inc'], bins = (5.82, 7.32, 7.415, 7.4
input['dti'] = pd.cut(input['dti'], bins = (-0.001, 6.4, 9.46, 11.88, 14.03, 16.
input['revol_util'] = pd.cut(input['revol_util'], bins= (-0.001, 20.1, 32.5, 41.
input['loan_issue_m'] = pd.cut(input['loan_issue_m'], bins = (2.172, 2.388, 2.55
```

In [85]:
```
for col in list(input.columns):
    key = list(woe['Cutoff'][woe['Variable']==  col])
    value = list(woe['WoE'][woe['Variable']== col])
    d = dict(zip(key, value))
    input[col] = input[col].map(d)
```

In [86]:
```
input.head()
```

Out[86]:

	loan_amnt	term	int_rate	grade	annual_inc	verification_status	purpose	dti
0	-0.218	-0.243	-0.568	-0.440	0.306	0.158	-0.152	0.571
1	-0.258	0.679	0.201	0.132	0.306	0.122	-0.557	-0.475
2	-0.258	-0.243	0.430	0.132	0.306	-0.305	0.526	-0.364
3	-0.047	-0.243	-0.134	0.132	0.092	0.122	0.135	0.080
4	-0.218	-0.243	-0.792	-1.241	0.247	0.122	-0.567	-0.255

Lựa chọn mô hình phân loại và tối ưu hóa

In [104…
```
from sklearn.model_selection import train_test_split, KFold, cross_validate, cro
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
#from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
```

In [115…
```
#Chia train -test data
from sklearn.model_selection import train_test_split
RANDOM_STATE = 50
x_train, x_test, y_train, y_test = train_test_split(input, target, test_size=0.2

#kết quả
print("Variables in Train Set : {} & Test Set : {}".format(len(x_train), len(x_t
y_train.value_counts()
```

```
Variables in Train Set : 190122 & Test Set : 47531
```

Out[115]:
```
loan_status
0              148833
1               41289
dtype: int64
```

In [116…
```
DT_model = DecisionTreeClassifier(random_state=RANDOM_STATE)
RF_model= RandomForestClassifier(random_state=RANDOM_STATE,n_jobs=-1)
LR_model= LogisticRegression(random_state=RANDOM_STATE,n_jobs=-1)
XGB_model = XGBClassifier(random_state=RANDOM_STATE, n_jobs=-1)
model = [DT_model, RF_model, LR_model]
```

In [117…
```
for i in range(len(model)):
    kfolds = 4
    split = KFold(n_splits= kfolds, shuffle=True, random_state= RANDOM_STATE)
    output = cross_val_score(model[i], x_train, y_train, cv= split, scoring ='ac

    min_score = round(min(output), 4)
    max_score = round(max(output), 4)
    mean_score = round(np.mean(output), 4)
    std_dev = round(np.std(output), 4)

    print(f"{model[i]} cross validation accuarcy score: {mean_score} +/- {std_de
```

```
DecisionTreeClassifier(random_state=50) cross validation accuarcy score: 0.6823
+/- 0.0027 (std) min: 0.6792, max: 0.685,
RandomForestClassifier(n_jobs=-1, random_state=50) cross validation accuarcy scor
e: 0.7707 +/- 0.0014 (std) min: 0.7686, max: 0.7724,
LogisticRegression(n_jobs=-1, random_state=50) cross validation accuarcy score:
0.786 +/- 0.0018 (std) min: 0.7837, max: 0.7886,
```

LR có sai số nhỏ và accuracy score lớn nhất, lựa chọn `LR làm model chính`

In [118…
```
from sklearn.datasets import make_blobs
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
#model.fit(x_train, y_train)

# define models and parameters
model = LogisticRegression(random_state=RANDOM_STATE,n_jobs=-1)
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
# define grid search
grid = dict(solver=solvers,penalty=penalty,C=c_values)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats= 10, random_state= RANDOM_ST
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, s
grid_result = grid_search.fit(x_train, y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.785990 using {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.785987 (0.001144) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.785987 (0.001144) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.785988 (0.001145) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.785987 (0.001144) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.785987 (0.001143) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.785990 (0.001144) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.785986 (0.001147) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.785987 (0.001148) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.785985 (0.001148) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.785965 (0.001142) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.785965 (0.001142) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.785972 (0.001135) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.785870 (0.001126) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
0.785869 (0.001125) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
0.785871 (0.001134) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

In [119…
```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve,
# Dự đoán mô hình
model  = LogisticRegression(random_state=RANDOM_STATE,n_jobs=-1, C =10, penalty
y_preds = model.predict(x_test)
#classification report
print(classification_report(y_test, y_preds))
```

```
              precision    recall  f1-score   support

           0       0.79      0.98      0.88     37249
           1       0.55      0.08      0.14     10282

    accuracy                           0.79     47531
   macro avg       0.67      0.53      0.51     47531
weighted avg       0.74      0.79      0.72     47531
```

In [120…

```
#tạo đầu vào cho Roc-auc curve
y_hat_test_proba = model.predict_proba(x_test)
y_hat_test_proba = y_hat_test_proba[:][: , 1]
y_test_temp = y_test.copy()
y_test_temp.reset_index(drop = True, inplace = True)
y_test_proba = pd.concat([y_test_temp, pd.DataFrame(y_hat_test_proba), pd.DataFr
y_test_proba.columns = ['y_test_class_actual', 'y_hat_test_proba', 'y_hat_test']
y_test_proba.index = x_test.index
y_test_proba.head()
```

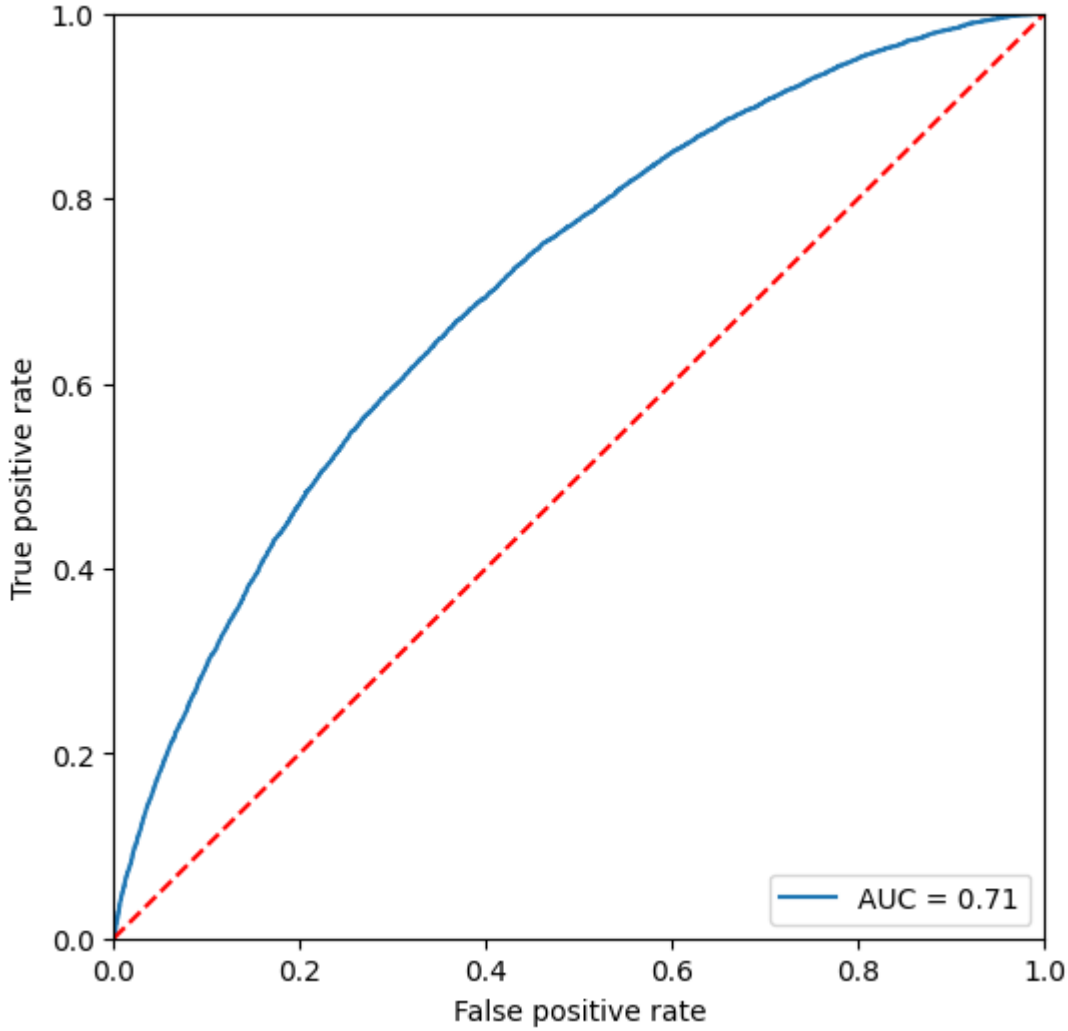
Out[120]:

	y_test_class_actual	y_hat_test_proba	y_hat_test
39267	1	0.230	0
15969	1	0.112	0
48832	0	0.109	0
32610	1	0.140	0
95300	0	0.297	0

In [121…

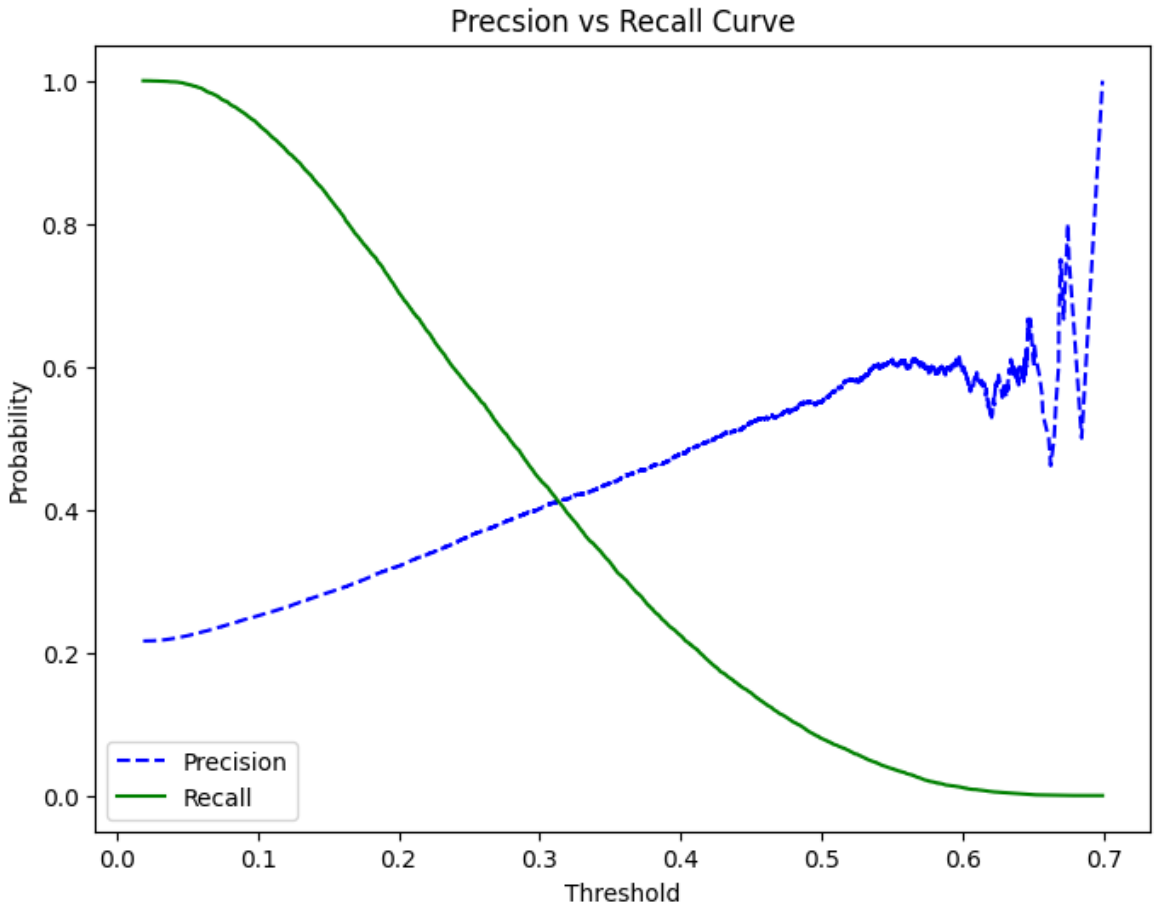
```
# get the values required to plot a ROC curve
import sklearn.metrics as metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test_proba['y_test_class_actual'], y_
roc_auc = metrics.auc(fpr, tpr)

# roc_curve n gini
Gini_index = round((2* roc_auc -1),2)
print('Hệ số gini của mô hình là {}'.format(Gini_index))

plt.figure(figsize = (6, 6))
plt.plot(fpr, tpr, label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc =  'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.show()
```

```
Hệ số gini của mô hình là 0.41
```



In [122…

```
from sklearn.metrics import precision_recall_curve
precision, recall, thres = precision_recall_curve(y_test, y_hat_test_proba)

def _plot_prec_rec_curve(prec, rec, thres):
    plt.figure(figsize = (8, 6))
    plt.plot(thres, prec[:-1], 'b--', label = 'Precision')
    plt.plot(thres, rec[:-1], 'g-', label = 'Recall')
    plt.xlabel('Threshold')
    plt.ylabel('Probability')
    plt.title('Precsion vs Recall Curve')
    plt.legend()

_plot_prec_rec_curve(precision, recall, thres)
```



Kiểm định Kolmogorov-Smirnov về phân phối 2 nhóm đầu ra

In [123…

```
def _KM(y_pred, n_bins):
  _, thresholds = pd.qcut(y_pred, q=n_bins, retbins=True)
  cmd_BAD = []
  cmd_GOOD = []
  BAD_id = set(np.where(y_test == 0)[0])
  GOOD_id = set(np.where(y_test == 1)[0])
  total_BAD = len(BAD_id)
  total_GOOD = len(GOOD_id)
  for thres in thresholds:
    pred_id = set(np.where(y_pred <= thres)[0])
    # Đếm % số lượng hồ sơ BAD có xác suất dự báo nhỏ hơn hoặc bằng thres
    per_BAD = len(pred_id.intersection(BAD_id))/total_BAD
    cmd_BAD.append(per_BAD)
    # Đếm % số lượng hồ sơ GOOD có xác suất dự báo nhỏ hơn hoặc bằng thres
    per_GOOD = len(pred_id.intersection(GOOD_id))/total_GOOD
    cmd_GOOD.append(per_GOOD)
  cmd_BAD = np.array(cmd_BAD)
  cmd_GOOD = np.array(cmd_GOOD)
  return cmd_BAD, cmd_GOOD, thresholds

cmd_BAD, cmd_GOOD, thresholds = _KM(y_hat_test_proba, n_bins=20)


from scipy import stats
stats.ks_2samp(cmd_BAD, cmd_GOOD)
```

Out[123]:

```
KstestResult(statistic=0.2857142857142857, pvalue=0.36497950870925666, statisti
c_location=0.2709589574012838, statistic_sign=-1)
```

Nhận xét

- Accuracy Score = 0.79, tuy vậy precise và recall của `1` rất thấp, trong khi ngược lại đối với `0` thì cao, mô hình có khả năng nhận biết tốt đối với các `khoản vay tốt`
- AUC = 0.71, thể hiện `khả năng phân loại` của mô hình ở mức `tương đối - kém` dựa trên bảng chất lượng.
- `p-value > 0.05` cho thấy phân phối tích lũy giữa tỷ lệ BAD và GOOD là chưa có sự khác biệt nhau. Do đó mô hình có chưa có ý nghĩa trong phân loại hồ sơ.

Tính điểm Scorecard

$$CreditScore = (beta \cdot WOE + \frac{alpha}{n}) \cdot Factor + \frac{Offset}{n}$$

Note:

- $Odds = 1:50$
- $Base_score = 600$
- pdo là $mức$ điểm để $gấp$ đôi $odds (mặc\ định = 20)$
- $Factor = \frac{pdo}{ln(2)}$
- $Offset = Base_score - Factor \cdot ln(Odds)$

In [124…
```
#Tạo hàm tính điểm
def CreditScore(beta, alpha, woe, n = 12, odds = 1/4, pdo = -50, thres_score = 6
  factor = pdo/np.log(2)
  offset = thres_score - factor*np.log(odds)
  score = (beta*woe+alpha/n)*factor+offset/n
  return score
```

In [125…
```
betas_dict = dict(zip(list(x_train.columns), model.coef_[0]))
alpha = model.intercept_[0]
betas_dict
```

Out[125]:
```
{'loan_amnt': 0.6647805234646367,
 'term': 0.41446877823012535,
 'int_rate': 0.37385693618440485,
 'grade': 0.38904946263696877,
 'annual_inc': 1.3907282467445767,
 'verification_status': 0.07958199670649607,
 'purpose': 0.5215111146285062,
 'dti': 0.44157871219896083,
 'revol_util': 0.2845571754224391,
 'loan_issue_m': 0.6652292700381502}
```

In [126…
```
columns = list(woe['Variable'])

beta = []
alpha = model.intercept_[0]
for col in columns:
    beta.append(betas_dict[col])
woe['beta'] = beta
```

```
#tính điểm cho mỗi loại dữ liệu
score = CreditScore(beta = woe['beta'], alpha =  alpha, woe = woe['WoE'], n = 10
woe['score'] = score
```

In [127…
```
woe = woe.reset_index(drop = True)
```

In [128…
```
# Giả sử có bộ dữ liệu
test_case = input_score.iloc[5:6, :]
test_case['term'] = test_case['term'].replace(36, '36')

woe['Cutoff'] = woe['Cutoff'].astype('object')
test_case
```

Out[128]:

	loan_amnt	term	int_rate	grade	annual_inc	verification_status	purpose	dti	re
5	42.754	36	7.023	E	7.495	Source Verified	car	5.350	

In [129…
```
# Hàm tính toán cho 1 trường dữ liệu
def scoring(obs, col):
    for i in list(woe[woe['Variable'] == col].index):
        if (obs[col].values[0] in woe['Cutoff'][i]) == True:
            score = round(woe[(woe['Variable'] == col)]['score'][i],3)
            return score

scoring(test_case, 'int_rate')
```

Out[129]: 73.654

In [130…
```
# Tính tổng điểm cho test_case
def total_score(obs, columns = columns):
    scores = dict()
    for col in columns:
        scores[col] = scoring(obs, col)
    total_score = round(sum(scores.values()), 3)
    return total_score

total_score(test_case, columns = list(test_case.columns))
```

Out[130]: 671.888