data-preprocessing-business

November 17, 2023

1 Permasalahan Bisnis:

Sumanto seorang kredit analis sebuah Bank ABC sedang memiliki masalah karena banyaknya nasabah yang mengalami kredit macet. Untuk mengantisipasi masalah tersebut, dia mencoba melakukan analisis terhadap data nasabah dan status pembayaran cicilan kreditnya agar dapat memprediksi profile debitur (penghutang) dari aspek lancar atau macet kreditnya.

2 Tujuan Bisnis:

Untuk memprediksi calon nasabah apakah dapat membayar kredit lancar atau macet berdasarkan data history tahun lalu.(data terlampir)

3 Tujuan Teknis Data Science:

Membuat model klasifikasi (decission tree atau naïve bayes) untuk memprediksi seorang calon debitur , apakah dapat lancar membayar cicilan kredit atau tidak.

Ukuran keberhasilan pengembangan model klasifikasi sebagai berikut: nilai accuracy, precision, recall dan F-1 score harus diatas 80%.

[1]:
```python
# Visual Python: Data Analysis > File
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Asesmen Data Science/
 ↪creditapproval-data kotor.csv', sep=';')
df.head(10)

N, P = df.shape # Ukuran Data
print('baris = ', N, ', Kolom (jumlah variabel) = ', P)
print("Tipe Variabe df = ", type(df))
df
```

```
baris =  766 , Kolom (jumlah variabel) =  16
Tipe Variabe df =  <class 'pandas.core.frame.DataFrame'>
```

[1]:
```
  nama_nasabah jenis_kelamin  umur jml_pinjaman   jkw  \
0           x1             P  40.0       345000   1.0
1           x2             L  31.0       350000   7.0
2           x3             L   NaN       649926   6.0
```

3	x4	P	2.0	459168	NaN
4	x5	WANITA	34.0	3055499	8.0
..	…	…	…	…	…
761	x762	L	38.0	1000000	16.0
762	x763	P	36.0	1000000	12.0
763	x764	L	28.0	2000000	10.0
764	x765	P	31.0	1312500	7.0
765	x766	P	36.0	2000000	4.0

	jml_angsuran_per_bulan	type_pinjaman	jenis_pinjaman	bi_sektor_ekonomi \
0	345000	100	301	6000.0
1	55716	100	301	6000.0
2	108321	100	301	6000.0
3	38264	100	301	6000.0
4	381937,41	100	301	6000.0
..	…	…	…	…
761	70000	100	301	6000.0
762	90833,37	100	301	6000.0
763	260000	100	301	6000.0
764	198750	100	301	6000.0
765	550000	100	301	6000.0

	col	bi_golongan_debitur	bi_gol_penjamin	saldo_nominatif \
0	1	874	875	345000
1	1	874	875	390000
2	1	874	875	649926
3	1	874	875	459168
4	1	874	875	3055499
..	…	…	…	…
761	2	874	0	812500
762	2	874	0	429000
763	2	874	0	600000
764	2	874	0	1312500
765	2	874	0	1000000

	tunggakan_pokok	tunggakan_bunga	status kredit
0	345000	0	MACET
1	111428	0	MACET
2	216642	0	MACET
3	382640	0	MACET
4	1527749,48	0	MACET
..	…	…	…
761	812500	97500	MACET
762	429000	45000	MACET
763	600000	180000	MACET
764	1312500	78750	MACET
765	1000000	100000	MACET

```
[766 rows x 16 columns]
```

[2]:
```python
# Daftar Nama Kolom
df.columns
```

[2]:
```
Index(['nama_nasabah', 'jenis_kelamin', 'umur', 'jml_pinjaman', 'jkw',
       'jml_angsuran_per_bulan', 'type_pinjaman', 'jenis_pinjaman',
       'bi_sektor_ekonomi', 'col', 'bi_golongan_debitur', 'bi_gol_penjamin',
       'saldo_nominatif', 'tunggakan_pokok', 'tunggakan_bunga',
       'status kredit'],
      dtype='object')
```

[3]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 766 entries, 0 to 765
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   nama_nasabah            766 non-null    object
 1   jenis_kelamin           766 non-null    object
 2   umur                    757 non-null    float64
 3   jml_pinjaman            766 non-null    object
 4   jkw                     758 non-null    float64
 5   jml_angsuran_per_bulan  766 non-null    object
 6   type_pinjaman           766 non-null    int64
 7   jenis_pinjaman          766 non-null    int64
 8   bi_sektor_ekonomi       765 non-null    float64
 9   col                     766 non-null    int64
 10  bi_golongan_debitur     766 non-null    int64
 11  bi_gol_penjamin         766 non-null    int64
 12  saldo_nominatif         766 non-null    object
 13  tunggakan_pokok         766 non-null    object
 14  tunggakan_bunga         766 non-null    object
 15  status kredit           766 non-null    object
dtypes: float64(3), int64(5), object(8)
memory usage: 95.9+ KB
```

[4]:
```python
df.describe()
```

[4]:
```
              umur         jkw  type_pinjaman  jenis_pinjaman  \
count   757.000000  758.000000          766.0      766.000000
mean     29.073976   19.011873          100.0      301.197128
std     264.552192   32.231431            0.0        0.822267
min   -7162.000000    1.000000          100.0      301.000000
25%      32.000000    8.000000          100.0      301.000000
```

```
50%       38.000000   12.000000          100.0      301.000000
75%       43.000000   20.000000          100.0      301.000000
max     1043.000000  679.000000          100.0      305.000000

       bi_sektor_ekonomi         col  bi_golongan_debitur  bi_gol_penjamin
count         765.000000  766.000000           766.000000       766.000000
mean         6013.045752    1.216710           873.968668       281.300261
std           216.196305    0.412273             1.460257       408.099019
min          6000.000000    1.000000           834.000000         0.000000
25%          6000.000000    1.000000           874.000000         0.000000
50%          6000.000000    1.000000           874.000000         0.000000
75%          6000.000000    1.000000           874.000000       875.000000
max          9990.000000    2.000000           876.000000       875.000000
```

4 Alert: Data Noise

Dilihat dari info diatas, ditemukan bahwa kolom 'jml_pinjaman', 'jml_angsuran_per_bulan', 'saldo_nominatif', 'tunggakan_pokok', 'tunggakan_bunga' bertipe data object.

Data tersebut harusnya bersifat numerik. Maka dari itu diubah dulu ke dalam bentuk 'float' atau 'int'.

[5]:
```python
# Salin data untuk dapat menjaga keaslian data
df_clean=df.copy()

def clean_noise(value):
    try:
        # Mengonversi nilai ke float dan menangani kesalahan dengan 'coerce'
        return pd.to_numeric(value, errors='coerce')
    except ValueError:
        return value

# Membersihkan koma dan titik dari seluruh DataFrame
columns_to_clean = ['jml_pinjaman', 'jml_angsuran_per_bulan',␣
 ↪'saldo_nominatif', 'tunggakan_pokok', 'tunggakan_bunga']

for col in columns_to_clean:
    df_clean[col] = df_clean[col].apply(clean_noise)

df_clean.head(10)
```

[5]:
```
  nama_nasabah jenis_kelamin  umur  jml_pinjaman   jkw  \
0           x1             P  40.0      345000.0   1.0
1           x2             L  31.0      350000.0   7.0
2           x3             L   NaN      649926.0   6.0
3           x4             P   2.0      459168.0   NaN
4           x5        WANITA  34.0     3055499.0   8.0
```

```
5          x6            L  49.0     2000000.0   NaN
6          x7            L   NaN     8333334.0  10.0
7          x8            L  27.0     4435001.0   8.0
8          x9            L   NaN      560000.0   NaN
9         x10     LAKI-LAKI  49.0     1443750.0  15.0

   jml_angsuran_per_bulan  type_pinjaman  jenis_pinjaman  bi_sektor_ekonomi  \
0                345000.0            100             301             6000.0
1                 55716.0            100             301             6000.0
2                108321.0            100             301             6000.0
3                 38264.0            100             301             6000.0
4                     NaN            100             301             6000.0
5                     0.0            100             301             6000.0
6                     NaN            100             301             6000.0
7                671098.0            100             301             6000.0
8                 95221.0            100             301             6000.0
9                107800.0            100             301             6000.0

   col  bi_golongan_debitur  bi_gol_penjamin  saldo_nominatif  \
0    1                  874              875         345000.0
1    1                  874              875         390000.0
2    1                  874              875         649926.0
3    1                  874              875         459168.0
4    1                  874              875        3055499.0
5    1                  874              875         -85000.0
6    1                  874              875        8333334.0
7    1                  874              875        4435001.0
8    1                  874              875         660800.0
9    1                  874              875        1617000.0

   tunggakan_pokok  tunggakan_bunga status kredit
0         345000.0              0.0         MACET
1         111428.0              0.0         MACET
2         216642.0              0.0         MACET
3         382640.0              0.0         MACET
4              NaN              0.0         MACET
5              0.0              0.0        LANCAR
6              NaN              0.0         MACET
7              0.0              0.0        LANCAR
8         100800.0              0.0         MACET
9        1078000.0              0.0         MACET
```

[6]:
```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 766 entries, 0 to 765
Data columns (total 16 columns):
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   nama_nasabah            766 non-null    object
 1   jenis_kelamin           766 non-null    object
 2   umur                    757 non-null    float64
 3   jml_pinjaman            701 non-null    float64
 4   jkw                     758 non-null    float64
 5   jml_angsuran_per_bulan  425 non-null    float64
 6   type_pinjaman           766 non-null    int64
 7   jenis_pinjaman          766 non-null    int64
 8   bi_sektor_ekonomi       765 non-null    float64
 9   col                     766 non-null    int64
 10  bi_golongan_debitur     766 non-null    int64
 11  bi_gol_penjamin         766 non-null    int64
 12  saldo_nominatif         607 non-null    float64
 13  tunggakan_pokok         544 non-null    float64
 14  tunggakan_bunga         750 non-null    float64
 15  status kredit           766 non-null    object
dtypes: float64(8), int64(5), object(3)
memory usage: 95.9+ KB
```

Untuk data object sebaiknya kita ubah menjadi data kategori

[7]:
```python
# Lihat data yang berbentuk Objek
df_clean_objects = df_clean.copy()
df_object_variable = df_clean_objects.select_dtypes(include = ['object'])

# Lakukan looping untuk kolom pada variabel "df_objects"
for col in df_object_variable.columns:
    df_clean_objects[col] = df_clean_objects[col].astype('category')

df_clean_objects.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 766 entries, 0 to 765
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   nama_nasabah            766 non-null    category
 1   jenis_kelamin           766 non-null    category
 2   umur                    757 non-null    float64
 3   jml_pinjaman            701 non-null    float64
 4   jkw                     758 non-null    float64
 5   jml_angsuran_per_bulan  425 non-null    float64
 6   type_pinjaman           766 non-null    int64
 7   jenis_pinjaman          766 non-null    int64
 8   bi_sektor_ekonomi       765 non-null    float64
 9   col                     766 non-null    int64
```

```
 10  bi_golongan_debitur     766 non-null    int64
 11  bi_gol_penjamin         766 non-null    int64
 12  saldo_nominatif         607 non-null    float64
 13  tunggakan_pokok         544 non-null    float64
 14  tunggakan_bunga         750 non-null    float64
 15  status kredit           766 non-null    category
dtypes: category(3), float64(8), int64(5)
memory usage: 103.4 KB
```

[8]:

```python
# Check Noise pada data kategorikal atau object
df_miss_val = df_clean_objects.copy()
categoric_variable = df_clean_objects.select_dtypes(include = ['object',␣
 ↪'category'])

for col in categoric_variable.columns:
    print(col,': ', set(df_clean_objects[col].unique()))
```

```
nama_nasabah :  {'x348', 'x705', 'x523', 'x39', 'x139', 'x121', 'x546', 'x663',
'x676', 'x40', 'x410', 'x474', 'x644', 'x259', 'x687', 'x576', 'x87', 'x285',
'x750', 'x419', 'x149', 'x669', 'x512', 'x696', 'x35', 'x156', 'x171', 'x421',
'x634', 'x709', 'x316', 'x253', 'x589', 'x36', 'x77', 'x199', 'x657', 'x483',
'x306', 'x630', 'x8', 'x234', 'x11', 'x393', 'x563', 'x103', 'x403', 'x489',
'x667', 'x577', 'x727', 'x127', 'x219', 'x221', 'x409', 'x567', 'x753', 'x135',
'x292', 'x720', 'x29', 'x492', 'x328', 'x736', 'x581', 'x284', 'x688', 'x167',
'x422', 'x269', 'x99', 'x240', 'x236', 'x43', 'x395', 'x142', 'x514', 'x186',
'x617', 'x484', 'x682', 'x143', 'x762', 'x627', 'x640', 'x649', 'x12', 'x575',
'x612', 'x511', 'x658', 'x66', 'x700', 'x718', 'x245', 'x734', 'x32', 'x560',
'x343', 'x517', 'x1', 'x86', 'x545', 'x542', 'x222', 'x487', 'x64', 'x50',
'x322', 'x189', 'x551', 'x303', 'x58', 'x329', 'x494', 'x464', 'x607', 'x588',
'x7', 'x425', 'x298', 'x177', 'x493', 'x144', 'x715', 'x763', 'x296', 'x258',
'x368', 'x647', 'x744', 'x488', 'x326', 'x188', 'x73', 'x299', 'x573', 'x535',
'x591', 'x604', 'x597', 'x379', 'x386', 'x183', 'x74', 'x185', 'x170', 'x373',
'x541', 'x480', 'x584', 'x465', 'x592', 'x151', 'x89', 'x302', 'x308', 'x137',
'x418', 'x61', 'x264', 'x375', 'x83', 'x122', 'x438', 'x286', 'x41', 'x220',
'x247', 'x134', 'x37', 'x637', 'x197', 'x254', 'x237', 'x615', 'x712', 'x349',
'x48', 'x75', 'x646', 'x620', 'x211', 'x232', 'x407', 'x428', 'x656', 'x162',
'x210', 'x698', 'x390', 'x88', 'x173', 'x387', 'x499', 'x553', 'x454', 'x632',
'x354', 'x265', 'x400', 'x304', 'x314', 'x146', 'x661', 'x94', 'x365', 'x160',
'x446', 'x15', 'x534', 'x599', 'x638', 'x441', 'x451', 'x20', 'x213', 'x100',
'x266', 'x434', 'x98', 'x246', 'x714', 'x225', 'x746', 'x665', 'x442', 'x648',
'x583', 'x689', 'x406', 'x556', 'x628', 'x274', 'x693', 'x120', 'x668', 'x57',
'x323', 'x473', 'x158', 'x420', 'x337', 'x206', 'x606', 'x526', 'x363', 'x312',
'x80', 'x537', 'x215', 'x350', 'x115', 'x295', 'x28', 'x181', 'x17', 'x513',
'x678', 'x377', 'x209', 'x21', 'x643', 'x747', 'x325', 'x578', 'x707', 'x605',
'x324', 'x174', 'x228', 'x450', 'x520', 'x46', 'x138', 'x433', 'x677', 'x408',
'x145', 'x697', 'x692', 'x107', 'x527', 'x106', 'x4', 'x305', 'x624', 'x518',
'x716', 'x429', 'x281', 'x290', 'x140', 'x179', 'x358', 'x760', 'x728', 'x500',
'x703', 'x208', 'x362', 'x293', 'x675', 'x561', 'x691', 'x244', 'x53', 'x748',
```

```
'x572', 'x743', 'x190', 'x218', 'x273', 'x111', 'x745', 'x685', 'x370', 'x372',
'x486', 'x462', 'x338', 'x490', 'x664', 'x327', 'x633', 'x550', 'x471', 'x544',
'x670', 'x496', 'x564', 'x481', 'x445', 'x610', 'x193', 'x150', 'x721', 'x710',
'x524', 'x683', 'x59', 'x216', 'x67', 'x90', 'x128', 'x204', 'x291', 'x704',
'x756', 'x453', 'x497', 'x463', 'x766', 'x580', 'x571', 'x113', 'x72', 'x726',
'x608', 'x367', 'x169', 'x272', 'x383', 'x227', 'x502', 'x44', 'x271', 'x63',
'x287', 'x374', 'x184', 'x548', 'x187', 'x509', 'x178', 'x424', 'x38', 'x263',
'x618', 'x68', 'x396', 'x702', 'x335', 'x510', 'x16', 'x566', 'x270', 'x436',
'x261', 'x635', 'x659', 'x443', 'x629', 'x331', 'x437', 'x207', 'x469', 'x516',
'x384', 'x742', 'x31', 'x569', 'x738', 'x376', 'x426', 'x600', 'x116', 'x540',
'x18', 'x405', 'x522', 'x459', 'x223', 'x356', 'x24', 'x201', 'x217', 'x730',
'x662', 'x723', 'x330', 'x733', 'x97', 'x26', 'x10', 'x168', 'x241', 'x202',
'x25', 'x22', 'x759', 'x60', 'x505', 'x623', 'x506', 'x532', 'x276', 'x85',
'x262', 'x267', 'x440', 'x673', 'x175', 'x536', 'x765', 'x695', 'x125', 'x666',
'x55', 'x76', 'x533', 'x382', 'x549', 'x378', 'x78', 'x582', 'x14', 'x722',
'x352', 'x342', 'x344', 'x625', 'x226', 'x2', 'x639', 'x155', 'x62', 'x596',
'x381', 'x153', 'x457', 'x282', 'x447', 'x70', 'x461', 'x224', 'x585', 'x531',
'x112', 'x401', 'x508', 'x467', 'x519', 'x587', 'x313', 'x764', 'x525', 'x195',
'x81', 'x427', 'x650', 'x621', 'x755', 'x631', 'x739', 'x19', 'x248', 'x402',
'x238', 'x392', 'x507', 'x653', 'x336', 'x626', 'x515', 'x539', 'x166', 'x45',
'x165', 'x231', 'x371', 'x96', 'x359', 'x547', 'x485', 'x200', 'x504', 'x717',
'x470', 'x157', 'x558', 'x191', 'x713', 'x278', 'x255', 'x411', 'x598', 'x602',
'x198', 'x52', 'x203', 'x289', 'x101', 'x56', 'x230', 'x229', 'x297', 'x283',
'x398', 'x315', 'x415', 'x141', 'x448', 'x729', 'x491', 'x495', 'x475', 'x79',
'x105', 'x27', 'x47', 'x23', 'x530', 'x757', 'x622', 'x725', 'x674', 'x301',
'x708', 'x148', 'x130', 'x126', 'x394', 'x751', 'x346', 'x686', 'x754', 'x397',
'x71', 'x117', 'x404', 'x472', 'x154', 'x319', 'x645', 'x353', 'x366', 'x737',
'x42', 'x758', 'x521', 'x318', 'x256', 'x477', 'x34', 'x388', 'x586', 'x252',
'x233', 'x260', 'x345', 'x478', 'x562', 'x735', 'x161', 'x559', 'x654', 'x132',
'x279', 'x498', 'x593', 'x711', 'x732', 'x30', 'x93', 'x334', 'x460', 'x9',
'x311', 'x385', 'x435', 'x159', 'x309', 'x124', 'x731', 'x574', 'x399', 'x431',
'x590', 'x136', 'x501', 'x95', 'x684', 'x679', 'x616', 'x694', 'x680', 'x110',
'x257', 'x333', 'x5', 'x439', 'x214', 'x108', 'x503', 'x172', 'x182', 'x476',
'x288', 'x355', 'x239', 'x275', 'x557', 'x456', 'x129', 'x243', 'x109', 'x332',
'x642', 'x13', 'x152', 'x423', 'x133', 'x82', 'x131', 'x251', 'x280', 'x341',
'x652', 'x277', 'x570', 'x147', 'x176', 'x543', 'x671', 'x681', 'x613', 'x724',
'x364', 'x300', 'x91', 'x417', 'x164', 'x361', 'x416', 'x595', 'x430', 'x33',
'x104', 'x614', 'x699', 'x92', 'x69', 'x339', 'x601', 'x482', 'x603', 'x740',
'x118', 'x310', 'x452', 'x119', 'x690', 'x701', 'x320', 'x307', 'x752', 'x412',
'x65', 'x555', 'x250', 'x538', 'x651', 'x294', 'x660', 'x552', 'x123', 'x565',
'x594', 'x741', 'x357', 'x611', 'x84', 'x180', 'x391', 'x449', 'x351', 'x235',
'x249', 'x528', 'x194', 'x414', 'x609', 'x761', 'x455', 'x192', 'x317', 'x268',
'x579', 'x49', 'x636', 'x102', 'x114', 'x468', 'x458', 'x242', 'x321', 'x749',
'x672', 'x347', 'x706', 'x619', 'x360', 'x380', 'x529', 'x568', 'x554', 'x389',
'x719', 'x432', 'x196', 'x340', 'x369', 'x163', 'x641', 'x212', 'x413', 'x3',
'x655', 'x51', 'x6', 'x466', 'x479', 'x444', 'x205', 'x54'}
jenis_kelamin :  {'WANITA', 'L', 'P', 'LAKI-LAKI', 'PRIA', 'PEREMPUAN'}
status kredit :  {'MACET', 'LANCAR'}
```

[9]:
```
# Mengonversi jenis kelamin
df_miss_val['jenis_kelamin'] = df_miss_val['jenis_kelamin'].
 ↪replace(to_replace=["PRIA", "WANITA", 'LAKI-LAKI', 'PEREMPUAN'],
                                                                    value=["L",␣
 ↪"P", "L", "P"])
for col in categoric_variable.columns:
  print(col,': ', set(df_miss_val[col].unique()))
```

```
nama_nasabah :  {'x348', 'x705', 'x523', 'x39', 'x139', 'x121', 'x546', 'x663',
'x676', 'x40', 'x410', 'x474', 'x644', 'x259', 'x687', 'x576', 'x87', 'x285',
'x750', 'x419', 'x149', 'x669', 'x512', 'x696', 'x35', 'x156', 'x171', 'x421',
'x634', 'x709', 'x316', 'x253', 'x589', 'x36', 'x77', 'x199', 'x657', 'x483',
'x306', 'x630', 'x8', 'x234', 'x11', 'x393', 'x563', 'x103', 'x403', 'x489',
'x667', 'x577', 'x727', 'x127', 'x219', 'x221', 'x409', 'x567', 'x753', 'x135',
'x292', 'x720', 'x29', 'x492', 'x328', 'x736', 'x581', 'x284', 'x688', 'x167',
'x422', 'x269', 'x99', 'x240', 'x236', 'x43', 'x395', 'x142', 'x514', 'x186',
'x617', 'x484', 'x682', 'x143', 'x762', 'x627', 'x640', 'x649', 'x12', 'x575',
'x612', 'x511', 'x658', 'x66', 'x700', 'x718', 'x245', 'x734', 'x32', 'x560',
'x343', 'x517', 'x1', 'x86', 'x545', 'x542', 'x222', 'x487', 'x64', 'x50',
'x322', 'x189', 'x551', 'x303', 'x58', 'x329', 'x494', 'x464', 'x607', 'x588',
'x7', 'x425', 'x298', 'x177', 'x493', 'x144', 'x715', 'x763', 'x296', 'x258',
'x368', 'x647', 'x744', 'x488', 'x326', 'x188', 'x73', 'x299', 'x573', 'x535',
'x591', 'x604', 'x597', 'x379', 'x386', 'x183', 'x74', 'x185', 'x170', 'x373',
'x541', 'x480', 'x584', 'x465', 'x592', 'x151', 'x89', 'x302', 'x308', 'x137',
'x418', 'x61', 'x264', 'x375', 'x83', 'x122', 'x438', 'x286', 'x41', 'x220',
'x247', 'x134', 'x37', 'x637', 'x197', 'x254', 'x237', 'x615', 'x712', 'x349',
'x48', 'x75', 'x646', 'x620', 'x211', 'x232', 'x407', 'x428', 'x656', 'x162',
'x210', 'x698', 'x390', 'x88', 'x173', 'x387', 'x499', 'x553', 'x454', 'x632',
'x354', 'x265', 'x400', 'x304', 'x314', 'x146', 'x661', 'x94', 'x365', 'x160',
'x446', 'x15', 'x534', 'x599', 'x638', 'x441', 'x451', 'x20', 'x213', 'x100',
'x266', 'x434', 'x98', 'x246', 'x714', 'x225', 'x746', 'x665', 'x442', 'x648',
'x583', 'x689', 'x406', 'x556', 'x628', 'x274', 'x693', 'x120', 'x668', 'x57',
'x323', 'x473', 'x158', 'x420', 'x337', 'x206', 'x606', 'x526', 'x363', 'x312',
'x80', 'x537', 'x215', 'x350', 'x115', 'x295', 'x28', 'x181', 'x17', 'x513',
'x678', 'x377', 'x209', 'x21', 'x643', 'x747', 'x325', 'x578', 'x707', 'x605',
'x324', 'x174', 'x228', 'x450', 'x520', 'x46', 'x138', 'x433', 'x677', 'x408',
'x145', 'x697', 'x692', 'x107', 'x527', 'x106', 'x4', 'x305', 'x624', 'x518',
'x716', 'x429', 'x281', 'x290', 'x140', 'x179', 'x358', 'x760', 'x728', 'x500',
'x703', 'x208', 'x362', 'x293', 'x675', 'x561', 'x691', 'x244', 'x53', 'x748',
'x572', 'x743', 'x190', 'x218', 'x273', 'x111', 'x745', 'x685', 'x370', 'x372',
'x486', 'x462', 'x338', 'x490', 'x664', 'x327', 'x633', 'x550', 'x471', 'x544',
'x670', 'x496', 'x564', 'x481', 'x445', 'x610', 'x193', 'x150', 'x721', 'x710',
'x524', 'x683', 'x59', 'x216', 'x67', 'x90', 'x128', 'x204', 'x291', 'x704',
'x756', 'x453', 'x497', 'x463', 'x766', 'x580', 'x571', 'x113', 'x72', 'x726',
'x608', 'x367', 'x169', 'x272', 'x383', 'x227', 'x502', 'x44', 'x271', 'x63',
'x287', 'x374', 'x184', 'x548', 'x187', 'x509', 'x178', 'x424', 'x38', 'x263',
'x618', 'x68', 'x396', 'x702', 'x335', 'x510', 'x16', 'x566', 'x270', 'x436',
```

```
'x261', 'x635', 'x659', 'x443', 'x629', 'x331', 'x437', 'x207', 'x469', 'x516',
'x384', 'x742', 'x31', 'x569', 'x738', 'x376', 'x426', 'x600', 'x116', 'x540',
'x18', 'x405', 'x522', 'x459', 'x223', 'x356', 'x24', 'x201', 'x217', 'x730',
'x662', 'x723', 'x330', 'x733', 'x97', 'x26', 'x10', 'x168', 'x241', 'x202',
'x25', 'x22', 'x759', 'x60', 'x505', 'x623', 'x506', 'x532', 'x276', 'x85',
'x262', 'x267', 'x440', 'x673', 'x175', 'x536', 'x765', 'x695', 'x125', 'x666',
'x55', 'x76', 'x533', 'x382', 'x549', 'x378', 'x78', 'x582', 'x14', 'x722',
'x352', 'x342', 'x344', 'x625', 'x226', 'x2', 'x639', 'x155', 'x62', 'x596',
'x381', 'x153', 'x457', 'x282', 'x447', 'x70', 'x461', 'x224', 'x585', 'x531',
'x112', 'x401', 'x508', 'x467', 'x519', 'x587', 'x313', 'x764', 'x525', 'x195',
'x81', 'x427', 'x650', 'x621', 'x755', 'x631', 'x739', 'x19', 'x248', 'x402',
'x238', 'x392', 'x507', 'x653', 'x336', 'x626', 'x515', 'x539', 'x166', 'x45',
'x165', 'x231', 'x371', 'x96', 'x359', 'x547', 'x485', 'x200', 'x504', 'x717',
'x470', 'x157', 'x558', 'x191', 'x713', 'x278', 'x255', 'x411', 'x598', 'x602',
'x198', 'x52', 'x203', 'x289', 'x101', 'x56', 'x230', 'x229', 'x297', 'x283',
'x398', 'x315', 'x415', 'x141', 'x448', 'x729', 'x491', 'x495', 'x475', 'x79',
'x105', 'x27', 'x47', 'x23', 'x530', 'x757', 'x622', 'x725', 'x674', 'x301',
'x708', 'x148', 'x130', 'x126', 'x394', 'x751', 'x346', 'x686', 'x754', 'x397',
'x71', 'x117', 'x404', 'x472', 'x154', 'x319', 'x645', 'x353', 'x366', 'x737',
'x42', 'x758', 'x521', 'x318', 'x256', 'x477', 'x34', 'x388', 'x586', 'x252',
'x233', 'x260', 'x345', 'x478', 'x562', 'x735', 'x161', 'x559', 'x654', 'x132',
'x279', 'x498', 'x593', 'x711', 'x732', 'x30', 'x93', 'x334', 'x460', 'x9',
'x311', 'x385', 'x435', 'x159', 'x309', 'x124', 'x731', 'x574', 'x399', 'x431',
'x590', 'x136', 'x501', 'x95', 'x684', 'x679', 'x616', 'x694', 'x680', 'x110',
'x257', 'x333', 'x5', 'x439', 'x214', 'x108', 'x503', 'x172', 'x182', 'x476',
'x288', 'x355', 'x239', 'x275', 'x557', 'x456', 'x129', 'x243', 'x109', 'x332',
'x642', 'x13', 'x152', 'x423', 'x133', 'x82', 'x131', 'x251', 'x280', 'x341',
'x652', 'x277', 'x570', 'x147', 'x176', 'x543', 'x671', 'x681', 'x613', 'x724',
'x364', 'x300', 'x91', 'x417', 'x164', 'x361', 'x416', 'x595', 'x430', 'x33',
'x104', 'x614', 'x699', 'x92', 'x69', 'x339', 'x601', 'x482', 'x603', 'x740',
'x118', 'x310', 'x452', 'x119', 'x690', 'x701', 'x320', 'x307', 'x752', 'x412',
'x65', 'x555', 'x250', 'x538', 'x651', 'x294', 'x660', 'x552', 'x123', 'x565',
'x594', 'x741', 'x357', 'x611', 'x84', 'x180', 'x391', 'x449', 'x351', 'x235',
'x249', 'x528', 'x194', 'x414', 'x609', 'x761', 'x455', 'x192', 'x317', 'x268',
'x579', 'x49', 'x636', 'x102', 'x114', 'x468', 'x458', 'x242', 'x321', 'x749',
'x672', 'x347', 'x706', 'x619', 'x360', 'x380', 'x529', 'x568', 'x554', 'x389',
'x719', 'x432', 'x196', 'x340', 'x369', 'x163', 'x641', 'x212', 'x413', 'x3',
'x655', 'x51', 'x6', 'x466', 'x479', 'x444', 'x205', 'x54'}
jenis_kelamin :  {'L', 'P'}
status kredit :  {'MACET', 'LANCAR'}
```

[10]:
```
df_miss_val.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 766 entries, 0 to 765
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
```

```
 0   nama_nasabah            766 non-null    category
 1   jenis_kelamin           766 non-null    category
 2   umur                    757 non-null    float64
 3   jml_pinjaman            701 non-null    float64
 4   jkw                     758 non-null    float64
 5   jml_angsuran_per_bulan  425 non-null    float64
 6   type_pinjaman           766 non-null    int64
 7   jenis_pinjaman          766 non-null    int64
 8   bi_sektor_ekonomi       765 non-null    float64
 9   col                     766 non-null    int64
 10  bi_golongan_debitur     766 non-null    int64
 11  bi_gol_penjamin         766 non-null    int64
 12  saldo_nominatif         607 non-null    float64
 13  tunggakan_pokok         544 non-null    float64
 14  tunggakan_bunga         750 non-null    float64
 15  status kredit           766 non-null    category
dtypes: category(3), float64(8), int64(5)
memory usage: 103.3 KB
```

5 Remove Missing Value

[11]:
```python
df_miss_val.isnull().sum()
```

[11]:
```
nama_nasabah                0
jenis_kelamin               0
umur                        9
jml_pinjaman               65
jkw                         8
jml_angsuran_per_bulan    341
type_pinjaman               0
jenis_pinjaman              0
bi_sektor_ekonomi           1
col                         0
bi_golongan_debitur         0
bi_gol_penjamin             0
saldo_nominatif           159
tunggakan_pokok           222
tunggakan_bunga            16
status kredit               0
dtype: int64
```

[12]:
```python
df_miss_val_drop=df_miss_val.copy()

# Drop missing values in the 'umur' column
df_miss_val_drop = df_miss_val_drop.dropna(subset=['umur','jkw',␣
 ↪'bi_sektor_ekonomi', 'tunggakan_bunga'])
```

```python
print("Ukuran setelah di drop missing value:\n",df_miss_val_drop.shape)
df_miss_val_drop.head()
```

```
Ukuran setelah di drop missing value:
 (738, 16)
```

[12]:

```
  nama_nasabah jenis_kelamin  umur  jml_pinjaman   jkw  \
0           x1             P  40.0      345000.0   1.0
1           x2             L  31.0      350000.0   7.0
4           x5             P  34.0     3055499.0   8.0
7           x8             L  27.0     4435001.0   8.0
9          x10             L  49.0     1443750.0  15.0

   jml_angsuran_per_bulan  type_pinjaman  jenis_pinjaman  bi_sektor_ekonomi  \
0                345000.0            100             301             6000.0
1                 55716.0            100             301             6000.0
4                     NaN            100             301             6000.0
7                671098.0            100             301             6000.0
9                107800.0            100             301             6000.0

   col  bi_golongan_debitur  bi_gol_penjamin  saldo_nominatif  \
0    1                  874              875         345000.0
1    1                  874              875         390000.0
4    1                  874              875        3055499.0
7    1                  874              875        4435001.0
9    1                  874              875        1617000.0

   tunggakan_pokok  tunggakan_bunga status kredit
0         345000.0              0.0         MACET
1         111428.0              0.0         MACET
4              NaN              0.0         MACET
7              0.0              0.0        LANCAR
9        1078000.0              0.0         MACET
```

[13]:

```python
df_miss_val_drop.isnull().sum()
```

[13]:

```
nama_nasabah                0
jenis_kelamin               0
umur                        0
jml_pinjaman               58
jkw                         0
jml_angsuran_per_bulan    322
type_pinjaman               0
jenis_pinjaman              0
bi_sektor_ekonomi           0
col                         0
```

```
bi_golongan_debitur          0
bi_gol_penjamin              0
saldo_nominatif            147
tunggakan_pokok            207
tunggakan_bunga              0
status kredit                0
dtype: int64
```

Dalam pendeskripsian di atas, umur memiliki nilai 'anomali'. Ada baiknya dibersihkan

[14]:
```python
# Jadikan sebagai nilai int terlebih dahulu
df_miss_val_drop['umur'] = df_miss_val_drop['umur'].astype('int64')
df_miss_val_drop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 738 entries, 0 to 765
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   nama_nasabah          738 non-null    category
 1   jenis_kelamin         738 non-null    category
 2   umur                  738 non-null    int64
 3   jml_pinjaman          680 non-null    float64
 4   jkw                   738 non-null    float64
 5   jml_angsuran_per_bulan 416 non-null    float64
 6   type_pinjaman         738 non-null    int64
 7   jenis_pinjaman        738 non-null    int64
 8   bi_sektor_ekonomi     738 non-null    float64
 9   col                   738 non-null    int64
 10  bi_golongan_debitur   738 non-null    int64
 11  bi_gol_penjamin       738 non-null    int64
 12  saldo_nominatif       591 non-null    float64
 13  tunggakan_pokok       531 non-null    float64
 14  tunggakan_bunga       738 non-null    float64
 15  status kredit         738 non-null    category
dtypes: category(3), float64(7), int64(6)
memory usage: 106.0 KB
```

[15]:
```python
print("Data unik dalam umur:\n", df_miss_val_drop['umur'].unique())
```

```
Data unik dalam umur:
 [   40    31    34    27    49    42    26    55    38    41    35     2
    39    50    57    36    58    43    44    52    46    28    47    32
    48    45    30    67    29    37    25    21    33    23    19     3
    53    51     0    54    61   -42     1    24    56   -48    68  1043
    60   -49    76    22    80 -7162   -47    65   -44    64]
```

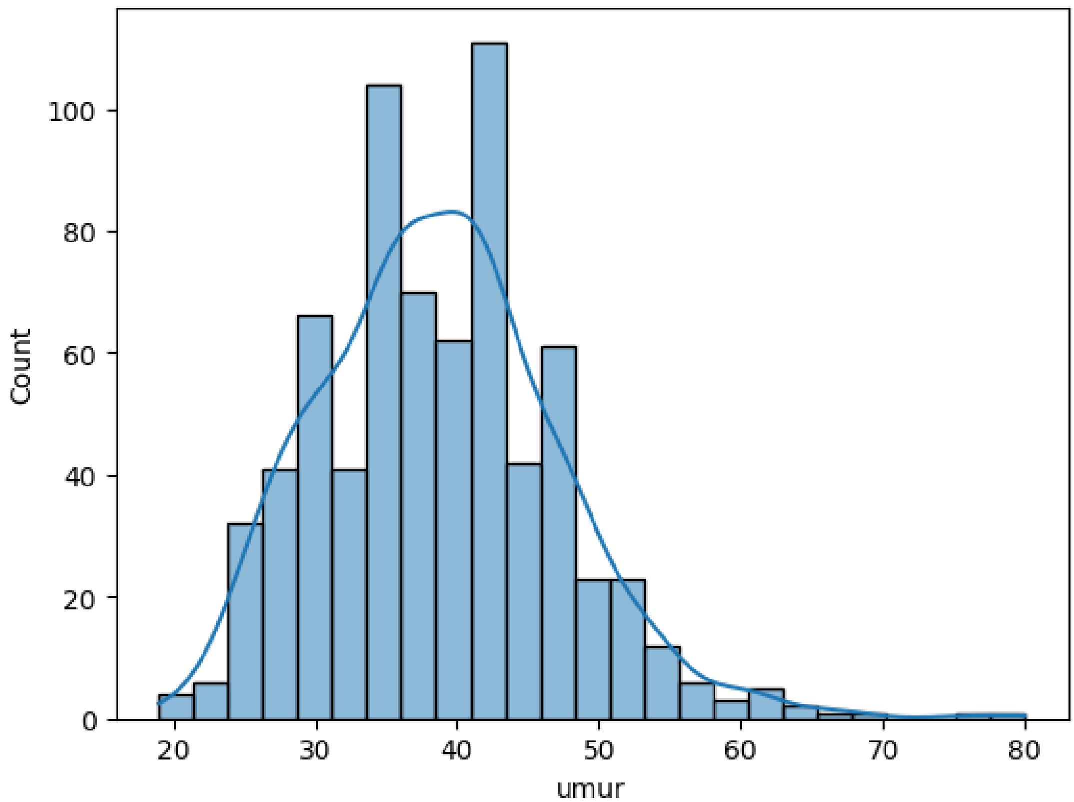
Terlihat bahwasannya data memiliki anomali. Sebaiknya tangani hal ini.

[16]:
```python
import numpy as np

import numpy as np

def clean_umur(column):
    # Replace specified values with NaN
    column = column.replace([1043, -7162, 2, 1, 0, 3], np.nan)

    # Take the absolute value
    column = column.abs()

    return column

# Apply the clean_umur function to the 'umur' column
df_miss_val_drop['umur'] = clean_umur(df_miss_val_drop['umur'])

# Check the updated DataFrame
print("Updated dataframe:\n", df_miss_val_drop['umur'])
```

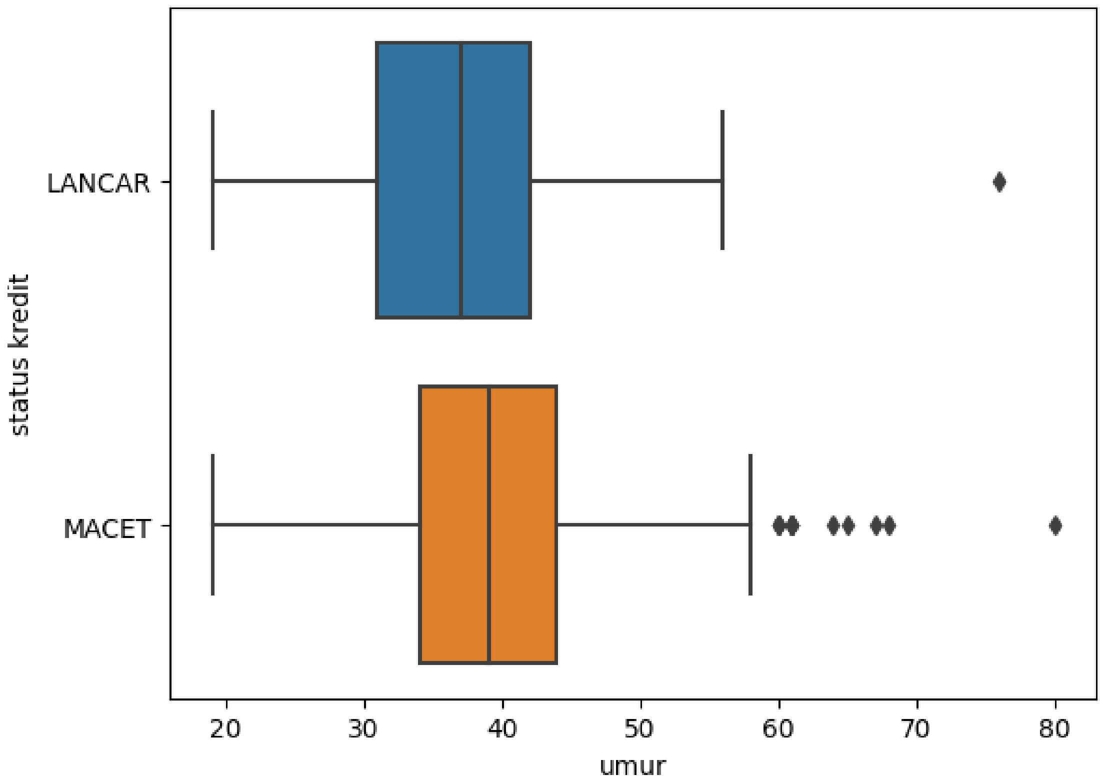
```
Updated dataframe:
 0      40.0
1      31.0
4      34.0
7      27.0
9      49.0
       …
761    38.0
762    36.0
763    28.0
764    31.0
765    36.0
Name: umur, Length: 738, dtype: float64
```

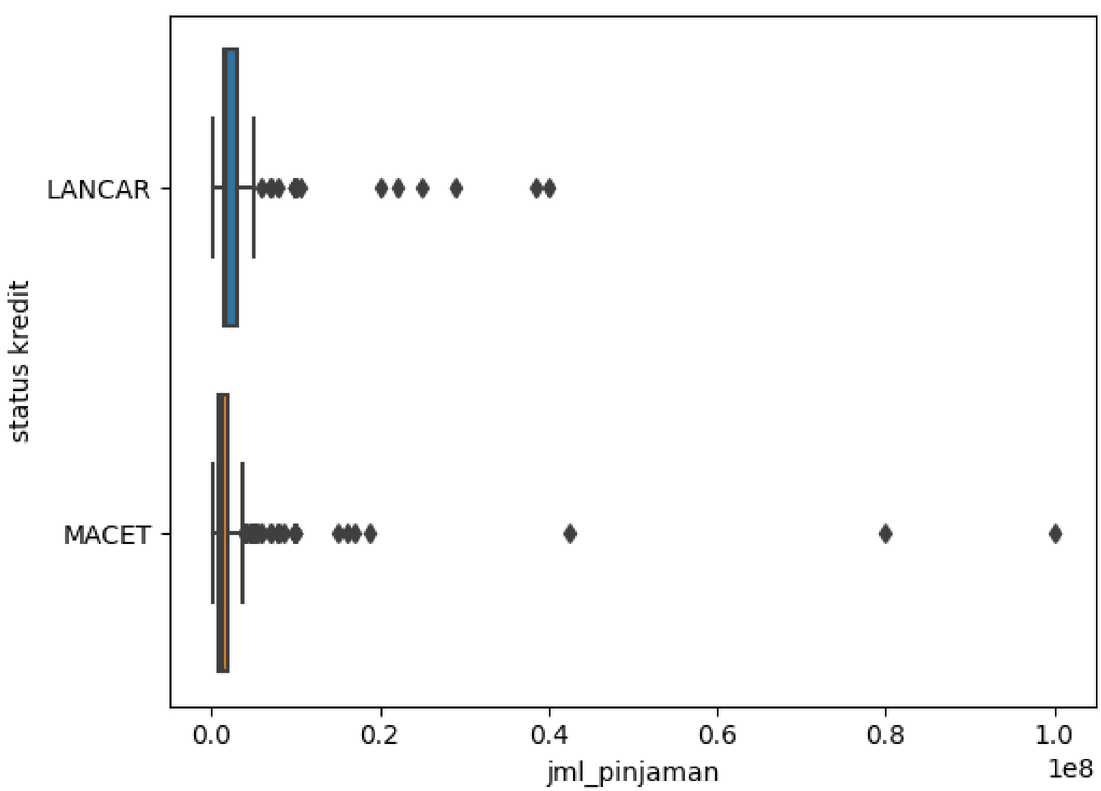
Visualize the data

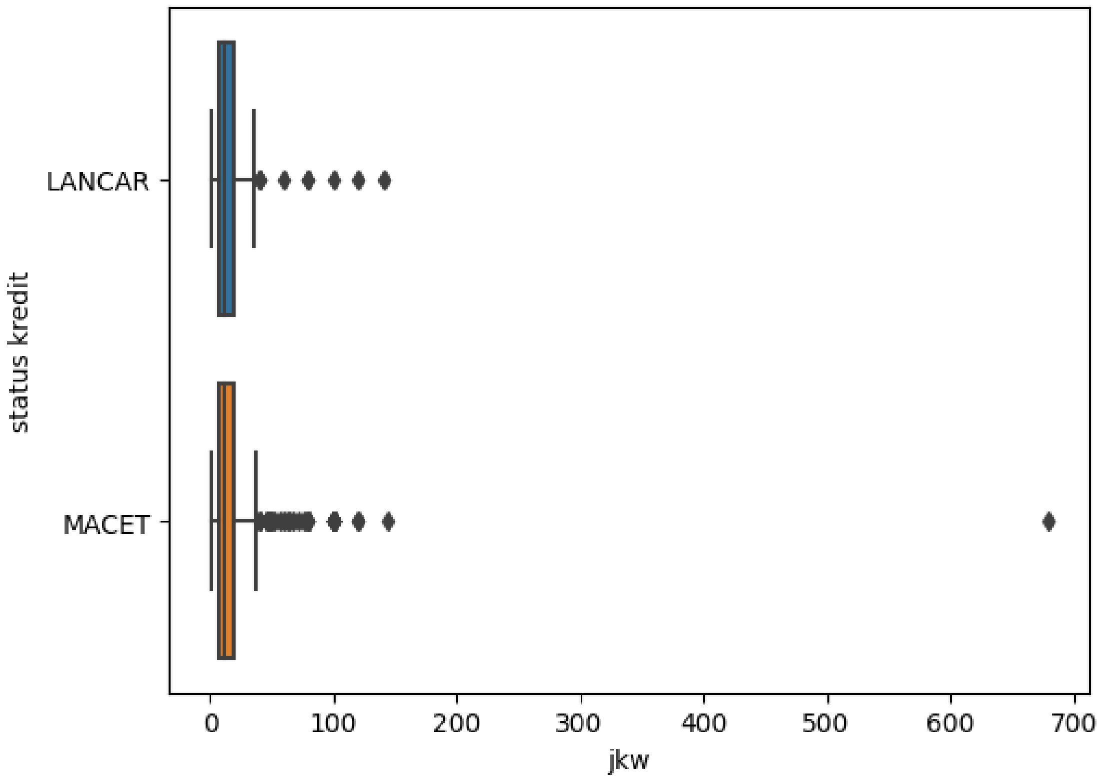
[17]:
```python
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt

sns.histplot(data=df_miss_val_drop, x='umur', kde=True)
plt.show()
```

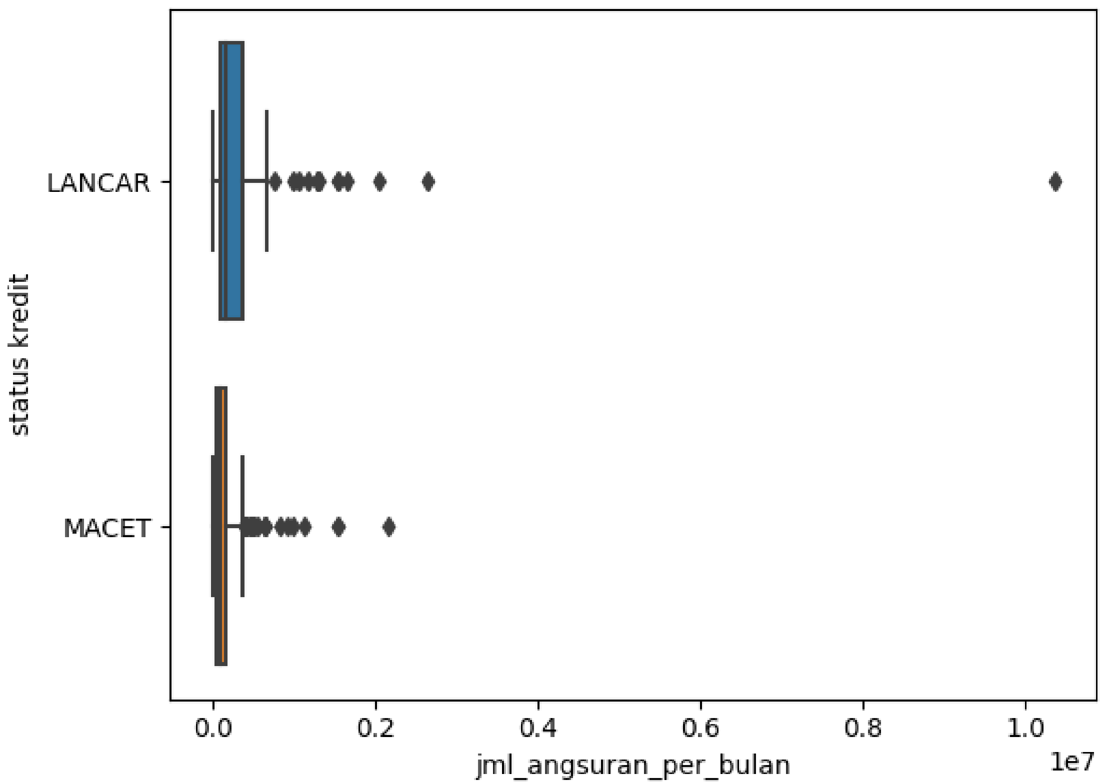


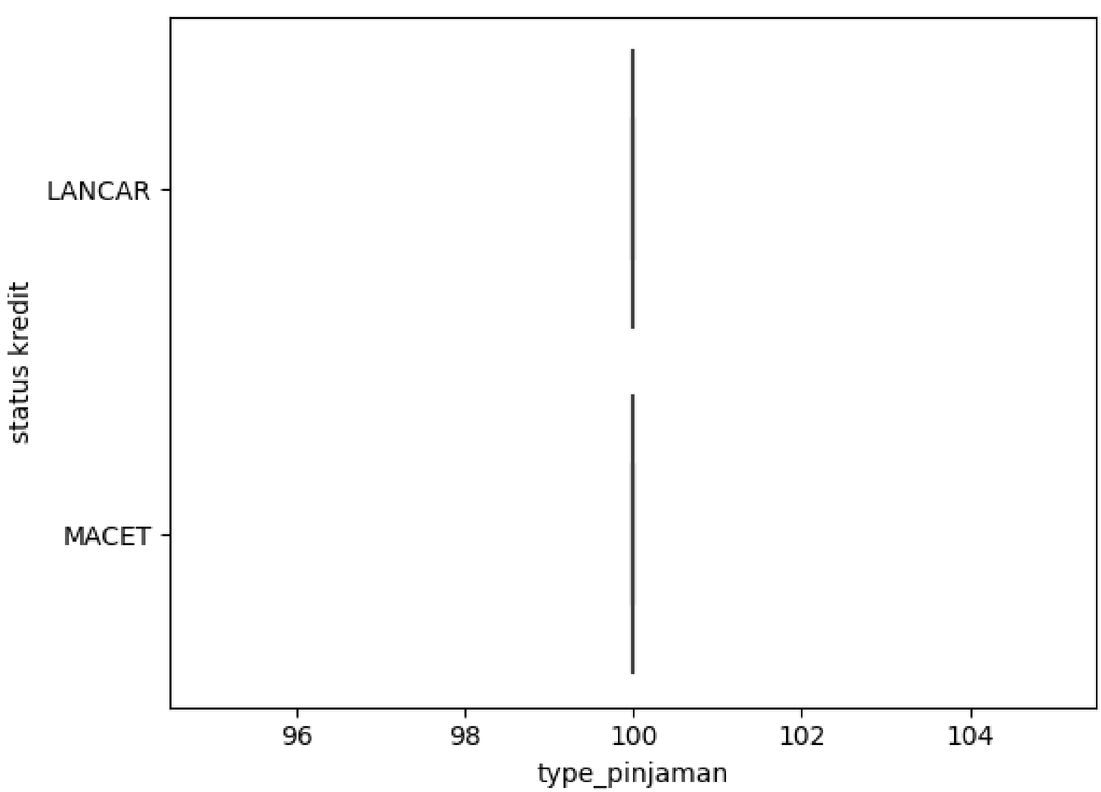
[18]:
```
# Visual Python: Visualization > Seaborn
intVar = df_miss_val_drop.select_dtypes(include = ['int64', 'float64'])
for col in intVar.columns:
    p = sns.boxplot(x=col, y="status kredit", data=df_miss_val_drop)
    plt.show()
```

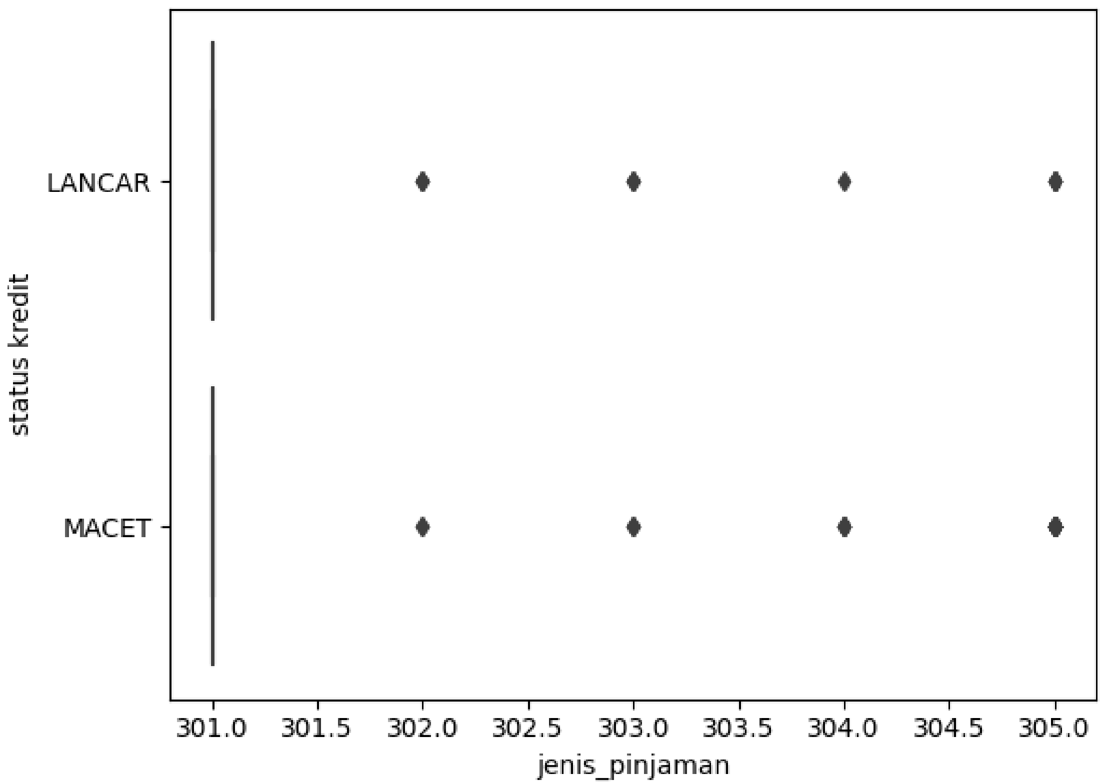

LANCAR
MACET
status kredit
20
30
40
50
60
70
80
umur

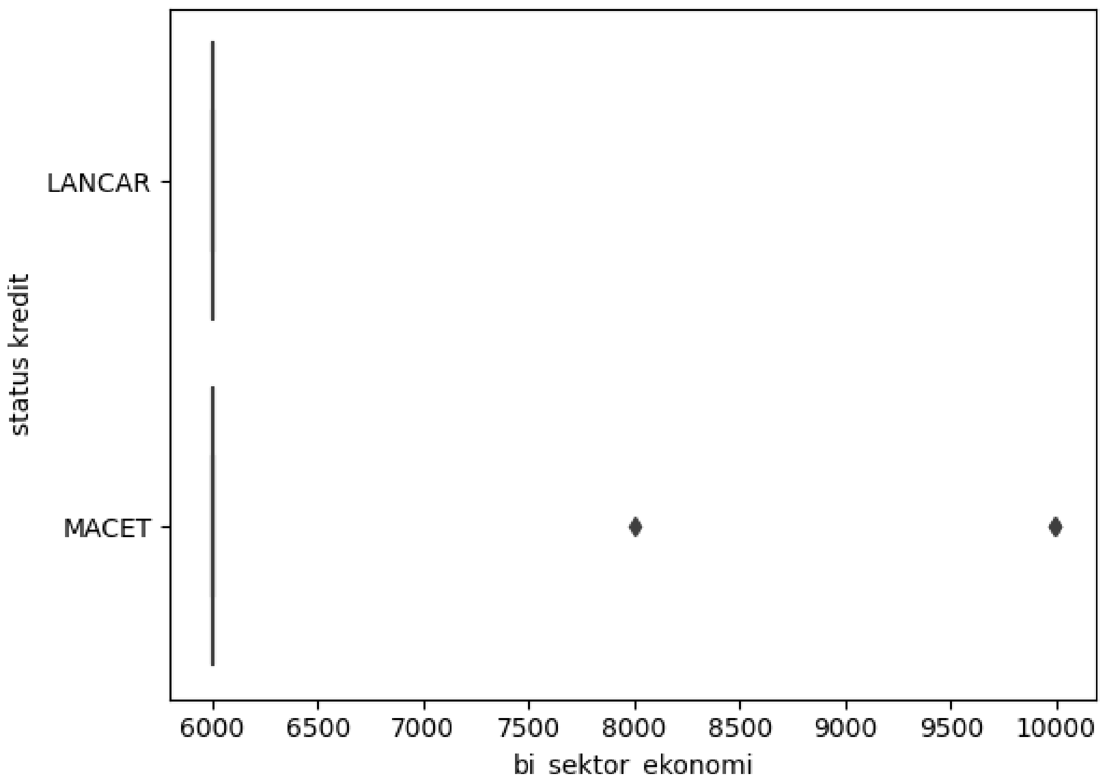

LANCAR
MACET
status kredit
0.0
0.2
0.4
0.6
0.8
1.0
1e8
jml_pinjaman


LANCAR
MACET
status kredit
0
100
200
300
400
500
600
700
jkw

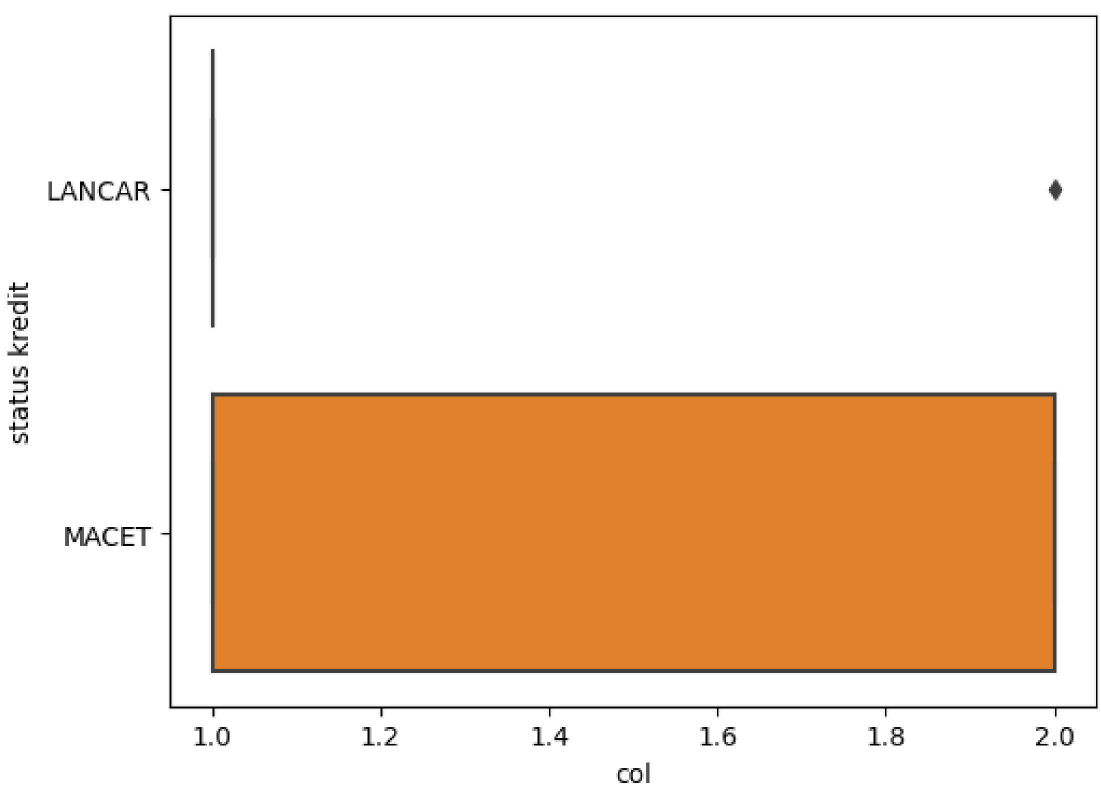

status kredit
LANCAR
MACET
0.0
0.2
0.4
0.6
0.8
1.0
1e7
jml_angsuran_per_bulan


status kredit
LANCAR
MACET
96
98
100
102
104
type_pinjaman

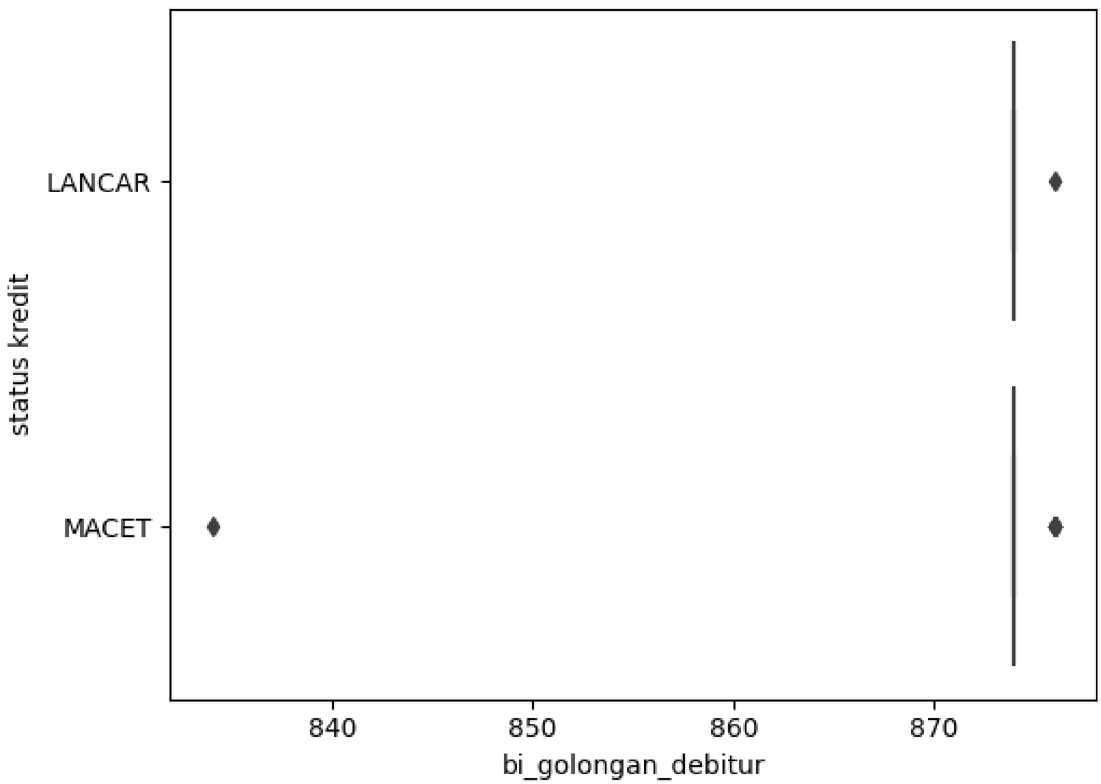

LANCAR
MACET
status kredit
301.0
301.5
302.0
302.5
303.0
303.5
304.0
304.5
305.0
jenis_pinjaman

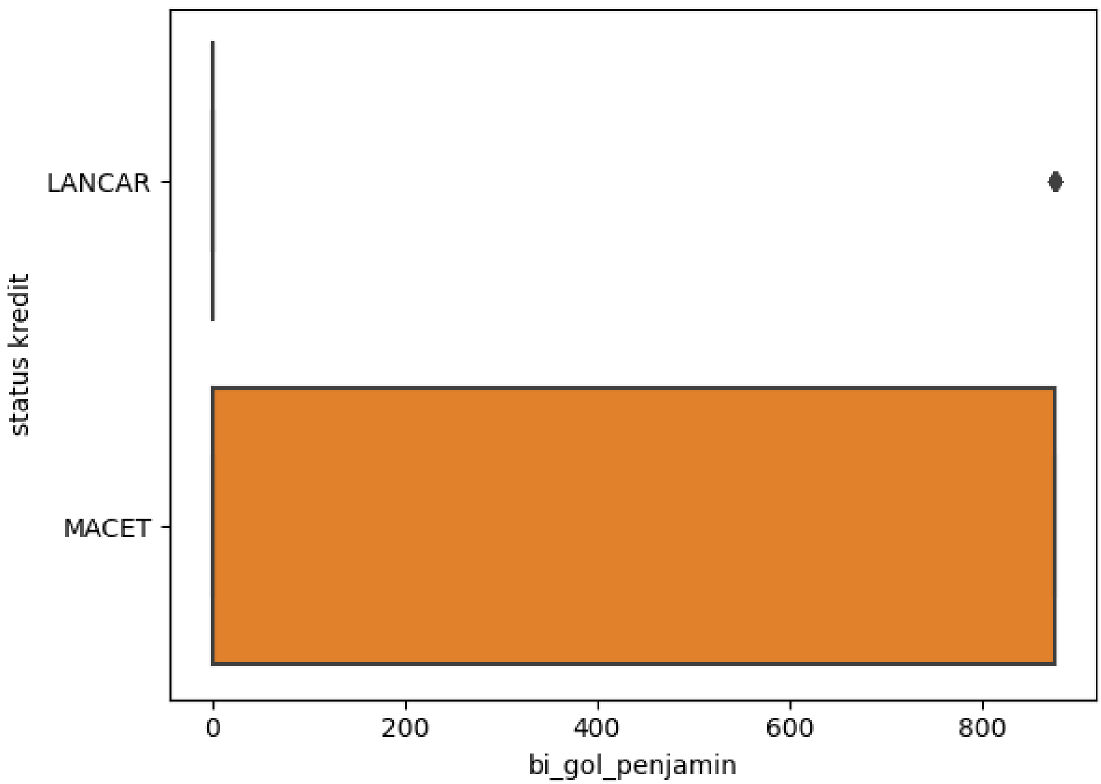


status kredit
LANCAR
MACET
6000
6500
7000
7500
8000
8500
9000
9500
10000
bi_sektor_ekonomi

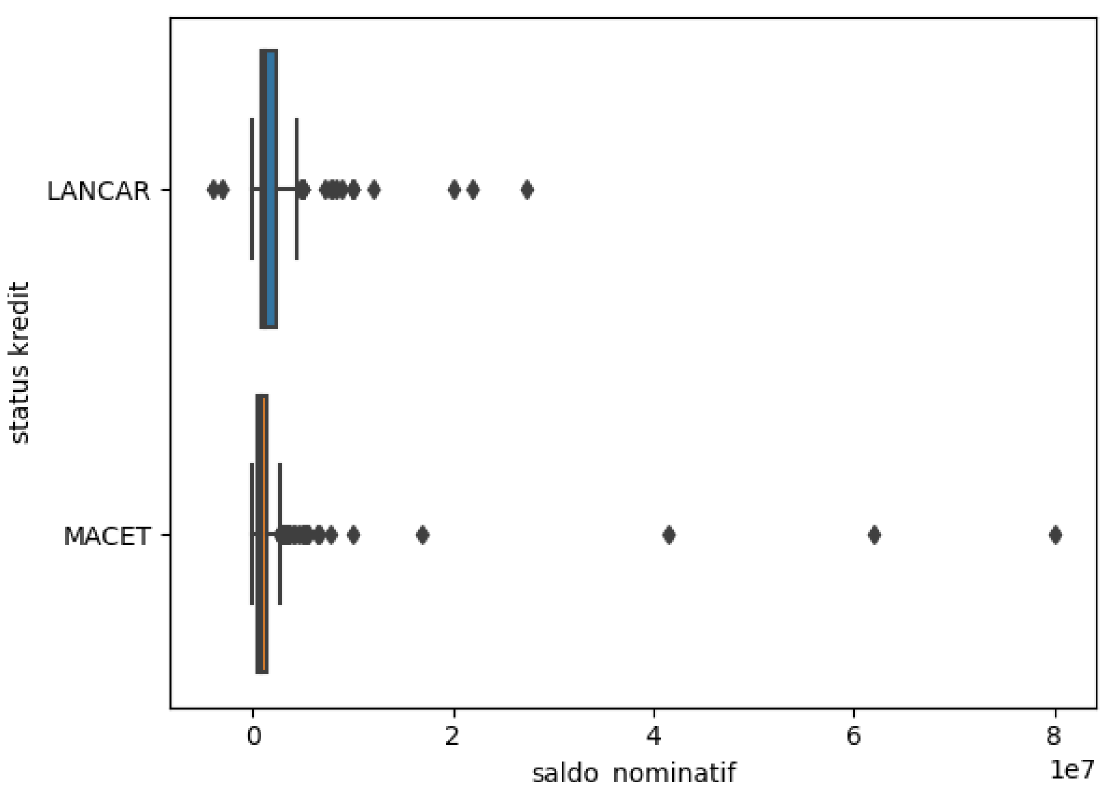


status kredit
LANCAR
MACET
1.0
1.2
1.4
1.6
1.8
2.0
col

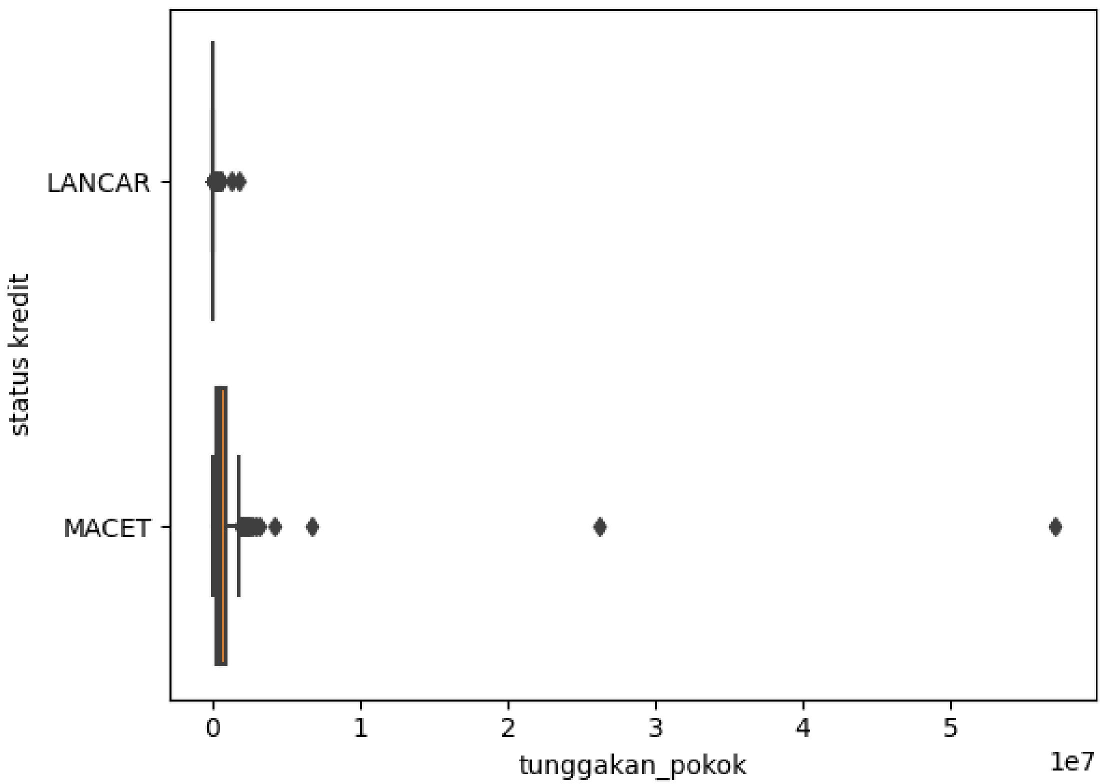

LANCAR
MACET
status kredit
840
850
860
870
bi_golongan_debitur

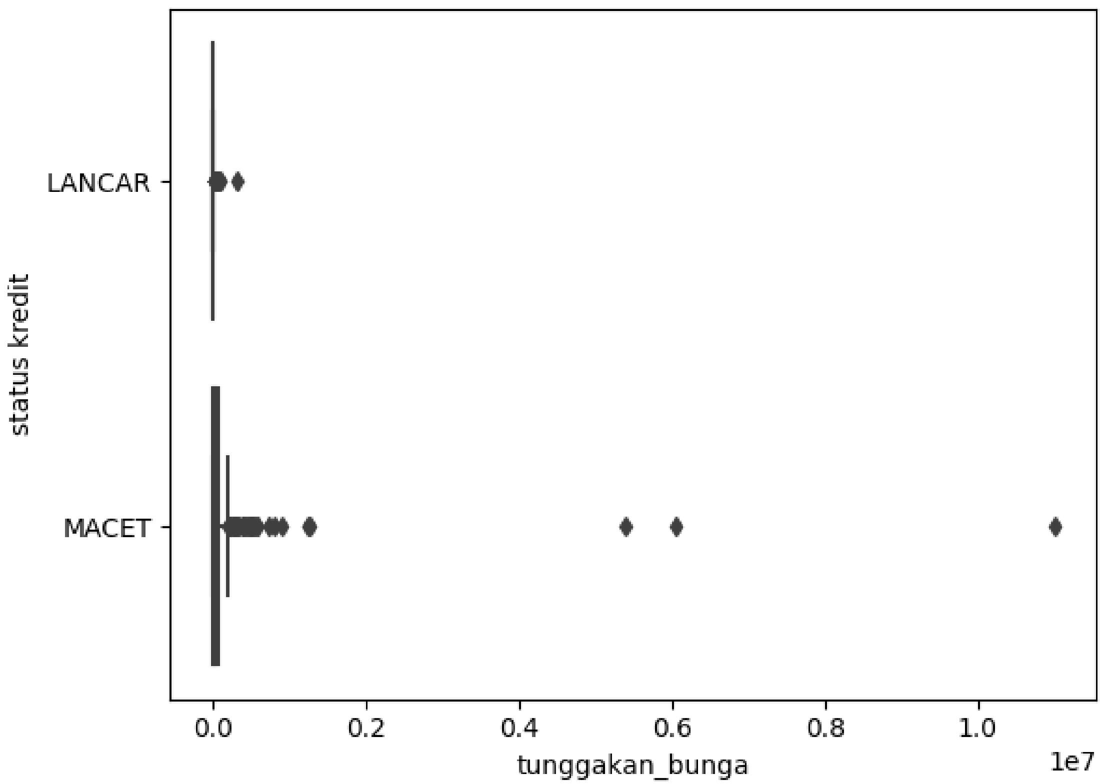


LANCAR
MACET
status kredit
0
200
400
600
800
bi_gol_penjamin



LANCAR
MACET
status kredit
0
2
4
6
8
1e7
saldo_nominatif

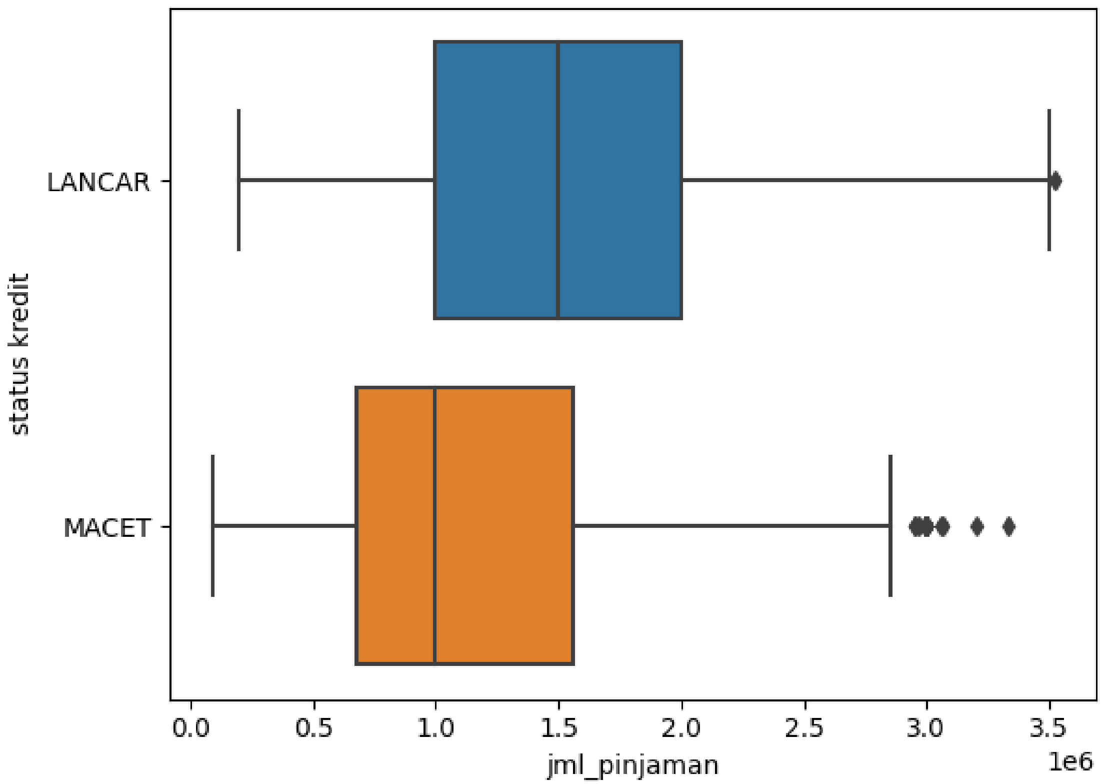

LANCAR
MACET
status kredit
0
1
2
3
4
5
1e7
tunggakan_pokok

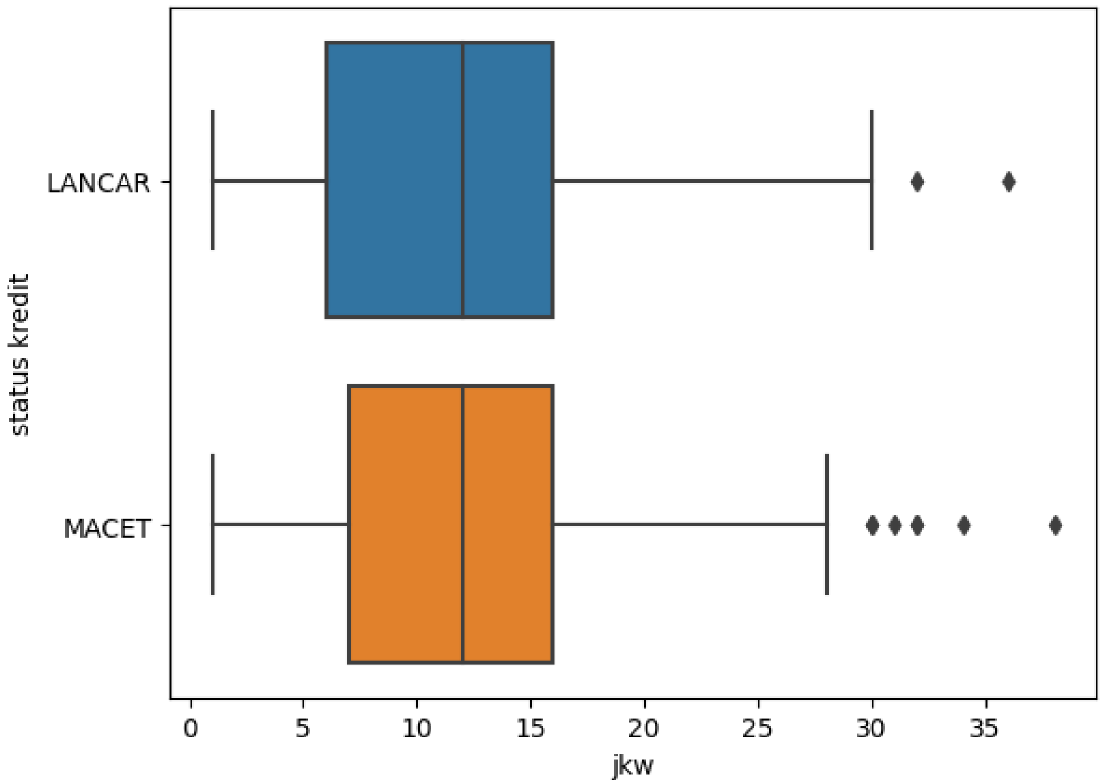


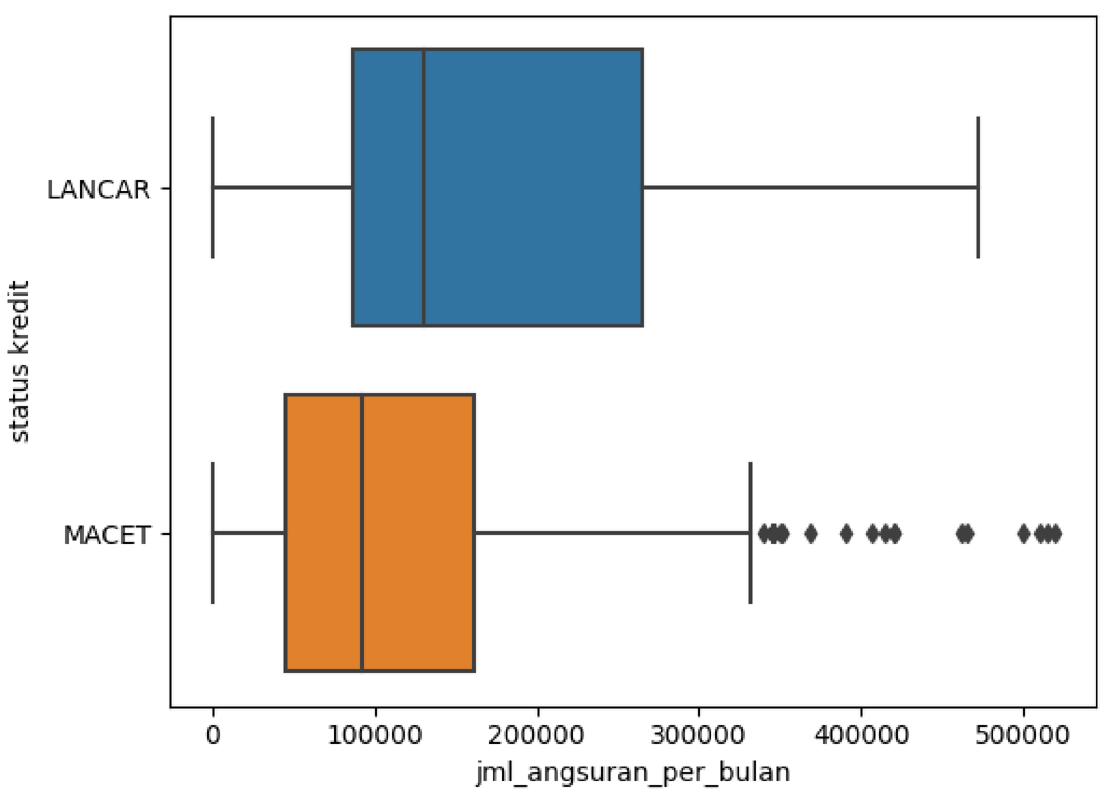
Bisa dilihat bahwasannya ada beberapa data yang sebenarnya bisa kita hapus

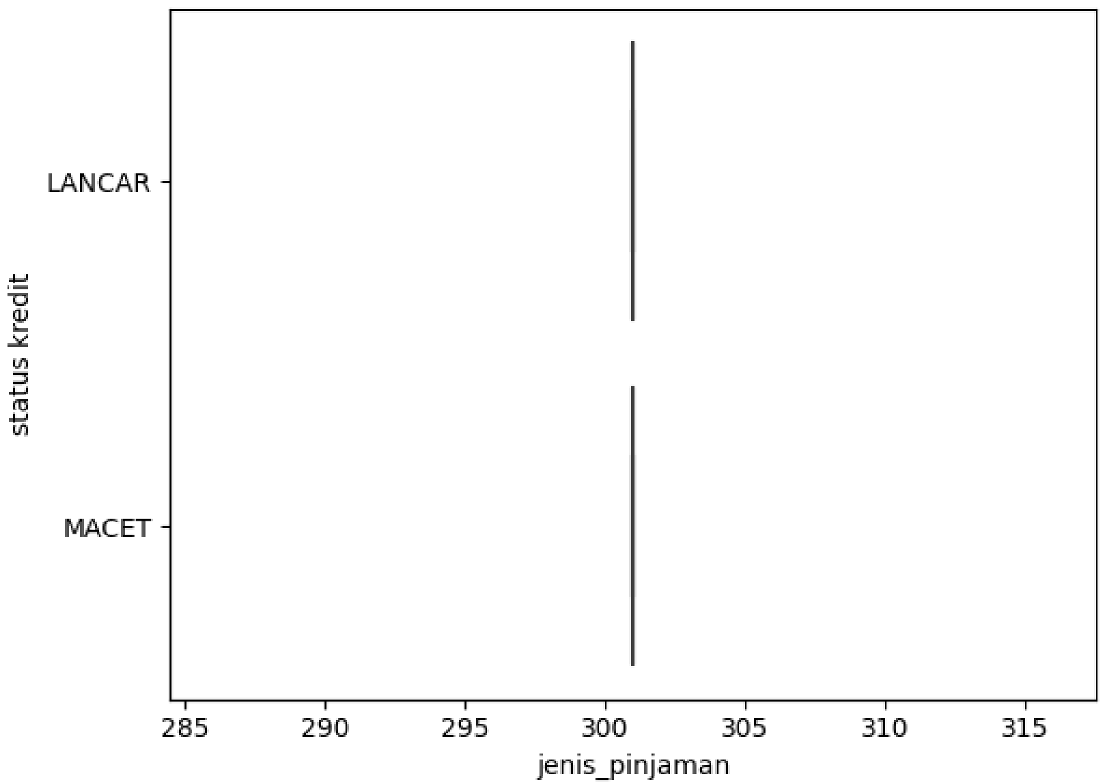
[19]:
```
df_outlier_cleaned = df_miss_val_drop.copy()

# Tentukan kolom nya
columns_outlier = ['jml_pinjaman',
                   'jkw',
                   'jml_angsuran_per_bulan',
                   'jenis_pinjaman',
                   'saldo_nominatif',
                   'tunggakan_pokok',
                   'tunggakan_bunga']

for col in columns_outlier:
  Q1 = df_outlier_cleaned[col].quantile(0.25)
  Q3 = df_outlier_cleaned[col].quantile(0.75)
  IQR = Q3 - Q1  # IQR is interquartile range.

  # Menggunakan operator bitwise & untuk menggabungkan kondisi
  DfNoOutliers = df_outlier_cleaned[~((df_outlier_cleaned[col] < Q1 - 1.5 *␣
 ↪IQR) | (df_outlier_cleaned[col] > Q3 + 1.5 * IQR))]
```

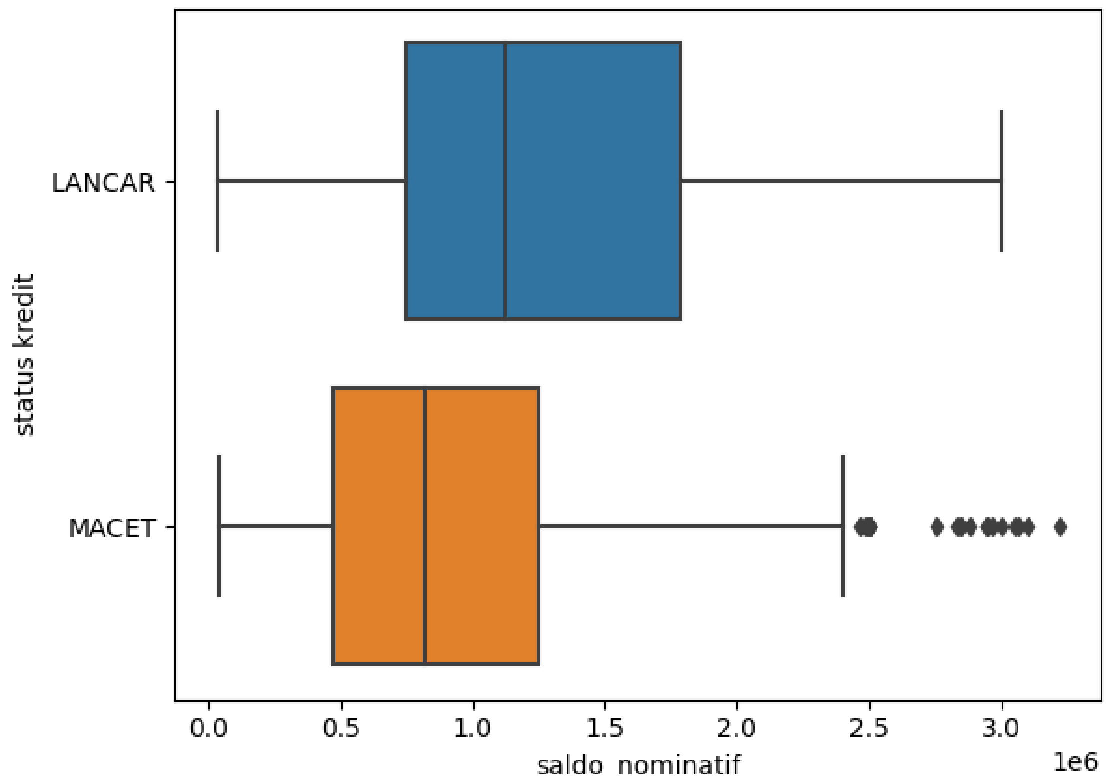
```python
# Memasukkan perintah plot ke dalam loop
p = sns.boxplot(x=col, y="status kredit", data=DfNoOutliers)
plt.show()
```

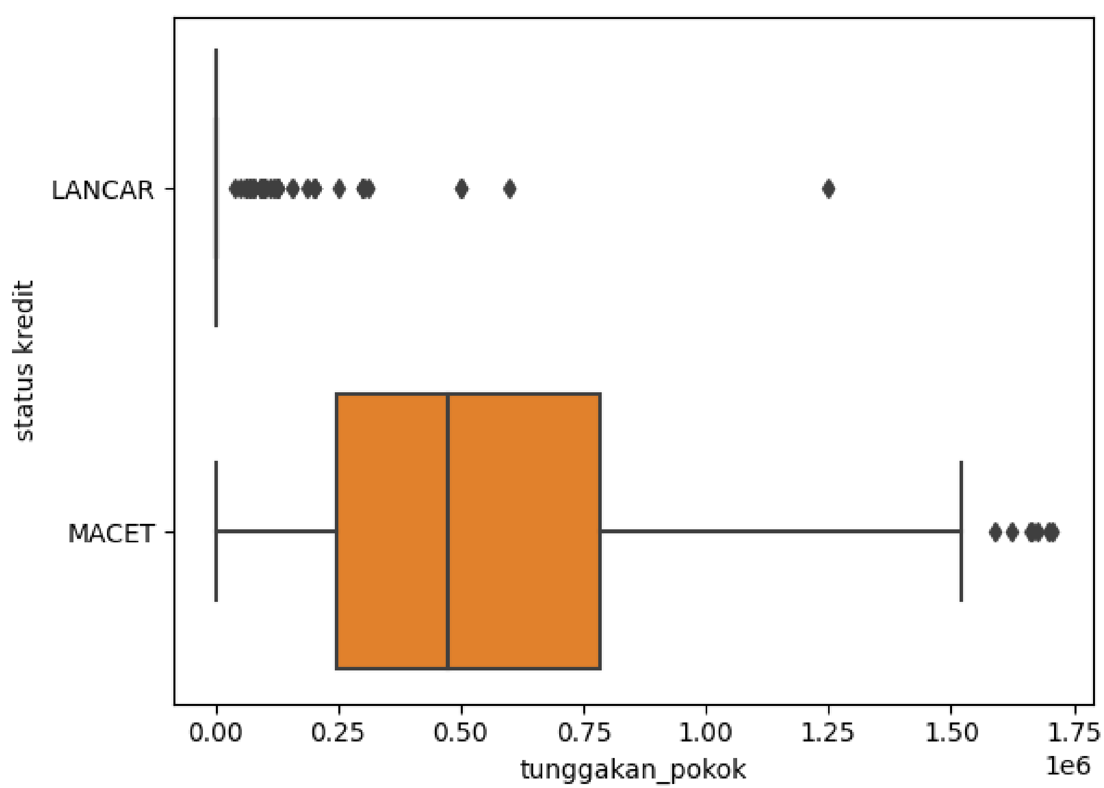


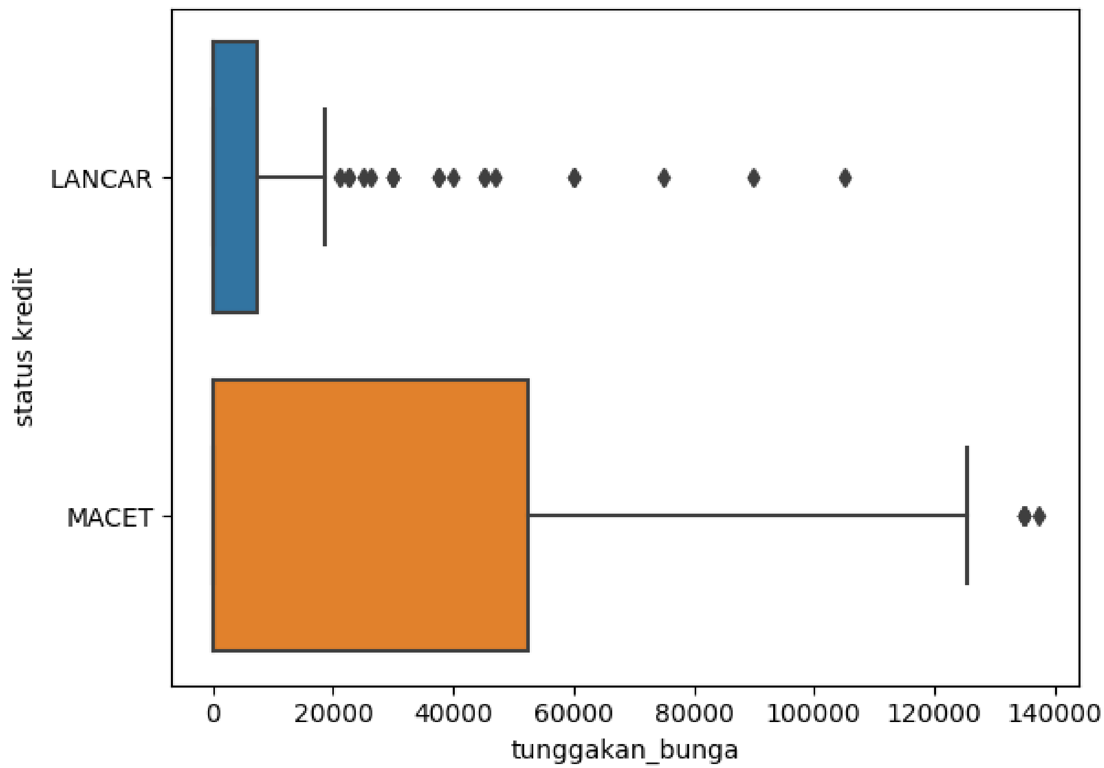

LANCAR
MACET
status kredit
0
5
10
15
20
25
30
35
jkw


LANCAR
MACET
status kredit
0
100000
200000
300000
400000
500000
jml_angsuran_per_bulan


LANCAR
MACET
status kredit
285
290
295
300
305
310
315
jenis_pinjaman


LANCAR
MACET
status kredit
0.0
0.5
1.0
1.5
2.0
2.5
3.0
saldo_nominatif
1e6


LANCAR
MACET
status kredit
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
tunggakan_pokok
1e6

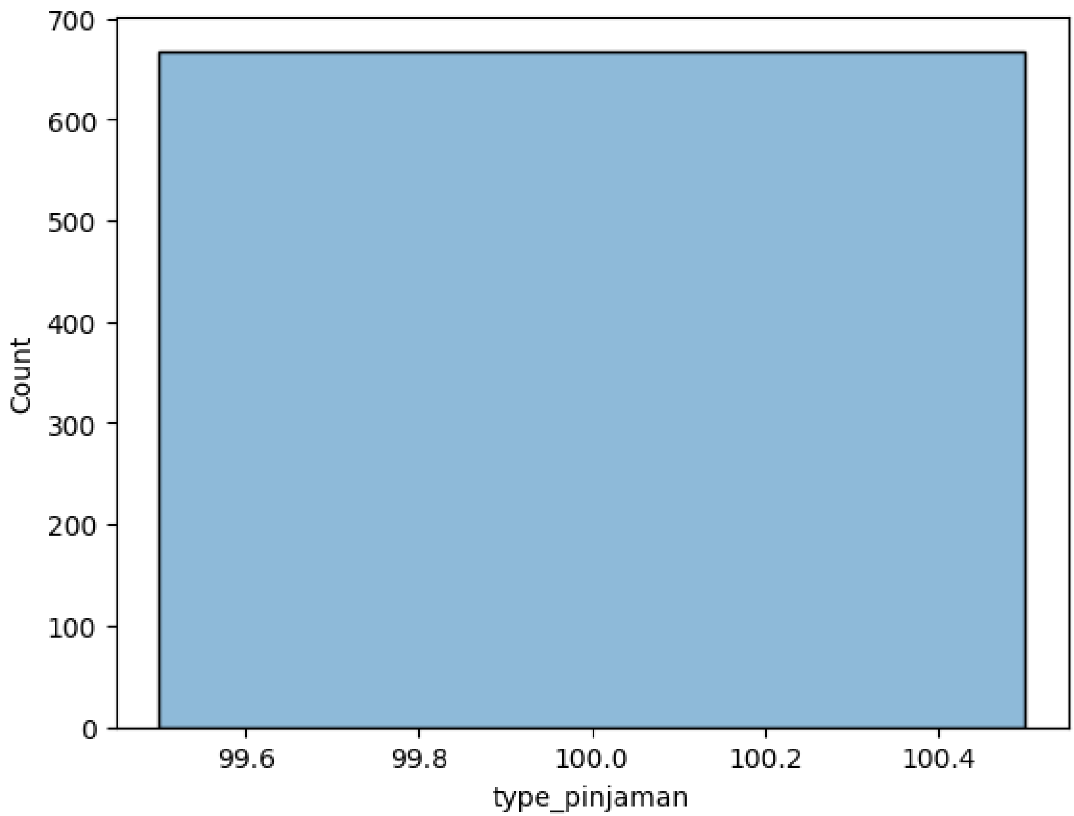


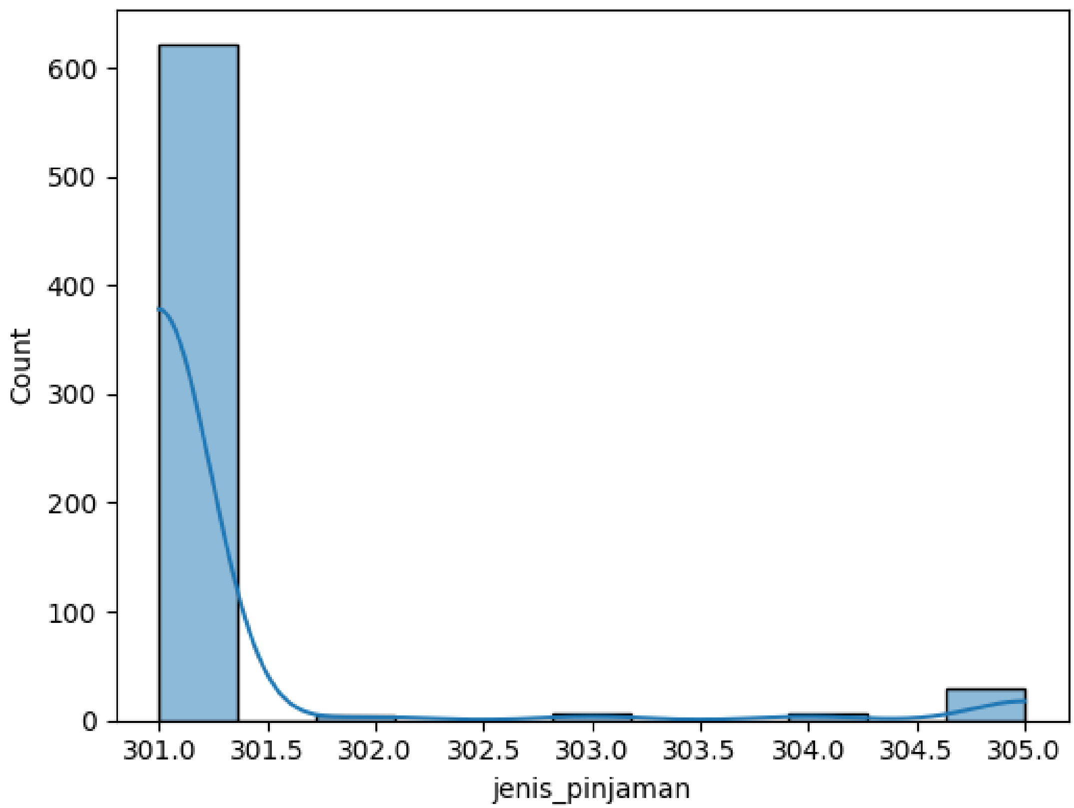
[20]:
```
DfNoOutliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 668 entries, 0 to 765
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   nama_nasabah           668 non-null    category
 1   jenis_kelamin          668 non-null    category
 2   umur                   651 non-null    float64
 3   jml_pinjaman           614 non-null    float64
 4   jkw                    668 non-null    float64
 5   jml_angsuran_per_bulan 379 non-null    float64
 6   type_pinjaman          668 non-null    int64
 7   jenis_pinjaman         668 non-null    int64
 8   bi_sektor_ekonomi      668 non-null    float64
 9   col                    668 non-null    int64
 10  bi_golongan_debitur    668 non-null    int64
 11  bi_gol_penjamin        668 non-null    int64
 12  saldo_nominatif        544 non-null    float64
```

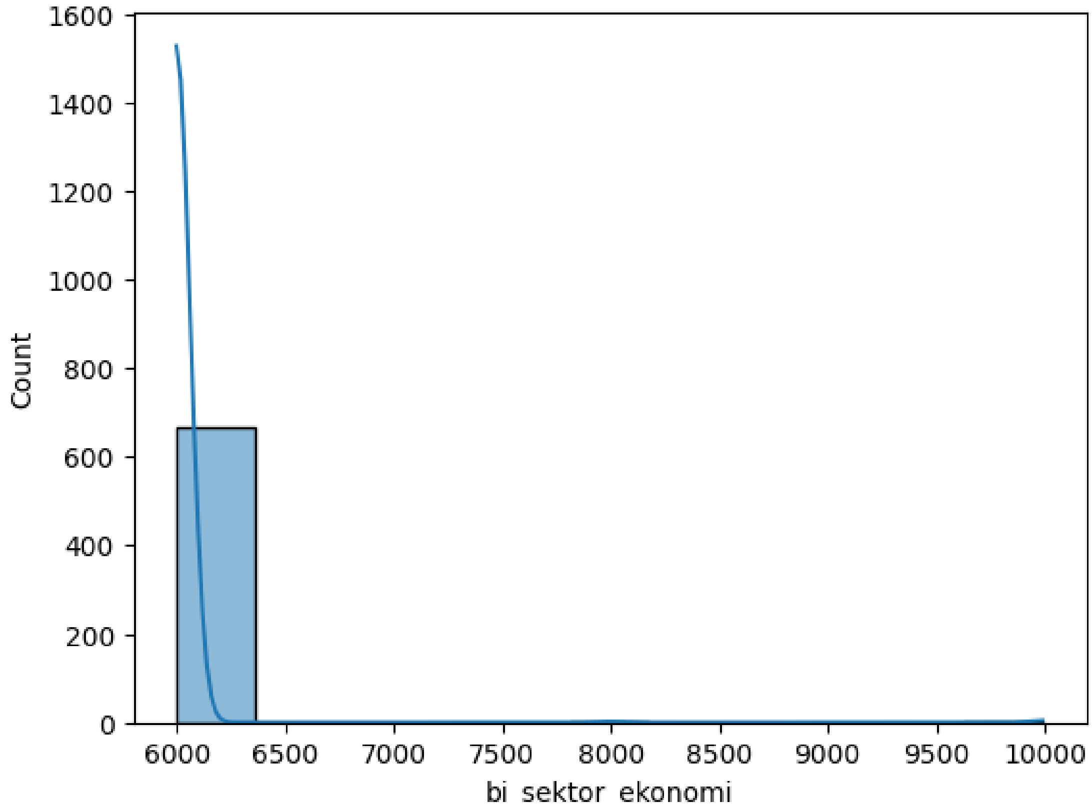
```
 13  tunggakan_pokok         487 non-null    float64
 14  tunggakan_bunga         668 non-null    float64
 15  status kredit           668 non-null    category
dtypes: category(3), float64(8), int64(5)
memory usage: 98.1 KB
```

[21]:
```
columns_categ = ['type_pinjaman',
                'jenis_pinjaman',
                'bi_sektor_ekonomi',
                'col',
                'bi_golongan_debitur',
                'bi_gol_penjamin']

for col in columns_categ:
    sns.histplot(data=DfNoOutliers, x=col, kde=True)
    plt.show()
```

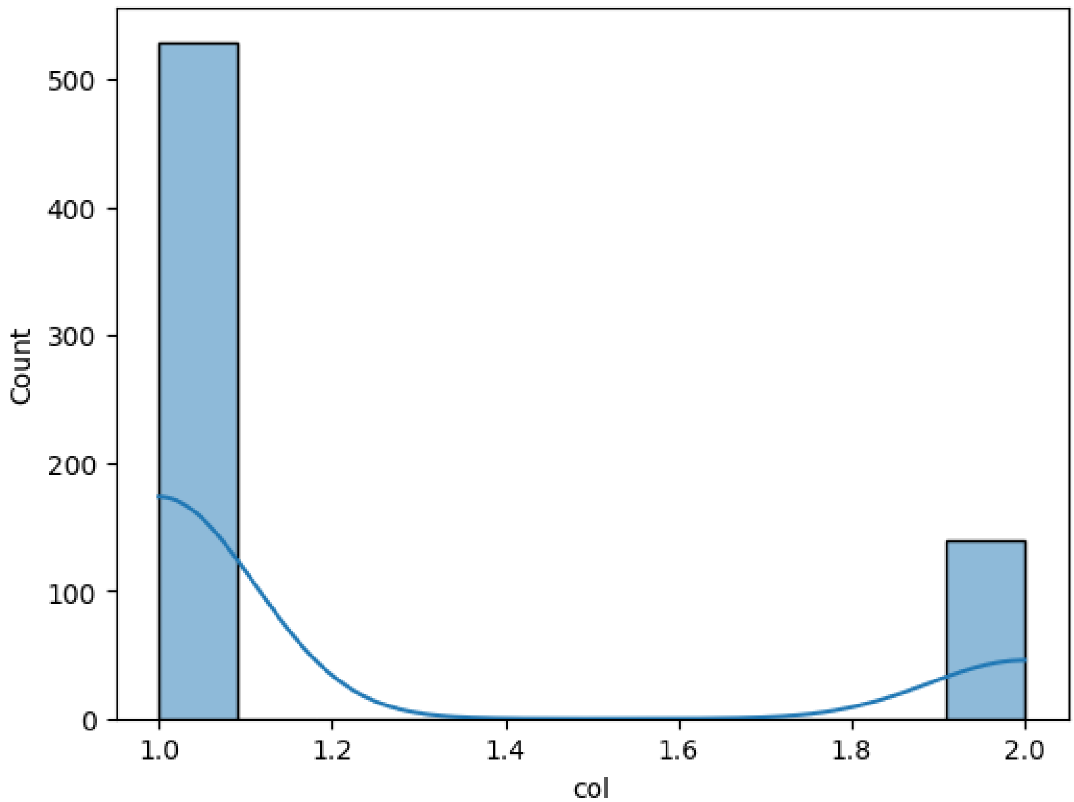


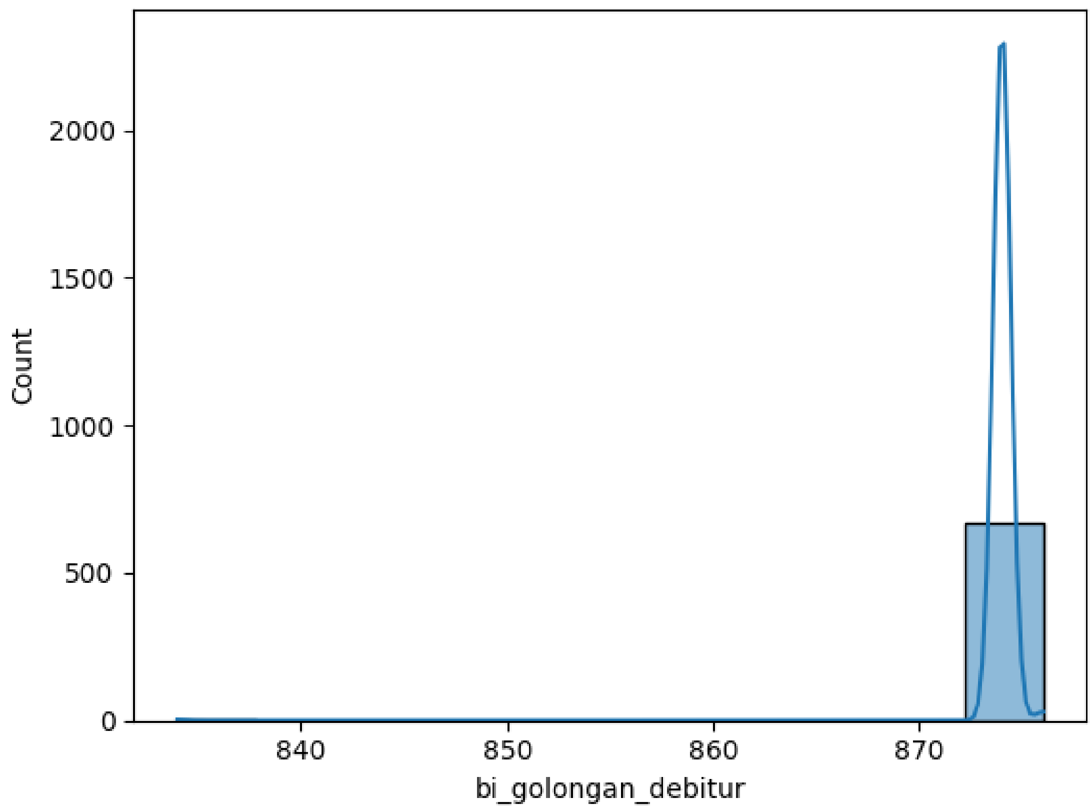


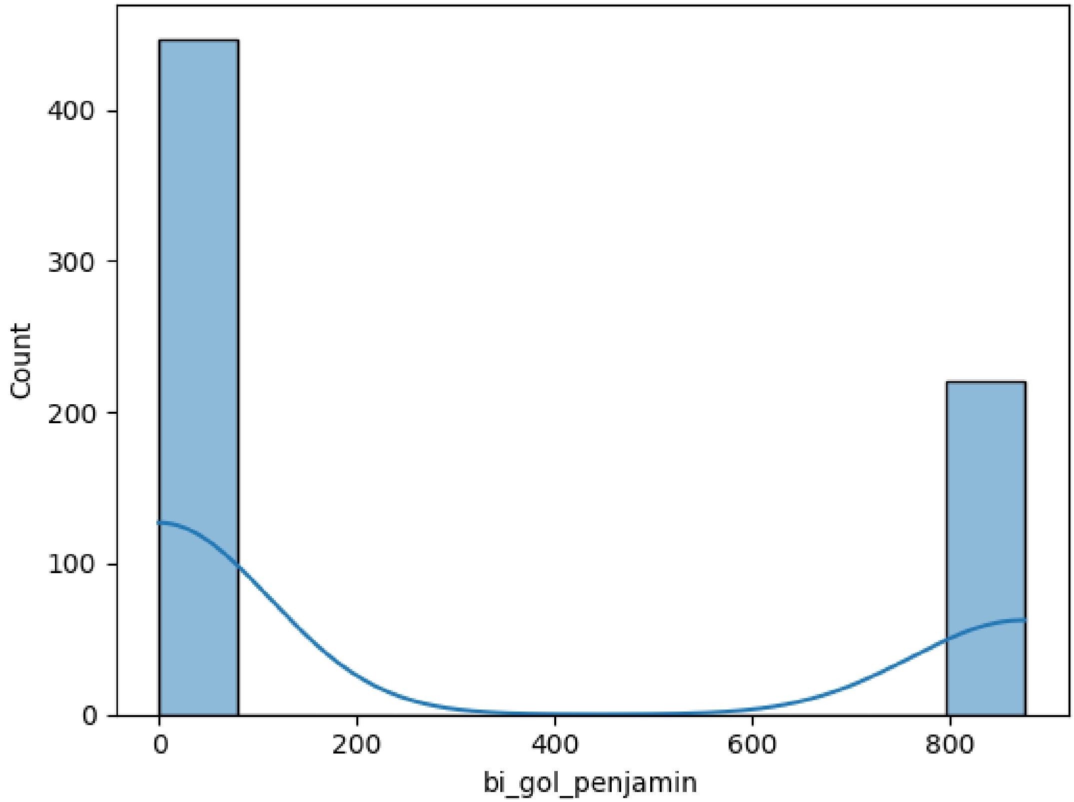
600
500
400
300
200
100
0
Count
301.0
301.5
302.0
302.5
303.0
303.5
304.0
304.5
305.0
jenis_pinjaman



1600
1400
1200
1000
800
600
400
200
0
Count
6000
6500
7000
7500
8000
8500
9000
9500
10000
bi_sektor_ekonomi


Count
col
0
100
200
300
400
500
1.0
1.2
1.4
1.6
1.8
2.0


2000
1500
1000
500
0
Count
840
850
860
870
bi_golongan_debitur



6 Drop Fitur yang kurang berguna

[22]:
```
column_to_remove = 'bi_gol_penjamin'

if column_to_remove in columns_categ:
    columns_categ.remove(column_to_remove)


DfNoOutliers = DfNoOutliers.drop(columns_categ, axis=1)
DfNoOutliers = DfNoOutliers.drop('nama_nasabah', axis=1)

DfNoOutliers
```

[22]:

	jenis_kelamin	umur	jml_pinjaman	jkw	jml_angsuran_per_bulan	\
0	P	40.0	345000.0	1.0	345000.0	
1	L	31.0	350000.0	7.0	55716.0	
4	P	34.0	3055499.0	8.0	NaN	
7	L	27.0	4435001.0	8.0	671098.0	
9	L	49.0	1443750.0	15.0	107800.0	
..	…	…	…	…	…	

```
760             P  24.0     1500000.0  16.0                 105000.0
761             L  38.0     1000000.0  16.0                  70000.0
762             P  36.0     1000000.0  12.0                      NaN
764             P  31.0     1312500.0   7.0                 198750.0
765             P  36.0     2000000.0   4.0                 550000.0

     bi_gol_penjamin  saldo_nominatif  tunggakan_pokok  tunggakan_bunga  \
0                875         345000.0         345000.0              0.0
1                875         390000.0         111428.0              0.0
4                875        3055499.0              NaN              0.0
7                875        4435001.0              0.0              0.0
9                875        1617000.0        1078000.0              0.0
..               …                …                …                …
760                0         700000.0         700000.0          90000.0
761                0         812500.0         812500.0          97500.0
762                0         429000.0         429000.0          45000.0
764                0        1312500.0        1312500.0          78750.0
765                0        1000000.0        1000000.0         100000.0

    status kredit
0           MACET
1           MACET
4           MACET
7          LANCAR
9           MACET
..            …
760         MACET
761         MACET
762         MACET
764         MACET
765         MACET

[668 rows x 10 columns]
```

[23]:

```
for col in DfNoOutliers.columns:
    print(col,DfNoOutliers[col].unique())
```

```
jenis_kelamin ['P', 'L']
Categories (2, object): ['L', 'P']
umur [40. 31. 34. 27. 49. 42. 26. 55. 41. 35. nan 39. 50. 57. 36. 58. 43. 44.
 52. 46. 28. 48. 45. 30. 67. 32. 38. 37. 25. 29. 21. 33. 23. 19. 53. 47.
 51. 54. 61. 24. 56. 68. 60. 76. 22. 80. 65. 64.]
jml_pinjaman [  345000.   350000.  3055499.  4435001.  1443750.  3066000.
4071669.
   840000.       nan  2000000.  1000000.  4000000.   775811.   860000.
   270000.   583335.   506835.  1500000.   185169.   486000.   666667.
    90000. 10550000. 28950000.   535835.  6933332.  1116263.   610000.
```

```
  1820000.  1458339.   495000.  2040000.  1430666.   187500.  1060000.
  1400000.   145000.   124000.   603750.   200000.  1086000.   900000.
  1590000.  1200000.   875000.  1406250.  5000000.  1425000. 10000000.
   675000.  2500000.  3000000.  7000000.   800000.   500000.  3500000.
  2350000.   750000.   530000.   147667.  1600000.  2850000.   680000.
   496668.   930000.   555000.   880000.   320000.  2540000.   593334.
  2380000.   920000.   333334.   725000.   136900.  1567500.   383750.
   561250.   511252.  1670000.   126670.   840278.  1910000.  1125000.
   845000.   975000.  2450000.  1655000.   525000.  2330402.  1499333.
  1190000.   666668.   567501.   215835.  1559581.   932500.   650000.
  2950001.  1340000.  1739996.  1499995.  1335330.   566668.   373000.
  2057588.  1260000.  2965000.   665000.   550000.   225000. 40000000.
   690000.  1944442.  1864998.   772500.  1095000.   711250.   315000.
  2170000.   264284.  1013932.   405000.   425000.   275000.   415002.
   788120.   584250.  1575000.  1960000.  1830000.   247500.   468750.
   893500.    93750.   700000. 20000000. 22000000.   131670.  1385000.
  1042000.   780834.   645000.   451668.   461671.   590834.  1608750.
  2062500.   682500.  4600000.   154169.  1178000.   865000.  1031250.
  3771000.  1766001.   267669.   769167.   899550.  1395000.   190000.
   488894.   684800. 16975000. 38500000.  5500000.  1950000.  1660000.
   945000.   430000.   780000.   303750.   635000.   812500.  2665000.
   713500.  1741750.  1137250.  1300000.  1066260.  2083332. 25000000.
  6000000.   245166.   340000.   330000.  1140000.   940000.   116663.
  3520000.  4520000.  1275000.  1420000.  2312000.  3200000.   950000.
   239668.   400000.   581335.  1320000.  8000000.   175000.   310000.
  1380000.   695000.   406667.  2725000.   783500.   850000.  1123750.
  1090000.   600000.  1710000.  1750000.   336461.  2333332.   670000.
   390000.   447500.  2950000.   538000.   220000.  2300000.   896000.
   100000.  1030000.  2255000.   630000.   203000.   300000.   472500.
   582500.   499999.  2480000.   921260.  1020000.   195000.  1130000.
   521000.   347000.   297500.   218750.   812750.  1312500.]
jkw [  1.   7.   8.  15.  10.  20.   4.  12.  24.   6.   5. 141.   2.   3.
  53.  17.  31.  21.  19.  13.  27. 100.  40.  16. 120.  80.  46.  57.
  28.  50.   9.  26.  11.  18.  63.  60.  36.  67.  30.  38. 679.  64.
  76.  32.  25.  14.  47.  72.  34.]
jml_angsuran_per_bulan [3.45000e+05 5.57160e+04         nan 6.71098e+05
1.07800e+05 3.51670e+05
 6.00000e+04 1.15000e+05 4.50000e+04 6.07500e+04 1.80000e+04 1.82000e+05
 4.37250e+04 2.29500e+05 4.55000e+04 2.90000e+04 2.48000e+04 1.20750e+05
 1.90000e+04 5.43000e+04 3.45000e+04 1.05000e+05 7.75000e+05 8.62500e+04
 1.55000e+06 1.72500e+04 1.15000e+04 3.25000e+05 1.40000e+05 2.30000e+04
 1.47500e+05 2.10000e+04 1.16000e+04 1.47500e+04 2.65000e+06 6.75000e+04
 1.54500e+06 5.42500e+05 6.50000e+05 5.75000e+05 5.90000e+05 2.33433e+05
 7.47500e+04 6.62500e+04 1.75056e+05 1.00000e+05 1.60056e+05 5.39450e+04
 1.60000e+05 2.80250e+05 4.38890e+04 3.22500e+04 7.50000e+04 7.80000e+04
 2.95000e+05 4.42000e+04 4.15000e+05 9.88890e+04 1.84000e+05 1.83750e+05
 7.56890e+04 0.00000e+00 1.81500e+04 4.72000e+04 2.45000e+05 4.96600e+04
 6.77500e+04 5.00000e+04 3.25646e+05 7.03720e+04 7.46000e+04 7.87500e+04
```

```
 2.70770e+04 3.90000e+05 3.31250e+05 2.30000e+05 2.10000e+05 4.20000e+05
 1.75000e+05 2.80000e+05 1.36250e+05 1.28750e+06 2.65000e+05 2.87500e+05
 2.83250e+04 1.35940e+04 5.00000e+05 1.03611e+05 3.86250e+04 3.43750e+04
 5.50000e+04 4.62000e+05 7.03130e+04 2.72500e+05 9.30000e+04 2.36000e+05
 2.52000e+04 4.72000e+05 4.60000e+05 9.20000e+05 7.00000e+04 1.72500e+05
 2.60000e+05 5.30000e+04 3.10000e+05 1.65000e+06 7.93750e+04 3.41250e+04
 4.32500e+04 1.15500e+05 2.33289e+05 5.00630e+04 5.00000e+03 3.83200e+04
 2.50000e+04 5.18750e+05 1.21875e+05 4.72500e+04 1.27000e+05 7.10030e+04
 5.28570e+04 8.64060e+04 7.76400e+04 5.75000e+04 3.59375e+05 5.15000e+05
 1.40000e+04 5.45000e+05 1.55000e+05 1.18750e+06 1.30000e+05 6.60000e+05
 5.25000e+05 1.32000e+06 1.06000e+06 2.06000e+06 4.08610e+04 6.05000e+05
 4.25000e+04 4.12500e+04 2.00000e+04 1.03500e+07 9.50000e+04 1.25000e+05
 2.50000e+05 2.33330e+04 1.00000e+06 1.27500e+05 1.10000e+05 4.62400e+04
 3.20000e+05 2.37500e+05 2.00000e+05 2.64000e+04 4.37500e+04 3.88000e+04
 1.38000e+05 6.03750e+04 7.85000e+04 6.80000e+04 8.40000e+05 1.42500e+05
 1.03000e+05 2.65000e+04 1.13542e+05 2.72500e+04 1.09000e+05 6.90000e+04
 1.90000e+05 6.70000e+04 4.80000e+04 4.47500e+04 3.68750e+05 4.75000e+04
 2.25500e+05 7.17500e+04 4.35000e+04 1.65000e+05 3.50000e+05 3.50000e+04
 7.84000e+03 1.54000e+05 2.48000e+05 9.21260e+04 1.21325e+05 2.00850e+04
 1.13000e+05 8.64380e+04 6.45000e+04 9.36500e+04 1.68000e+05 1.98750e+05
 5.50000e+05]
bi_gol_penjamin [875   0 800 874 835]
saldo_nominatif [  345000.   390000.  3055499.  4435001.  1617000.  3066000.
3671669.
   240000.       nan  1800000.   810000.   742662.   775811.   708857.
   270000.   635835.   551835.   200169.   486000.   693667.    90000.
   332500.  1090000. 27399322.   535835.  6433332.   945263.   640000.
  1585000.   524700.  5500000.  2754000.  1430666.   227500.   947000.
   915000.   145000.   124000.   603750.    38000.  1086000.   900000.
  1590000.  1000000.   875000.   690000.  1406250.   311500.   209500.
  2000000.  1425000.   129000.   111750.   450000.  3750000.   625000.
   300000.   963500.   310000.  1457500.   860250.  7250000.   750000.
  1244000.  1100000.  1600000.  2250000.   850000.  1050000.   732500.
  1125000.  1068750.   686500.   592000.  1045500.   666000.   665000.
  1398000.  2500000.  1200000.  1500000.   248000.  3500000.  1218750.
 10000000.   833000.   917500.  5000000.  1350000.  1312500.  3000000.
  4000000.  2350000.   598000.   530000.   147667.   500000.  2850000.
   604000.   334668.   258000.   600000.  3304000.   221000.   320000.
  2490000.   593334.   937500.   363334.   725000.   136900.  1837500.
   343750.   529821. -3000000. -4000000.  1770000.   131670.   726000.
  1962606.   924000.  2400000.   971250.  1681316.   510250.  1975402.
  1619333.  1355000.   666668.   347501.   165835.  1859581.   254996.
   226667.   650000.  2950001.  1025000.  1539996.  1524995.   895330.
   576668.   273000.  2057588.  2965000.  1250000.  2062500.  1875000.
  2750000.  1375000.  4750000.   566500.   108752.  1644442.  1864998.
   772500.   315000.  1240000.   264284.  1067512.   412500.   425000.
   275000.   445002.   788120.   334250.   860000.  1175000.  1960000.
  1830000.   247500.   375000.  1036000.   215000.   616000.    93750.
```

```
  885000.   250000.   400000.   800000.   437500.   812500.   952500.
  656250.  3220000.  1235000.   823500.  1650000.  3666000.  1424250.
  700000. 20000000. 22000000.  1587500.  2010000.  1066204.  1380000.
  848334.   451668.   216671.   590834.  1458750.  2310004.   603500.
  562500.   154169.  1178000.   765000.   843750.  1031250.  3771000.
 1766001.  4600000.   290169.   726668.   744167.   934550.  2406250.
 1235689.   190000.   494554.   566400. 16975000.  7823334.  5206250.
 1940000.  1660000.   945000.   430000.   520000.    30750.   535000.
  655000.  1515000.   426700.   526750.   969500.   666750.   995500.
  442500.  1320000.  1947500.   519750.  1345000.   969750.   332000.
  816500.   720000.  1321500.  2495000.   749500.   350000.  1750000.
 1300000.  1066260.  2114332. 12000000.  3600000.   512500.  9000000.
   95166.  8334000.   330000.    75000.  1140000.  2291000.   940000.
  116663.  2920000.  4320000.  1275000.  1420000.   880000.  2312000.
 2880000.   176668.   581335.   499000.  1280000.  1165000.  8000000.
  175000.   388000.  2124000.  1150000.   641000.   288000.  6700000.
  890000.   406667.  2465000.   783500.  1123750.   460000.   100000.
  167000.   787000.  1144000.   554000.  1520000.  1678000.  1115000.
  608500.   336461.  2333332.   427500.  1450000.   399250.   220000.
  711000.   882500.   594000.   352500.   115000.  1030000.  1055000.
  563750.   132000.   472500.   214000.   125000.   582500.   550000.
  178000.   235440.   935000.  1111250.   595000.   395000.   434000.
  848750.   782500.   516000.   589999.  1988000.   771260.   673000.
  200850.   930000.   481000.   347000.   297500.   218750.   812750.
  468750.   475000.   136000.   753500.   810250.   612000.   312500.
  692750.   823000.   894000.   218250.   572500.   429000.]
tunggakan_pokok [ 345000.  111428.      nan       0. 1078000.  613200.  240000.
700000.
  708857.   90000.  200169.  486000.   36000.  437250. 2295000.  403500.
   58000.   49600.  172500.   38000.  651600.  600000.  330000.  400000.
  937500.  311500.  209500.  750000.  129000.  111750.  450000.  225000.
  918500.  260000.  305250.  494000.  100000.   75000.  440000.  250000.
  187500.  102000.  235500.   93750.  108452.   44114.  391666.  598000.
  530000.  147667.  350112.  200000.  320112.  107890.  475000.  246890.
  258000.  156000. 2714000.  221000.  320000. 2490000.  593334.  136000.
  363334.  136900.  735000.  151378.  109725.  181500.  452000.  930000.
  784375.  454225.  677500.  241670.  226667.  500000. 2950001.  227665.
  576668.  570600. 2057588.  577500.   41340.  300000.  156250.  375000.
 1250000.   50000.  283250.  108752. 1000000.  690000.  414444.  386250.
  196875.  533756.  343750.  110000.  370835.   42125.  860000.  584375.
  411111. 1960000.  184762.  732000.  123750.   95000.  236000.  215000.
  210000.  900000.  768750.  625000.  427500.  281250.  820000.  125000.
  325000.  275000.   62500.  793750. 1150000.  706945.  183125.  654375.
  434446.  262250. 1700000.  332500. 3142500. 1766001.  193446.  584689.
  227277.  183200.  840000. 1660000.  472500.  520000.  154000. 1472222.
  467500.  569375.  182500.  825000.  687500.  516750.  695500.  242500.
  360000.  697500.   69750.  312500.  370500.  219750.  562500.  246500.
   70500.  120000. 2000000.   37500.  533130.  362500.   13444.   70000.
```

```
  82500.  142222.   46666. 1147500.  462400.  140000.  264000.  175000.
 388000. 1380000. 2124000.  603750.  641000. 1590000.  288000. 6700000.
1425000.  890000.  406667. 2465000.  783500.  850000. 1123750.  460000.
 167000.  787000.  666000. 1144000.  940000.  554000. 1520000. 1678000.
1115000.  608500.  336461. 2333332.  390000. 1450000.  399250.  220000.
 711000.  882500.  594000. 1125000.  352500.  115000. 1030000. 1055000.
 563750.  132000.  214000.  833000.  582500.  550000.  178000.  235440.
 935000. 1111250.  595000.  395000.  434000.  848750.  782500.  516000.
 589999. 1988000.  771260.  673000.  200850.  875000.  481000.  347000.
 297500.  218750.  812750.  468750.  753500.  810250.  612000.  692750.
 823000.  894000.  218250.  843750.  572500.  812500.  429000. 1312500.]
tunggakan_bunga [     0.  90140. 105000.  80000. 120000.  90000.  60000.  30000.
49500.
  33649. 112500.  46500.  33000.  19500.  18000.  67500.  13709. 137250.
  18750.  38500.  75000.  37500.  47250.  15000.   7500.  11250.  66000.
  45000.  37000.  22500.  17600.  42750.  75200.  33750.  16500.  56250.
  26250.  25000. 109800.  16876.  36000.  27000. 135000.  92250.  78750.
   3512.  21000.   5000.  95000.  46319.  42000. 125466.  52500.  82500.
 116250.  83250.  38984.  54000.  46875.   4500.   9000.  20000. 123750.
  51000. 101250.  69000.  15500.  25500. 118500.  91500.  48000. 100000.
  67000.  22000.  28560.  65500.  97500. 114380.  93000. 105750.  40000.]
status kredit ['MACET', 'LANCAR']
Categories (2, object): ['LANCAR', 'MACET']
```

[24]:
```
DfNoOutliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 668 entries, 0 to 765
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   jenis_kelamin          668 non-null    category
 1   umur                   651 non-null    float64
 2   jml_pinjaman           614 non-null    float64
 3   jkw                    668 non-null    float64
 4   jml_angsuran_per_bulan 379 non-null    float64
 5   bi_gol_penjamin        668 non-null    int64
 6   saldo_nominatif        544 non-null    float64
 7   tunggakan_pokok        487 non-null    float64
 8   tunggakan_bunga        668 non-null    float64
 9   status kredit          668 non-null    category
dtypes: category(2), float64(7), int64(1)
memory usage: 48.5 KB
```

7 Data Encoding

[25]:
```
encode_data = DfNoOutliers.copy()
encode_data['bi_gol_penjamin'] = encode_data['bi_gol_penjamin'].astype('object')
columns = ['bi_gol_penjamin','jenis_kelamin']
transformasi_gol = pd.get_dummies(encode_data['bi_gol_penjamin'],␣
 ↪prefix='bi_gol_penjamin_')
transformasi_kelamin = pd.get_dummies(encode_data['jenis_kelamin'], prefix='')
encode_data = pd.concat([encode_data, transformasi_gol,transformasi_kelamin],␣
 ↪axis = 1)
try:
    encode_data.drop(columns, axis=1, inplace=True)
except Exception as err_:
    print(err_)

print(encode_data.shape)
encode_data.head()
```

```
(668, 15)
```

```
<ipython-input-25-c9f40a9a9d88>:4: FutureWarning: In a future version, the Index
constructor will not infer numeric dtypes when passed object-dtype sequences
(matching Series behavior)
  transformasi_gol = pd.get_dummies(encode_data['bi_gol_penjamin'],
prefix='bi_gol_penjamin_')
```

[25]:
```
   umur  jml_pinjaman   jkw  jml_angsuran_per_bulan  saldo_nominatif  \
0  40.0      345000.0   1.0                345000.0         345000.0
1  31.0      350000.0   7.0                 55716.0         390000.0
4  34.0     3055499.0   8.0                     NaN        3055499.0
7  27.0     4435001.0   8.0                671098.0        4435001.0
9  49.0     1443750.0  15.0                107800.0        1617000.0

   tunggakan_pokok  tunggakan_bunga status kredit  bi_gol_penjamin__0  \
0         345000.0              0.0         MACET                   0
1         111428.0              0.0         MACET                   0
4              NaN              0.0         MACET                   0
7              0.0              0.0        LANCAR                   0
9        1078000.0              0.0         MACET                   0

   bi_gol_penjamin__800  bi_gol_penjamin__835  bi_gol_penjamin__874  \
0                     0                     0                     0
1                     0                     0                     0
4                     0                     0                     0
7                     0                     0                     0
9                     0                     0                     0

   bi_gol_penjamin__875  _L  _P
```

```
0                      1   0   1
1                      1   1   0
4                      1   0   1
7                      1   1   0
9                      1   1   0
```

[28]:

```
# Saving the preprocessed Data for future use/analysis
final_data = encode_data.copy()
final_data.to_csv("/content/drive/MyDrive/Asesmen Data Science/
 ↪Data_PreProcessed.csv", encoding='utf8', index=False)
```

[ ]:

```
```

modelling-data

November 17, 2023

[27]:
```python
import warnings; warnings.simplefilter('ignore')
import pandas as pd, matplotlib.pyplot as plt
import time, numpy as np, seaborn as sns
from sklearn import  tree
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import cross_val_score
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
sns.set(style="ticks", color_codes=True)
"Done"
```

[27]: 'Done'

[8]:
```python
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/Asesmen Data Science/Data_PreProcessed.
 ↪csv')
df.head(10)

N, P = df.shape # Ukuran Data
print('baris = ', N, ', Kolom (jumlah variabel) = ', P)
print("Tipe Variabe df = ", type(df))
df
```

```
baris =  668 , Kolom (jumlah variabel) =  15
Tipe Variabe df =  <class 'pandas.core.frame.DataFrame'>
```

[8]:
```
     umur  jml_pinjaman   jkw  jml_angsuran_per_bulan  saldo_nominatif  \
0    40.0      345000.0   1.0                345000.0         345000.0
1    31.0      350000.0   7.0                 55716.0         390000.0
2    34.0     3055499.0   8.0                     NaN        3055499.0
3    27.0     4435001.0   8.0                671098.0        4435001.0
4    49.0     1443750.0  15.0                107800.0        1617000.0
..    …             …    …                     …                …
```

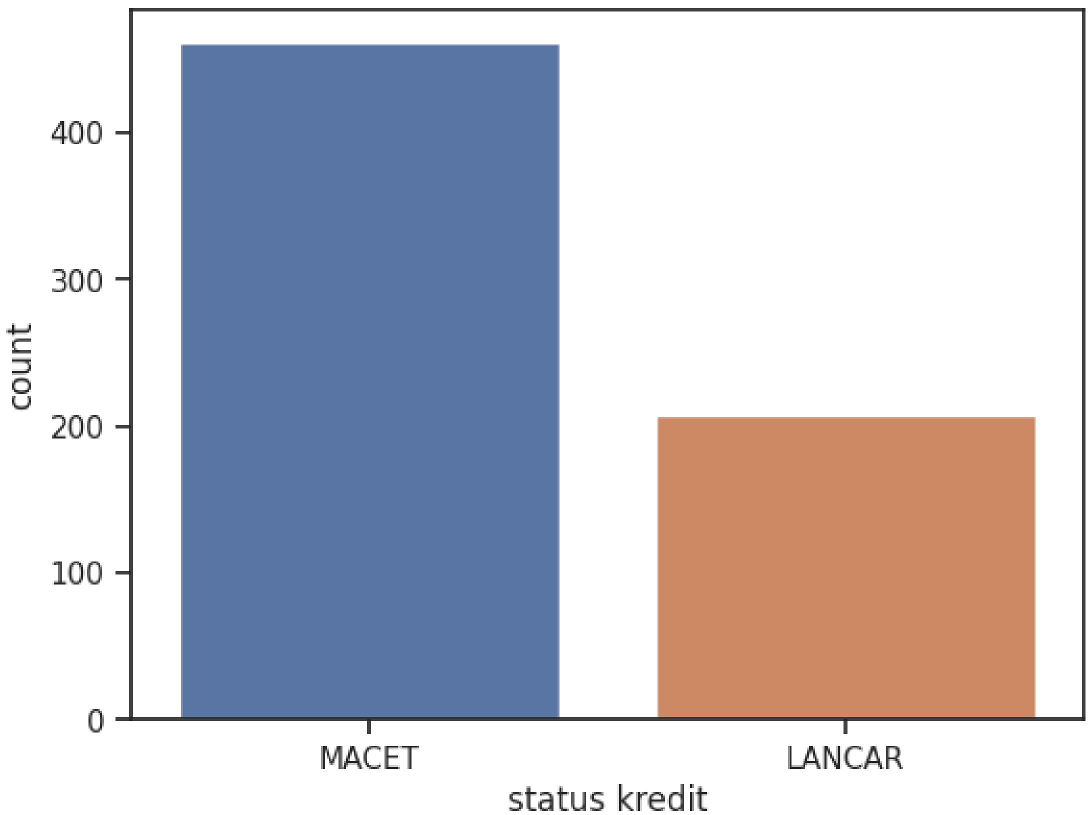
```
663  24.0     1500000.0  16.0                  105000.0         700000.0
664  38.0     1000000.0  16.0                   70000.0         812500.0
665  36.0     1000000.0  12.0                       NaN         429000.0
666  31.0     1312500.0   7.0                  198750.0        1312500.0
667  36.0     2000000.0   4.0                  550000.0        1000000.0

     tunggakan_pokok  tunggakan_bunga status kredit  bi_gol_penjamin__0  \
0           345000.0              0.0         MACET                   0
1           111428.0              0.0         MACET                   0
2                NaN              0.0         MACET                   0
3                0.0              0.0        LANCAR                   0
4          1078000.0              0.0         MACET                   0
..               …                …            …                   …
663         700000.0          90000.0         MACET                   1
664         812500.0          97500.0         MACET                   1
665         429000.0          45000.0         MACET                   1
666        1312500.0          78750.0         MACET                   1
667        1000000.0         100000.0         MACET                   1

     bi_gol_penjamin__800  bi_gol_penjamin__835  bi_gol_penjamin__874  \
0                       0                     0                     0
1                       0                     0                     0
2                       0                     0                     0
3                       0                     0                     0
4                       0                     0                     0
..                    …                     …                     …
663                     0                     0                     0
664                     0                     0                     0
665                     0                     0                     0
666                     0                     0                     0
667                     0                     0                     0

     bi_gol_penjamin__875  _L  _P
0                       1   0   1
1                       1   1   0
2                       1   0   1
3                       1   1   0
4                       1   1   0
..                    …  ..  ..
663                     0   0   1
664                     0   1   0
665                     0   0   1
666                     0   0   1
667                     0   0   1

[668 rows x 15 columns]
```

Lakukan Splitting Data

[9]:
```
predictor = df.loc[:, ~df.columns.isin(['status kredit'])]
target = df['status kredit']

# Splitting into train-test split
xTrain, xTest, yTrain, yTest = train_test_split(predictor, target, test_size=0.
↪3, random_state=33)
print(xTrain.shape, yTrain.shape)
print(xTest.shape, yTest.shape)
```

```
(467, 14) (467,)
(201, 14) (201,)
```

[10]:
```
# Visual Python: Visualization > Seaborn
from collections import Counter

sns.countplot(data=df, x='status kredit')
plt.show()

D = Counter(df['status kredit'])
print(D)
print("MACET = ", D['MACET']*100/(len(df['status kredit'])), '% LANCAR =␣
↪',D['LANCAR']*100/(len(df['status kredit'])) ,'%')
```




```
Counter({'MACET': 461, 'LANCAR': 207})
MACET =  69.0119760479042 % LANCAR =  30.98802395209581 %
```

Logistic Regression

[14]:
```python
# Membuat pipeline dengan SimpleImputer dan Logistic Regression
pipeline = make_pipeline(SimpleImputer(strategy='mean'), LogisticRegression())

# Melatih model menggunakan pipeline
pipeline.fit(xTrain, yTrain)

# Melakukan prediksi
prediksi_regLog = pipeline.predict(xTest)

# Evaluasi model
print(confusion_matrix(yTest, prediksi_regLog))
print(classification_report(yTest, prediksi_regLog))
```

```
[[ 54  13]
 [  8 126]]
              precision    recall  f1-score   support
```

```
      LANCAR       0.87      0.81      0.84        67
       MACET       0.91      0.94      0.92       134

    accuracy                           0.90       201
   macro avg       0.89      0.87      0.88       201
weighted avg       0.89      0.90      0.89       201
```

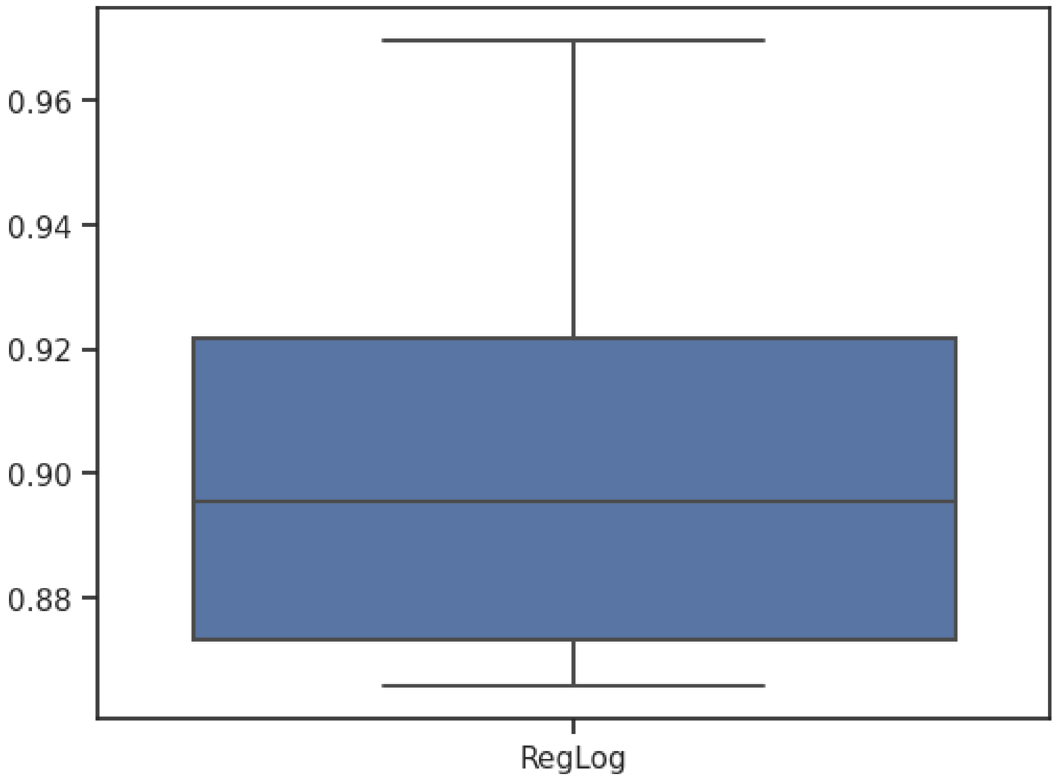
Cross Validation

[16]:
```python
# Mengukur waktu eksekusi
mulai = time.time()

# Menghitung skor cross-validation
scores_regLog = cross_val_score(pipeline, predictor, target, cv=10)

# Mengukur waktu eksekusi
waktu = time.time() - mulai

# Menampilkan hasil
print("Accuracy Regresi Logistik: %0.2f (+/- %0.2f), Waktu = %0.3f detik" %␣
 ↪(scores_regLog.mean(), scores_regLog.std() * 2, waktu))
```

```
Accuracy Regresi Logistik: 0.90 (+/- 0.07), Waktu = 0.108 detik
```

[17]:
```python
# Visualisasi untuk mengevaluasi & membandingkan model dengan lebih baik lagi
df_ = pd.DataFrame({'RegLog': scores_regLog})
p = sns.boxplot(data = df_)
df_.min()
```

[17]:
```
RegLog    0.865672
dtype: float64
```



[21]:
```
# Melatih model menggunakan pipeline
pipeline.fit(predictor, target)

# Mendapatkan koefisien dari model
koefisien_reglog = pipeline.named_steps['logisticregression'].coef_[0]

# Menampilkan koefisien
print("Koefisien Regresi Logistik:", koefisien_reglog)
```

```
Koefisien Regresi Logistik: [ 5.04484881e-09 -7.15361276e-07  2.72062282e-08
-6.21188527e-06
  2.40649782e-07  1.13897425e-05  4.29157191e-06 -1.08412202e-09
  1.52664662e-11  7.35802842e-12  3.67715783e-11  1.08312788e-09
 -2.37741784e-10  2.96143715e-10]
```

Decision Tree

[29]:
```
# Decision Tree Algorithm
# Decision Tree: http://scikit-learn.org/stable/modules/tree.html
# Membagi data menjadi data latih dan data uji
xTrain, xTest, yTrain, yTest = train_test_split(predictor, target, test_size=0.
↪2, random_state=42)
```

```
# Menggunakan SimpleImputer untuk mengisi nilai-nilai yang hilang
imputer = SimpleImputer(strategy='mean')
xTrain_imputed = imputer.fit_transform(xTrain)
xTest_imputed = imputer.transform(xTest)

# Membuat dan melatih model DecisionTreeClassifier
DT = DecisionTreeClassifier(random_state=0)
DT.fit(xTrain_imputed, yTrain)

# Melakukan prediksi
prediksi_DT = DT.predict(xTest_imputed)

# Evaluasi model
print(confusion_matrix(yTest, prediksi_DT))
print(classification_report(yTest, prediksi_DT))
```

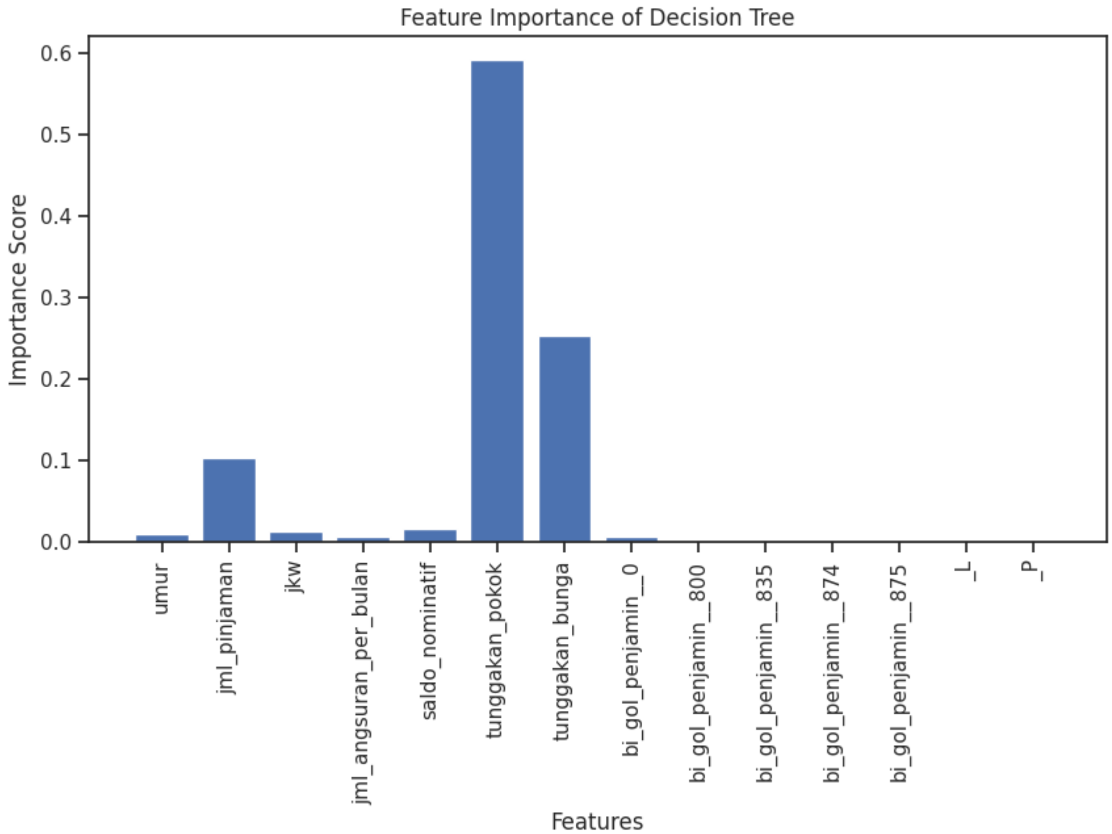
```
[[33  4]
 [ 3 94]]
              precision    recall  f1-score   support

      LANCAR       0.92      0.89      0.90        37
       MACET       0.96      0.97      0.96        97

    accuracy                           0.95       134
   macro avg       0.94      0.93      0.93       134
weighted avg       0.95      0.95      0.95       134
```

[30]:
```
# Varible importance - Salah satu kelebihan Decision Tree
DT.feature_importances_
```

[30]:
```
array([0.00985725, 0.10363044, 0.01368315, 0.00621651, 0.01635083,
       0.5914697 , 0.25303905, 0.00575307, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        ])
```

[31]:
```
# Assuming your model is fitted, you can access feature importances
feature_importances = DT.feature_importances_

# Assuming you have feature names (replace feature_names with your actual␣
↪feature names)
feature_names = df.drop('status kredit', axis=1).columns

# Visualize the feature importances
plt.figure(figsize=(10, 5))
plt.bar(feature_names, feature_importances)
plt.xlabel('Features')
plt.ylabel('Importance Score')
```

```
plt.title('Feature Importance of Decision Tree')
plt.xticks(rotation=90)
plt.show()
```

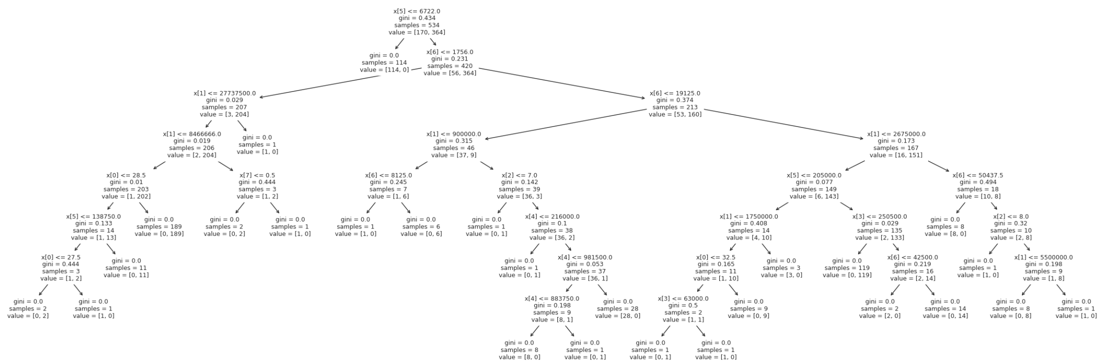


[33]:

```
# Membuat pipeline dengan SimpleImputer dan DecisionTreeClassifier
pipeline = make_pipeline(SimpleImputer(strategy='mean'),␣
 ↪DecisionTreeClassifier(random_state=0))

# Menghitung skor cross-validation
mulai = time.time()
scores_dt = cross_val_score(pipeline, predictor, target, cv=10)
waktu = time.time() - mulai

# Menampilkan hasil
print("Accuracy Decision Tree: %0.2f (+/- %0.2f), Waktu = %0.3f detik" %␣
 ↪(scores_dt.mean(), scores_dt.std() * 2, waktu))
```

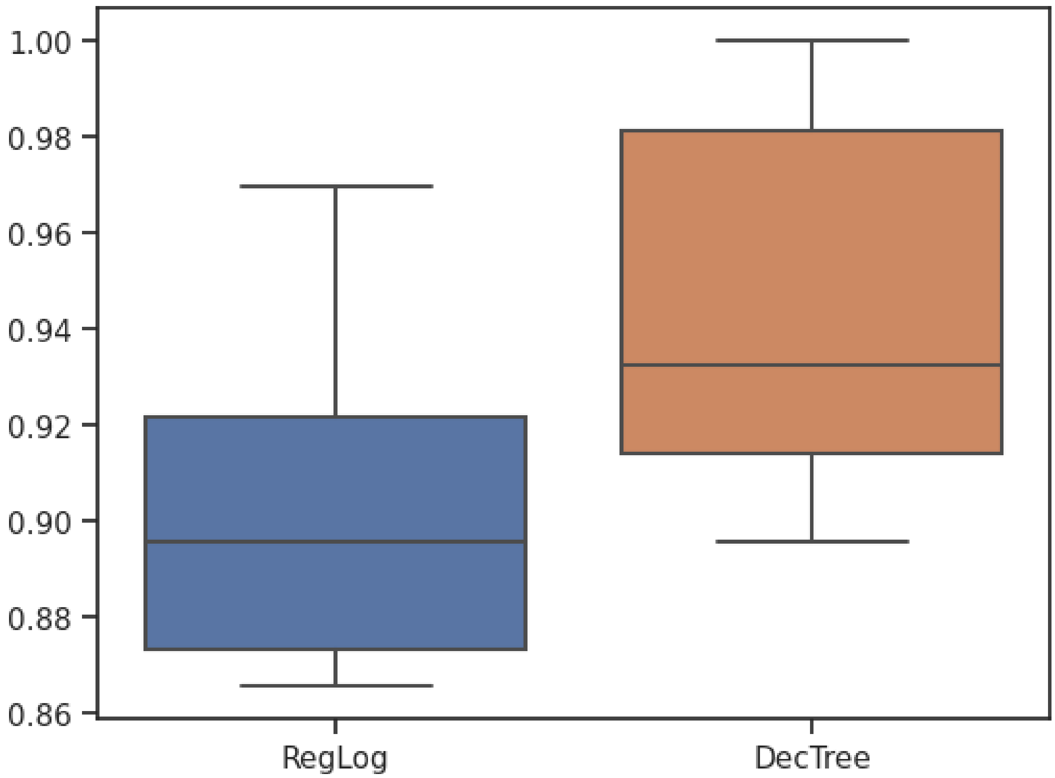
```
Accuracy Decision Tree: 0.94 (+/- 0.08), Waktu = 0.322 detik
```

[35]:
```
plt.figure(figsize=(30,10))
p = tree.plot_tree(DT)
```



[39]:
```
# Visualisasi untuk mengevaluasi & membandingkan model dengan lebih baik lagi
df_ = pd.DataFrame({'RegLog': scores_regLog, "DecTree":scores_dt})
p = sns.boxplot(data = df_)

# Menampilkan nilai minimum dari kedua model
print("Minimum Score RegLog:", df_scores['RegLog'].min())
print("Minimum Score DecTree:", df_scores['DecTree'].min())
print("Maximum Score RegLog:", df_scores['RegLog'].max())
print("Maximum Score DecTree:", df_scores['DecTree'].max())
```

```
Minimum Score RegLog: 0.8656716417910447
Minimum Score DecTree: 0.8955223880597015
Maximum Score RegLog: 0.9696969696969697
Maximum Score DecTree: 1.0
```


1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
RegLog
DecTree

[ ]: