UTS Model Deployment Carrissa Gloria Herman 2702322411 Dataset A

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

In [2]:
```
pd.set_option('display.max_rows', None)
```

EDA Data

Load dataset

In [3]:
```
df = pd.read_csv("Dataset_A_loan.csv")
df.head()
```

Out[3]:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_hon
0	22.0	female	Master	71948.0	0	
1	21.0	female	High School	12282.0	0	
2	25.0	female	High School	12438.0	3	
3	23.0	female	Bachelor	79753.0	0	
4	24.0	male	Master	66135.0	1	

In [4]:
```
df.shape
```

Out[4]:
```
(45000, 14)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['person_age', 'person_gender', 'person_education', 'person_income',
       'person_emp_exp', 'person_home_ownership', 'loan_amnt', 'loan_inten
t',
       'loan_int_rate', 'loan_percent_income', 'cb_person_cred_hist_length',
       'credit_score', 'previous_loan_defaults_on_file', 'loan_status'],
      dtype='object')
```

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   person_age                      45000 non-null  float64
 1   person_gender                   45000 non-null  object 
 2   person_education                45000 non-null  object 
 3   person_income                   42750 non-null  float64
 4   person_emp_exp                  45000 non-null  int64  
 5   person_home_ownership           45000 non-null  object 
 6   loan_amnt                       45000 non-null  float64
 7   loan_intent                     45000 non-null  object 
 8   loan_int_rate                   45000 non-null  float64
 9   loan_percent_income             45000 non-null  float64
 10  cb_person_cred_hist_length      45000 non-null  float64
 11  credit_score                    45000 non-null  int64  
 12  previous_loan_defaults_on_file  45000 non-null  object 
 13  loan_status                     45000 non-null  int64  
dtypes: float64(6), int64(3), object(5)
memory usage: 4.8+ MB
```

In [7]:
```python
for i in df.columns:
    print(df[i].value_counts())
    print("")
```

```
Streaming output truncated to the last 5000 lines.
4252.0         1
11651.0        1
7665.0         1
7362.0         1
13725.0        1
9942.0         1
29274.0        1
2697.0         1
5569.0         1
18330.0        1
10495.0        1
7769.0         1
11869.0        1
14814.0        1
2106.0         1
1665.0         1
7717.0         1
10886.0        1
```

Hal yang perlu diperbaiki:

Menyamakan penulisan gender "male", "female", "Male", dan "fe male"

In [8]:
```
df['person_gender'].value_counts()
```

Out[8]:

person_gender	count
male	24799
female	20111
Male	45
fe male	45

dtype: int64

In [9]:
```
df['person_gender'] = df['person_gender'].replace({'Male':'male', 'fe male': '
df['person_gender'].value_counts()
```

Out[9]:

person_gender	count
male	24844
female	20156

dtype: int64

Dataset Splitting

In [10]:
```
from sklearn.model_selection import train_test_split

x = df.drop('loan_status', axis=1).copy()
y = df['loan_status'].copy()

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =0.2, ranc
```

Missing Values

In [11]:
```
df.isna().sum()
```

Out[11]:

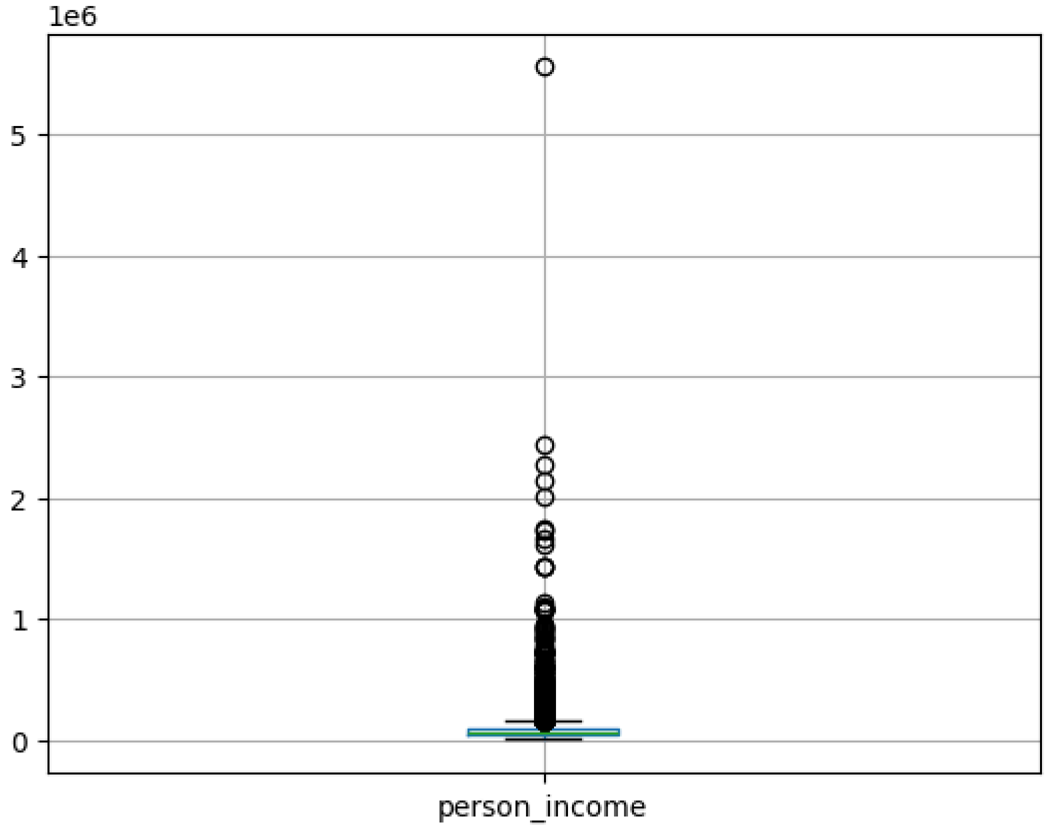
	0
person_age	0
person_gender	0
person_education	0
person_income	2250
person_emp_exp	0
person_home_ownership	0
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0
loan_status	0

dtype: int64

Sebelum impute missing value pada column 'person_income', kita perlu melakukan outlier detection untuk mengetahui cara terbaik untuk melakukan impute missing value dan memperbaiki value-value yang tidak masuk akal.

In [12]:
```
x_train.boxplot('person_income')
```

Out[12]: <Axes: >



Buat function untuk mendeteksi outlier berdasarkan peraturan IQR

In [13]:
```
def detect_outliers_iqr(data):
    outliers = []
    data = sorted(data)
    q1 = np.nanpercentile(data, 25)
    q3 = np.nanpercentile(data, 75)
    IQR = q3-q1
    lower_bound = q1-(1.5*IQR)
    upper_bound = q3+(1.5*IQR)
    for i in data:
        if (i < lower_bound or i > upper_bound):
            outliers.append(i)
    return outliers

income_outlier = detect_outliers_iqr(x_train['person_income'])
print("Outliers di column 'person_income:", income_outlier)
print("Jumlah outliers di column 'person_income:", len(income_outlier))
```

```
Outliers di column 'person_income: [270972.0, 171265.0, 183548.0, 20504
6.0, 211005.0, 356880.0, 177269.0, 199596.0, 216897.0, 301158.0, 420652.
0, 493048.0, 173294.0, 180646.0, 180875.0, 192674.0, 207525.0, 207697.0,
216923.0, 241075.0, 180643.0, 611327.0, 204962.0, 210914.0, 180752.0, 24
1153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 3225
97.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 27692
2.0, 198957.0, 179877.0, 181221.0, 241041.0, 203531.0, 211022.0, 253004.
0, 577186.0, 169439.0, 181014.0, 181110.0, 231450.0, 169438.0, 241966.0,
270797.0, 180986.0, 193014.0, 193168.0, 270875.0, 173345.0, 217215.0, 28
6397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 2267
51.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 72524
2.0, 187709.0, 193124.0, 238837.0, 175387.0, 168935.0, 180846.0, 194828.
0, 271193.0, 184645.0, 361181.0, 363682.0, 185905.0, 213645.0, 226664.0,
293525.0, 174825.0, 182130.0, 188387.0, 209988.0, 228907.0, 270729.0, 30
0843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 1691
27.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 21090
3.0, 241005.0, 313076.0, 241278.0, 174919.0, 178777.0, 206229.0, 181692.
0, 192841.0, 228928.0, 169058.0, 213479.0, 226517.0, 361226.0, 217194.0,
180779.0, 241245.0, 178447.0, 282277.0, 177226.0, 199013.0, 249925.0, 17
```

In [14]:
```
pd.set_option('display.float_format', '{:.0f}'.format)
x_train['person_income'].describe()
```

Out[14]:

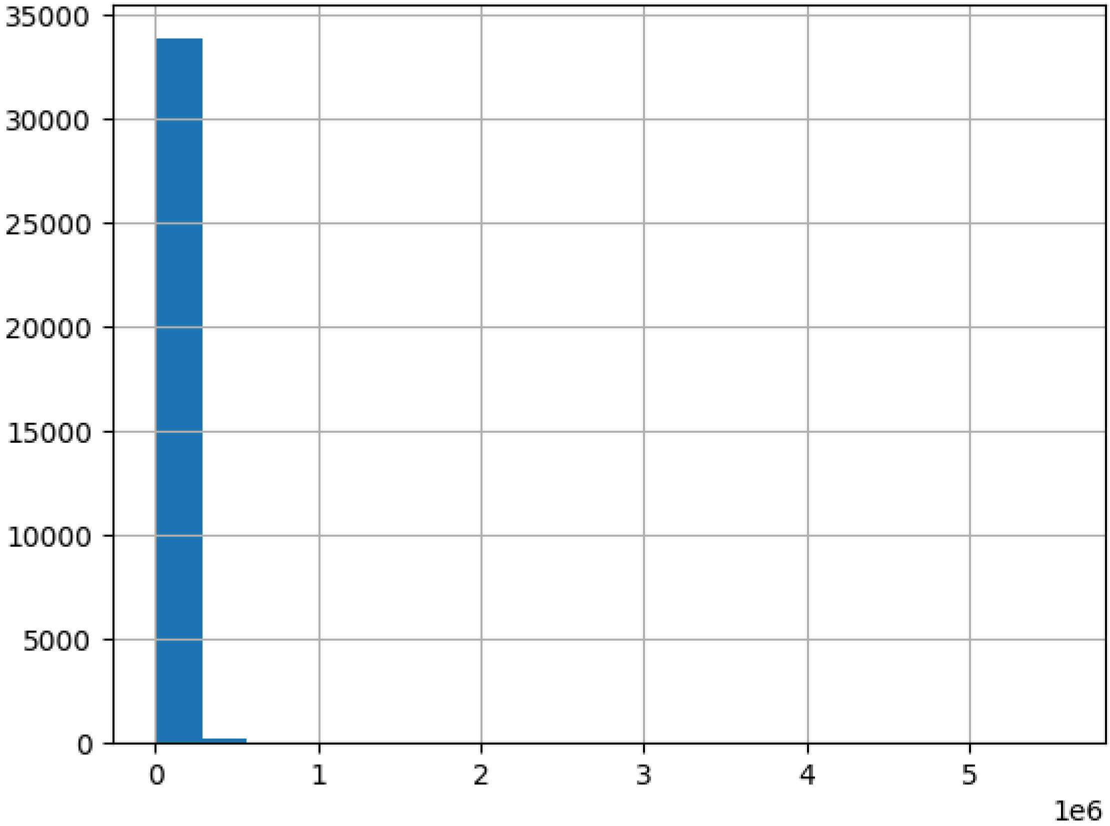
	person_income
count	34181
mean	80112
std	71186
min	8000
25%	47087
50%	67002
75%	95698
max	5556399

dtype: float64

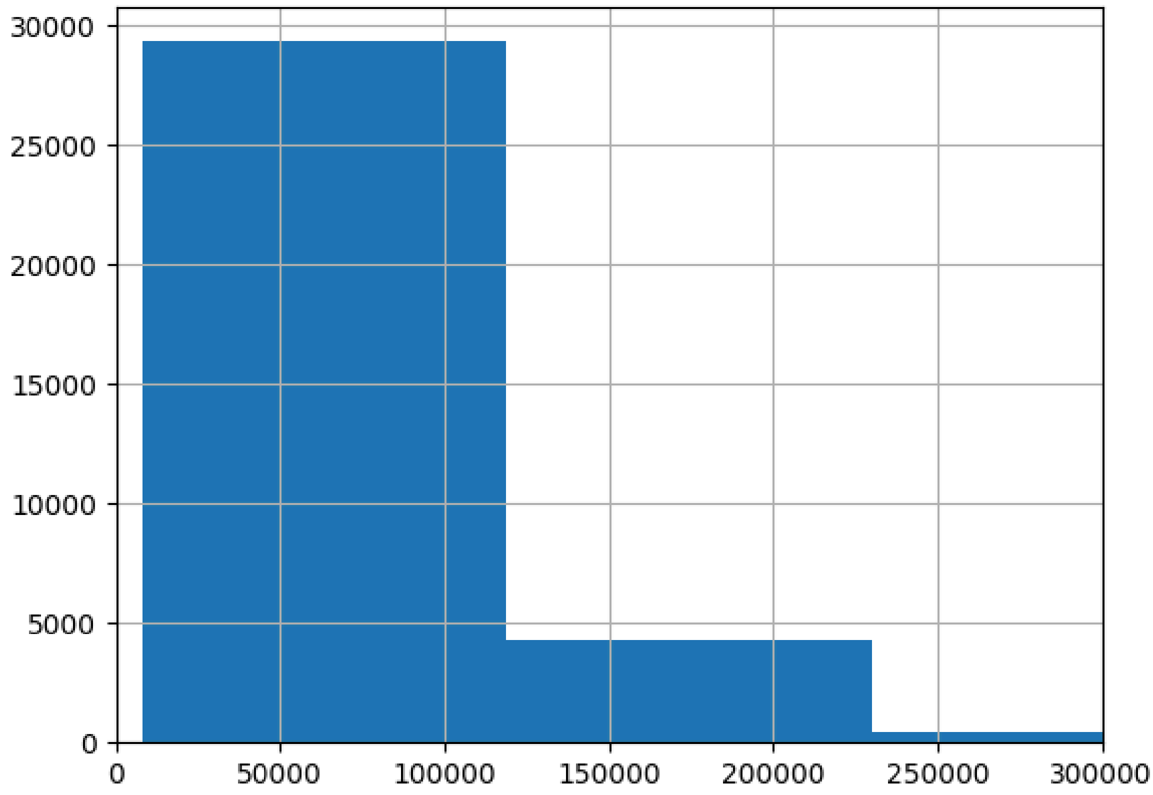
In [15]:
```
print('Min outlier:', pd.DataFrame(income_outlier).min()[0])
print('Max outlier:', pd.DataFrame(income_outlier).max()[0])
print('Mean outlier:', pd.DataFrame(income_outlier).mean()[0])
```

```
Min outlier: 168633.0
Max outlier: 5556399.0
Mean outlier: 259862.46978672987
```

In [16]:
```
x_train['person_income'].hist(bins=20)
plt.show()
```



In [17]:
```
x_train['person_income'].hist(bins=50)
plt.xlim(0, 300000)
plt.show()
```



Kita bisa melihat bahwa nilai min max, dan mean outlier jauh lebih besar daripada nilai min, max, dan mean dataset seluruhnya. Oleh karena itu, kita bisa menganggap bahwa nilai-nilai yang terdapat pada list outlier berupa true outlier dan harus di handle terlebih dahulu. Selain itu, lewat histogram, kita bisa melihat bahwa data 'person_income' sangat skewed dan jumlah outliernya sangat banyak (2112). Maka, outlier akan di-impute dengan nilai median dari training dataset.

In [18]:
```
income_median = x_train['person_income'].median()
income_median
```

Out[18]:
```
67002.0
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

In [19]:
```
def lower_upper(data):
  q1 = np.nanpercentile(data, 25)
  q3 = np.nanpercentile(data, 75)
  IQR = q3-q1
  lower_bound = q1-(1.5*IQR)
  upper_bound = q3+(1.5*IQR)
  return lower_bound, upper_bound
```

In [20]:
```
income_lower_bound, income_upper_bound = lower_upper(x_train['person_income'])
print(income_lower_bound)
print(income_upper_bound)
```

```
-25829.5
168614.5
```

Impute outlier pada semua dataset dengan median dari training dataset (income_median)

In [21]:
```
def impute_outlier(data, lower_bound, upper_bound, median):
  for i in data:
    if (i < lower_bound or i > upper_bound):
      data = data.replace(i, median)
  return data

x_train['person_income'] = impute_outlier(x_train['person_income'], income_low
x_test['person_income'] = impute_outlier(x_test['person_income'], income_lower
```

In [22]:
```
x_train['person_income'].describe()
```

Out[22]:

	person_income
count	34181
mean	70588
std	31562
min	8000
25%	47087
50%	67002
75%	88433
max	168591

dtype: float64

In [23]:
```
x_test['person_income'].describe()
```

Out[23]:

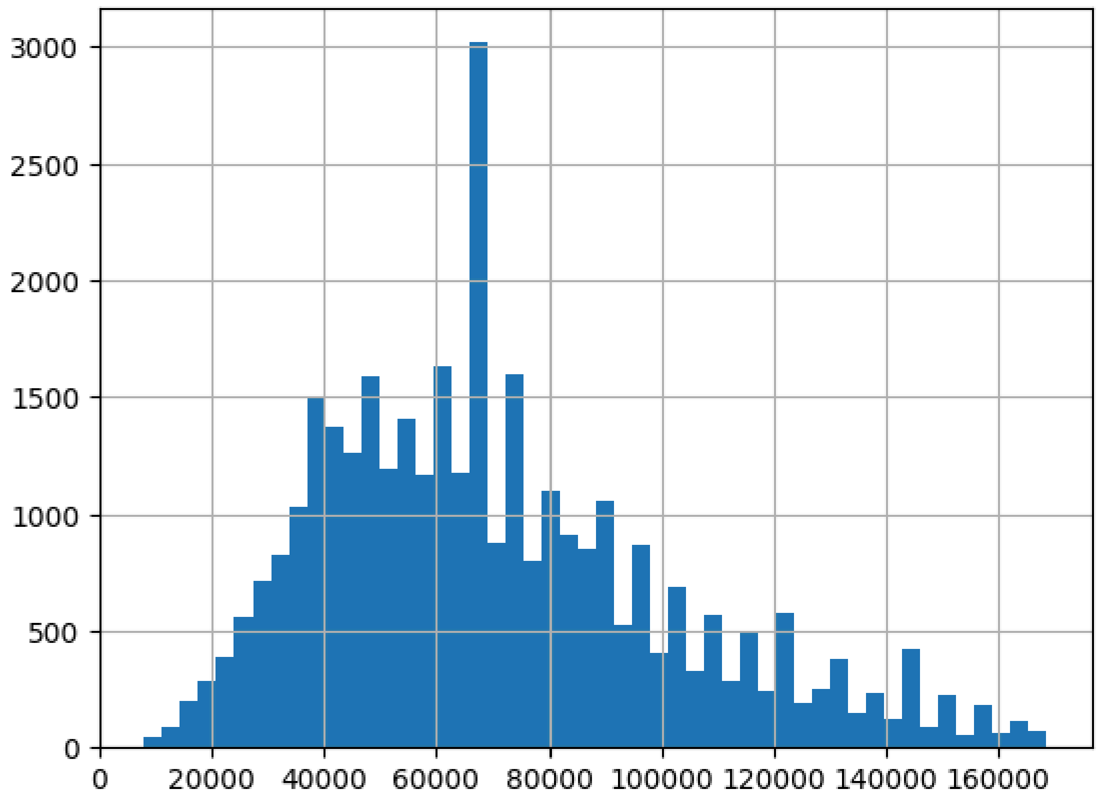
	person_income
count	8569
mean	71087
std	31926
min	8000
25%	47850
50%	67002
75%	89438
max	167935

dtype: float64

Missing value handling untuk column 'person_income'. In

In [24]:
```
x_train['person_income'].hist(bins=50)
```

Out[24]: <Axes: >



Karena dataset setelah handling outlier pun masih skewed, maka kita akan impute missing value dengan median.

In [25]:
```
new_income_median = x_train['person_income'].median()
x_train['person_income'] = x_train['person_income'].fillna(new_income_median)
x_test['person_income'] = x_test['person_income'].fillna(new_income_median)
```

In [26]:
```
x_train.isna().sum()
```

Out[26]:

	0
person_age	0
person_gender	0
person_education	0
person_income	0
person_emp_exp	0
person_home_ownership	0
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0

dtype: int64

In [27]:
```
x_test.isna().sum()
```

Out[27]:

	0
person_age	0
person_gender	0
person_education	0
person_income	0
person_emp_exp	0
person_home_ownership	0
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0

dtype: int64

In [28]:
```
y_train.isna().sum()
```

Out[28]:
```
np.int64(0)
```

In [29]:
```
y_test.isna().sum()
```

Out[29]:
```
np.int64(0)
```

Outlier Detection

Kita perlu melakukan outlier detection terhadap column/variabel numerical lain yang belum di testing. Categorical variabel tidak perlu dilakukan outlier detection karena outlier detection dilakukan saat value_counts di awal notebook.

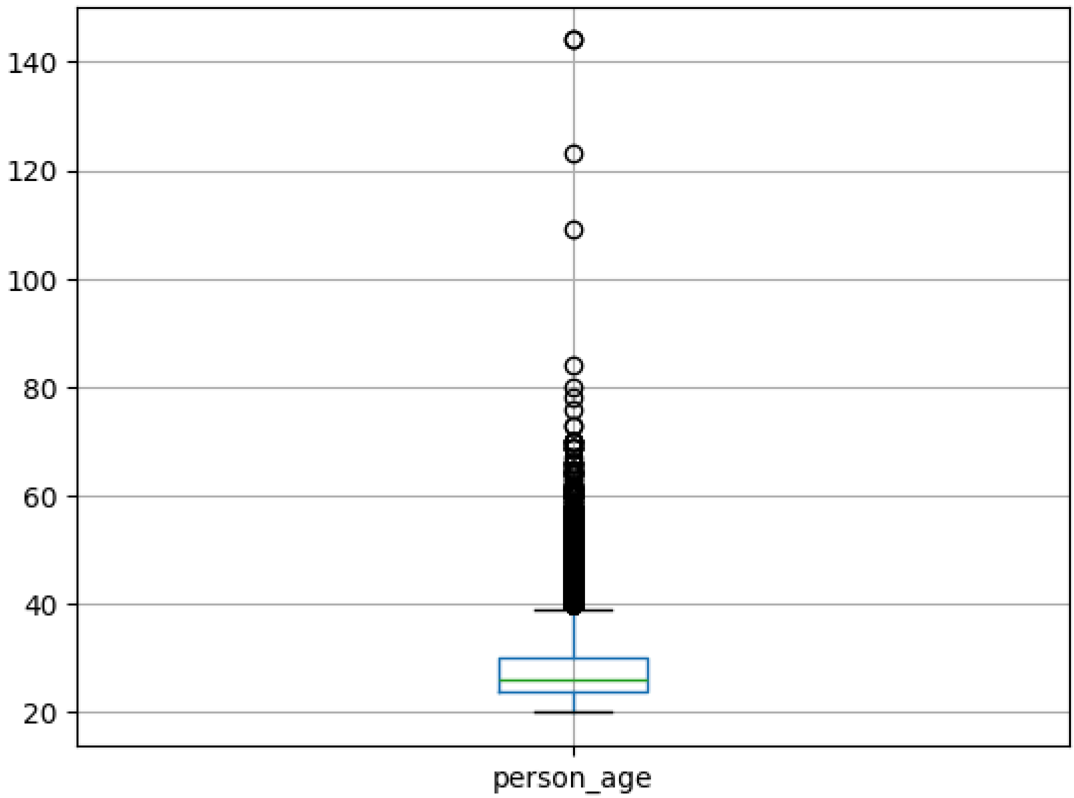
List numerical variabel:

1. person_age
2. person_emp_exp
3. loan_amnt
4. loan_int_rate
5. loan_percent_income
6. cb_person_cred_hist_length
7. credit_score

In [30]:
```
pd.reset_option('display.float_format')
```

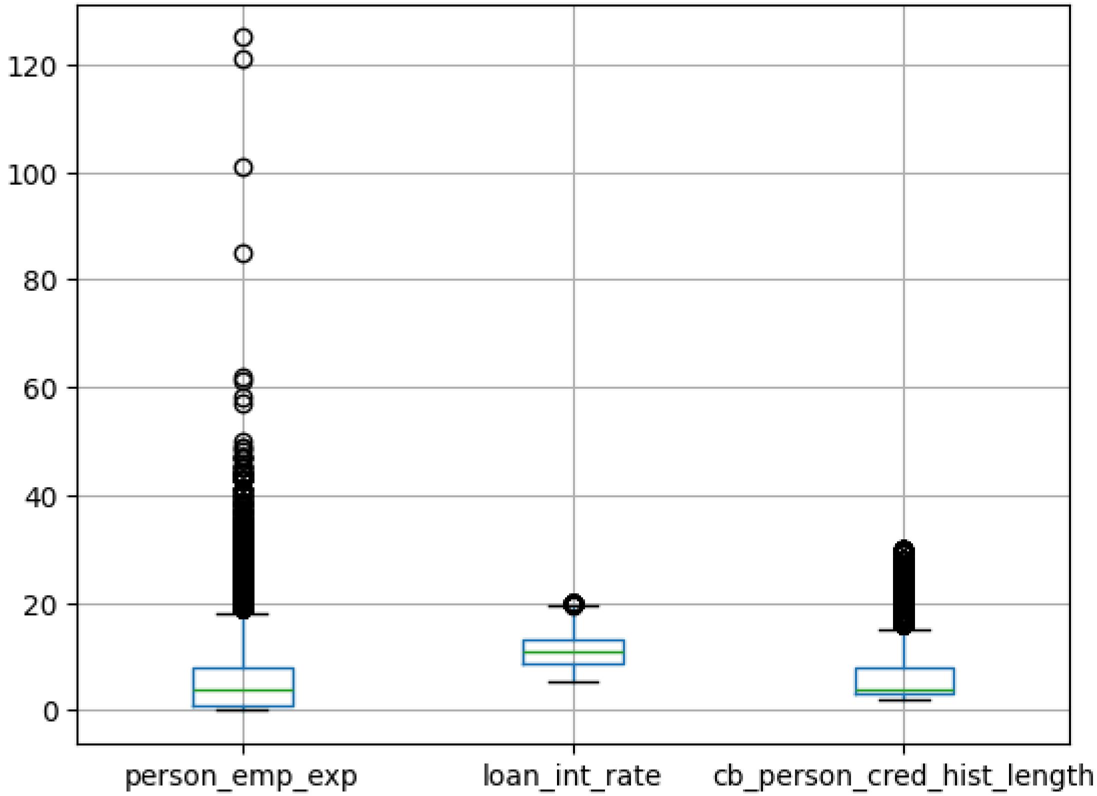
In [31]:
```
x_train.boxplot('person_age')
```

Out[31]:
```
<Axes: >
```



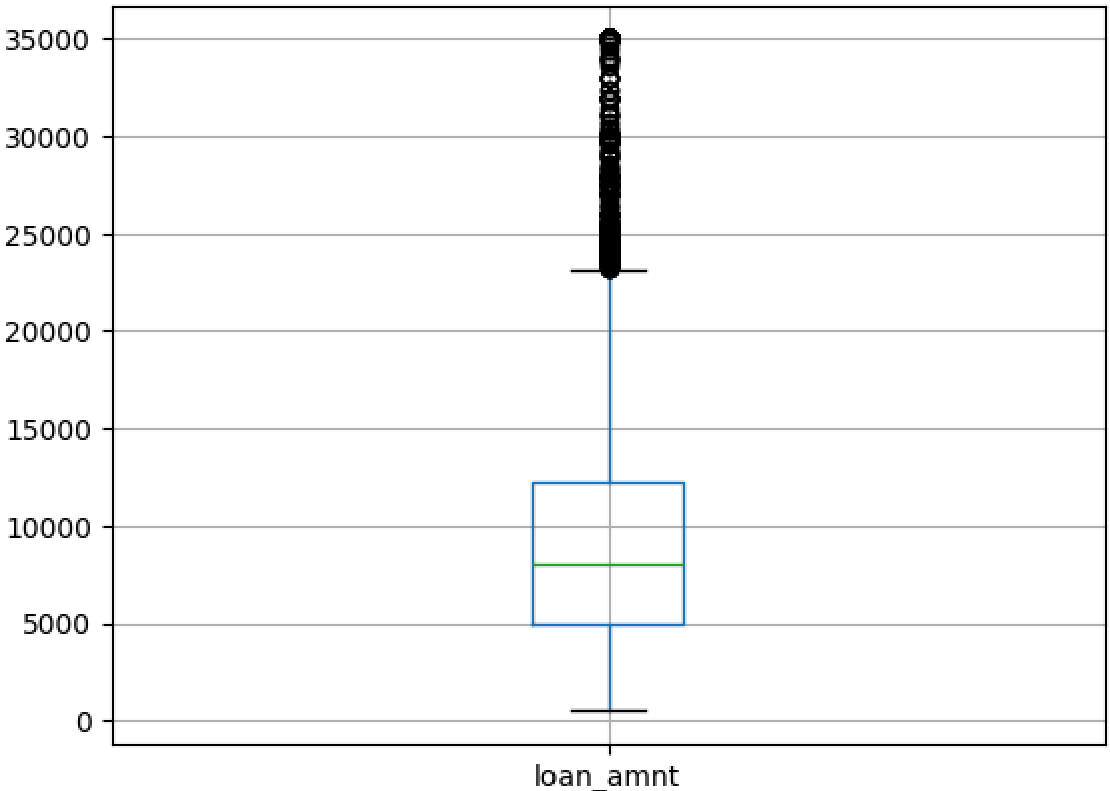
In [32]:
```
x_train.boxplot(['person_emp_exp', 'loan_int_rate', 'cb_person_cred_hist_lengt
```

Out[32]: <Axes: >



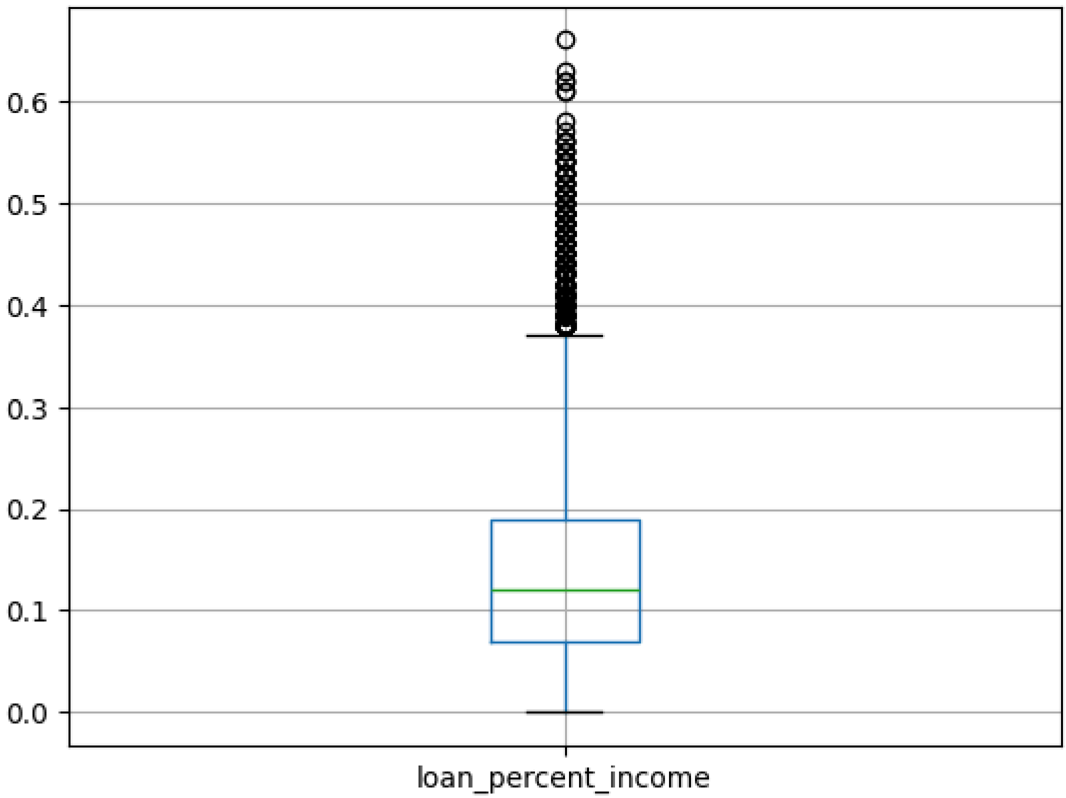
In [33]:
```
x_train.boxplot('loan_amnt')
```

Out[33]: <Axes: >



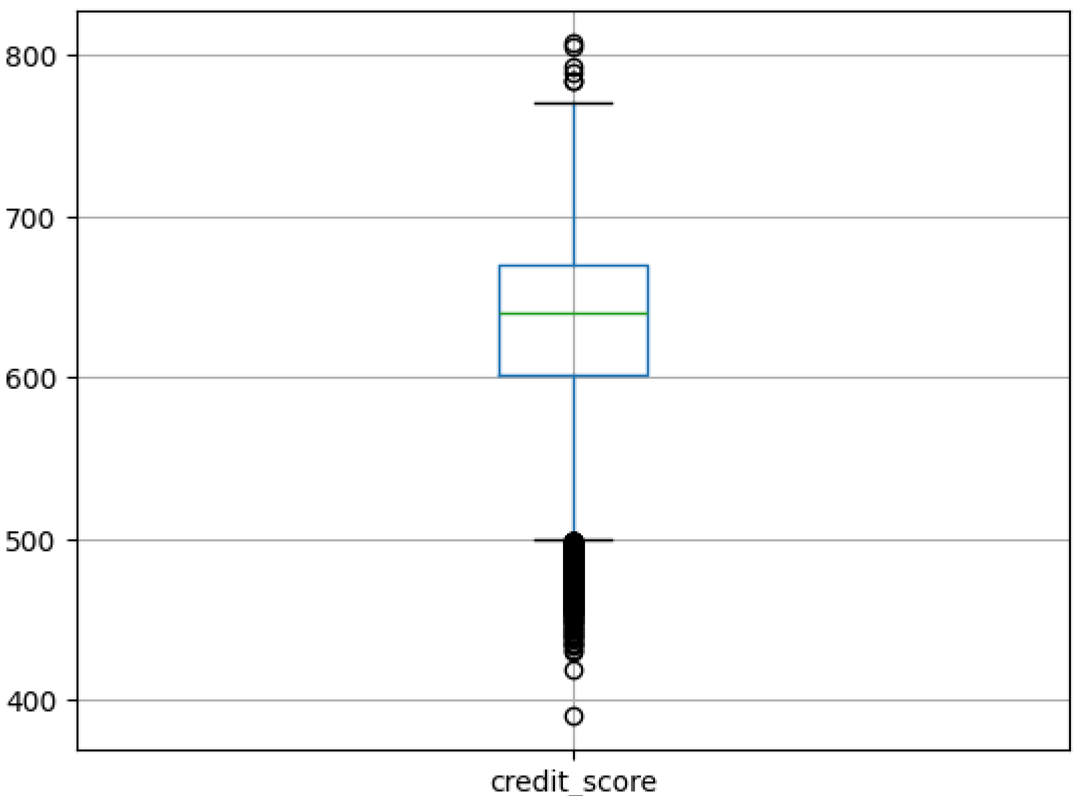
In [34]:
```
x_train.boxplot('loan_percent_income')
```

Out[34]: <Axes: >



In [35]:
```
x_train.boxplot('credit_score')
```

Out[35]: <Axes: >



Semua numerical variable memiliki banyak outlier yang perlu di handle

person_age

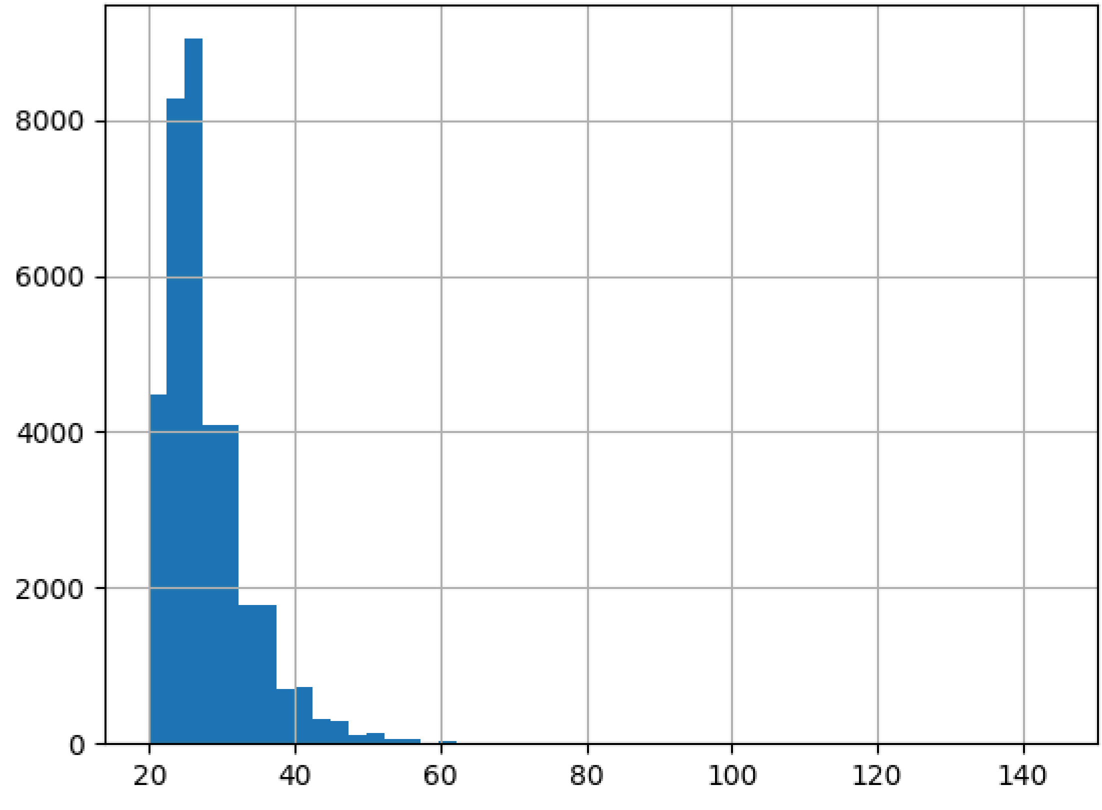
In [36]:
```
age_outlier = detect_outliers_iqr(x_train['person_age'])
print("Outliers di column 'person_age:", age_outlier)
print("Jumlah outliers di column 'person_age:", len(age_outlier))
```

```
Outliers di column 'person_age: [40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0,
40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.
0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 40.0, 4
0.0, 40.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.
0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 4
1.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.
0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 4
1.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.
0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 4
1.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.
0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 4
1.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.
0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 4
1.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0,
41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 41.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0,
42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0,
42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0,
42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0,
42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0,
42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.
0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 42.0, 4
2.0, 42.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0,
43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.
0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 4
```

```
3.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0,
43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.
0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 4
3.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0,
43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.
0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 4
3.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0,
43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.
0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 4
3.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0,
43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 43.0, 44.
0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 4
4.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0,
44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.
0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 4
4.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0,
44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.
0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 4
4.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0,
44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.
0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 4
4.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 44.0,
44.0, 44.0, 44.0, 44.0, 44.0, 44.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.
0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 4
5.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0,
45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.
0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 4
5.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0,
45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.
0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 4
5.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0, 45.0,
45.0, 45.0, 45.0, 45.0, 45.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.
0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 4
6.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0,
46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.
0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 4
6.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0,
46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.
0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 46.0, 47.0, 47.0, 47.0, 47.0, 4
7.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0,
47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.
0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 4
7.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0,
47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.
0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 4
7.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0, 47.0,
47.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.
0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 4
8.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0,
48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.
0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 4
8.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 48.0, 49.0, 49.0, 49.0, 49.0,
49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.
0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 4
9.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0, 49.0,
49.0, 49.0, 49.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.
0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 5
0.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0,
50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.0, 50.
0, 50.0, 50.0, 50.0, 50.0, 50.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 5
1.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0,
```

```
51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.
0, 51.0, 51.0, 51.0, 51.0, 51.0, 51.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 5
2.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0,
52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 52.0, 53.
0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 5
3.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0, 53.0,
53.0, 53.0, 53.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.
0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 54.0, 55.0, 5
5.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0, 55.0,
55.0, 55.0, 55.0, 55.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.
0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 56.0, 57.0, 57.0, 5
7.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0, 57.0,
58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.0, 58.
0, 58.0, 59.0, 59.0, 59.0, 59.0, 59.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 6
0.0, 60.0, 60.0, 60.0, 60.0, 60.0, 60.0, 61.0, 61.0, 61.0, 61.0, 61.0, 61.0,
61.0, 61.0, 61.0, 62.0, 62.0, 62.0, 62.0, 62.0, 62.0, 63.0, 63.0, 64.0, 64.
0, 64.0, 64.0, 64.0, 64.0, 64.0, 65.0, 65.0, 65.0, 65.0, 66.0, 66.0, 66.0, 6
6.0, 66.0, 66.0, 67.0, 69.0, 69.0, 69.0, 69.0, 69.0, 70.0, 70.0, 70.0, 70.0,
70.0, 70.0, 73.0, 73.0, 76.0, 78.0, 80.0, 84.0, 109.0, 123.0, 144.0, 144.0]
Jumlah outliers di column 'person_age: 1780
```

In [37]:
```python
x_train['person_age'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [38]:
```python
age_median = x_train['person_age'].median()
age_median
```

Out[38]: 26.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

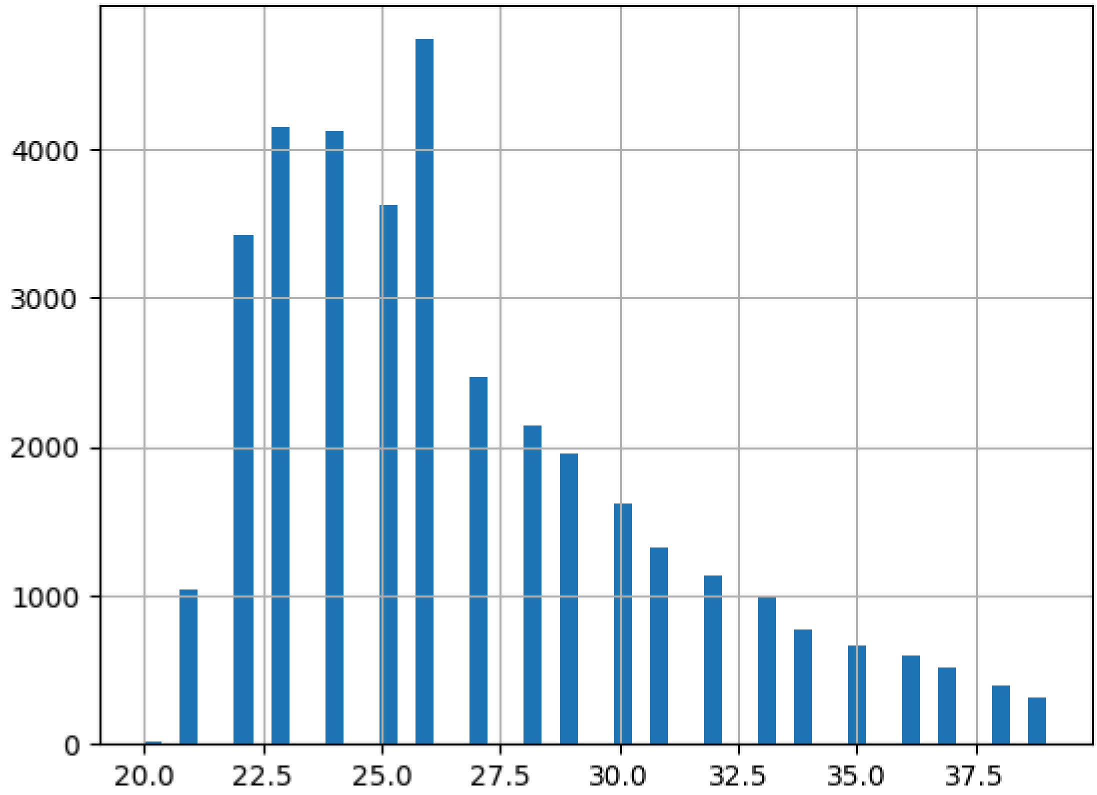
In [39]:
```
age_lower_bound, age_upper_bound = lower_upper(x_train['person_age'])
print(age_lower_bound)
print(age_upper_bound)
```

```
15.0
39.0
```

Impute outlier pada semua dataset dengan median dari training dataset

In [40]:
```
x_train['person_age'] = impute_outlier(x_train['person_age'], age_lower_bound,
x_test['person_age'] = impute_outlier(x_test['person_age'], age_lower_bound, a
```

In [41]:
```
x_train['person_age'].hist(bins=50)
plt.show()
```



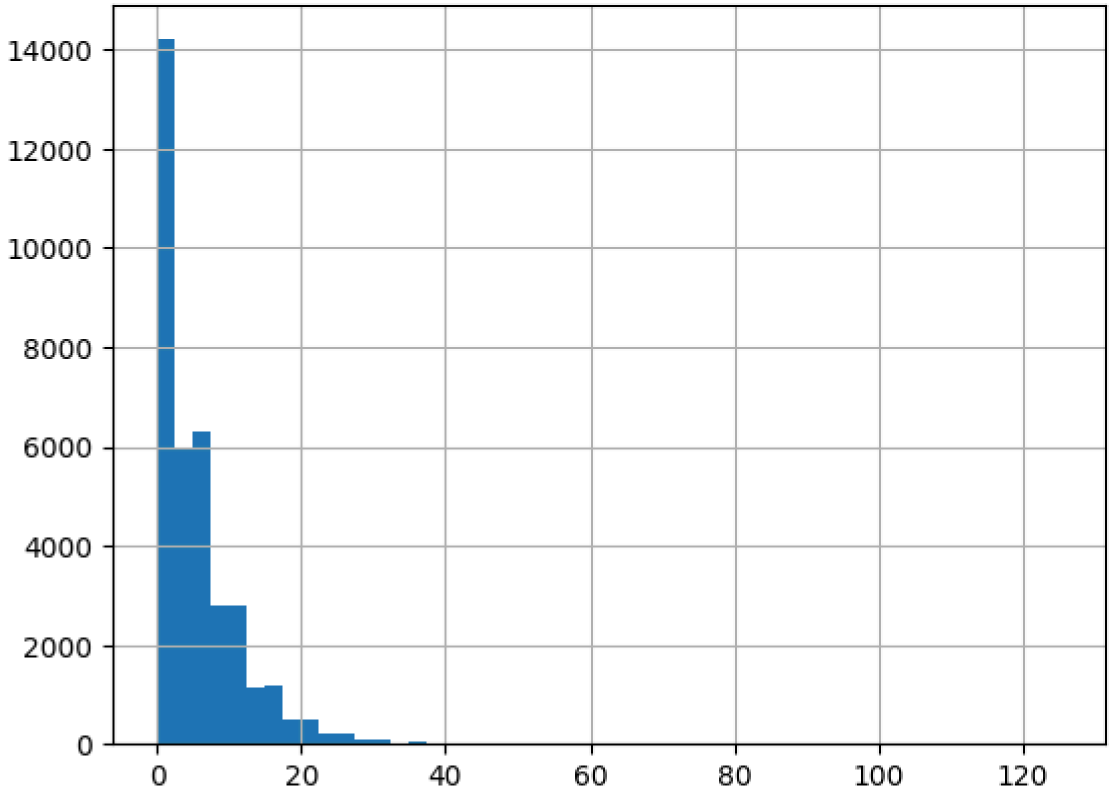
person_emp_exp

In [42]:
```
emp_exp_outlier = detect_outliers_iqr(x_train['person_emp_exp'])
print("Outliers di column 'person_emp_exp:", emp_exp_outlier)
print("Jumlah outliers di column 'person_emp_exp:", len(emp_exp_outlier))
```

```
Outliers di column 'person_emp_exp: [19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19,
19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
20, 20, 20, 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
```

```
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 30, 30,
30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
30, 30, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31,
31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33,
33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 34, 34, 34, 34, 34, 34, 34,
34, 34, 34, 34, 34, 34, 34, 34, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 35, 36, 36, 36, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 37, 37, 37, 37,
37, 37, 38, 38, 38, 38, 38, 38, 39, 39, 39, 39, 39, 39, 39, 40, 40, 40, 40,
40, 40, 40, 41, 41, 41, 41, 41, 41, 41, 43, 43, 43, 43, 43, 43, 43, 43, 44,
44, 44, 44, 44, 44, 44, 44, 44, 45, 45, 45, 45, 45, 46, 47, 47, 47, 47, 47,
48, 49, 49, 50, 57, 58, 61, 62, 85, 101, 121, 125]
Jumlah outliers di column 'person_emp_exp: 1409
```

In [43]:
```
x_train['person_emp_exp'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [44]:
```
emp_exp_median = x_train['person_emp_exp'].median()
emp_exp_median
```

Out[44]:
```
4.0
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

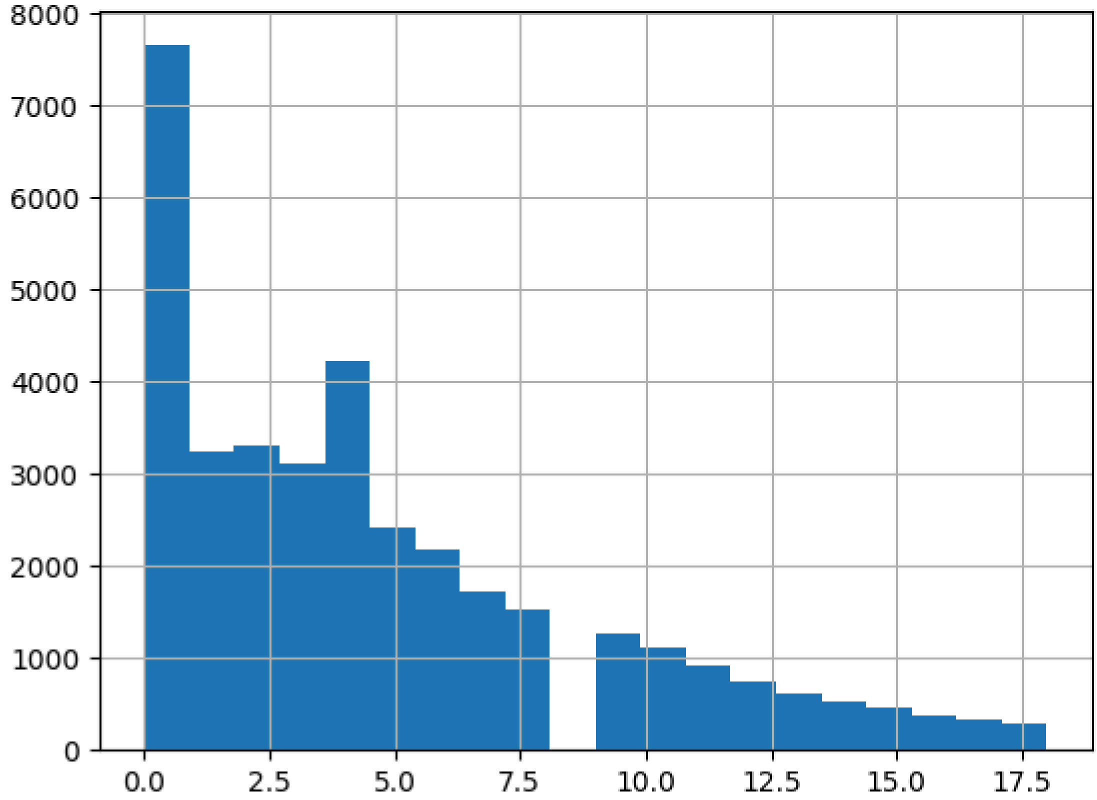
In [45]:
```
emp_exp_lower_bound, emp_exp_upper_bound = lower_upper(x_train['person_emp_exp
print(emp_exp_lower_bound)
print(emp_exp_upper_bound)
```

```
-9.5
18.5
```

Impute outlier pada semua dataset dengan median dari training dataset

In [46]:
```
x_train['person_emp_exp'] = impute_outlier(x_train['person_emp_exp'], emp_exp_
x_test['person_emp_exp'] = impute_outlier(x_test['person_emp_exp'], emp_exp_lo
```

In [47]:
```
x_train['person_emp_exp'].hist(bins=20)
plt.show()
```

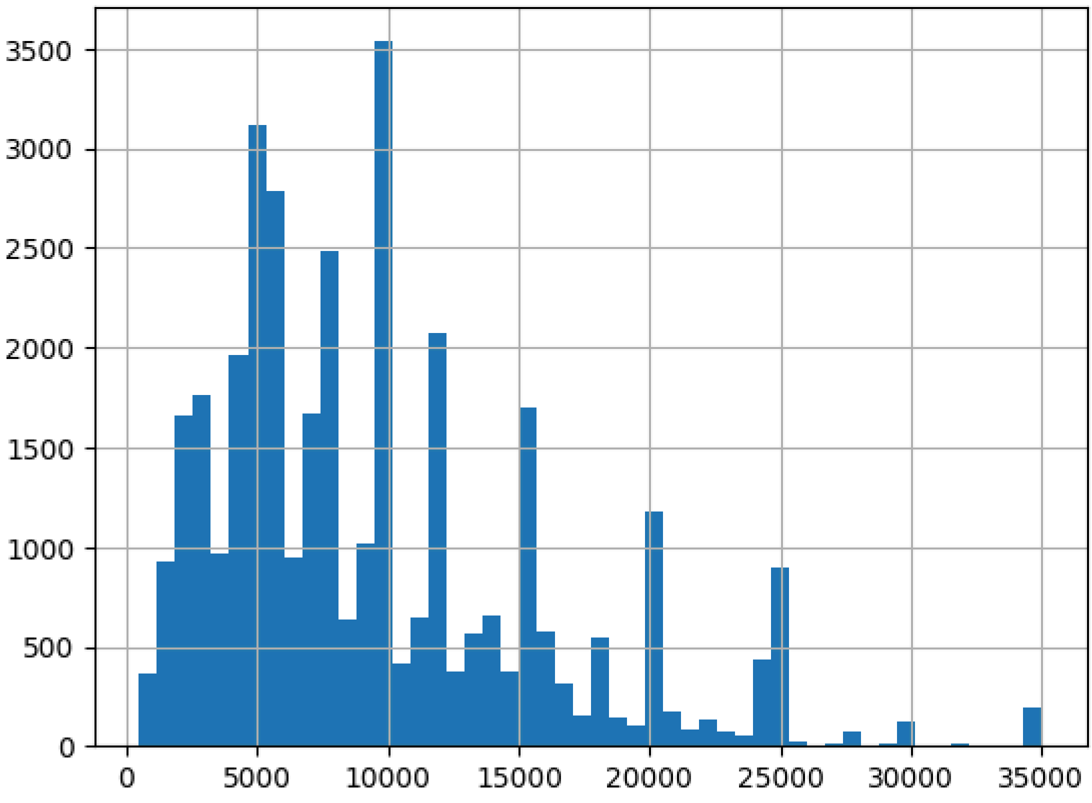


loan_amnt

In [48]:
```
amnt_outlier = detect_outliers_iqr(x_train['loan_amnt'])
print("Outliers di column 'loan_amnt:", amnt_outlier)
print("Jumlah outliers di column 'loan_amnt:", len(amnt_outlier))
```

```
Outliers di column 'loan_amnt: [23200.0, 23200.0, 23200.0, 23225.0, 2332
5.0, 23325.0, 23400.0, 23450.0, 23450.0, 23450.0, 23462.0, 23487.0, 2350
0.0, 23500.0, 23500.0, 23500.0, 23500.0, 23500.0, 23500.0, 23500.0, 2350
0.0, 23500.0, 23500.0, 23525.0, 23558.0, 23575.0, 23600.0, 23600.0, 2360
0.0, 23638.0, 23663.0, 23666.0, 23687.0, 23700.0, 23700.0, 23737.0, 2373
7.0, 23748.0, 23750.0, 23774.0, 23776.0, 23786.0, 23800.0, 23800.0, 2380
0.0, 23800.0, 23802.0, 23821.0, 23839.0, 23857.0, 23873.0, 23890.0, 2391
5.0, 23925.0, 23928.0, 23929.0, 23942.0, 23949.0, 23953.0, 23963.0, 2397
0.0, 23975.0, 23975.0, 23982.0, 23987.0, 23989.0, 23990.0, 23990.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
0.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 24000.0, 2400
```

In [49]:
```
x_train['loan_amnt'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [50]:
```
amnt_median = x_train['loan_amnt'].median()
amnt_median
```

Out[50]:
```
8000.0
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

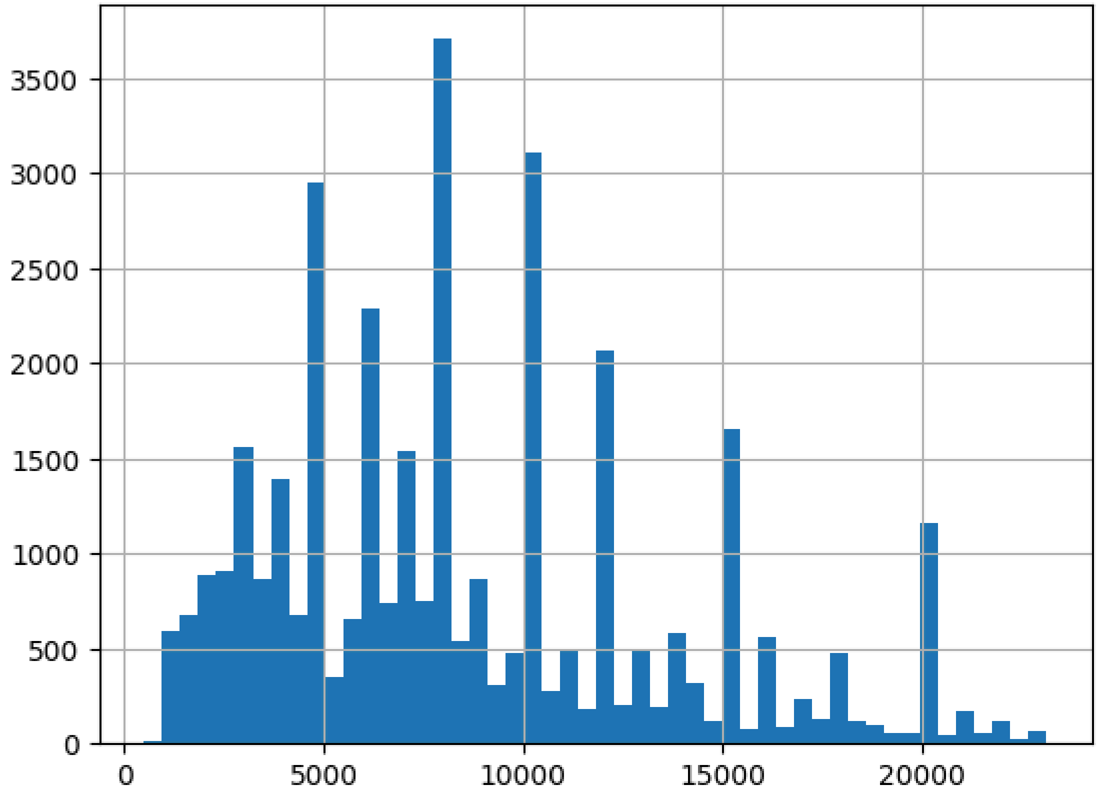
In [51]:
```
amnt_lower_bound, amnt_upper_bound = lower_upper(x_train['loan_amnt'])
print(amnt_lower_bound)
print(amnt_upper_bound)
```

```
-5875.0
23125.0
```

Impute outlier pada semua dataset dengan median dari training dataset

In [52]:
```
x_train['loan_amnt'] = impute_outlier(x_train['loan_amnt'], amnt_lower_bound,
x_test['loan_amnt'] = impute_outlier(x_test['loan_amnt'], amnt_lower_bound, am
```

In [53]:
```
x_train['loan_amnt'].hist(bins=50)
plt.show()
```

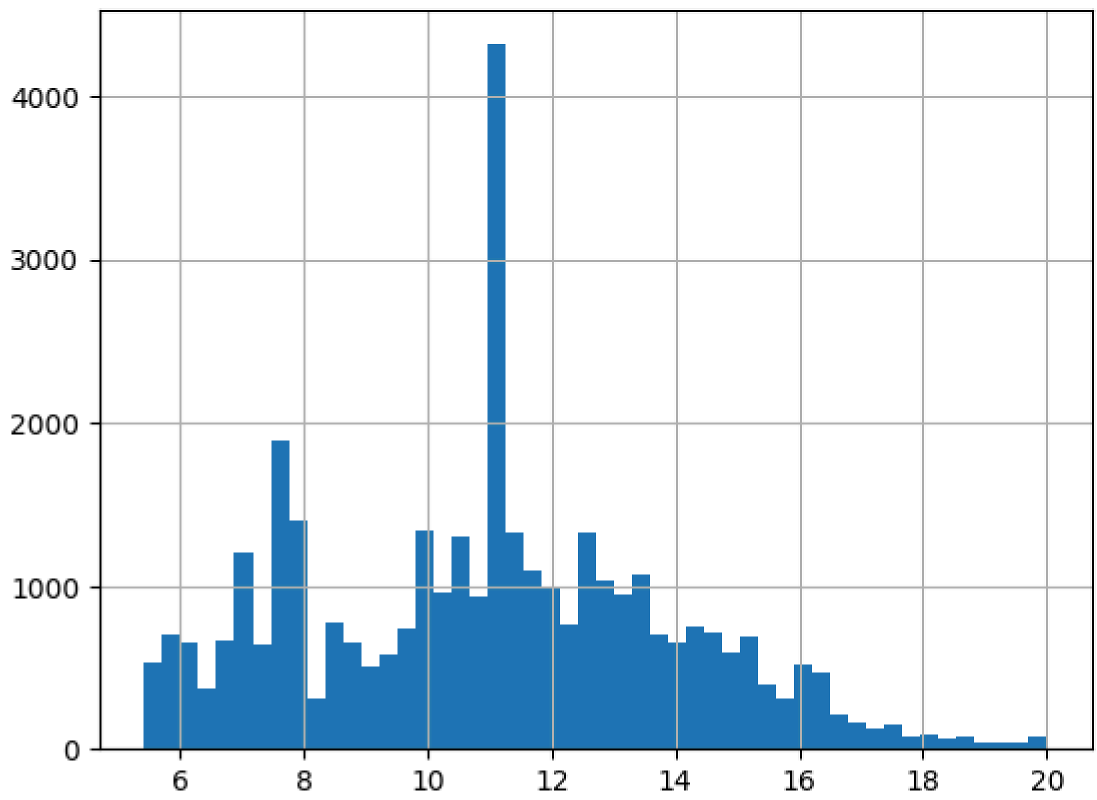


loan_int_rate

In [54]:
```
int_rate_outlier = detect_outliers_iqr(x_train['loan_int_rate'])
print("Outliers di column 'loan_int_rate:", int_rate_outlier)
print("Jumlah outliers di column 'loan_int_rate:", len(int_rate_outlier))
```

```
Outliers di column 'loan_int_rate: [19.62, 19.69, 19.69, 19.69, 19.69, 19.6
9, 19.69, 19.69, 19.69, 19.69, 19.69, 19.69, 19.74, 19.74, 19.74, 19.79, 19.
79, 19.8, 19.82, 19.82, 19.82, 19.82, 19.91, 19.91, 19.91, 19.91, 19.91, 19.
91, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0,
20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.
0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 2
0.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0,
20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.
0, 20.0, 20.0, 20.0]
Jumlah outliers di column 'loan_int_rate: 94
```

In [55]:
```
x_train['loan_int_rate'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [56]:
```
int_rate_median = x_train['loan_int_rate'].median()
int_rate_median
```

Out[56]: 11.01

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

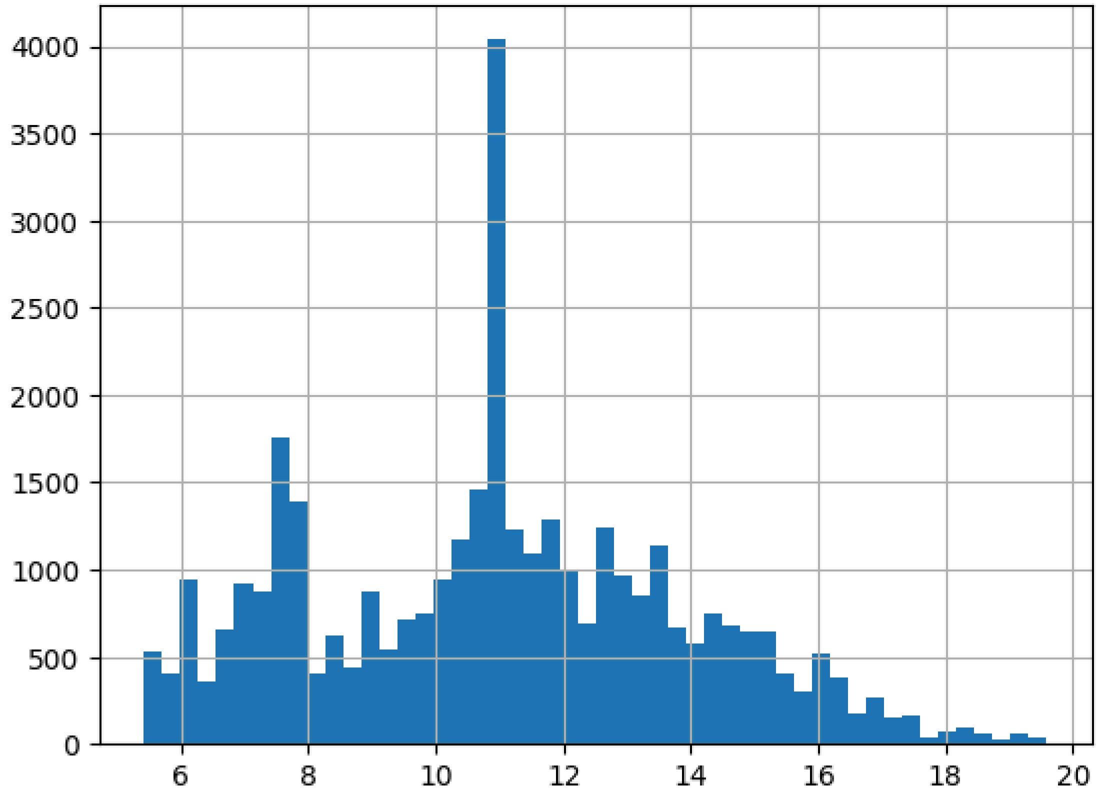
In [57]:
```
int_rate_lower_bound, int_rate_upper_bound = lower_upper(x_train['loan_int_rat
print(int_rate_lower_bound)
print(int_rate_upper_bound)
```

```
1.9899999999999993
19.59
```

Impute outlier pada semua dataset dengan median dari training dataset

In [58]:
```
x_train['loan_int_rate'] = impute_outlier(x_train['loan_int_rate'], int_rate_l
x_test['loan_int_rate'] = impute_outlier(x_test['loan_int_rate'], int_rate_low
```

In [59]:
```
x_train['loan_int_rate'].hist(bins=50)
plt.show()
```

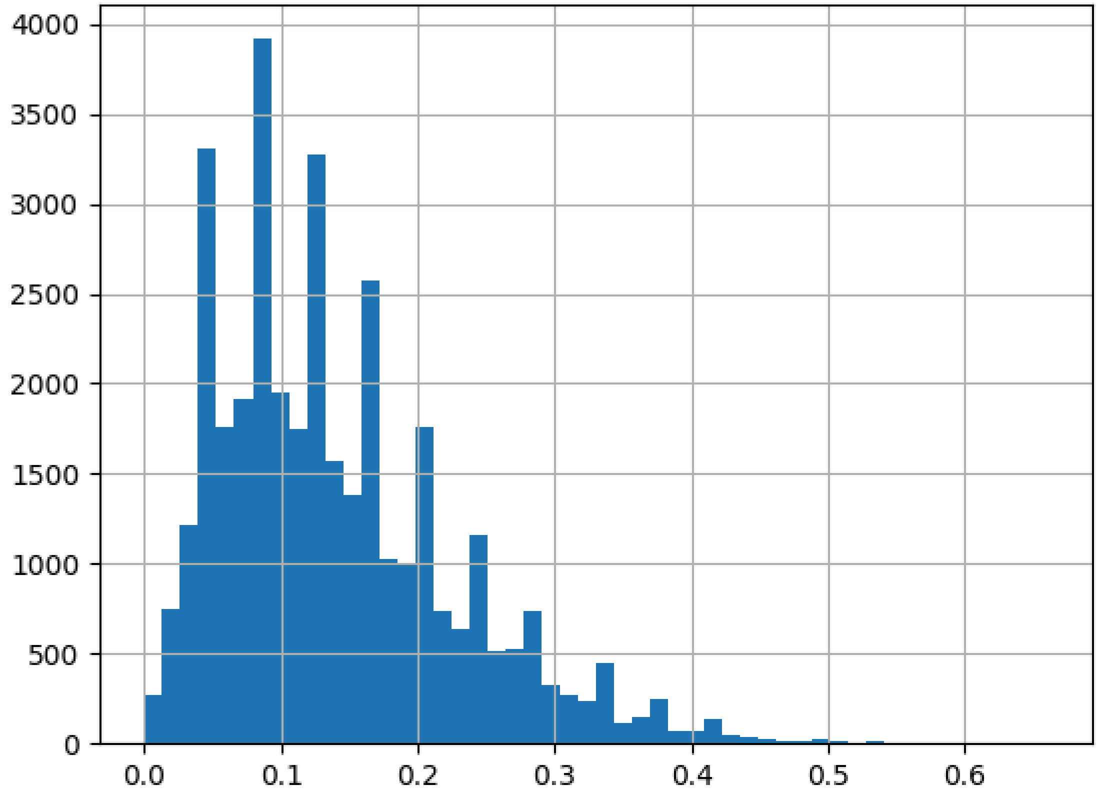


loan_percent_income

In [60]:
```
loan_percent_outlier = detect_outliers_iqr(x_train['loan_percent_income'])
print("Outliers di column 'loan_percent_income:", loan_percent_outlier)
print("Jumlah outliers di column 'loan_percent_income:", len(loan_percent_outl
```

```
Outliers di column 'loan_percent_income: [0.38, 0.38, 0.38, 0.38, 0.38, 0.3
8, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38,
0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.3
8, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38,
0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.3
8, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38,
0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.3
8, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38,
0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.3
8, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38,
0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.38, 0.39, 0.39, 0.39, 0.3
9, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39,
0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.3
9, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39,
0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.3
9, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39,
0.39, 0.39, 0.39, 0.39, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.
4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.
4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.
4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.
4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.41, 0.41, 0.41, 0.41,
0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.4
1, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41,
0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.4
1, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41,
0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.4
1, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41,
0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.41, 0.42, 0.42, 0.42, 0.42, 0.4
2, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42,
0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.4
2, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42,
0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.42, 0.4
3, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43,
0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.4
3, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43,
0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.43, 0.4
3, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44,
0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.4
4, 0.44, 0.44, 0.44, 0.44, 0.44, 0.44, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45,
0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.45, 0.46, 0.46, 0.4
6, 0.46, 0.46, 0.46, 0.46, 0.46, 0.46, 0.46, 0.46, 0.46, 0.47, 0.47, 0.47,
0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.47, 0.4
7, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48,
0.48, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.49, 0.4
9, 0.49, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.51, 0.51, 0.51, 0.51, 0.51, 0.
51, 0.51, 0.51, 0.51, 0.52, 0.52, 0.52, 0.52, 0.52, 0.52, 0.53, 0.53, 0.53,
0.53, 0.53, 0.53, 0.53, 0.54, 0.54, 0.54, 0.54, 0.54, 0.54, 0.55, 0.55, 0.5
5, 0.55, 0.56, 0.56, 0.56, 0.56, 0.57, 0.58, 0.61, 0.61, 0.62, 0.62, 0.63,
0.66]
Jumlah outliers di column 'loan_percent_income: 607
```

In [61]:
```
x_train['loan_percent_income'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [62]:
```
loan_percent_median = x_train['loan_percent_income'].median()
loan_percent_median
```

Out[62]:
```
0.12
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

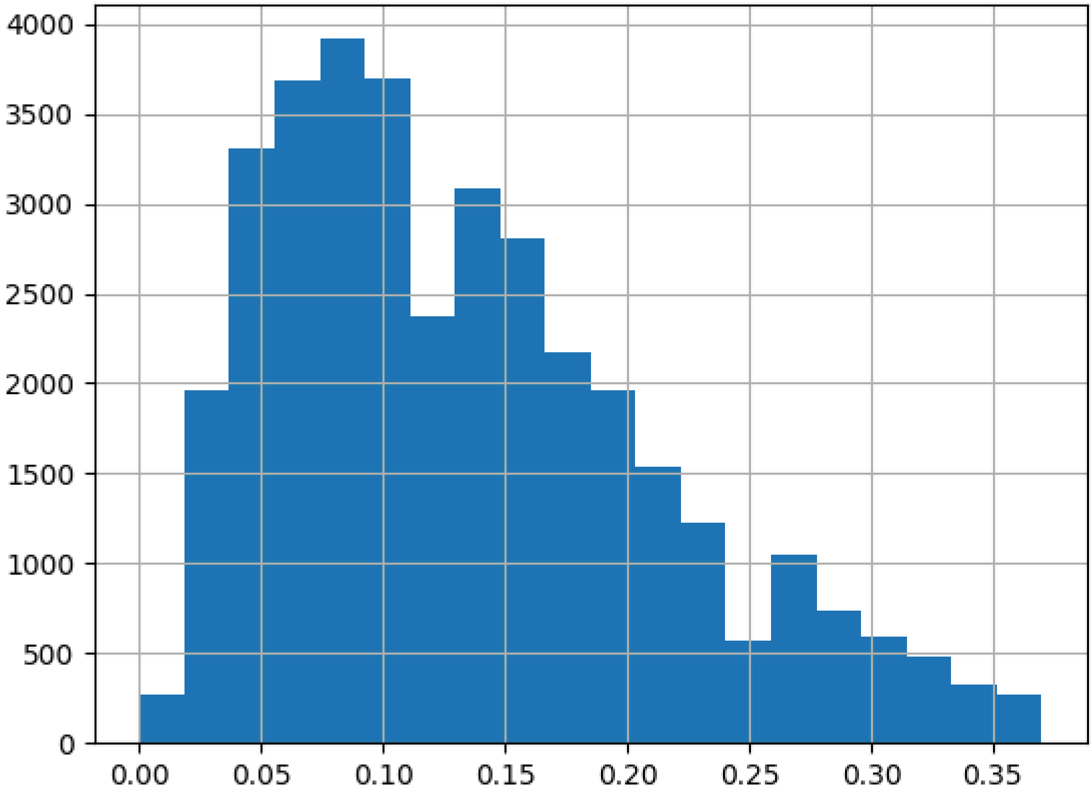
In [63]:
```
loan_percent_lower_bound, loan_percent_upper_bound = lower_upper(x_train['loar
print(loan_percent_lower_bound)
print(loan_percent_upper_bound)
```

```
-0.10999999999999999
0.37
```

Impute outlier pada semua dataset dengan median dari training dataset

In [64]:
```
x_train['loan_percent_income'] = impute_outlier(x_train['loan_percent_income']
x_test['loan_percent_income'] = impute_outlier(x_test['loan_percent_income'],
```

In [65]:
```
x_train['loan_percent_income'].hist(bins=20)
plt.show()
```



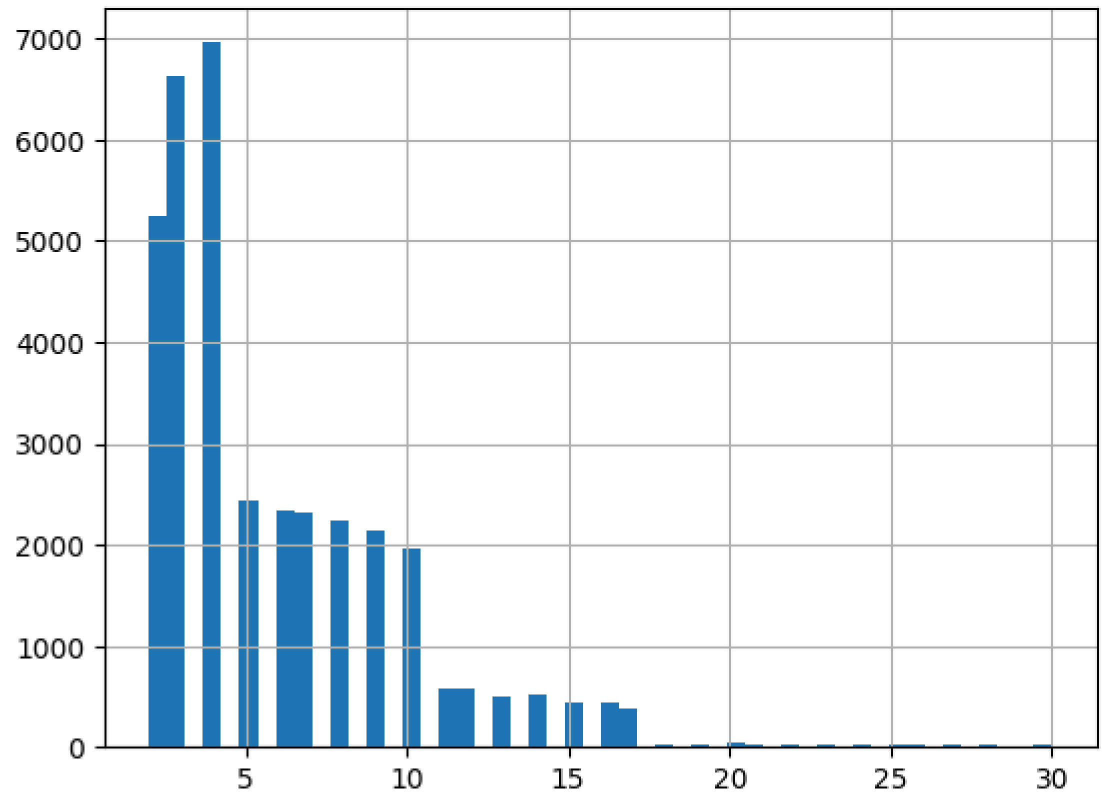
cb_person_cred_hist_length

In [66]:
```
cred_hist_outlier = detect_outliers_iqr(x_train['cb_person_cred_hist_length'])
print("Outliers di column 'cb_person_cred_hist_length:", cred_hist_outlier)
print("Jumlah outliers di column 'cb_person_cred_hist_length:", len(cred_hist_
```

```
Outliers di column 'cb_person_cred_hist_length: [16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0,
16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.
0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 1
6.0, 16.0, 16.0, 16.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
```

```
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0,
17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.
0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 18.0, 1
8.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0,
18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.
0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 1
9.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0, 19.0,
19.0, 19.0, 19.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.
0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 2
0.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0, 20.0,
20.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.
0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 2
2.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0,
22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 22.0, 23.0, 23.
0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 2
3.0, 23.0, 23.0, 23.0, 23.0, 23.0, 23.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0,
24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.
0, 24.0, 24.0, 24.0, 24.0, 24.0, 24.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 2
5.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0, 25.0,
26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.0, 26.
0, 26.0, 26.0, 26.0, 26.0, 27.0, 27.0, 27.0, 27.0, 27.0, 27.0, 27.0, 27.0, 2
7.0, 27.0, 27.0, 27.0, 27.0, 27.0, 27.0, 27.0, 28.0, 28.0, 28.0, 28.0, 28.0,
28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.0, 28.
0, 28.0, 28.0, 28.0, 28.0, 29.0, 29.0, 29.0, 29.0, 29.0, 29.0, 29.0, 29.0, 2
9.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0,
30.0, 30.0, 30.0, 30.0, 30.0, 30.0, 30.0]
Jumlah outliers di column 'cb_person_cred_hist_length: 1101
```

In [67]:
```python
x_train['cb_person_cred_hist_length'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [68]:
```
cred_hist_median = x_train['cb_person_cred_hist_length'].median()
cred_hist_median
```

Out[68]:
```
4.0
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

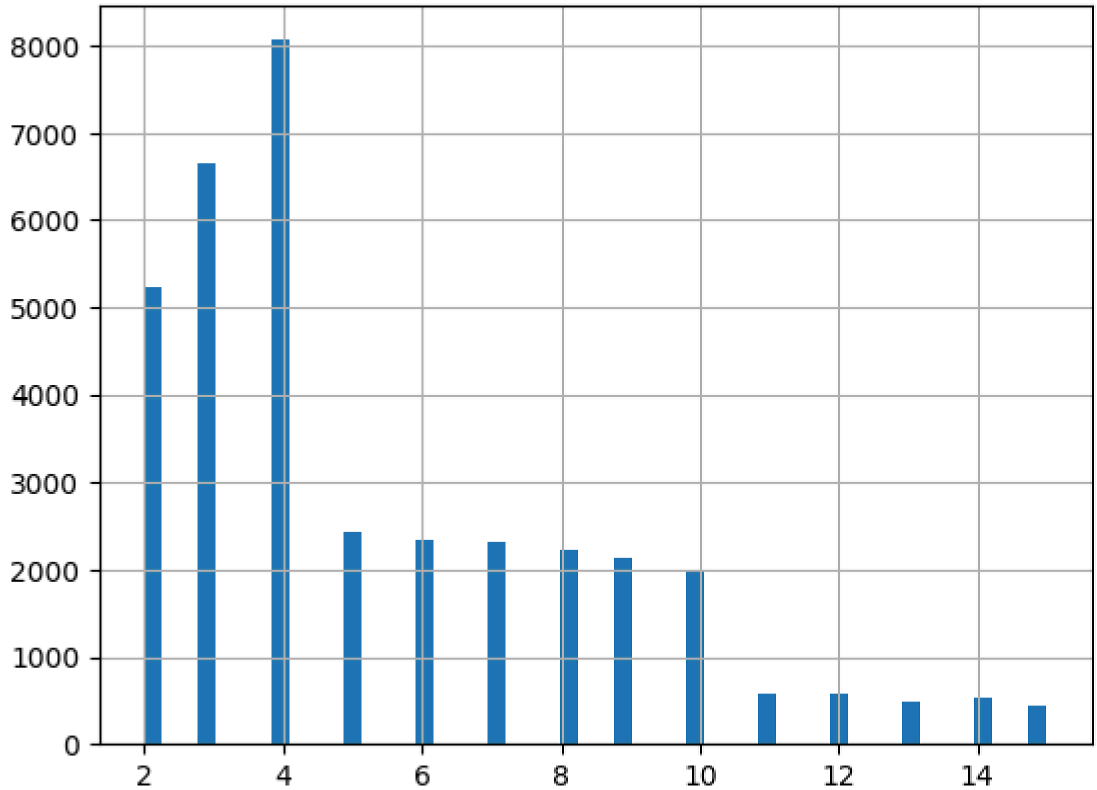
In [69]:
```
cred_hist_lower_bound, cred_hist_upper_bound = lower_upper(x_train['cb_person_
print(cred_hist_lower_bound)
print(cred_hist_upper_bound)
```

```
-4.5
15.5
```

Impute outlier pada semua dataset dengan median dari training dataset

In [70]:
```
x_train['cb_person_cred_hist_length'] = impute_outlier(x_train['cb_person_cred
x_test['cb_person_cred_hist_length'] = impute_outlier(x_test['cb_person_cred_h
```

In [71]:
```
x_train['cb_person_cred_hist_length'].hist(bins=50)
plt.show()
```

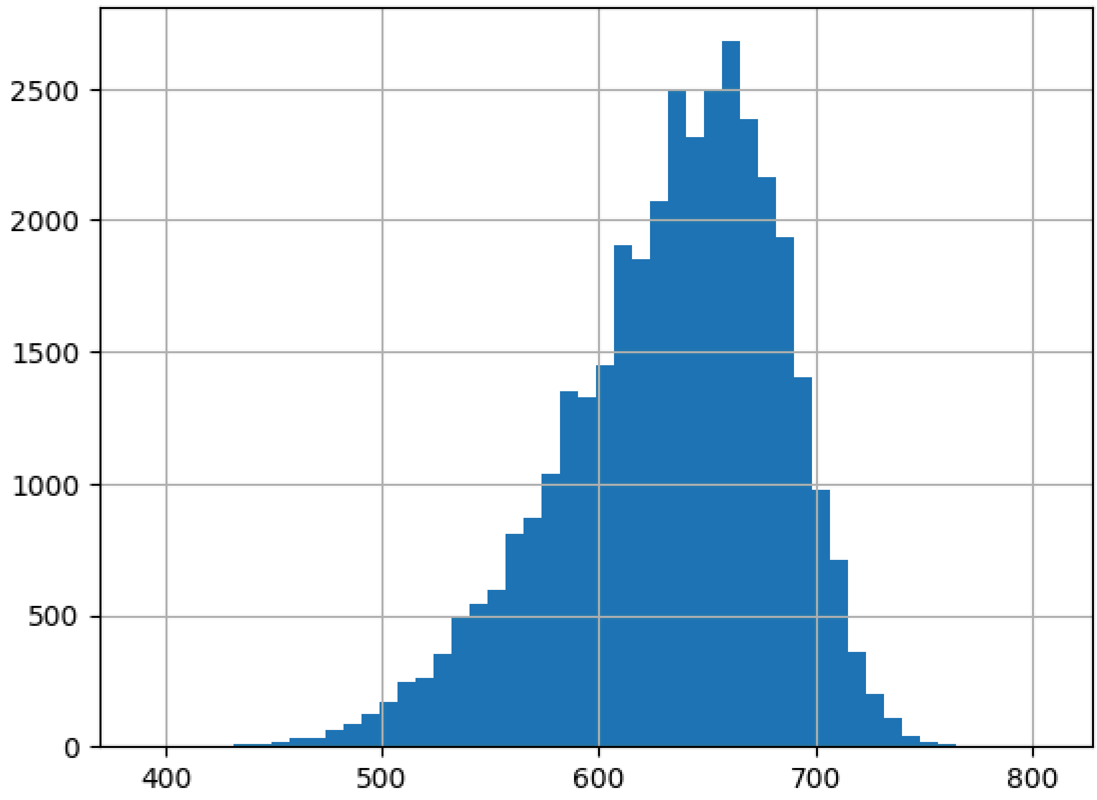


credit_score

In [72]:
```
credit_score_outlier = detect_outliers_iqr(x_train['credit_score'])
print("Outliers di column 'credit_score:", credit_score_outlier)
print("Jumlah outliers di column 'credit_score:", len(credit_score_outlier))
```

```
Outliers di column 'credit_score: [390, 419, 430, 431, 434, 435, 435, 435, 4
37, 439, 439, 440, 441, 441, 441, 444, 444, 444, 445, 446, 448, 448, 449, 44
9, 450, 450, 450, 451, 453, 453, 453, 454, 454, 454, 455, 455, 455, 455, 45
6, 456, 456, 457, 458, 458, 458, 458, 458, 459, 459, 459, 460, 460, 460, 46
0, 460, 460, 460, 460, 461, 461, 461, 461, 461, 461, 462, 462, 462, 463, 46
3, 463, 464, 464, 465, 465, 465, 465, 466, 466, 467, 467, 467, 467, 468, 46
8, 468, 469, 469, 469, 469, 469, 469, 470, 470, 470, 470, 470, 471, 471, 47
1, 471, 472, 472, 472, 472, 472, 472, 472, 472, 473, 473, 473, 474, 474, 47
5, 475, 475, 475, 475, 475, 475, 475, 476, 476, 476, 476, 476, 476, 476, 47
6, 477, 477, 477, 477, 477, 477, 477, 477, 477, 477, 477, 477, 477, 477, 47
7, 477, 477, 478, 478, 478, 478, 478, 478, 479, 479, 479, 479, 479, 479, 47
9, 479, 480, 480, 480, 480, 480, 480, 480, 480, 480, 481, 481, 481, 481, 48
1, 481, 482, 482, 482, 482, 482, 483, 483, 483, 483, 483, 483, 483, 483, 48
4, 484, 484, 484, 484, 484, 484, 484, 484, 484, 484, 484, 485, 485, 485, 48
5, 486, 486, 486, 486, 486, 486, 486, 486, 486, 486, 487, 487, 487, 487, 48
7, 487, 487, 487, 487, 488, 488, 488, 488, 488, 488, 488, 488, 488, 488, 48
8, 489, 489, 489, 489, 489, 489, 489, 489, 489, 489, 489, 489, 489, 489, 48
9, 490, 490, 490, 490, 490, 490, 490, 490, 490, 490, 490, 491, 491, 491, 49
1, 491, 491, 491, 491, 491, 491, 491, 491, 491, 492, 492, 492, 492, 492, 49
2, 492, 492, 492, 492, 492, 492, 492, 492, 492, 493, 493, 493, 493, 493, 49
3, 493, 493, 493, 493, 493, 494, 494, 494, 494, 494, 494, 494, 494, 494, 49
4, 494, 494, 494, 494, 494, 494, 494, 494, 494, 494, 495, 495, 495, 495, 49
5, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 495, 496, 496, 496, 49
6, 496, 496, 496, 496, 496, 496, 496, 496, 496, 496, 496, 497, 497, 497, 49
7, 497, 497, 497, 497, 497, 497, 497, 498, 498, 498, 498, 498, 498, 498, 49
8, 498, 498, 498, 498, 498, 498, 498, 498, 498, 498, 498, 498, 498, 498, 49
9, 499, 499, 499, 499, 499, 499, 499, 499, 499, 499, 499, 499, 499, 499, 78
4, 784, 789, 792, 805, 807]
Jumlah outliers di column 'credit_score: 404
```

In [73]:
```
x_train['credit_score'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

In [74]:
```
credit_score_median = x_train['credit_score'].median()
credit_score_median
```

Out[74]:
```
640.0
```

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

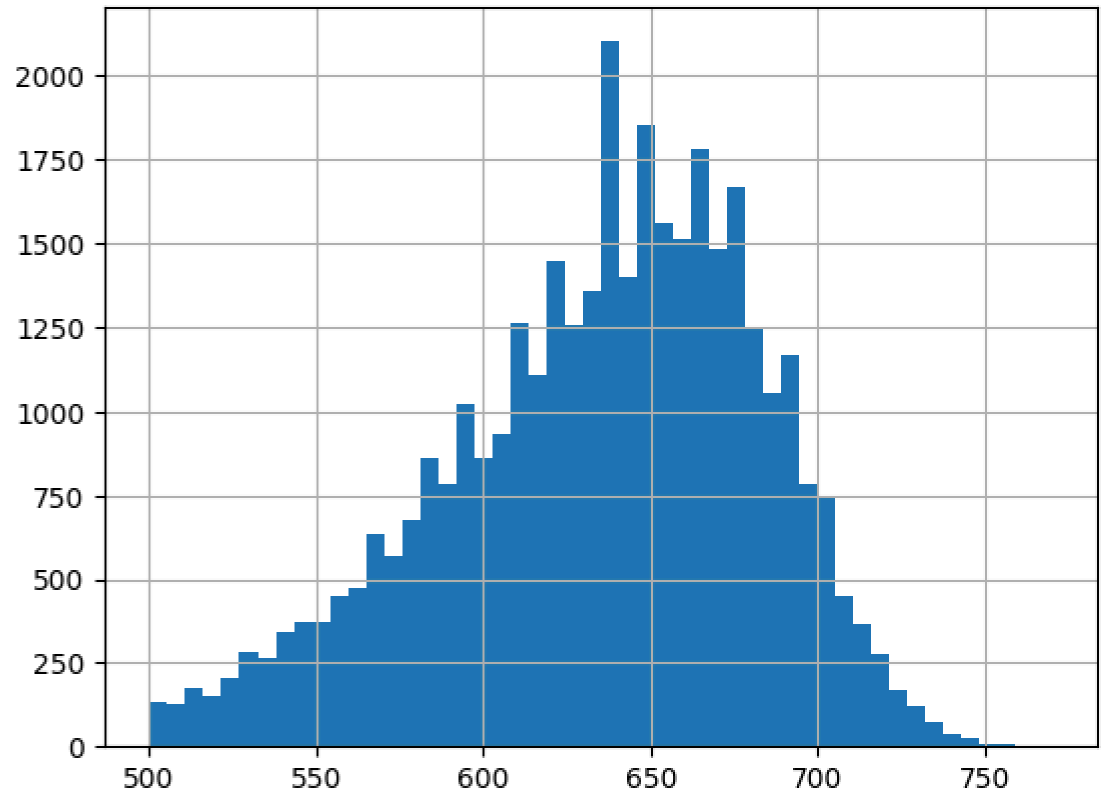
In [75]:
```
credit_score_lower_bound, credit_score_upper_bound = lower_upper(x_train['cred
print(credit_score_lower_bound)
print(credit_score_upper_bound)
```

```
500.0
772.0
```

Impute outlier pada semua dataset dengan median dari training dataset

In [76]:
```
x_train['credit_score'] = impute_outlier(x_train['credit_score'], credit_score
x_test['credit_score'] = impute_outlier(x_test['credit_score'], credit_score_l
```

In [77]:
```
x_train['credit_score'].hist(bins=50)
plt.show()
```



Encoding

In [78]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 36000 entries, 42205 to 39200
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   person_age                      36000 non-null  float64
 1   person_gender                   36000 non-null  object 
 2   person_education                36000 non-null  object 
 3   person_income                   36000 non-null  float64
 4   person_emp_exp                  36000 non-null  int64  
 5   person_home_ownership           36000 non-null  object 
 6   loan_amnt                       36000 non-null  float64
 7   loan_intent                     36000 non-null  object 
 8   loan_int_rate                   36000 non-null  float64
 9   loan_percent_income             36000 non-null  float64
 10  cb_person_cred_hist_length      36000 non-null  float64
 11  credit_score                    36000 non-null  int64  
 12  previous_loan_defaults_on_file  36000 non-null  object 
dtypes: float64(6), int64(2), object(5)
memory usage: 3.8+ MB
```

Lakukan encoding pada variabel yang masih memiliki dtype object

In [79]:
```
from sklearn.preprocessing import LabelEncoder
le_gender = LabelEncoder()
le_edu = LabelEncoder()
le_hownership = LabelEncoder()
le_intent = LabelEncoder()
le_prev_loan = LabelEncoder()

x_train['person_gender'] = le_gender.fit_transform(x_train['person_gender'])
x_train['person_education'] = le_edu.fit_transform(x_train['person_education']
x_train['person_home_ownership'] = le_hownership.fit_transform(x_train['persor
x_train['loan_intent'] = le_intent.fit_transform(x_train['loan_intent'])
x_train['previous_loan_defaults_on_file'] = le_prev_loan.fit_transform(x_trair

x_test['person_gender'] = le_gender.fit_transform(x_test['person_gender'])
x_test['person_education'] = le_edu.fit_transform(x_test['person_education'])
x_test['person_home_ownership'] = le_hownership.fit_transform(x_test['person_h
x_test['loan_intent'] = le_intent.fit_transform(x_test['loan_intent'])
x_test['previous_loan_defaults_on_file'] = le_prev_loan.fit_transform(x_test['

# lihat kembali hasilnya
x_train.head(5)
```

Out[79]:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_
42205	25.0	1	3	67002.0	1	
42935	23.0	0	3	71677.0	0	
10748	24.0	1	0	48829.0	2	
30883	36.0	1	0	42940.0	15	
33233	30.0	1	1	95344.0	8	

Scaling

Scaling tidak perlu dilakukan karena model yang ingin dibuat itu random forest dan xgboosting dan keduanya tidak butuh scaled data. Scale data yang berbeda tidak berpengaruh terhadap efisiensi model.

Random Forest

In [80]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

parameters = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'min_samples_split': [5,8,10],
    'max_depth': [3,5,7]
}
```

In [81]:
```
Random_forest_class= GridSearchCV(RandomForestClassifier(), param_grid = paran
```

In [82]:
```
Random_forest_class.fit(x_train,y_train)
print("Tuned Hyperparameters :", Random_forest_class.best_params_)
print("Accuracy score :",Random_forest_class.best_score_)
```

```
Tuned Hyperparameters : {'criterion': 'entropy', 'max_depth': 7, 'min_sample
s_split': 10}
Accuracy score : 0.9116944444444444
```

In [83]:
```
Random_forest_class_best = RandomForestClassifier(criterion = 'entropy', max_c
Random_forest_class_best.fit(x_train, y_train)
```

Out[83]:
```
RandomForestClassifier(criterion='entropy', max_depth=7, min_samples_split=
5,
                       random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [84]:
```
y_predict_rf = Random_forest_class_best.predict(x_test)

from sklearn.metrics import classification_report

print('\nClassification Report Random Forest Model\n')
print(classification_report(y_test, y_predict_rf))
```

```

Classification Report Random Forest Model

              precision    recall  f1-score   support

           0       0.92      0.98      0.95      6995
           1       0.91      0.69      0.78      2005

    accuracy                           0.91      9000
   macro avg       0.91      0.83      0.86      9000
weighted avg       0.91      0.91      0.91      9000
```

XGBoost

In [85]:
```
import xgboost as xgb

# set hyperparameters untuk tuning
parameters = {
    'eta': [0.2, 0.3, 0.5],
    'gamma': [0, 0.1, 0.3, 0.5],
    'max_depth': [3,5,7]
}
```

In [86]:
```
xgb_class = xgb.XGBClassifier()
grid_xgb_class = GridSearchCV(xgb_class, param_grid = parameters, scoring= 'ac
```

In [87]:
```
grid_xgb_class.fit(x_train,y_train)
print("Tuned Hyperparameters :", grid_xgb_class.best_params_)
print("Accuracy score :",grid_xgb_class.best_score_)
```

```
Tuned Hyperparameters : {'eta': 0.2, 'gamma': 0.1, 'max_depth': 7}
Accuracy score : 0.9284444444444443
```

In [88]:
```
xgb_class_best = xgb.XGBClassifier(eta = 0.2, gamma = 0.1, max_depth = 7)
xgb_class_best.fit(x_train, y_train)
```

Out[88]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=Non
e,
              enable_categorical=False, eta=0.2, eval_metric=None,
              feature_types=None, gamma=0.1, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=7,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=N
one,
              n_jobs=None, num_parallel_tree=None, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [89]:
```
y_predict_xgb = xgb_class_best.predict(x_test)

print('\nClassification Report XG Boosting model\n')
print(classification_report(y_test, y_predict_xgb))
```

```
Classification Report XG Boosting model

              precision    recall  f1-score   support

           0       0.94      0.97      0.96      6995
           1       0.88      0.80      0.84      2005

    accuracy                           0.93      9000
   macro avg       0.91      0.88      0.90      9000
weighted avg       0.93      0.93      0.93      9000
```

Hasil weighted avg dari XG Boosting model 0.02 lebih baik dari weighted avg dari random forest model dari ketiga kriteria (precision, recall, dan f-1 score). Maka, model yang akan digunakan adalah XG Boosting model dengan parameter:
eta: 0.2,
gamma: 0.1,
max_depth: 7