

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]: `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]: `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
1000.0      1
```

Hal yang perlu diperbaiki:

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

```
In [8]: df['person_gender'] = df['person_gender'].replace({'Male':'male', 'fe male': 'fe male'})
df['person_gender'].value_counts()
```

```
Out[8]:
```

	count
person_gender	
male	24844
female	20156

dtype: int64

Dataset Splitting

```
In [9]: 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, random_state=42)
```

Missing Values

```
In [10]: df.isna().sum()
```

```
Out[10]:
```

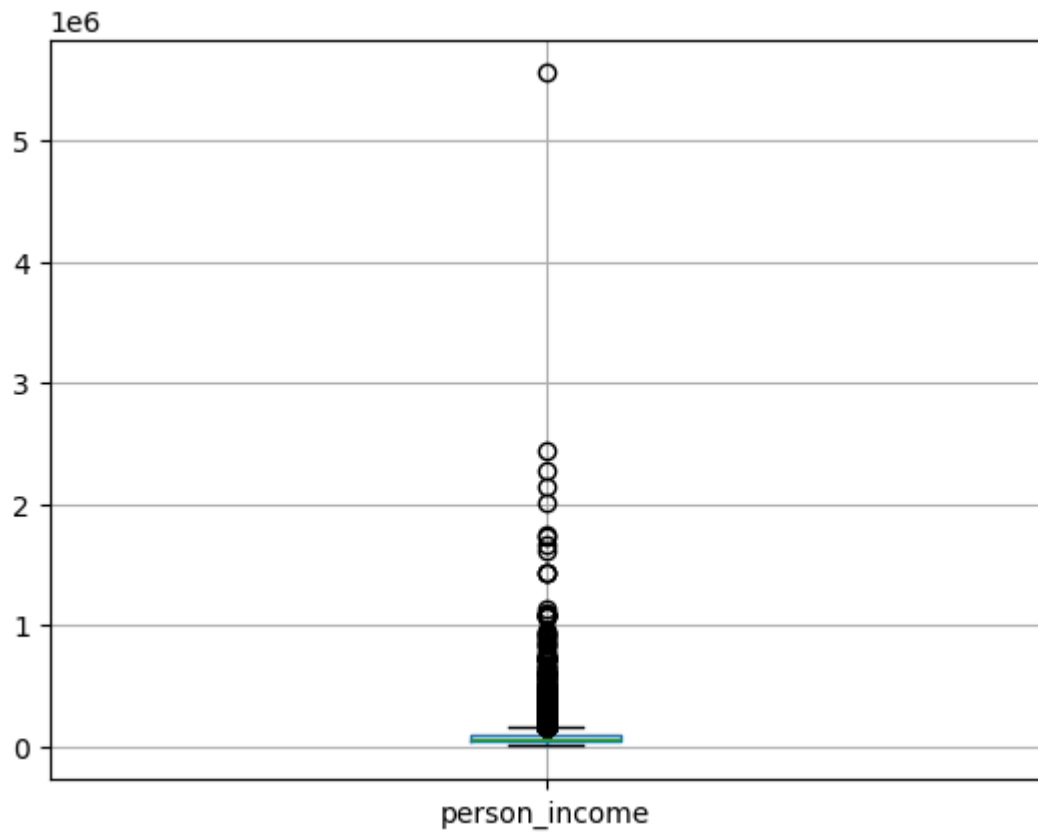
	0
person_age	0
person_gender	0
person_education	0
person_income	2250
person_emp_exp	0
person_home_ownership	0
loan_amnt	0
loan_intent	0
loan_int_rate	0
loan_percent_income	0
cb_person_cred_hist_length	0
credit_score	0
previous_loan_defaults_on_file	0
loan_status	0

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 [11]: x_train.boxplot('person_income')
```

```
Out[11]: <Axes: >
```



Buat function untuk mendeteksi outlier berdasarkan peraturan IQR

```
In [12]: outliers = []
def detect_outliers_iqr(data):
    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))
```

0828.0, 180796.0, 187978.0, 216043.0, 217059.0, 225400.0, 265119.0, 180875.0, 241296.0, 300951.0, 192853.0, 193082.0, 205048.0, 361012.0, 1741243.0, 725801.0, 180805.0, 192954.0, 214863.0, 2139143.0, 174957.0, 240921.0, 205019.0, 181065.0, 174103.0, 187107.0, 219900.0, 361101.0, 180907.0, 180753.0, 199016.0, 181597.0, 195394.0, 228588.0, 270665.0, 181175.0, 179235.0, 198890.0, 217017.0, 279664.0, 172215.0, 168844.0, 187973.0, 193046.0, 234964.0, 168746.0, 169240.0, 181052.0, 187011.0, 193036.0, 195789.0, 224427.0, 265048.0, 421619.0, 325368.0, 400383.0, 1076036.0, 180960.0, 280535.0, 216944.0, 176144.0, 217336.0, 180638.0, 180936.0, 204871.0, 240491.0, 181151.0, 217061.0, 220872.0, 181118.0, 360630.0, 238525.0, 179061.0, 240659.0, 247174.0, 264950.0, 390999.0, 420852.0, 211093.0, 270644.0, 169228.0, 175787.0, 181022.0, 181108.0, 204788.0, 274421.0, 348890.0, 193233.0, 173781.0, 246870.0, 169229.0, 180946.0, 192956.0, 252699.0, 258024.0, 180871.0, 201823.0, 301183.0, 172318.0, 181002.0, 184678.0, 190083.0, 211272.0, 222975.0, 170147.0, 234318.0, 259694.0, 241222.0, 288930.0, 319355.0, 331037.0, 252929.0, 212397.0, 184703.0, 190098.0, 192792.0, 169181.0, 181267.0, 187273.0, 262431.0, 168690.0, 233498.0, 312553.0, 403758.0, 210982.0, 228016.0, 241002.0, 516892.0, 172618.0, 174718.0, 376616.0, 222629.0, 369446.0, 190978.0, 320081.0, 432619.0, 300983.0, 169127.0, 205028.0, 329417.0, 180823.0, 185652.0, 241073.0, 241172.0,

```
In [13]: pd.set_option('display.float_format', '{:.0f}'.format)
x_train['person_income'].describe()
```

```
Out[13]:
```

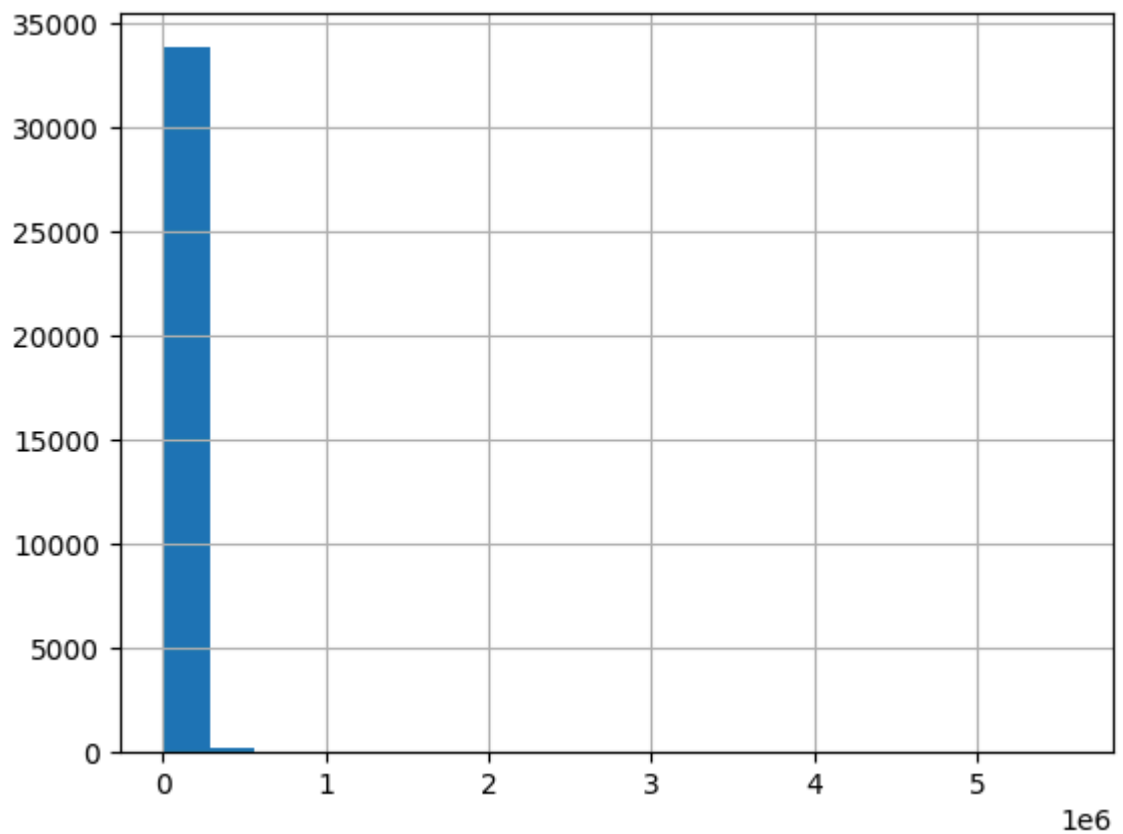
	person_income
count	34181
mean	80112
std	71186
min	8000
25%	47087
50%	67002
75%	95698
max	5556399

dtype: float64

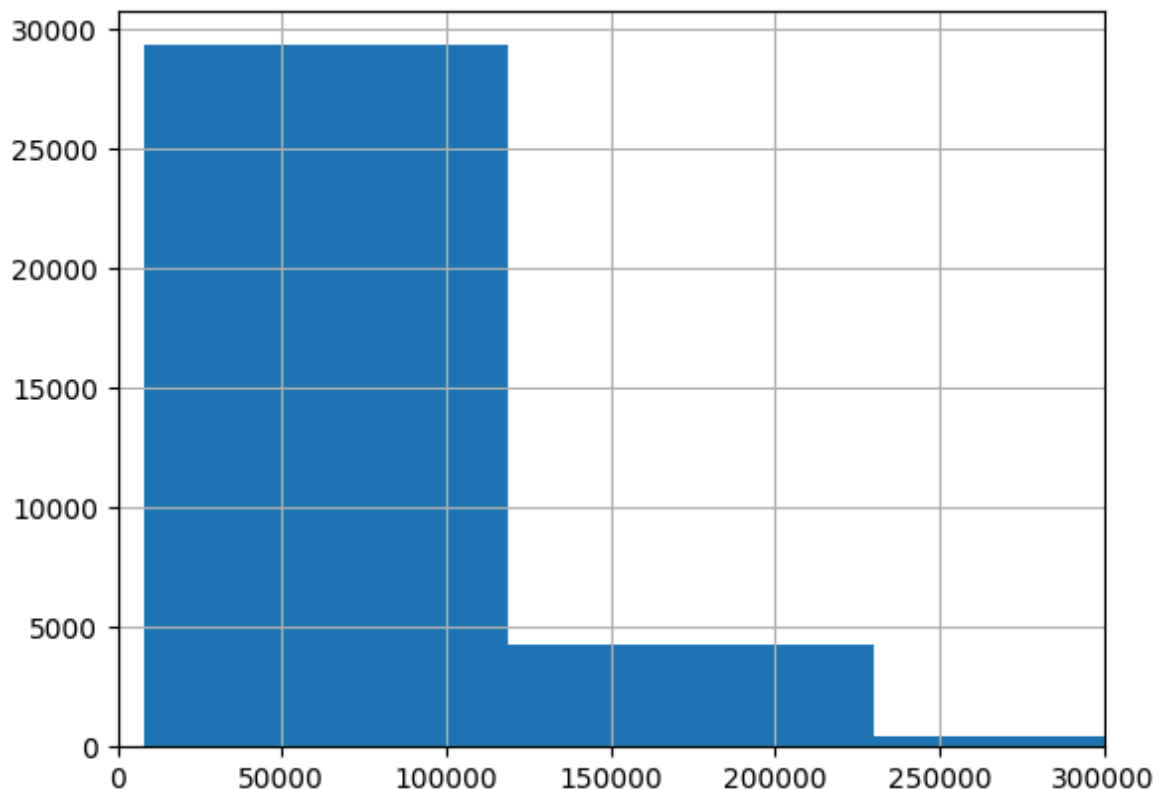
```
In [14]: print('Min outlier:', pd.DataFrame(income_outlier).min()[0])  
print('Max outlier:', pd.DataFrame(income_outlier).max()[0])  
print('Mean outlier:', pd.DataFrame(income_outlier).mean()[0])
```

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

```
In [15]: x_train['person_income'].hist(bins=20)  
plt.show()
```



```
In [16]: 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 [17]: income_median = x_train['person_income'].median()
income_median
```

Out[17]: 67002.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [18]: 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 [19]: 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 [20]: 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_lower_bound, income_upper_bound, income_median)
x_test['person_income'] = impute_outlier(x_test['person_income'], income_lower_bound, income_upper_bound, income_median)
```

```
In [21]: x_train['person_income'].describe()
```

```
Out[21]:
```

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

dtype: float64

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

```
Out[22]:
```

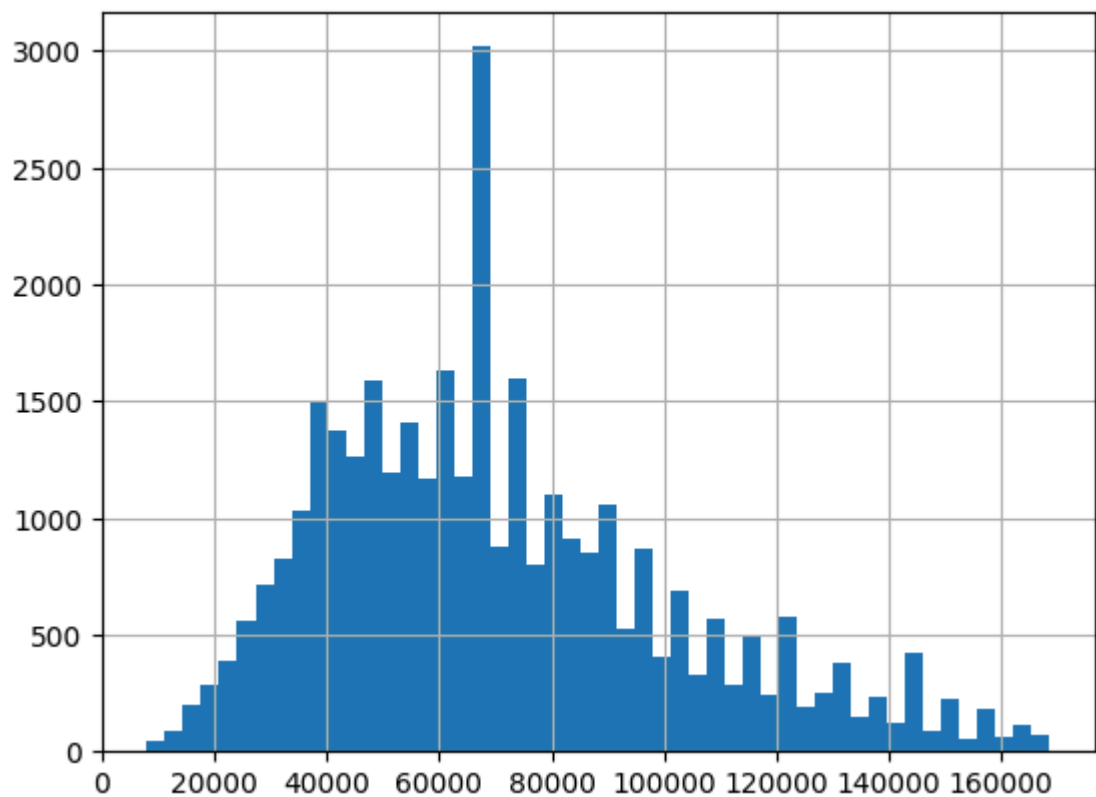
	person_income
count	8569
mean	71087
std	31926
min	8000
25%	47850
50%	67002
75%	89438
max	167935

dtype: float64

Missing value handling untuk column 'person_income'. In

```
In [23]: x_train['person_income'].hist(bins=50)
```

Out[23]: <Axes: >



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

```
In [24]: 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 [25]: x_train.isna().sum()
```

```
Out[25]:
```

	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 [26]: x_test.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]: y_train.isna().sum()
```

```
Out[27]: np.int64(0)
```

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

```
Out[28]: 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.

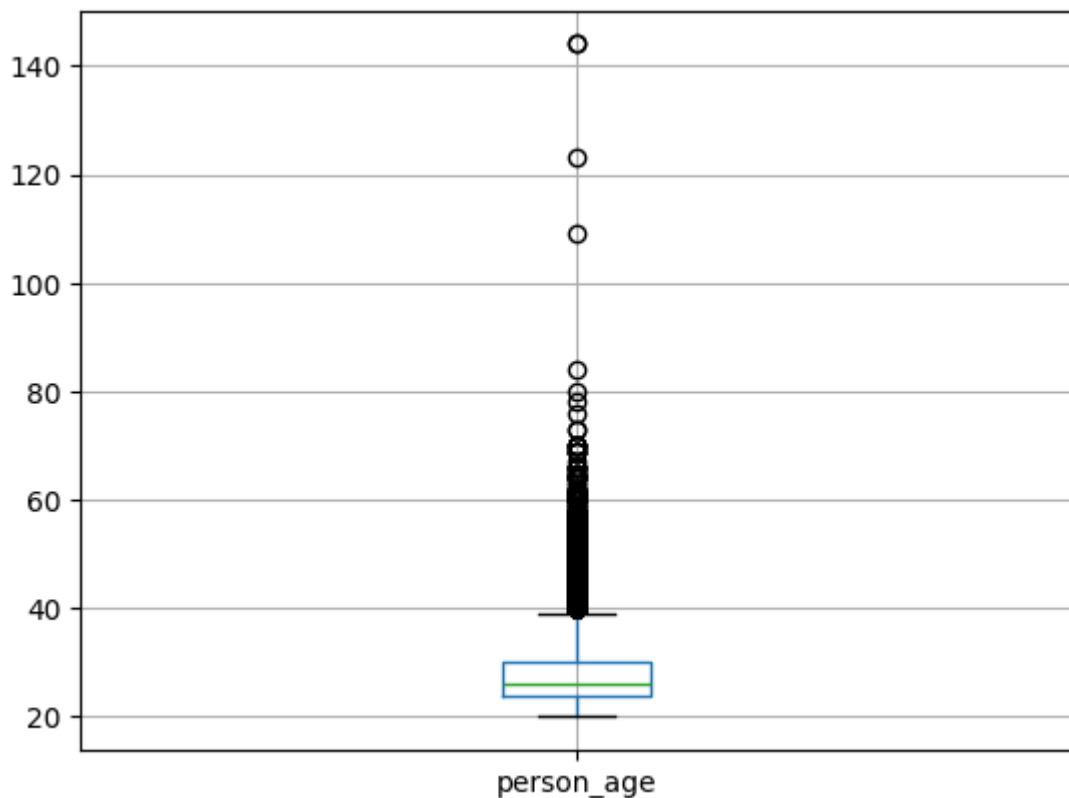
List numerical variabel:

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

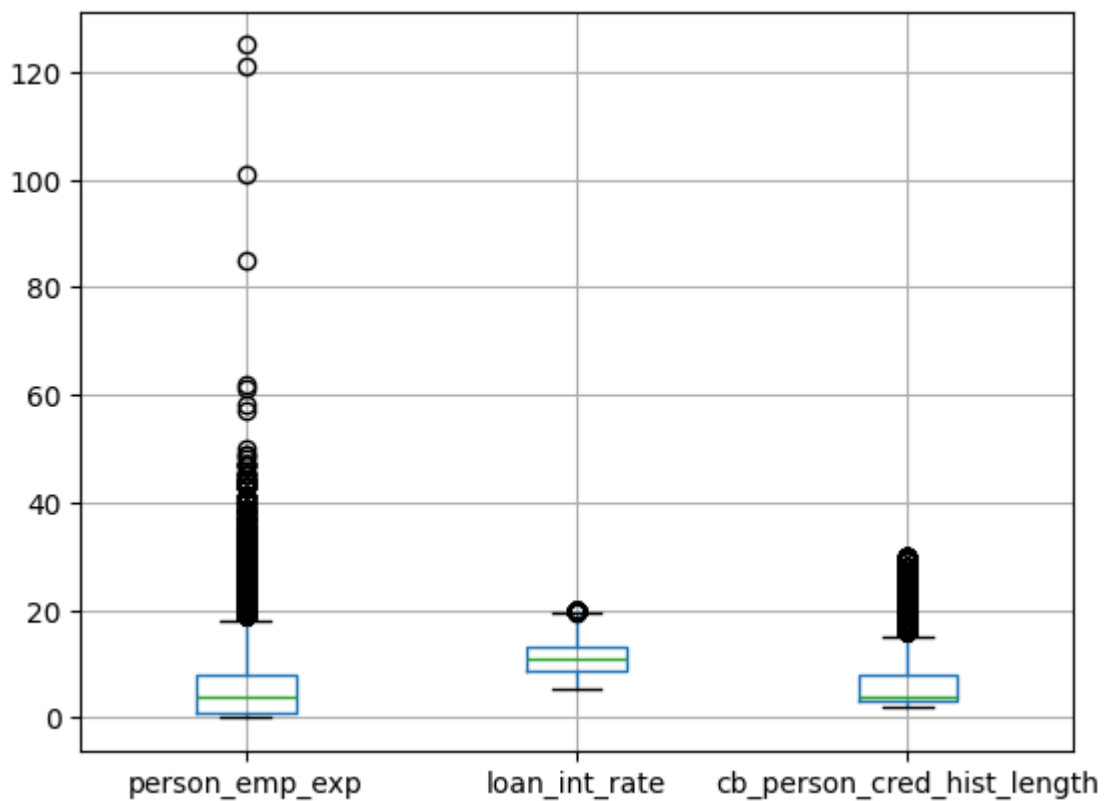
```
In [30]: x_train.boxplot('person_age')
```

```
Out[30]: <Axes: >
```



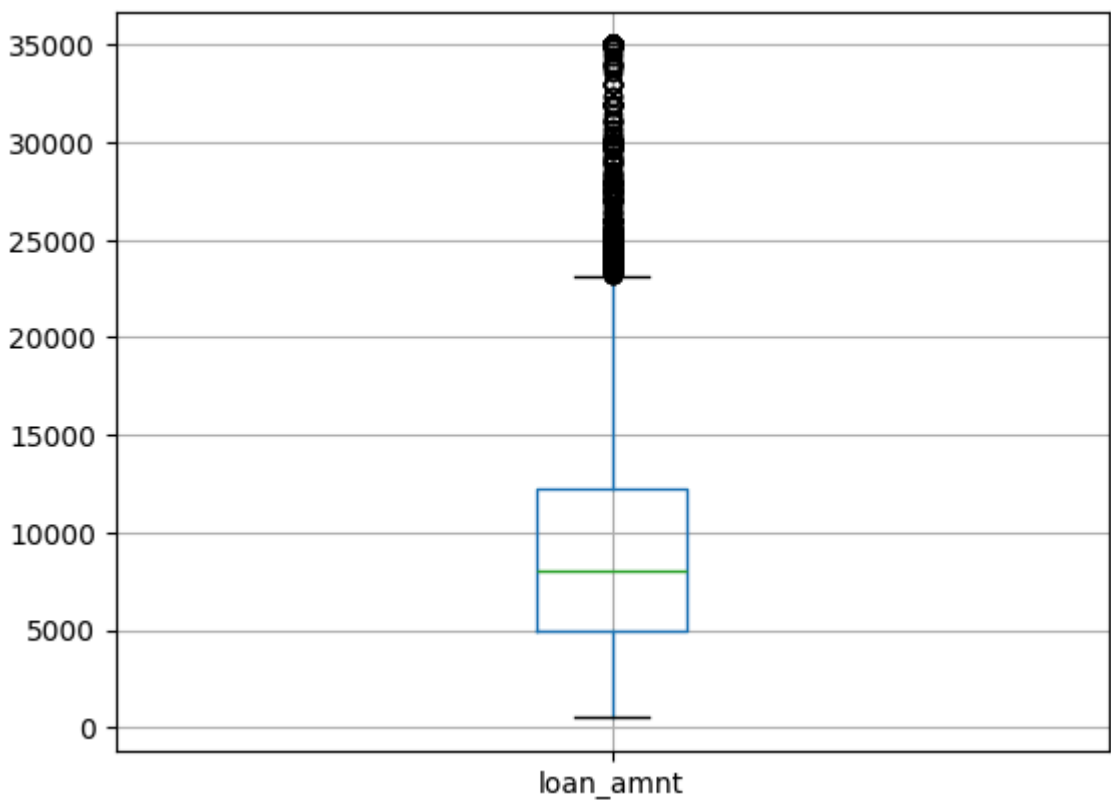
```
In [31]: x_train.boxplot(['person_emp_exp', 'loan_int_rate', 'cb_person_cred_hist_length'])
```

Out[31]: <Axes: >



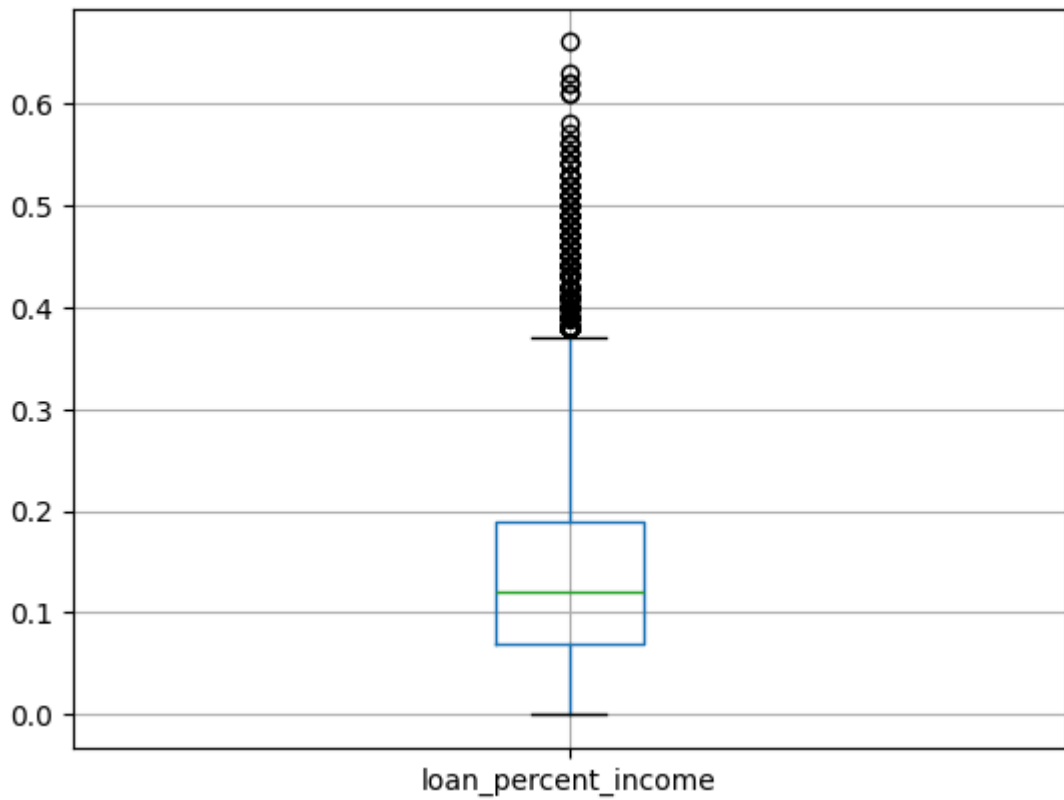
```
In [32]: x_train.boxplot('loan_amnt')
```

Out[32]: <Axes: >



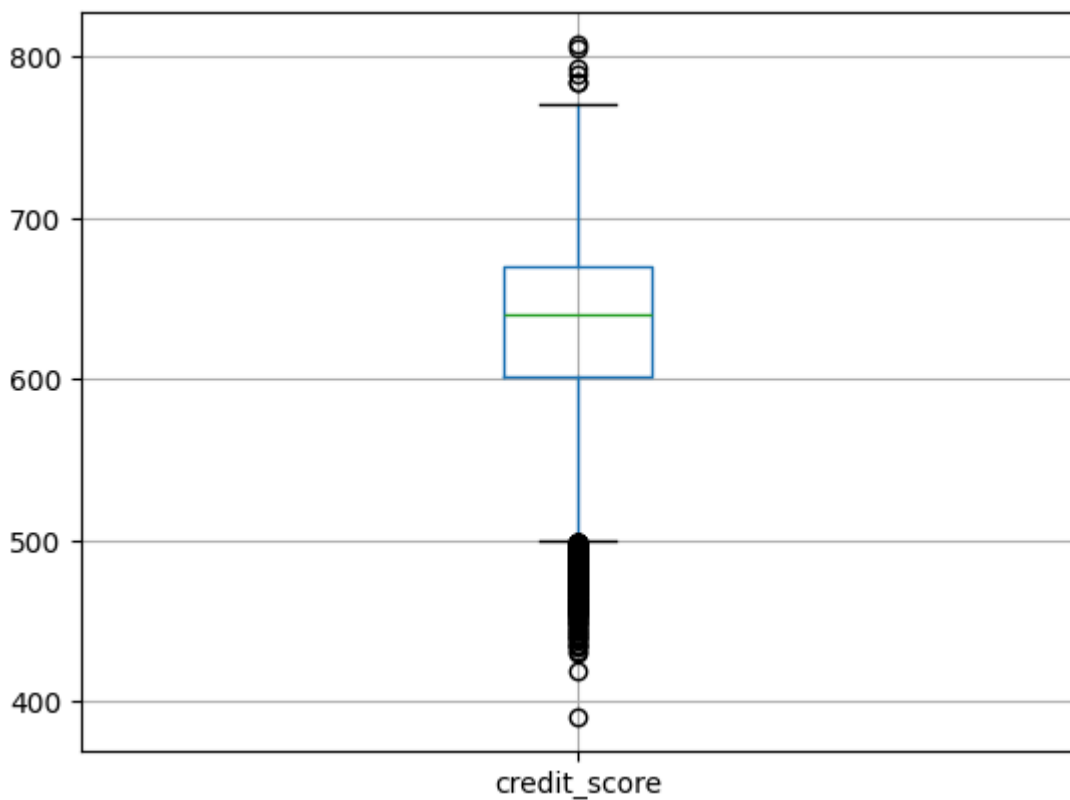
```
In [33]: x_train.boxplot('loan_percent_income')
```

```
Out[33]: <Axes: >
```



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

```
Out[34]: <Axes: >
```



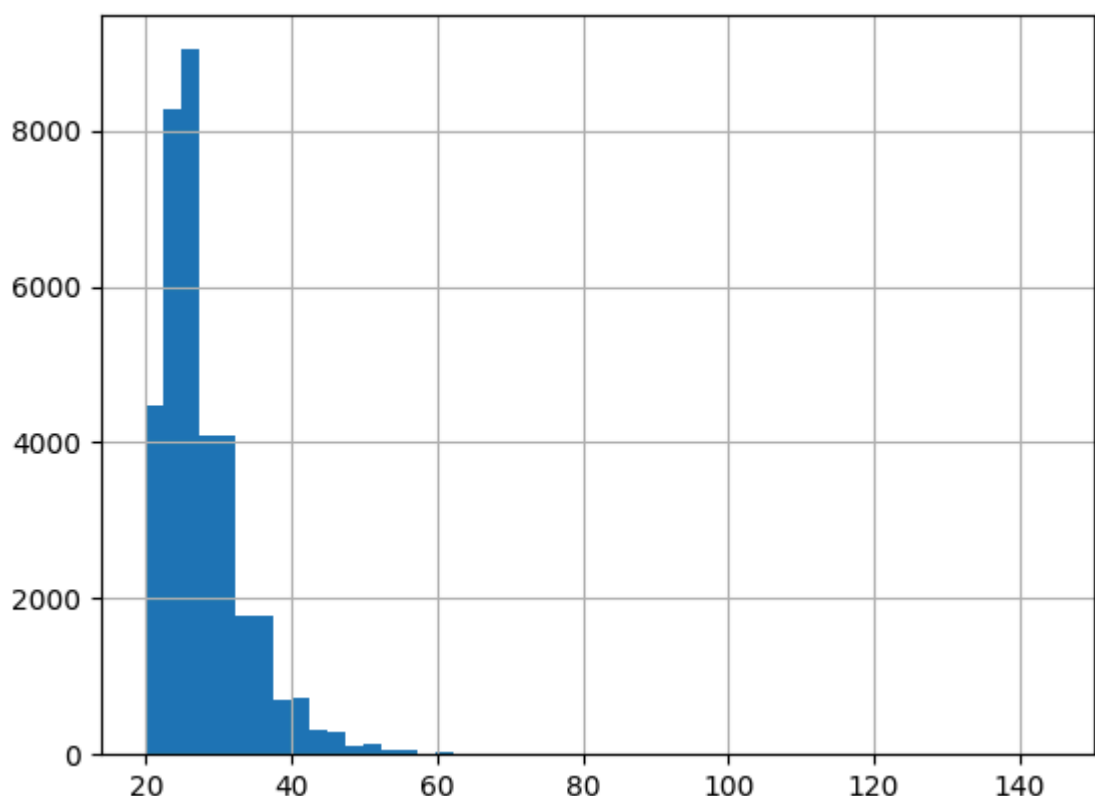
Semua numerical variable memiliki banyak outlier yang perlu di handle

person_age

```
In [35]: 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: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 173913.0, 174073.0, 350134.0, 361176.0, 160000.0, 313136.0, 175015.0, 160000.0]

```
In [36]: x_train['person_age'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

```
In [37]: age_median = x_train['person_age'].median()  
age_median
```

Out[37]: 26.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [38]: age_lower_bound, age_upper_bound = lower_upper(x_train['person_age'])  
print(age_lower_bound)  
print(age_upper_bound)
```

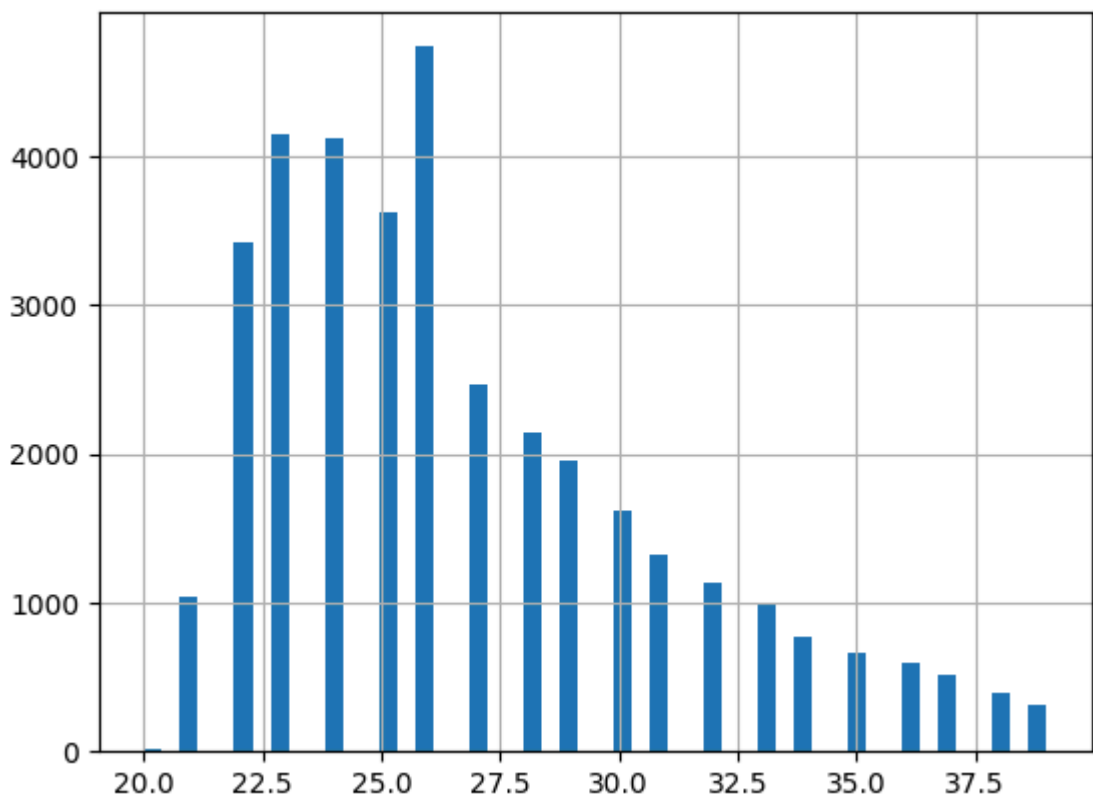
15.0

39.0

Impute outlier pada semua dataset dengan median dari training dataset

```
In [39]: 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 [40]: x_train['person_age'].hist(bins=50)  
plt.show()
```

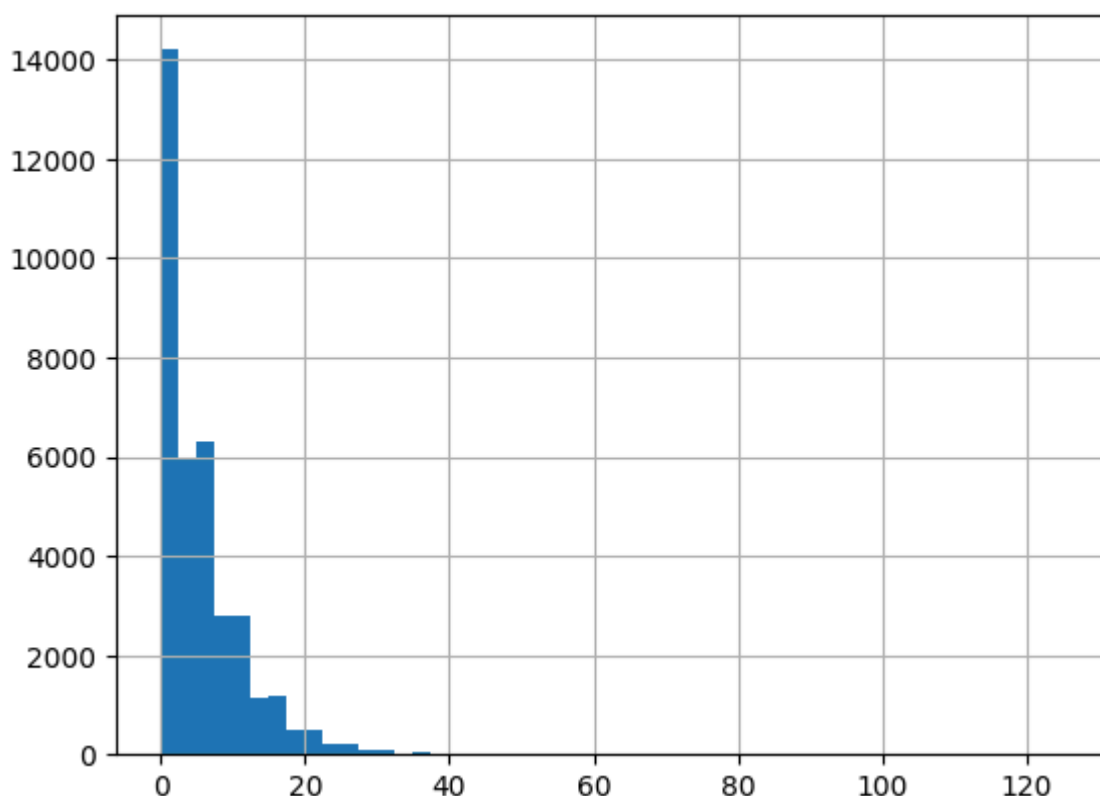


person_emp_exp

```
In [41]: 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: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 173013.0, 174070.0, 250121.0, 261170.0, 160000.0, 210120.0, 170015.0, 160000.0]

```
In [42]: x_train['person_emp_exp'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.


```
In [43]: emp_exp_median = x_train['person_emp_exp'].median()  
emp_exp_median
```

Out[43]: 4.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [44]: 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)
```

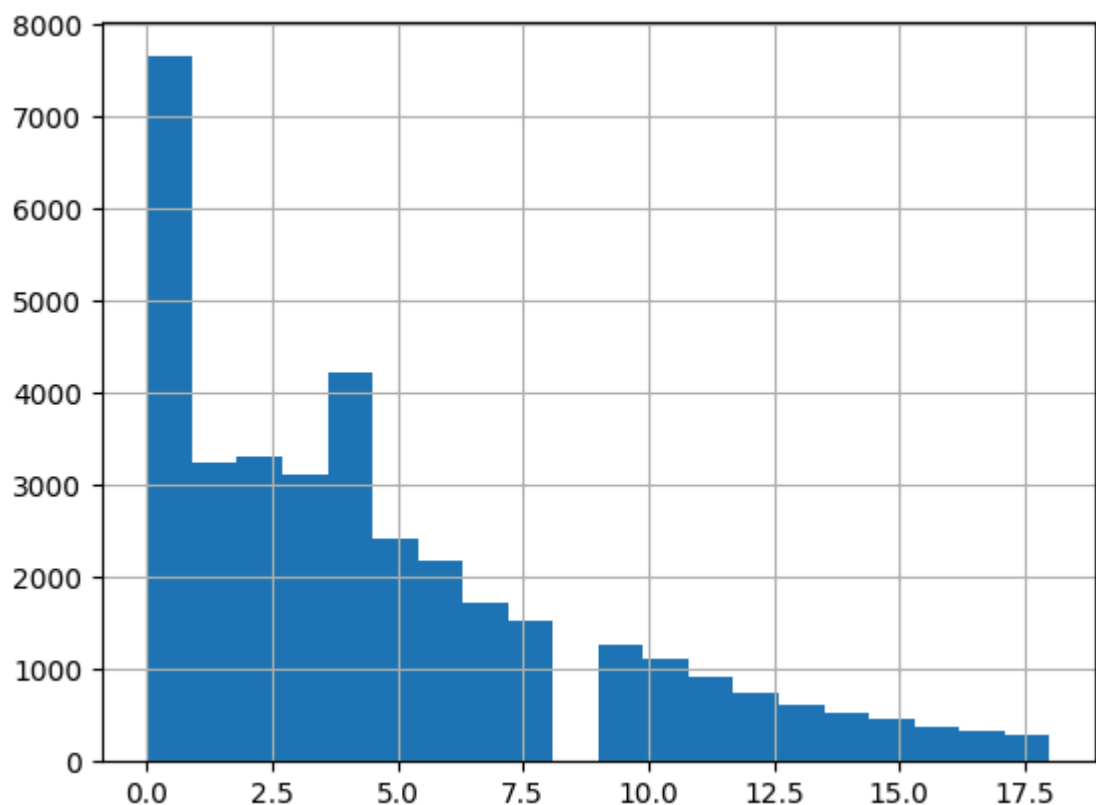
-9.5

18.5

Impute outlier pada semua dataset dengan median dari training dataset

```
In [45]: x_train['person_emp_exp'] = impute_outlier(x_train['person_emp_exp'], emp_exp_median)  
x_test['person_emp_exp'] = impute_outlier(x_test['person_emp_exp'], emp_exp_median)
```

```
In [46]: x_train['person_emp_exp'].hist(bins=20)  
plt.show()
```

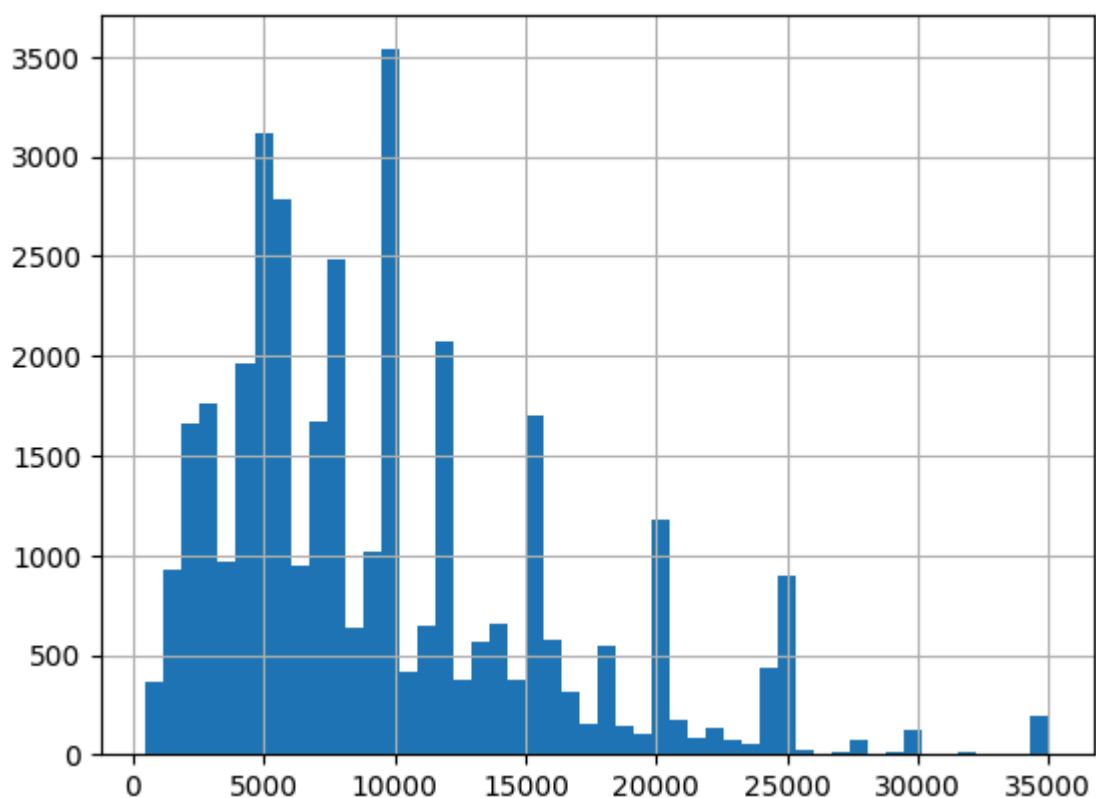


loan_amnt

```
In [47]: 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: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 173913.0, 171073.0, 356134.0, 361175.0, 168838.0, 313136.0, 176815.0, 168868.0]

```
In [48]: x_train['loan_amnt'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

```
In [49]: amnt_median = x_train['loan_amnt'].median()
amnt_median
```

Out[49]: 8000.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [50]: amnt_lower_bound, amnt_upper_bound = lower_upper(x_train['loan_amnt'])
print(amnt_lower_bound)
print(amnt_upper_bound)
```

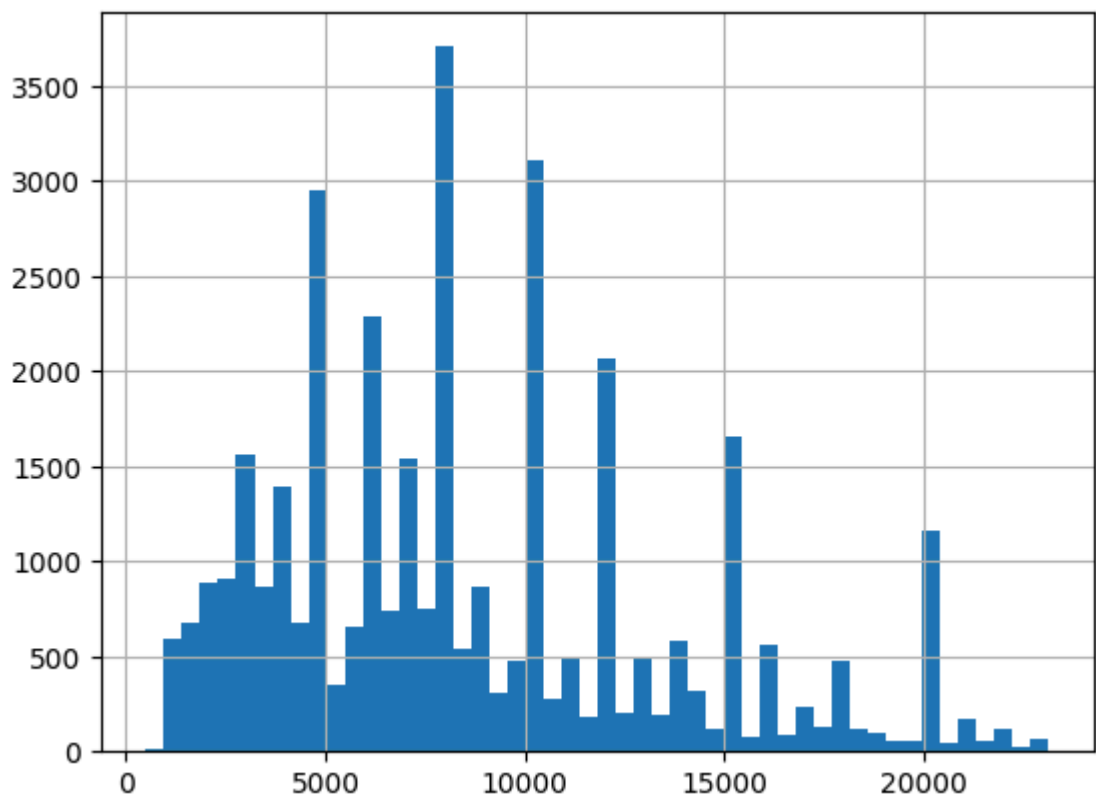
-5875.0

23125.0

Impute outlier pada semua dataset dengan median dari training dataset

```
In [51]: 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, an
```

```
In [52]: x_train['loan_amnt'].hist(bins=50)
plt.show()
```

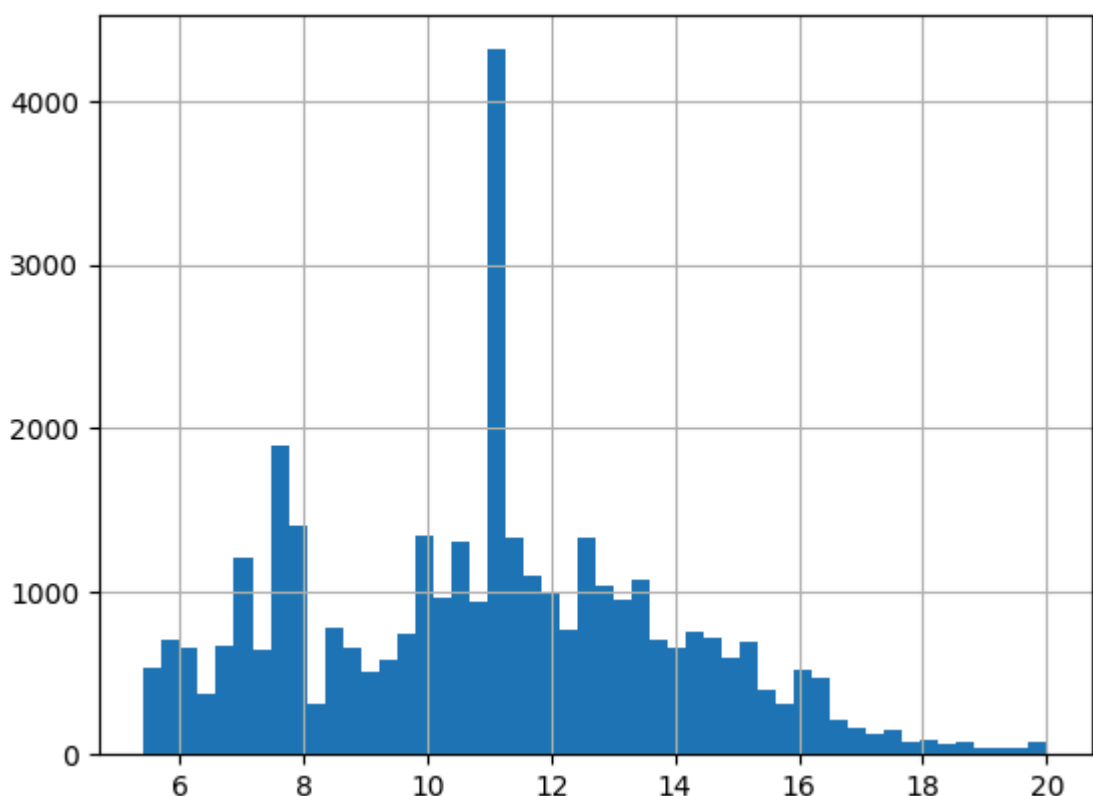


loan_int_rate

```
In [53]: 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: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 170013.0, 171070.0, 350134.0, 361170.0, 168830.0, 310130.0, 170015.0, 168830.0]

```
In [54]: x_train['loan_int_rate'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

```
In [55]: int_rate_median = x_train['loan_int_rate'].median()  
int_rate_median
```

Out[55]: 11.01

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

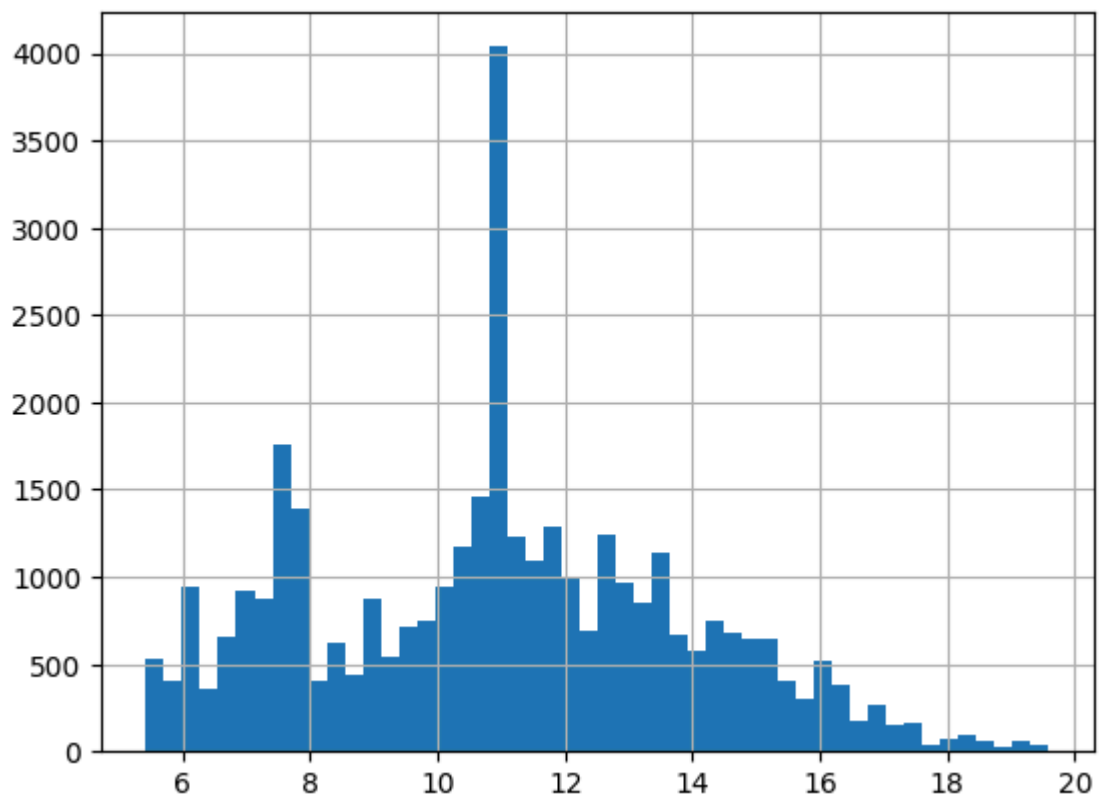
```
In [56]: int_rate_lower_bound, int_rate_upper_bound = lower_upper(x_train['loan_int_rate'])  
print(int_rate_lower_bound)  
print(int_rate_upper_bound)
```

1.9899999999999993
19.59

Impute outlier pada semua dataset dengan median dari training dataset

```
In [57]: x_train['loan_int_rate'] = impute_outlier(x_train['loan_int_rate'], int_rate_median)  
x_test['loan_int_rate'] = impute_outlier(x_test['loan_int_rate'], int_rate_median)
```

```
In [58]: x_train['loan_int_rate'].hist(bins=50)  
plt.show()
```

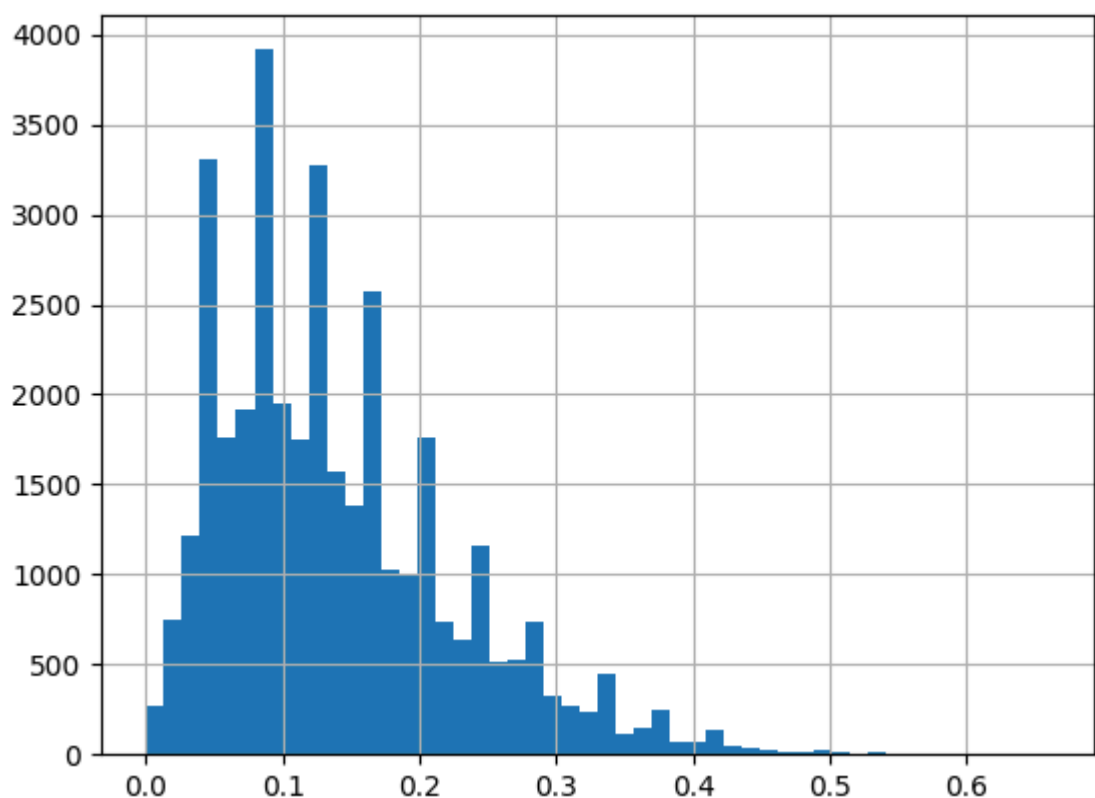


loan_percent_income

```
In [59]: 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_outlier))
```

Outliers di column 'loan_percent_income: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 173013.0, 174073.0, 250131.0, 261176.0, 168830.0, 213136.0, 176015.0]

```
In [60]: x_train['loan_percent_income'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

```
In [61]: loan_percent_median = x_train['loan_percent_income'].median()  
loan_percent_median
```

Out[61]: 0.12

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [62]: loan_percent_lower_bound, loan_percent_upper_bound = lower_upper(x_train['loan_percent_income'])  
print(loan_percent_lower_bound)  
print(loan_percent_upper_bound)
```

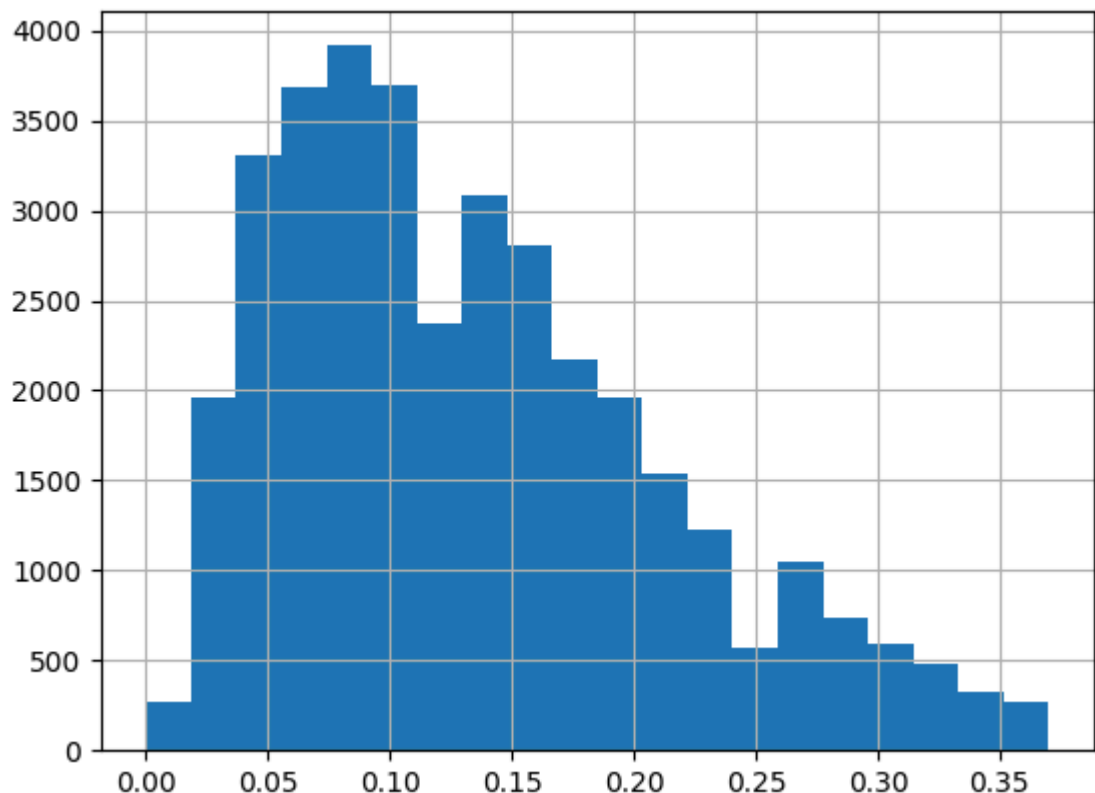
-0.10999999999999999
0.37

Impute outlier pada semua dataset dengan median dari training dataset

```
In [63]: 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 [64]: x_train['loan_percent_income'].hist(bins=20)  
plt.show()
```

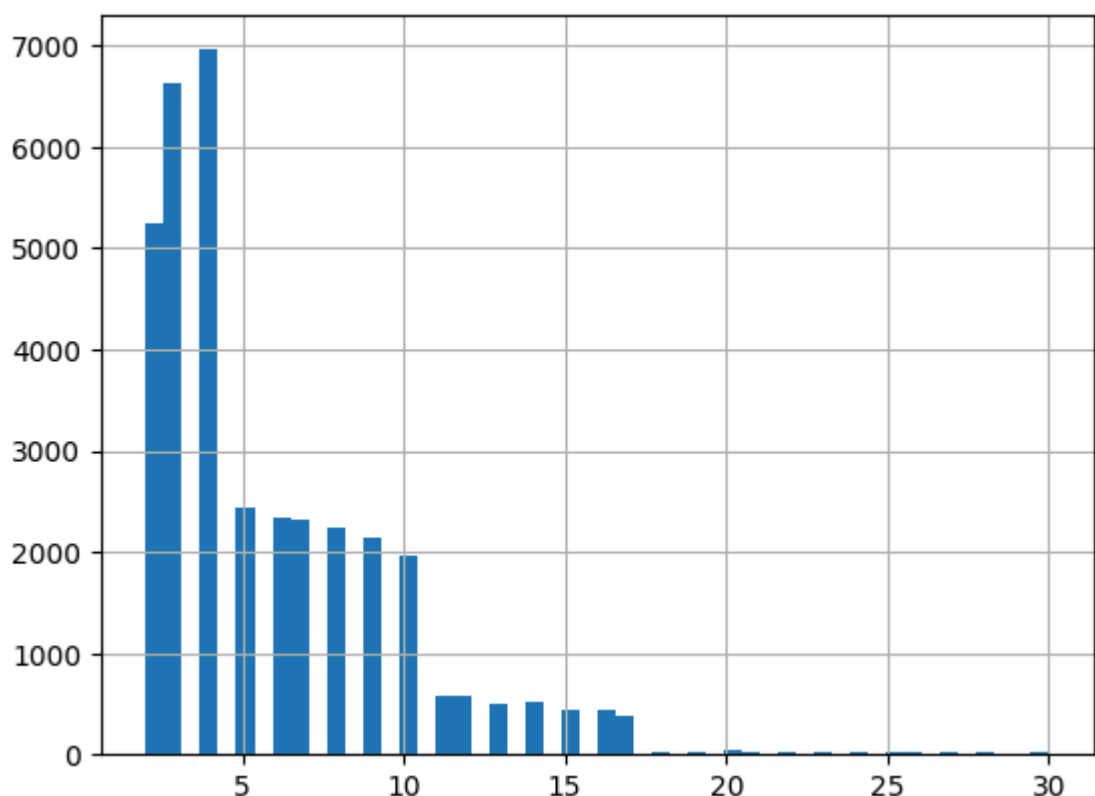


cb_person_cred_hist_length

```
In [65]: 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_outlier))
```

```
Outliers di column 'cb_person_cred_hist_length': [270972.0, 171265.0, 183
548.0, 205046.0, 211005.0, 356880.0, 177269.0, 199596.0, 216897.0, 30115
8.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 18
0670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 1808
68.0, 276922.0, 198957.0, 179877.0, 181221.0, 241041.0, 203531.0, 21102
2.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 21
0555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 2410
92.0, 725242.0, 187709.0, 193124.0, 238837.0, 175387.0, 168935.0, 18084
6.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 30
4778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 2519
92.0, 210903.0, 241005.0, 313076.0, 241278.0, 174919.0, 178777.0, 20622
9.0, 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,
210225.0, 173012.0, 171072.0, 250124.0, 261176.0, 160020.0, 212126.0, 17
```

```
In [66]: x_train['cb_person_cred_hist_length'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.


```
In [67]: cred_hist_median = x_train['cb_person_cred_hist_length'].median()  
cred_hist_median
```

Out[67]: 4.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [68]: cred_hist_lower_bound, cred_hist_upper_bound = lower_upper(x_train['cb_person_cred_hist_length'])  
print(cred_hist_lower_bound)  
print(cred_hist_upper_bound)
```

-4.5

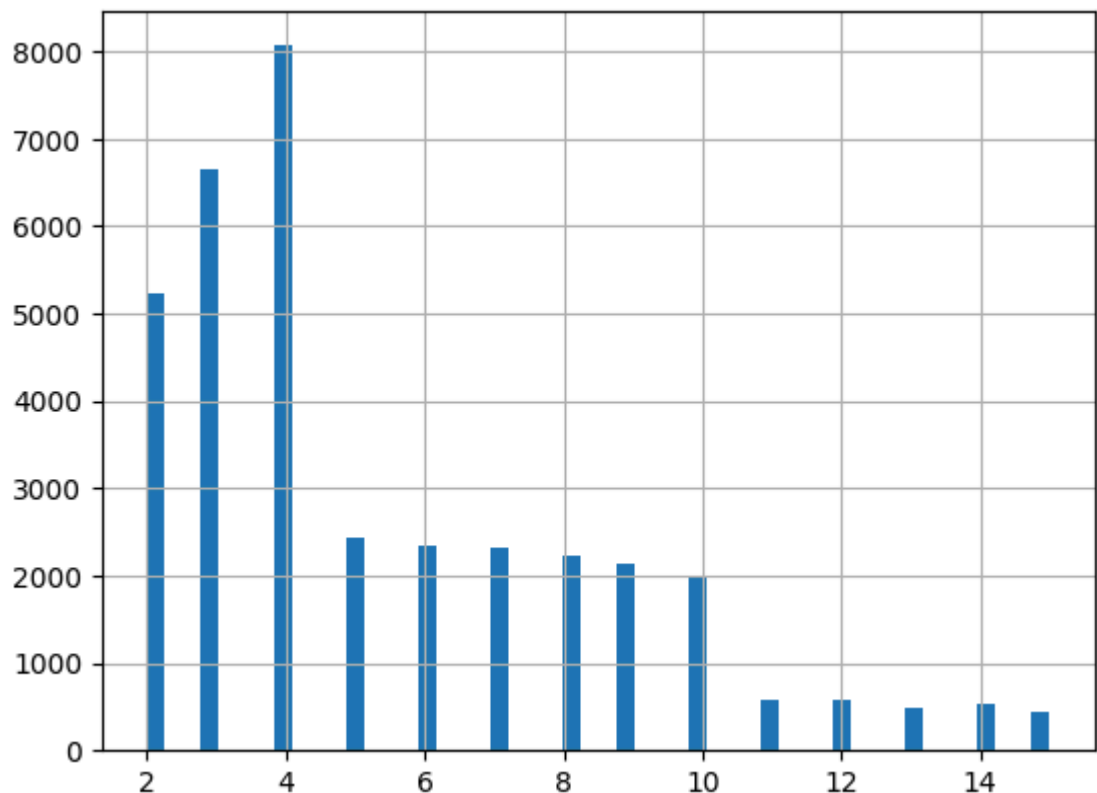
15.5

Impute outlier pada semua dataset dengan median dari training dataset

```
In [69]: x_train['cb_person_cred_hist_length'] = impute_outlier(x_train['cb_person_cred_hist_length'], cred_hist_median)  
x_test['cb_person_cred_hist_length'] = impute_outlier(x_test['cb_person_cred_hist_length'], cred_hist_median)
```



```
In [70]: x_train['cb_person_cred_hist_length'].hist(bins=50)  
plt.show()
```

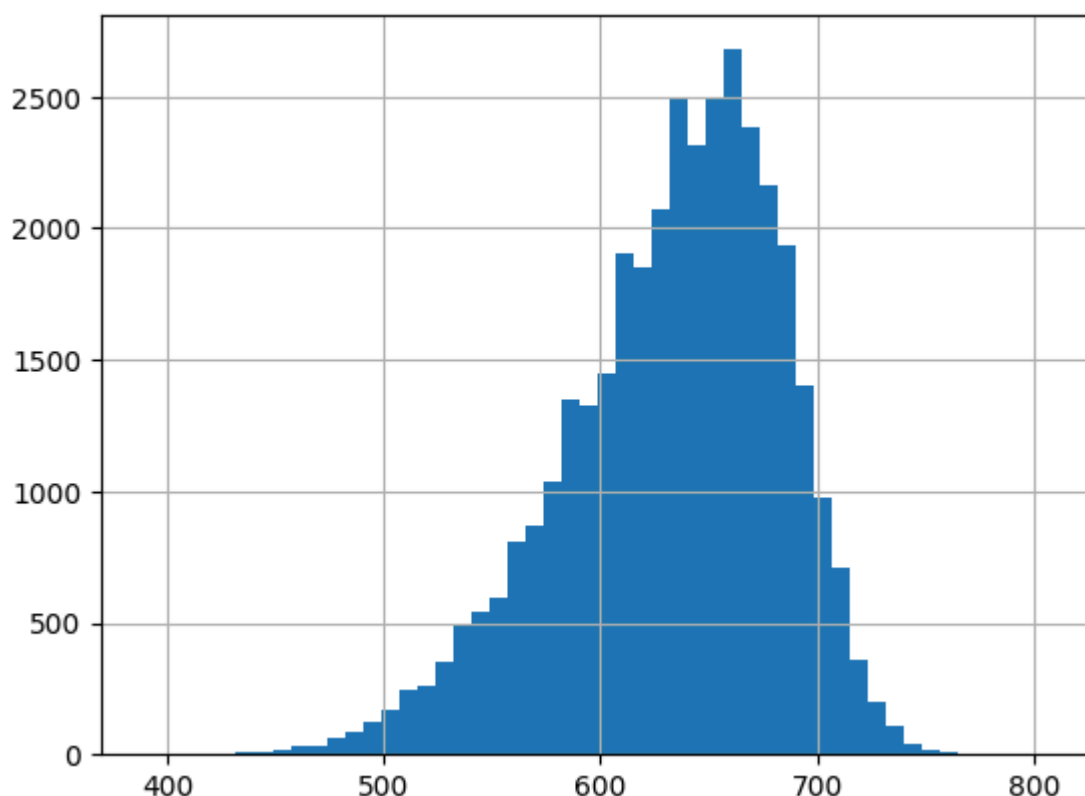


credit_score

```
In [71]: 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: [270972.0, 171265.0, 183548.0, 205046.0, 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, 241153.0, 301151.0, 422975.0, 193097.0, 228303.0, 289054.0, 180670.0, 322597.0, 178601.0, 209393.0, 291800.0, 360959.0, 240642.0, 180868.0, 276922.0, 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, 286397.0, 168827.0, 169315.0, 170578.0, 181214.0, 181353.0, 210555.0, 226751.0, 217115.0, 223194.0, 186971.0, 202682.0, 240810.0, 241092.0, 725242.0, 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, 300843.0, 321623.0, 366786.0, 735661.0, 266203.0, 178602.0, 304778.0, 169127.0, 174836.0, 180416.0, 360969.0, 391019.0, 173423.0, 251992.0, 210903.0, 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, 173912.0, 171070.0, 350131.0, 361170.0, 160000.0, 310130.0, 170015.0, 160000.0]

```
In [72]: x_train['credit_score'].hist(bins=50)
plt.show()
```



Karena data distributionnya skewed, kita impute outlier dengan median.

```
In [73]: credit_score_median = x_train['credit_score'].median()  
credit_score_median
```

Out[73]: 640.0

Cari tahu lower dan upper bound untuk outlier berdasarkan x_train

```
In [74]: credit_score_lower_bound, credit_score_upper_bound = lower_upper(x_train['cred  
print(credit_score_lower_bound)  
print(credit_score_upper_bound)
```

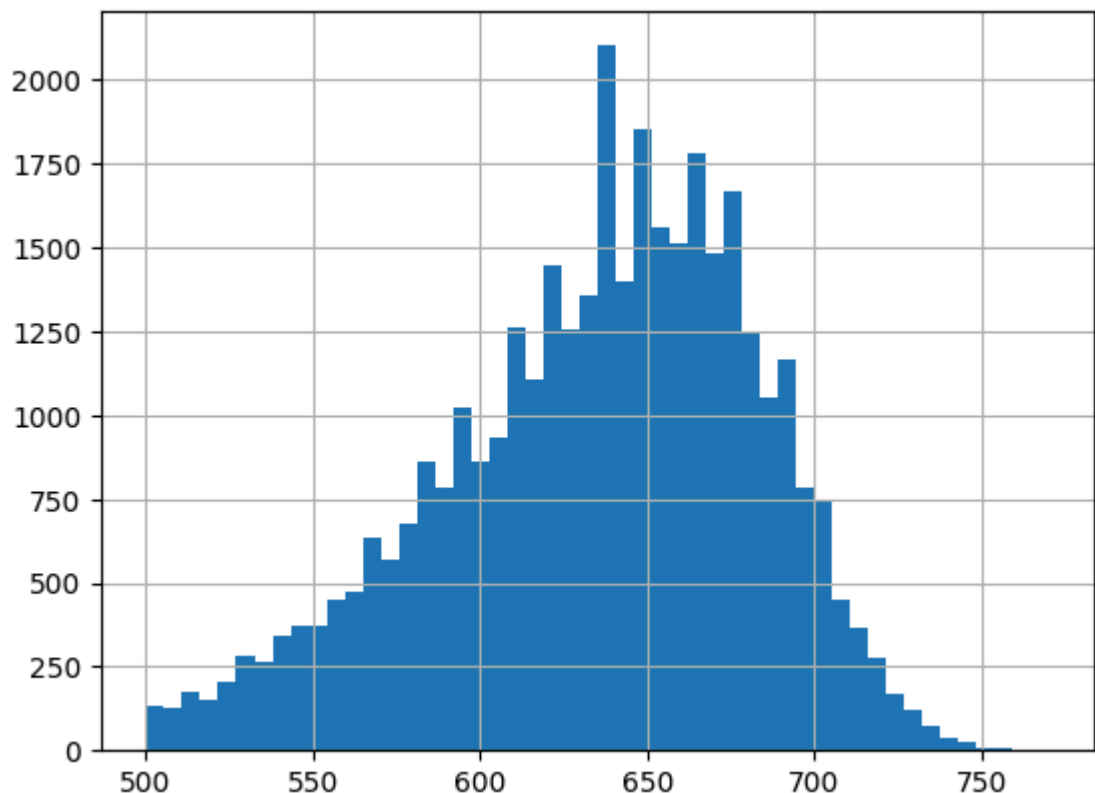
500.0

772.0

Impute outlier pada semua dataset dengan median dari training dataset

```
In [75]: x_train['credit_score'] = impute_outlier(x_train['credit_score'], credit_score  
x_test['credit_score'] = impute_outlier(x_test['credit_score'], credit_score_1
```

```
In [76]: x_train['credit_score'].hist(bins=50)  
plt.show()
```



Encoding

In [77]: `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 [78]: 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['person_h
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_train

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[78]:
```

	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 [79]: 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 [80]: Random_forest_class= GridSearchCV(RandomForestClassifier(), param_grid = param
```

```
In [81]: 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': 'log_loss', 'max_depth': 7, 'min_samples_split': 5}
Accuracy score : 0.9124166666666668

```
In [82]: Random_forest_class_best = RandomForestClassifier(criterion = 'entropy', max_depth=7, min_samples_split=5,
Random_forest_class_best.fit(x_train, y_train)
```

```
Out[82]: 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 [83]: 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 [84]: 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 [85]: xgb_class = xgb.XGBClassifier()
grid_xgb_class = GridSearchCV(xgb_class, param_grid = parameters, scoring= 'ac
```

```
In [86]: 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 [87]: xgb_class_best = xgb.XGBClassifier(eta = 0.2, gamma = 0.1, max_depth = 7)
xgb_class_best.fit(x_train, y_train)
```

```
Out[87]: XGBClassifier(base_score=None, booster=None, callbacks=None,
                      colsample_bylevel=None, colsample_bynode=None,
                      colsample_bytree=None, device=None, early_stopping_rounds=None,
                      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=None,
                      n_jobs=None, num_parallel_tree=None, ...)
one,
```

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 [88]: 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