```python
import pandas as pd
import numpy as np

df_kategori=pd.read_csv('Dataset UTS_Gasal 2425.csv')
df_kategori.head(20)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
2          86929            100     yes      no      11     98155
3          51522              3      no      no      61      9047
4          96470             74     yes      no      21     92029
5          79770              3      no     yes      69     54812
6          75985             60     yes      no      67      6517
7          64169             88      no     yes       6     61711
8          92383             12      no      no      78     71982
9          95121             46      no     yes       3      9382
10         76485             47     yes      no       9     90254
11         87060             27      no     yes      91     51803
12         66683             19     yes     yes       6     50801
13         84559             29      no     yes      69     53057
14         76091             38     yes      no      32     59451
15         92696             49     yes      no      38     74381
16         59800             47      no     yes      27     44815
17         54836             25      no     yes      53     64601
18         70021             52     yes      no      28     95678
19         54368             11     yes     yes      20     55761

    citypartrange  numprevowners  made isnewbuilt hasstormprotector
basement  \
0               3              8  2005        old               yes
4313
1               6              8  2021        old                no
2937
2               3              4  2003        new                no
6326
3               8              3  2012        new               yes
632
4               4              2  2011        new               yes
5414
5              10              5  2018        old               yes
8871
6               6              9  2009        new               yes
4878
7               3              9  2011        new               yes
3054
8               3              7  2000        old                no
7507
9               7              9  1994        old                no
615
```

```
10                2                 9  2008          new                   no
2860
11                8                10  2000          old                   no
6629
12                6                 2  2001          old                   no
7473
13                7                 7  2000          new                   no
3573
14                5                 8  2016          new                   no
8150
15                9                 2  2021          old                   no
1559
16                6                 9  2021          old                   no
5075
17               10                 5  2020          new                   no
5278
18                4                 6  1992          old                  yes
4480
19                3                 7  2021          old                   no
231

    attic  garage hasstorageroom  hasguestroom      price category
0    9005     956             no             7  7559081.5   Luxury
1    8852     135            yes             9  5574642.1   Middle
2    4748     654             no            10  8696869.3   Luxury
3    5792     807            yes             5  5154055.2   Middle
4    1172     716            yes             9  9652258.1   Luxury
5    7117     240             no             7  7986665.8   Luxury
6     281     384            yes             5  7607322.9   Luxury
7     129     726             no             9  6420823.1   Middle
8    9056     892            yes             1  9244344.0   Luxury
9    1221     328             no            10  9515440.4   Luxury
10   3129     982             no             1  7653300.8   Luxury
11    435     512             no             7  8711426.0   Luxury
12    796     237            yes             3  6677649.1   Middle
13   9556     918            yes             8  8460604.0   Luxury
14   6037     930             no             7  7614076.6   Luxury
15   5111     957            yes             2  9272740.1   Luxury
16   3104     864             no             4  5984462.1   Middle
17   1059     313            yes             6  5492532.0   Middle
18   6919     680            yes             1  7005572.2   Luxury
19   1939     223             no             8  5446398.1   Middle
```

```
df_kategori2=df_kategori.drop('price', axis=1)
df_kategori2.head(50)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
2          86929            100     yes      no      11     98155
```

3	51522	3	no	no	61	9047
4	96470	74	yes	no	21	92029
5	79770	3	no	yes	69	54812
6	75985	60	yes	no	67	6517
7	64169	88	no	yes	6	61711
8	92383	12	no	no	78	71982
9	95121	46	no	yes	3	9382
10	76485	47	yes	no	9	90254
11	87060	27	no	yes	91	51803
12	66683	19	yes	yes	6	50801
13	84559	29	no	yes	69	53057
14	76091	38	yes	no	32	59451
15	92696	49	yes	no	38	74381
16	59800	47	no	yes	27	44815
17	54836	25	no	yes	53	64601
18	70021	52	yes	no	28	95678
19	54368	11	yes	yes	20	55761
20	63053	6	yes	yes	28	45312
21	64393	8	no	no	51	95335
22	93876	60	no	yes	70	5484
23	84016	15	yes	no	55	63595
24	89768	48	yes	yes	17	71000
25	58478	5	no	yes	35	5898
26	66621	48	no	no	89	52165
27	73314	43	no	yes	38	49895
28	59972	28	no	yes	18	32083
29	71591	20	yes	no	58	46834
30	67311	67	no	no	10	45626
31	61534	73	yes	no	97	22943
32	84091	50	no	yes	72	22718
33	51434	64	no	no	23	79754
34	78960	55	no	yes	76	23408
35	81870	60	no	yes	100	58048
36	91559	36	no	yes	21	82521
37	72098	9	no	yes	67	91168
38	55232	23	no	no	8	849
39	53735	49	no	yes	92	2423
40	71397	71	no	no	93	68199
41	65151	81	yes	yes	3	66191
42	61484	91	yes	no	94	87015
43	57160	15	yes	yes	43	40786
44	55933	15	no	no	97	5800
45	81936	68	no	no	92	6143
46	99683	12	yes	no	77	18300
47	62887	45	no	yes	7	91125
48	73062	34	yes	yes	38	44770
49	84284	76	no	no	39	55723

```
    citypartrange  numprevowners  made isnewbuilt hasstormprotector
```

```
basement  \
0                3              8  2005         old            yes
4313
1                6              8  2021         old             no
2937
2                3              4  2003         new             no
6326
3                8              3  2012         new            yes
632
4                4              2  2011         new            yes
5414
5               10              5  2018         old            yes
8871
6                6              9  2009         new            yes
4878
7                3              9  2011         new            yes
3054
8                3              7  2000         old             no
7507
9                7              9  1994         old             no
615
10               2              9  2008         new             no
2860
11               8             10  2000         old             no
6629
12               6              2  2001         old             no
7473
13               7              7  2000         new             no
3573
14               5              8  2016         new             no
8150
15               9              2  2021         old             no
1559
16               6              9  2021         old             no
5075
17              10              5  2020         new             no
5278
18               4              6  1992         old            yes
4480
19               3              7  2021         old             no
231
20               3              1  1997         old            yes
8414
21               4              1  1990         new             no
3835
22               2              1  1999         new            yes
4086
23               1              7  2016         new             no
3284
```

```
24                 6                9  1993          old              yes
2485
25                 6               10  2016          old               no
8366
26                10                1  1995          new              yes
5024
27                10                1  2018          old              yes
3281
28                 9                8  2021          new              yes
8384
29                 7                4  1998          old               no
6486
30                 3                3  1990          new              yes
6928
31                 9                5  2001          old               no
9265
32                 7                5  1993          old               no
2668
33                10                2  2012          new              yes
2080
34                 8                4  2015          new              yes
7126
35                 3                8  2020          old               no
3632
36                 6                2  2007          old              yes
788
37                 2                3  2014          new               no
9080
38                 8                3  1991          old               no
1492
39                 4                7  2006          old               no
8654
40                 3               10  1995          new              yes
3477
41                 7                9  1991          old               no
3218
42                 8                8  2013          new               no
5486
43                 8                8  2002          old               no
4018
44                 9                8  2001          new              yes
7369
45                 3                1  2011          new               no
6393
46                 5                8  2002          old              yes
4034
47                 4                3  1993          new               no
292
48                 4                8  2016          old               no
```

```
9823
49              5               1  1998        old               no
4500

    attic  garage hasstorageroom  hasguestroom category
0    9005     956             no             7   Luxury
1    8852     135            yes             9   Middle
2    4748     654             no            10   Luxury
3    5792     807            yes             5   Middle
4    1172     716            yes             9   Luxury
5    7117     240             no             7   Luxury
6     281     384            yes             5   Luxury
7     129     726             no             9   Middle
8    9056     892            yes             1   Luxury
9    1221     328             no            10   Luxury
10   3129     982             no             1   Luxury
11    435     512             no             7   Luxury
12    796     237            yes             3   Middle
13   9556     918            yes             8   Luxury
14   6037     930             no             7   Luxury
15   5111     957            yes             2   Luxury
16   3104     864             no             4   Middle
17   1059     313            yes             6   Middle
18   6919     680            yes             1   Luxury
19   1939     223             no             8   Middle
20   6270     939            yes             8   Middle
21   2403     559             no             6   Middle
22   5991     494            yes             8   Luxury
23   9879     641             no             2   Luxury
24    108     864             no             7   Luxury
25   4799     979            yes             7   Middle
26   8103     388            yes             4   Middle
27   5020     968             no             8   Luxury
28   7226     226            yes             4   Middle
29   3310     366             no             0   Luxury
30   7808     774            yes             5   Middle
31   8974     755            yes             6   Middle
32   4669     766             no             8   Luxury
33   9575     753             no             7   Middle
34   5012     974            yes             0   Luxury
35   5960     723            yes             3   Luxury
36   4788     132            yes             8   Luxury
37   9356     740            yes             9   Luxury
38   5697     625             no             6   Middle
39   9588     290            yes             8   Middle
40   5530     342             no             2   Luxury
41   9119     849             no             4   Middle
42   3641     766             no             3   Middle
43   4871     836            yes             2   Middle
```

```
44   6739     686            yes             6   Middle
45   9082     734             no             0   Luxury
46   2877     787            yes             6   Luxury
47    744     675             no             4   Middle
48   7174     728            yes             0   Luxury
49   4877     480             no             6   Luxury
```

```
df_kategori2['category'].value_counts()
```

```
category
Basic     4344
Luxury    3065
Middle    2591
Name: count, dtype: int64
```

```
print("data null \n" ,df_kategori2.isnull().sum())
print("\ndata kosong \n" ,df_kategori2.empty)
print("\ndata nan \n" ,df_kategori2.isna().sum())
```

```
data null
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
dtype: int64

data kosong
 False

data nan
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
```

```
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
dtype: int64
```

```
median_chole = df_kategori2['category'].median()

print(median_chole)

df_kategori2['Cholesterol'] =
df_kategori2['Cholesterol'].fillna(median_chole)
```

```
222.0
```

```
print(df_kategori2['category'].value_counts())

print("data null \n" ,df_kategori2.isnull().sum())
print("\ndata kosong \n" ,df_kategori2.empty)
print("\ndata nan \n" ,df_kategori2.isna().sum())
```

```
category
Basic      4344
Luxury     3065
Middle     2591
Name: count, dtype: int64
data null
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
```

```
dtype: int64

data kosong
 False

data nan
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
dtype: int64
```

```
print("Sebelum drop missing value",df_kategori2.shape)
df_kategori2 = df_kategori2.dropna(how="any", inplace=False)
print("Sesudah drop missing value" , df_kategori2.shape)
```

```
Sebelum drop missing value (10000, 17)
Sesudah drop missing value (10000, 17)
```

```
df_kategori2['category'].value_counts()
```

```
category
Basic     4344
Luxury    3065
Middle    2591
Name: count, dtype: int64
```

```
print("Sebelum drop data dengan gender Bi",df_kategori2.shape)
df_kategori2=df_kategori2[df_kategori2['category']!='Bi']
print("Sesudah drop data dengan gender Bi" , df_kategori2.shape)
```

```
Sebelum drop data dengan gender Bi (10000, 17)
Sesudah drop data dengan gender Bi (10000, 17)
```

```
df_kategori2.head(20)
```

```
   squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0         75523              3      no     yes      63      9373
```

1	55712	58	no	yes	19	34457
2	86929	100	yes	no	11	98155
3	51522	3	no	no	61	9047
4	96470	74	yes	no	21	92029
5	79770	3	no	yes	69	54812
6	75985	60	yes	no	67	6517
7	64169	88	no	yes	6	61711
8	92383	12	no	no	78	71982
9	95121	46	no	yes	3	9382
10	76485	47	yes	no	9	90254
11	87060	27	no	yes	91	51803
12	66683	19	yes	yes	6	50801
13	84559	29	no	yes	69	53057
14	76091	38	yes	no	32	59451
15	92696	49	yes	no	38	74381
16	59800	47	no	yes	27	44815
17	54836	25	no	yes	53	64601
18	70021	52	yes	no	28	95678
19	54368	11	yes	yes	20	55761

	citypartrange	numprevowners	made	isnewbuilt	hasstormprotector	basement \
0	3	8	2005	old	yes	4313
1	6	8	2021	old	no	2937
2	3	4	2003	new	no	6326
3	8	3	2012	new	yes	632
4	4	2	2011	new	yes	5414
5	10	5	2018	old	yes	8871
6	6	9	2009	new	yes	4878
7	3	9	2011	new	yes	3054
8	3	7	2000	old	no	7507
9	7	9	1994	old	no	615
10	2	9	2008	new	no	2860
11	8	10	2000	old	no	6629
12	6	2	2001	old	no	7473
13	7	7	2000	new	no	

```
3573
14                5              8  2016        new                no
8150
15                9              2  2021        old                no
1559
16                6              9  2021        old                no
5075
17               10              5  2020        new                no
5278
18                4              6  1992        old               yes
4480
19                3              7  2021        old                no
231

    attic  garage hasstorageroom  hasguestroom category
0    9005     956             no             7   Luxury
1    8852     135            yes             9   Middle
2    4748     654             no            10   Luxury
3    5792     807            yes             5   Middle
4    1172     716            yes             9   Luxury
5    7117     240             no             7   Luxury
6     281     384            yes             5   Luxury
7     129     726             no             9   Middle
8    9056     892            yes             1   Luxury
9    1221     328             no            10   Luxury
10   3129     982             no             1   Luxury
11    435     512             no             7   Luxury
12    796     237            yes             3   Middle
13   9556     918            yes             8   Luxury
14   6037     930             no             7   Luxury
15   5111     957            yes             2   Luxury
16   3104     864             no             4   Middle
17   1059     313            yes             6   Middle
18   6919     680            yes             1   Luxury
19   1939     223             no             8   Middle
```

```python
print("Sebelum Pengecekan data Duplikat",df_kategori2.shape)
df_kategori3 = df_kategori2.drop_duplicates(keep='last')
print("Sesudah Pengecekan data Duplikat" , df_kategori3.shape)
```

```
Sebelum Pengecekan data Duplikat (10000, 17)
Sesudah Pengecekan data Duplikat (10000, 17)
```

```python
from sklearn.model_selection import train_test_split

x_regress = df_kategori3.drop(columns=['category'], axis = 1)
y_regress = df_kategori3['category']

x_train_category, x_test_category, y_train_category, y_test_category =
train_test_split(x_regress, y_regress, test_size= 0.25, random_state =
```

```
78)

print(x_train_category.shape)
print(x_test_category.shape)

(7500, 16)
(2500, 16)

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

kolom_kategori=['hasyard','haspool','isnewbuilt','hasstormprotector','
hasstorageroom']

transform=make_column_transformer(
    (OneHotEncoder(), kolom_kategori),remainder='passthrough'
)

x_train_category_enc = transform.fit_transform(x_train_category)

x_test_category_enc = transform.fit_transform(x_test_category)

df_train_enc = pd.DataFrame(x_train_category_enc,
columns=transform.get_feature_names_out())
df_test_enc = pd.DataFrame(x_test_category_enc,
columns=transform.get_feature_names_out())

df_train_enc.head(20)
df_test_enc.head(20)

    onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                         1.0                         0.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         1.0                         0.0
4                         0.0                         1.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         0.0                         1.0
9                         1.0                         0.0
10                        0.0                         1.0
11                        0.0                         1.0
12                        1.0                         0.0
13                        0.0                         1.0
14                        1.0                         0.0
15                        0.0                         1.0
16                        1.0                         0.0
17                        0.0                         1.0
18                        0.0                         1.0
19                        1.0                         0.0
```

```
    onehotencoder__haspool_no  onehotencoder__haspool_yes  \
0                         0.0                         1.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         0.0                         1.0
4                         1.0                         0.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         1.0                         0.0
9                         1.0                         0.0
10                        0.0                         1.0
11                        1.0                         0.0
12                        1.0                         0.0
13                        1.0                         0.0
14                        1.0                         0.0
15                        1.0                         0.0
16                        1.0                         0.0
17                        0.0                         1.0
18                        1.0                         0.0
19                        1.0                         0.0

    onehotencoder__isnewbuilt_new  onehotencoder__isnewbuilt_old  \
0                             0.0                            1.0
1                             0.0                            1.0
2                             0.0                            1.0
3                             1.0                            0.0
4                             1.0                            0.0
5                             0.0                            1.0
6                             0.0                            1.0
7                             0.0                            1.0
8                             1.0                            0.0
9                             1.0                            0.0
10                            0.0                            1.0
11                            0.0                            1.0
12                            1.0                            0.0
13                            1.0                            0.0
14                            0.0                            1.0
15                            1.0                            0.0
16                            1.0                            0.0
17                            1.0                            0.0
18                            1.0                            0.0
19                            1.0                            0.0

    onehotencoder__hasstormprotector_no
onehotencoder__hasstormprotector_yes  \
0                                   1.0
0.0
1                                   0.0
```

```
1.0
2                                    1.0
0.0
3                                    1.0
0.0
4                                    1.0
0.0
5                                    0.0
1.0
6                                    0.0
1.0
7                                    0.0
1.0
8                                    0.0
1.0
9                                    1.0
0.0
10                                   1.0
0.0
11                                   0.0
1.0
12                                   0.0
1.0
13                                   1.0
0.0
14                                   1.0
0.0
15                                   1.0
0.0
16                                   1.0
0.0
17                                   1.0
0.0
18                                   1.0
0.0
19                                   1.0
0.0

    onehotencoder__hasstorageroom_no
onehotencoder__hasstorageroom_yes  ...  \
0                                 1.0
0.0  ...
1                                 0.0
1.0  ...
2                                 0.0
1.0  ...
3                                 0.0
1.0  ...
4                                 1.0
```

```
0.0  ...
5                                  0.0
1.0  ...
6                                  0.0
1.0  ...
7                                  0.0
1.0  ...
8                                  0.0
1.0  ...
9                                  0.0
1.0  ...
10                                 0.0
1.0  ...
11                                 1.0
0.0  ...
12                                 0.0
1.0  ...
13                                 0.0
1.0  ...
14                                 1.0
0.0  ...
15                                 0.0
1.0  ...
16                                 0.0
1.0  ...
17                                 1.0
0.0  ...
18                                 0.0
1.0  ...
19                                 0.0
1.0  ...

    remainder__numberofrooms  remainder__floors
remainder__citycode  \
0                       73.0               13.0                 42855.0

1                       95.0                3.0                 75381.0

2                       39.0                8.0                 91674.0

3                       47.0               63.0                 58471.0

4                       64.0               83.0                 30779.0

5                       91.0               67.0                 65183.0

6                       48.0               70.0                 21012.0

7                       93.0               35.0                 12062.0
```

```
8                         74.0                 14.0               76662.0
9                         72.0                 76.0               87732.0
10                        18.0                 66.0               38920.0
11                        37.0                 26.0               96016.0
12                        76.0                 95.0               76985.0
13                        26.0                 99.0               38185.0
14                        69.0                 45.0               88591.0
15                         9.0                 25.0               33740.0
16                        29.0                 58.0               13202.0
17                        91.0                 43.0               93072.0
18                        68.0                 62.0               97608.0
19                        48.0                 53.0               34588.0

    remainder__citypartrange  remainder__numprevowners  
remainder__made  \
0                        9.0                       6.0  
2015.0
1                        5.0                       6.0  
2003.0
2                        2.0                       2.0  
2009.0
3                       10.0                       1.0  
1990.0
4                        6.0                       4.0  
1992.0
5                        1.0                      10.0  
2019.0
6                        5.0                       3.0  
2007.0
7                       10.0                       9.0  
1998.0
8                        9.0                       6.0  
2004.0
9                        2.0                       8.0  
2017.0
10                       7.0                       2.0  
2007.0
11                       3.0                       6.0  
```

```
2000.0
12                       4.0                         7.0
2009.0
13                       7.0                         7.0
1994.0
14                       1.0                         8.0
1995.0
15                       2.0                         1.0
2009.0
16                       6.0                         5.0
1993.0
17                       1.0                         7.0
2012.0
18                       1.0                         9.0
2001.0
19                       9.0                         6.0
2003.0

    remainder__basement  remainder__attic  remainder__garage  \
0                2560.0            6823.0              239.0
1                6810.0            1391.0              556.0
2                1477.0            3153.0              952.0
3                1730.0            7967.0              722.0
4                 594.0            8310.0              898.0
5                6824.0            7141.0              956.0
6                5861.0            2750.0              652.0
7                2064.0            2720.0              315.0
8                9501.0            9579.0              768.0
9                6414.0            6111.0              613.0
10               4949.0            5811.0              185.0
11               1261.0            6205.0              850.0
12               3586.0            6017.0              472.0
13               7053.0            1109.0              827.0
14               3429.0            8363.0              876.0
15               7818.0            1494.0              567.0
16               6740.0            5829.0              484.0
17               1528.0            4961.0              893.0
18                131.0             336.0              656.0
19                907.0            3418.0              162.0

    remainder__hasguestroom
0                      10.0
1                       3.0
2                       2.0
3                       0.0
4                       5.0
5                       6.0
6                      10.0
7                       5.0
```

```
8                          1.0
9                          7.0
10                         6.0
11                        10.0
12                         7.0
13                         3.0
14                         3.0
15                         0.0
16                         5.0
17                         8.0
18                        10.0
19                         5.0

[20 rows x 21 columns]
```

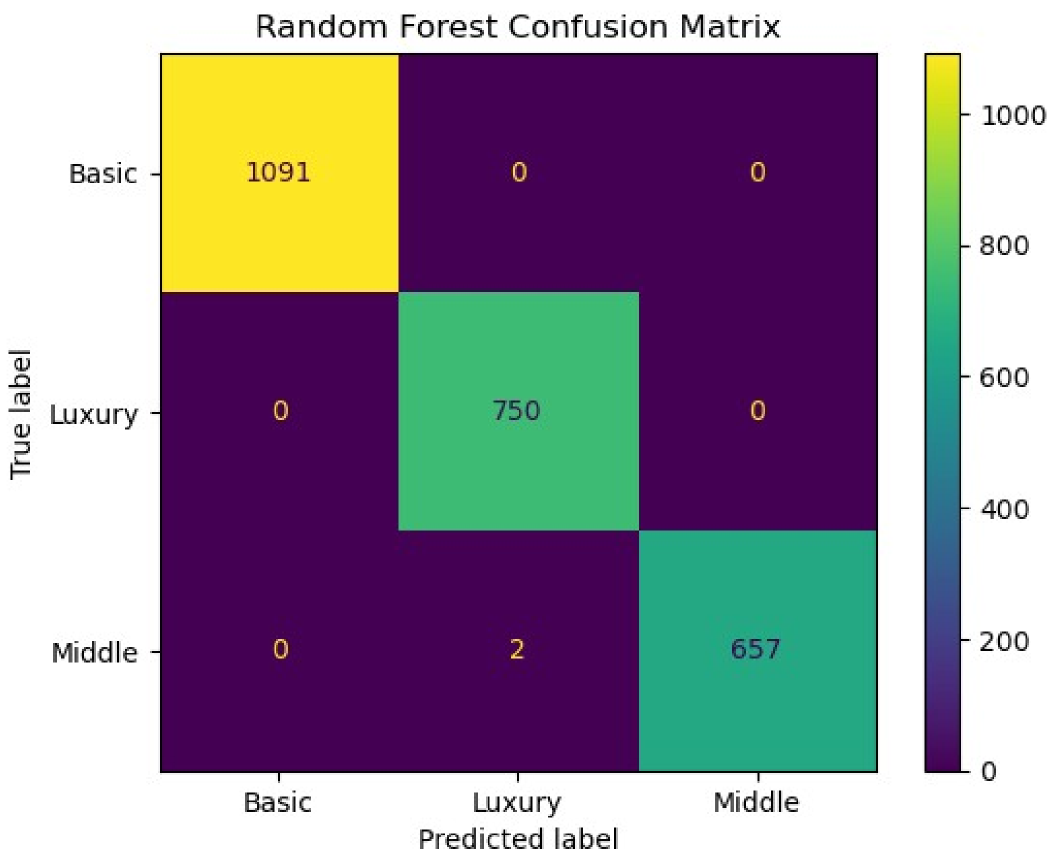
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.metrics import confusion_matrix, classification_report,
ConfusionMatrixDisplay
import numpy as np


pipe_RF = [
    ('data scaling', StandardScaler()),
    ('feature select', SelectKBest()),
    ('clf', RandomForestClassifier(random_state=78,
class_weight='balanced'))
]

params_grid_RF = [
    {
        'data scaling': [StandardScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [StandardScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select__k': np.arange(2, 6),
        'clf__max_depth': np.arange(4, 5),
```

```
        'clf__n_estimators': [100, 150]
    },
    {
        'data scaling': [MinMaxScaler()],
        'feature select': [SelectPercentile()],
        'feature select__percentile': np.arange(20, 50),
        'clf__max_depth': np.arange(4, 5),
        'clf__n_estimators': [100, 150]
    }
]

estimator_RF = Pipeline(pipe_RF)

GSCV_RF = GridSearchCV(estimator_RF, params_grid_RF,
cv=StratifiedKFold(n_splits=5))

GSCV_RF.fit(x_train_category_enc, y_train_category)

print("GSCV training finished")
```

```
GSCV training finished
```

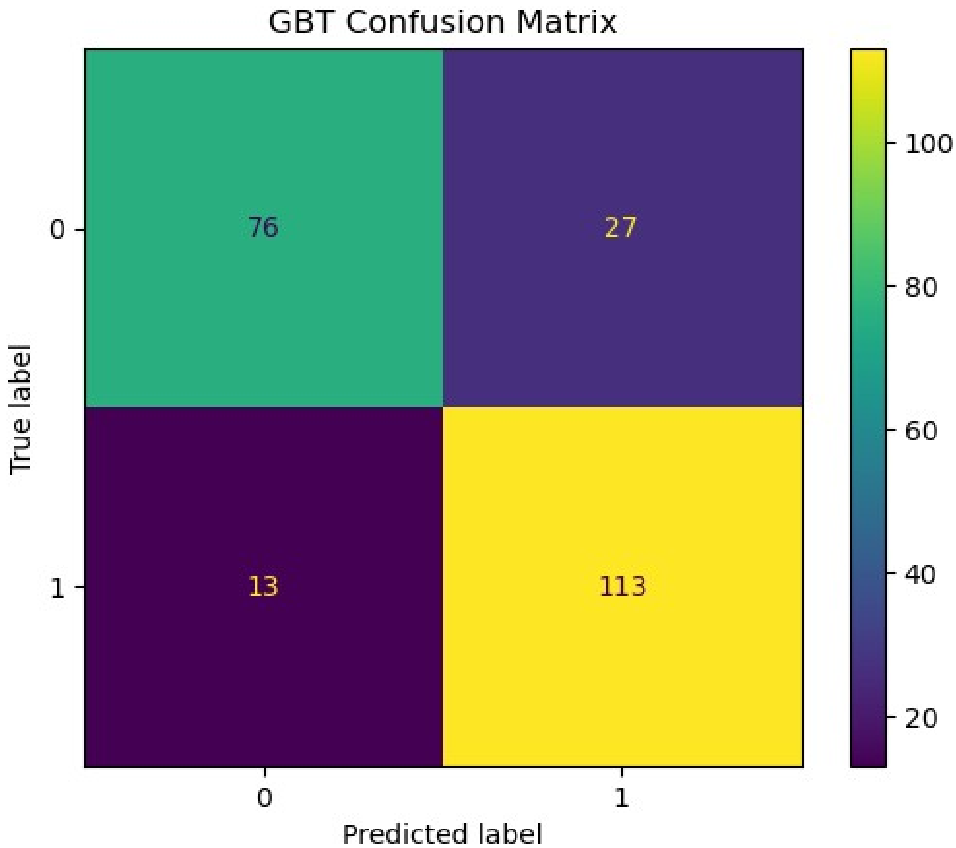
```
print("CV Score : {}".format(GSCV_RF.best_score_))

print("Test Score:
{}".format(GSCV_RF.best_estimator_.score(x_test_category_enc,
y_test_category)))

print("Best model:", GSCV_RF.best_estimator_)
mask = GSCV_RF.best_estimator_.named_steps['feature
select'].get_support()
print("Best features:", df_train_enc.columns[mask])

RF_pred = GSCV_RF.predict(x_test_category_enc)

import matplotlib.pyplot as plt

cm = confusion_matrix(y_test_category, RF_pred,
labels=GSCV_RF.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_RF.classes_)
disp.plot()

plt.title("Random Forest Confusion Matrix")
plt.show()

print("Classification report RF: \n",
classification_report(y_test_category, RF_pred))
```

```
CV Score : 0.9990666666666665
Test Score: 0.9992
Best model: Pipeline(steps=[('data scaling', StandardScaler()),
                ('feature select', SelectPercentile(percentile=29)),
                ('clf',
                 RandomForestClassifier(class_weight='balanced',
max_depth=4,
                                        random_state=78))])
Best features: Index(['onehotencoder__hasyard_no',
'onehotencoder__hasyard_yes',
       'onehotencoder__haspool_no', 'onehotencoder__haspool_yes',
       'onehotencoder__isnewbuilt_new', 'remainder__squaremeters'],
      dtype='object')
```




```
Classification report RF:
               precision    recall  f1-score   support

       Basic       1.00      1.00      1.00      1091
      Luxury       1.00      1.00      1.00       750
      Middle       1.00      1.00      1.00       659

    accuracy                           1.00      2500
```

```
   macro avg       1.00      1.00      1.00      2500
weighted avg       1.00      1.00      1.00      2500


print("CV Score : {}".format(GSCV_GBT.best_score_))

print("Test Score:
{}".format(GSCV_GBT.best_estimator_.score(x_test_enc, y_test)))

print("Best model:", GSCV_GBT.best_estimator_)
mask =
GSCV_GBT.best_estimator_.named_steps['feat_select'].get_support()
print("Best features:", df_train_enc.columns[mask])

RF_pred = GSCV_GBT.predict(x_test_enc)

import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, RF_pred, labels=GSCV_GBT.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_GBT.classes_)
disp.plot()
plt.title("GBT Confusion Matrix")
plt.show()

print("Classification report GBT: \n", classification_report(y_test,
RF_pred))

CV Score : 0.854437744631334
Test Score: 0.8253275109170306
Best model: Pipeline(steps=[('feat_select',
SelectPercentile(percentile=48)),
                ('clf',
                 GradientBoostingClassifier(learning_rate=0.01,
max_depth=4,
                                            n_estimators=150,
                                            random_state=47))])
Best features: Index(['onehotencoder__Sex_M',
'onehotencoder__ChestPainType_ASY',
       'onehotencoder__ChestPainType_ATA',
'onehotencoder__ExerciseAngina_N',
       'onehotencoder__ExerciseAngina_Y',
'onehotencoder__ST_Slope_Flat',
       'onehotencoder__ST_Slope_Up', 'remainder__Age',
'remainder__MaxHR',
       'remainder__Oldpeak'],
      dtype='object')
```




```
Classification report GBT:
              precision    recall  f1-score   support

           0       0.85      0.74      0.79       103
           1       0.81      0.90      0.85       126

    accuracy                           0.83       229
   macro avg       0.83      0.82      0.82       229
weighted avg       0.83      0.83      0.82       229
```

```
#import Library yang dibutuhkan untuk pipeline, GSCV, dan metrik evaluasi
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectPercentile, SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
import numpy as np

#buat rancangan pipeline mulai dari data scaling hingga classifier
```

```python
pipe_logreg = Pipeline(steps=[
    ('scale', MinMaxScaler()),
    ('feat_select', SelectKBest()),
    ('clf', LogisticRegression(class_weight='balanced',
max_iter=1000))
])

#buat parameter grid untuk step feature selection dan classifier
params_grid_logreg = [
    {
    'scale': [MinMaxScaler()],
    'feat_select__k':np.arange(2,6),
    'clf__penalty': ['l2', 'none'],
    'clf__C':[0.1, 1, 10],
    'clf__solver': ['lbfgs', 'saga']
    },
    {
    'scale': [MinMaxScaler()],
    'feat_select': [SelectPercentile()],
    'feat_select__percentile':np.arange(20,50),
    'clf__penalty': ['l2', 'none'],
    'clf__C':[0.1, 1, 10],
    'clf__solver': ['lbfgs', 'saga']
    },
    {
    'scale': [StandardScaler()],
    'feat_select__k':np.arange(2,6),
    'clf__penalty': ['l2', 'none'],
    'clf__C':[0.1, 1, 10],
    'clf__solver': ['lbfgs', 'saga']
    },
    {
    'scale': [StandardScaler()],
    'feat_select':[SelectPercentile()],
    'feat_select__percentile': np.arange(20,50),
    'clf__penalty': ['l2', 'none'],
    'clf__C':[0.1, 1, 10],
    'clf__solver': ['lbfgs', 'saga']
    }
]

#muat pipeline dan parameter grid ke dalam objek GridSearchCV dengan
Stratified 5-fold CV
SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=78)

GSCV_LogReg = GridSearchCV(pipe_logreg, params_grid_logreg, cv=SKF)
#jalankan objek GSCV untuk melatih model dengan train set menggunakan
fungsi fit
GSCV_LogReg.fit(x_train_category_enc, y_train_category)
print("GSCV training finished")
```

```
GSCV training finished

c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
model_selection\_validation.py:547: FitFailedWarning:
2040 fits failed out of a total of 4080.
The score on these train-test partitions for these parameters will be
set to nan.
If these failures are not expected, you can try to debug them by
setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
2040 fits failed with the following error:
Traceback (most recent call last):
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
model_selection\_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
base.py", line 1474, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
pipeline.py", line 475, in fit
    self._final_estimator.fit(Xt, y, **last_step_params["fit"])
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
base.py", line 1467, in wrapper
    estimator._validate_params()
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
base.py", line 666, in _validate_params
    validate_parameter_constraints(
  File "c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
utils\_param_validation.py", line 95, in
validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'penalty'
parameter of LogisticRegression must be a str among {'l2',
'elasticnet', 'l1'} or None. Got 'none' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
c:\Users\YAWE MANOLO\anaconda3\Lib\site-packages\sklearn\
model_selection\_search.py:1051: UserWarning: One or more of the test
scores are non-finite: [0.8604     0.85933333 0.84426667 0.84453333
0.8604     0.85933333
 0.84426667 0.8444            nan        nan        nan        nan
        nan        nan        nan        nan 0.86973333 0.87013333
 0.86706667 0.8672     0.87013333 0.87026667 0.8672     0.8672
        nan        nan        nan        nan        nan        nan
        nan        nan 0.8704     0.87013333 0.86946667 0.86933333
 0.87026667 0.87026667 0.8692     0.8692            nan        nan
```

```
        nan        nan        nan        nan        nan        nan
 0.8444     0.84453333 0.84453333 0.84453333 0.84453333 0.84453333
 0.86013333 0.86013333 0.86013333 0.87213333 0.87213333 0.87226667
 0.87226667 0.87226667 0.87226667 0.87226667 0.87146667 0.87146667
 0.87146667 0.87146667 0.87146667 0.87106667 0.87106667 0.87133333
 0.87133333 0.87133333 0.87106667 0.87106667 0.87106667 0.87106667
 0.8444     0.8444     0.8444     0.8444     0.84453333 0.84453333
 0.8604     0.86013333 0.86       0.87226667 0.87213333 0.87226667
 0.87226667 0.87226667 0.87226667 0.87226667 0.87106667 0.87133333
 0.8712     0.87133333 0.87133333 0.8708     0.87066667 0.87106667
 0.87093333 0.87106667 0.87133333 0.87093333 0.8708     0.87106667
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.86693333 0.8672     0.8672     0.8672     0.8672     0.8672
 0.87706667 0.87706667 0.87706667 0.8836     0.8836     0.88333333
 0.88333333 0.88333333 0.88333333 0.88333333 0.88346667 0.88346667
 0.88346667 0.88346667 0.88346667 0.8832     0.8832     0.8832
 0.8832     0.8832     0.8832     0.8832     0.8832     0.8832
 0.8672     0.8672     0.8672     0.8672     0.8672     0.8672
 0.87706667 0.87706667 0.87706667 0.8836     0.8836     0.88306667
 0.88306667 0.88306667 0.88306667 0.88306667 0.88346667 0.88346667
 0.88346667 0.88346667 0.88346667 0.88333333 0.88333333 0.88333333
 0.88333333 0.88333333 0.88266667 0.88266667 0.88266667 0.88266667
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.86933333 0.86933333 0.86933333 0.86933333 0.86933333 0.86933333
 0.8784     0.8784     0.8784     0.8852     0.8852     0.88506667
 0.88506667 0.88506667 0.88506667 0.88506667 0.88466667 0.88466667
 0.88466667 0.88466667 0.88466667 0.8844     0.8844     0.88453333
 0.88453333 0.88453333 0.88453333 0.88453333 0.88453333 0.88453333
 0.8692     0.8692     0.8692     0.8692     0.8692     0.8692
 0.8784     0.8784     0.8784     0.8852     0.8852     0.8852
 0.8852     0.8852     0.8852     0.8852     0.8848     0.8848
```

```
 0.8848     0.8848     0.8848     0.8844     0.8844     0.88453333
 0.88453333 0.88453333 0.8844     0.8844     0.8844     0.8844
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.8692     0.87013333 0.86733333 0.86746667 0.8692     0.87013333
 0.8672     0.8676            nan        nan        nan        nan
        nan        nan        nan        nan 0.87013333 0.8704
 0.8696     0.86946667 0.87       0.87026667 0.86946667 0.86946667
        nan        nan        nan        nan        nan        nan
        nan        nan 0.87026667 0.87026667 0.87       0.87
 0.87026667 0.87026667 0.87       0.87              nan        nan
        nan        nan        nan        nan        nan        nan
 0.86733333 0.86746667 0.86746667 0.86746667 0.86746667 0.86746667
 0.86746667 0.86746667 0.86893333 0.88386667 0.88386667 0.88373333
 0.88373333 0.88373333 0.88373333 0.88373333 0.88386667 0.88386667
 0.88386667 0.88386667 0.88386667 0.8836     0.8836     0.8836
 0.8836     0.8836     0.88266667 0.88266667 0.88266667 0.88266667
 0.86746667 0.8672     0.86733333 0.86746667 0.86746667 0.8676
 0.86746667 0.86746667 0.86893333 0.884      0.88386667 0.88373333
 0.88386667 0.88373333 0.88373333 0.88386667 0.88386667 0.884
 0.88386667 0.88386667 0.88386667 0.88346667 0.88346667 0.88346667
 0.88346667 0.8836     0.88266667 0.88266667 0.88253333 0.8828
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.8696     0.86946667 0.86946667 0.86946667 0.86946667 0.86946667
 0.86946667 0.86946667 0.8712     0.8852     0.8852     0.88533333
 0.88533333 0.88533333 0.88533333 0.88533333 0.88506667 0.88506667
 0.88506667 0.88506667 0.88506667 0.88453333 0.88453333 0.88466667
 0.88466667 0.88466667 0.88453333 0.88453333 0.88453333 0.88453333
 0.86946667 0.86946667 0.86946667 0.86946667 0.86946667 0.86946667
 0.86946667 0.86946667 0.8712     0.88533333 0.88533333 0.88533333
 0.88533333 0.88533333 0.88533333 0.88533333 0.88506667 0.88506667
 0.88506667 0.88506667 0.88506667 0.8844     0.88453333 0.88466667
```

```
 0.88466667 0.88453333 0.88466667 0.88466667 0.88466667 0.88466667
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.87       0.87       0.87       0.87       0.87       0.87
 0.87       0.87       0.8716     0.886      0.886      0.88613333
 0.88613333 0.88613333 0.88613333 0.88613333 0.886      0.886
 0.886      0.886      0.886      0.8852     0.8852     0.88506667
 0.88506667 0.88506667 0.88493333 0.88493333 0.88493333 0.88493333
 0.87       0.87       0.87       0.87       0.87       0.87
 0.87       0.87       0.8716     0.886      0.886      0.886
 0.886      0.886      0.886      0.886      0.886      0.886
 0.886      0.886      0.886      0.8852     0.8852     0.88506667
 0.88506667 0.88506667 0.8848     0.8848     0.8848     0.8848
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan]
  warnings.warn(

#tampilkan skor cross-validation
print("CV Score : {}".format(GSCV_LogReg.best_score_))
#tampilkan skor model terbaik GSCV pada test set
print("Test Score: 
{}".format(GSCV_LogReg.best_estimator_.score(x_test_category_enc, 
y_test_category)))
#tampilkan best model dan best features
print("Best model:", GSCV_LogReg.best_estimator_)
mask = 
GSCV_LogReg.best_estimator_.named_steps['feat_select'].get_support()
print("Best features:", df_train_enc.columns[mask])

#buat prediksi dari test set
LogReg_pred = GSCV_LogReg.predict(x_test_category_enc)

import matplotlib.pyplot as plt
#buat confusion matrix
```

```
cm = confusion_matrix(y_test_category, LogReg_pred,
labels=GSCV_LogReg.classes_)
#buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_LogReg.classes_)
disp.plot()
plt.title("Logistic Regression Confusion Matrix")
plt.show()

#tampilkan classification report
print("Classification report Logistic Regression:\n",
classification_report(y_test_category, LogReg_pred))
```

```
----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[1], line 2
      1 #tampilkan skor cross-validation
----> 2 print("CV Score : {}".format(GSCV_LogReg.best_score_))
      3 #tampilkan skor model terbaik GSCV pada test set
      4 print("Test Score:
{}".format(GSCV_LogReg.best_estimator_.score(x_test_category_enc,
y_test_category)))

NameError: name 'GSCV_LogReg' is not defined
```

```
import pickle

with open('RF_heartDisease_model.pkl','wb') as r:
    pickle.dump((GSCV_RF), r)

print("Model RF berhasil disimpan")
```

```
Model RF berhasil disimpan
```

Tugas Modul Model Training and Evaluation

```
import pandas as pd
import numpy as np

#Baca dataset dengan menggunakan function read_csv dari pandas
df_kategori=pd.read_csv('Dataset UTS_Gasal 2425.csv')
df_kategori.head(10000)
```

```
      squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0            75523              3      no     yes      63      9373
1            55712             58      no     yes      19     34457
2            86929            100     yes      no      11     98155
3            51522              3      no      no      61      9047
4            96470             74     yes      no      21     92029
...            ...            ...     ...     ...     ...       ...
9995           341             83      no      no       8      1960
9996         21514              5      no     yes      11     91373
9997          1726             89      no     yes       5     73133
9998         44403             29     yes     yes      12     34606
9999          1440             84      no      no      49     18412

      citypartrange  numprevowners  made isnewbuilt hasstormprotector
\
0                 3              8  2005        old               yes

1                 6              8  2021        old                no

2                 3              4  2003        new                no

3                 8              3  2012        new               yes

4                 4              2  2011        new               yes

...             ...            ...   ...        ...               ...

9995              4              4  1993        new               yes

9996              1              1  1999        old                no

9997              7              6  2009        old               yes

9998              9              4  1990        old               yes

9999              6             10  1994        new                no


      basement  attic  garage hasstorageroom  hasguestroom      price
category
0         4313   9005     956             no             7  7559081.5
```

```
Luxury
1         2937   8852      135            yes            9  5574642.1
Middle
2         6326   4748      654             no           10  8696869.3
Luxury
3          632   5792      807            yes            5  5154055.2
Middle
4         5414   1172      716            yes            9  9652258.1
Luxury
...        ...    ...      ...            ...          ...        ...
...
9995      2366   4016      229            yes            5    35371.3
Basic
9996      2584   5266      787             no            3  2153602.9
Basic
9997      9311   1698      218             no            4   176425.9
Basic
9998      9061   1742      230             no            0  4448474.0
Basic
9999      8485   2024      278            yes            6   146708.4
Basic

[10000 rows x 18 columns]
```

```
df_kategori2=df_kategori.drop('price',axis=1)
df_kategori2.head(50)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
2          86929            100     yes      no      11     98155
3          51522              3      no      no      61      9047
4          96470             74     yes      no      21     92029
5          79770              3      no     yes      69     54812
6          75985             60     yes      no      67      6517
7          64169             88      no     yes       6     61711
8          92383             12      no      no      78     71982
9          95121             46      no     yes       3      9382
10         76485             47     yes      no       9     90254
11         87060             27      no     yes      91     51803
12         66683             19     yes     yes       6     50801
13         84559             29      no     yes      69     53057
14         76091             38     yes      no      32     59451
15         92696             49     yes      no      38     74381
16         59800             47      no     yes      27     44815
17         54836             25      no     yes      53     64601
18         70021             52     yes      no      28     95678
19         54368             11     yes     yes      20     55761
20         63053              6     yes     yes      28     45312
21         64393              8      no      no      51     95335
```

```
22         93876               60       no      yes       70      5484
23         84016               15      yes       no       55     63595
24         89768               48      yes      yes       17     71000
25         58478                5       no      yes       35      5898
26         66621               48       no       no       89     52165
27         73314               43       no      yes       38     49895
28         59972               28       no      yes       18     32083
29         71591               20      yes       no       58     46834
30         67311               67       no       no       10     45626
31         61534               73      yes       no       97     22943
32         84091               50       no      yes       72     22718
33         51434               64       no       no       23     79754
34         78960               55       no      yes       76     23408
35         81870               60       no      yes      100     58048
36         91559               36       no      yes       21     82521
37         72098                9       no      yes       67     91168
38         55232               23       no       no        8       849
39         53735               49       no      yes       92      2423
40         71397               71       no       no       93     68199
41         65151               81      yes      yes        3     66191
42         61484               91      yes       no       94     87015
43         57160               15      yes      yes       43     40786
44         55933               15       no       no       97      5800
45         81936               68       no       no       92      6143
46         99683               12      yes       no       77     18300
47         62887               45       no      yes        7     91125
48         73062               34      yes      yes       38     44770
49         84284               76       no       no       39     55723

    citypartrange  numprevowners  made isnewbuilt hasstormprotector  
basement  \
0               3              8  2005        old               yes   
4313
1               6              8  2021        old                no   
2937
2               3              4  2003        new                no   
6326
3               8              3  2012        new               yes   
632
4               4              2  2011        new               yes   
5414
5              10              5  2018        old               yes   
8871
6               6              9  2009        new               yes   
4878
7               3              9  2011        new               yes   
3054
8               3              7  2000        old                no   
7507
```

```
9                7              9  1994        old               no
615
10               2              9  2008        new               no
2860
11               8             10  2000        old               no
6629
12               6              2  2001        old               no
7473
13               7              7  2000        new               no
3573
14               5              8  2016        new               no
8150
15               9              2  2021        old               no
1559
16               6              9  2021        old               no
5075
17              10              5  2020        new               no
5278
18               4              6  1992        old              yes
4480
19               3              7  2021        old               no
231
20               3              1  1997        old              yes
8414
21               4              1  1990        new               no
3835
22               2              1  1999        new              yes
4086
23               1              7  2016        new               no
3284
24               6              9  1993        old              yes
2485
25               6             10  2016        old               no
8366
26              10              1  1995        new              yes
5024
27              10              1  2018        old              yes
3281
28               9              8  2021        new              yes
8384
29               7              4  1998        old               no
6486
30               3              3  1990        new              yes
6928
31               9              5  2001        old               no
9265
32               7              5  1993        old               no
2668
33              10              2  2012        new              yes
```

```
2080
34              8             4  2015        new              yes
7126
35              3             8  2020        old               no
3632
36              6             2  2007        old              yes
788
37              2             3  2014        new               no
9080
38              8             3  1991        old               no
1492
39              4             7  2006        old               no
8654
40              3            10  1995        new              yes
3477
41              7             9  1991        old               no
3218
42              8             8  2013        new               no
5486
43              8             8  2002        old               no
4018
44              9             8  2001        new              yes
7369
45              3             1  2011        new               no
6393
46              5             8  2002        old              yes
4034
47              4             3  1993        new               no
292
48              4             8  2016        old               no
9823
49              5             1  1998        old               no
4500

    attic  garage hasstorageroom  hasguestroom category
0    9005     956             no             7   Luxury
1    8852     135            yes             9   Middle
2    4748     654             no            10   Luxury
3    5792     807            yes             5   Middle
4    1172     716            yes             9   Luxury
5    7117     240             no             7   Luxury
6     281     384            yes             5   Luxury
7     129     726             no             9   Middle
8    9056     892            yes             1   Luxury
9    1221     328             no            10   Luxury
10   3129     982             no             1   Luxury
11    435     512             no             7   Luxury
12    796     237            yes             3   Middle
13   9556     918            yes             8   Luxury
```

```
14   6037     930              no             7   Luxury
15   5111     957             yes             2   Luxury
16   3104     864              no             4   Middle
17   1059     313             yes             6   Middle
18   6919     680             yes             1   Luxury
19   1939     223              no             8   Middle
20   6270     939             yes             8   Middle
21   2403     559              no             6   Middle
22   5991     494             yes             8   Luxury
23   9879     641              no             2   Luxury
24    108     864              no             7   Luxury
25   4799     979             yes             7   Middle
26   8103     388             yes             4   Middle
27   5020     968              no             8   Luxury
28   7226     226             yes             4   Middle
29   3310     366              no             0   Luxury
30   7808     774             yes             5   Middle
31   8974     755             yes             6   Middle
32   4669     766              no             8   Luxury
33   9575     753              no             7   Middle
34   5012     974             yes             0   Luxury
35   5960     723             yes             3   Luxury
36   4788     132             yes             8   Luxury
37   9356     740             yes             9   Luxury
38   5697     625              no             6   Middle
39   9588     290             yes             8   Middle
40   5530     342              no             2   Luxury
41   9119     849              no             4   Middle
42   3641     766              no             3   Middle
43   4871     836             yes             2   Middle
44   6739     686             yes             6   Middle
45   9082     734              no             0   Luxury
46   2877     787             yes             6   Luxury
47    744     675              no             4   Middle
48   7174     728             yes             0   Luxury
49   4877     480              no             6   Luxury
```

```
df_kategori2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   squaremeters       10000 non-null  int64
 1   numberofrooms      10000 non-null  int64
 2   hasyard            10000 non-null  object
 3   haspool            10000 non-null  object
 4   floors             10000 non-null  int64
 5   citycode           10000 non-null  int64
```

```
 6   citypartrange      10000 non-null  int64
 7   numprevowners      10000 non-null  int64
 8   made               10000 non-null  int64
 9   isnewbuilt         10000 non-null  object
 10  hasstormprotector  10000 non-null  object
 11  basement           10000 non-null  int64
 12  attic              10000 non-null  int64
 13  garage             10000 non-null  int64
 14  hasstorageroom     10000 non-null  object
 15  hasguestroom       10000 non-null  int64
 16  category           10000 non-null  object
dtypes: int64(11), object(6)
memory usage: 1.3+ MB
```

```python
#cek deskripsi data
df_kategori2.describe()
```

```
       squaremeters  numberofrooms        floors      citycode  
citypartrange  \
count   10000.00000   10000.000000  10000.000000  10000.000000  
10000.000000
mean    49870.13120      50.358400     50.276300  50225.486100  
5.510100
std     28774.37535      28.816696     28.889171  29006.675799  
2.872024
min        89.00000       1.000000      1.000000      3.000000  
1.000000
25%     25098.50000      25.000000     25.000000  24693.750000  
3.000000
50%     50105.50000      50.000000     50.000000  50693.000000  
5.000000
75%     74609.75000      75.000000     76.000000  75683.250000  
8.000000
max     99999.00000     100.000000    100.000000  99953.000000  
10.000000

       numprevowners         made      basement         attic  
garage  \
count   10000.000000  10000.00000  10000.000000  10000.00000  
10000.00000
mean        5.521700   2005.48850   5033.103900   5028.01060  
553.12120
std         2.856667      9.30809   2876.729545   2894.33221  
262.05017
min         1.000000   1990.00000      0.000000      1.00000  
100.00000
25%         3.000000   1997.00000   2559.750000   2512.00000  
327.75000
50%         5.000000   2005.50000   5092.500000   5045.00000  
554.00000
```

```
75%          8.000000   2014.00000   7511.250000   7540.50000
777.25000
max         10.000000   2021.00000  10000.000000  10000.00000
1000.00000

       hasguestroom
count   10000.00000
mean        4.99460
std         3.17641
min         0.00000
25%         2.00000
50%         5.00000
75%         8.00000
max        10.00000
```

```
df_kategori2['category'].value_counts()
```

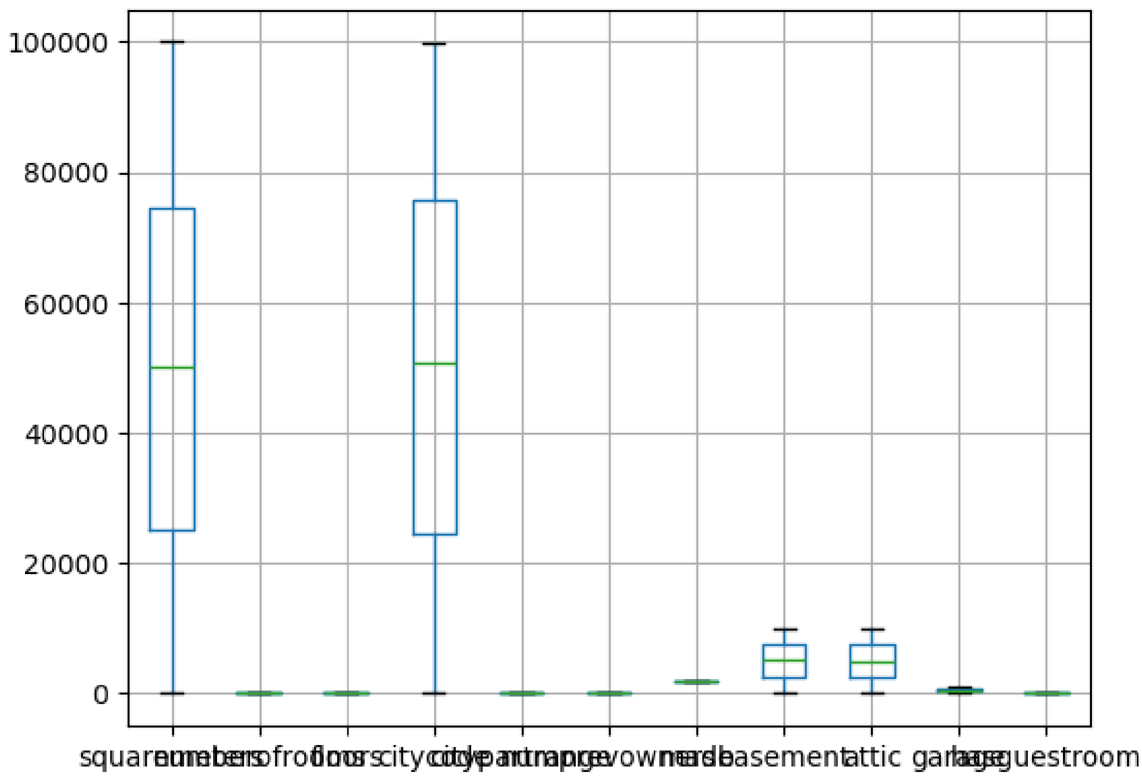
```
category
Basic      4344
Luxury     3065
Middle     2591
Name: count, dtype: int64
```

```
#gunakan fungsi isnull, empty, dan isna untuk mengecek data kosong
print("data null \n", df_kategori2.isnull().sum())
print("\ndata kosong \n", df_kategori2.empty)
print("\ndata nan \n", df_kategori2.isna().sum())
```

```
data null
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
dtype: int64

data kosong
 False
```

```
data nan
 squaremeters          0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
category             0
dtype: int64
```

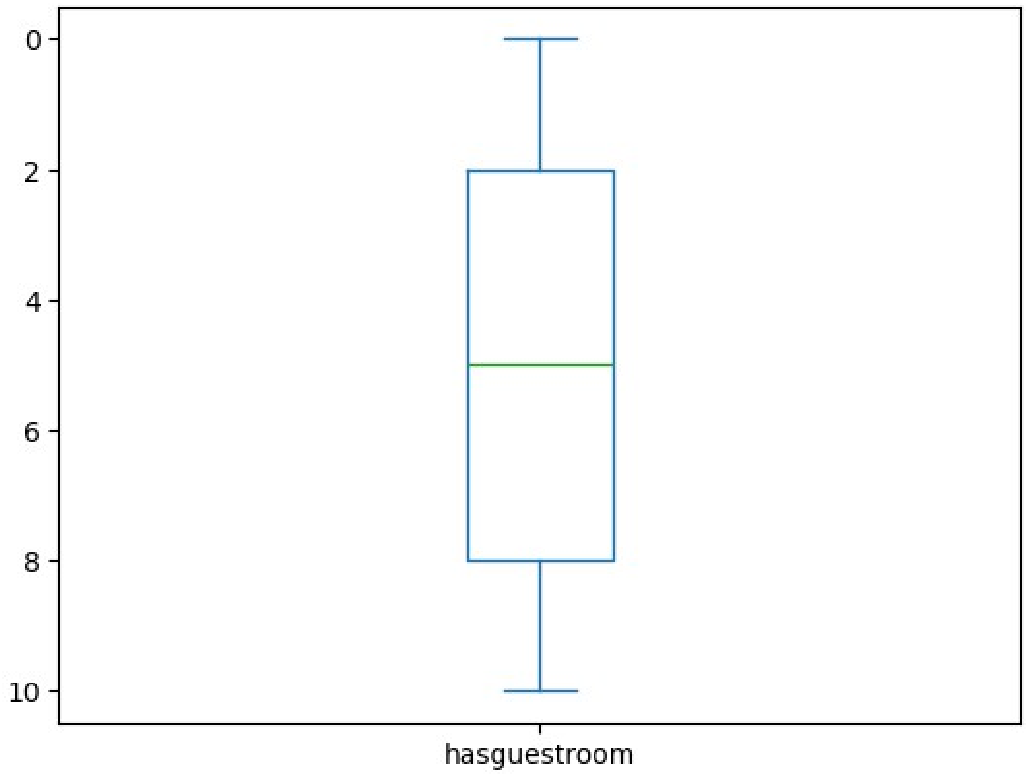
```
#cek data outlier
import matplotlib.pyplot as plt

df_kategori2.boxplot()
plt.show()
```




```
#menggunakan fungsi remove_outliner untuk menghilangkan outlier
from pandas.api.types import is_numeric_dtype
def remove_outlier(df_in):
    for col_name in list(df_in.columns):
        if is_numeric_dtype(df_in[col_name]):
            q1 = df_in[col_name].quantile(0.25)
            q3 = df_in[col_name].quantile(0.75)

            iqr = q3-q1
            batas_atas = q3 + (1.5*iqr)
            batas_bawah = q1 - (1.5*iqr)

            df_out = df_in.loc[(df_in[col_name] >= batas_bawah) &
(df_in[col_name] <= batas_atas)]
    return df_out

df_kategori_clean = remove_outlier(df_kategori2)
print("jumlah baris DataFrame sebelum dibuang outlier",
df_kategori2.shape[0])
print("jumlah baris DataFrame sebelum dibuang outlier",
df_kategori_clean.shape[0])
df_kategori_clean.hasguestroom.plot(kind='box', vert=True)

#untuk membalik sumbu y
```

```
plt.gca().invert_yaxis()
plt.show()
```

```
jumlah baris DataFrame sebelum dibuang outlier 10000
jumlah baris DataFrame sebelum dibuang outlier 10000
```




```
print(df_kategori_clean.dtypes)
```

```
squaremeters          int64
numberofrooms         int64
hasyard              object
haspool              object
floors                int64
citycode              int64
citypartrange         int64
numprevowners         int64
made                  int64
isnewbuilt           object
hasstormprotector    object
basement              int64
attic                 int64
garage                int64
hasstorageroom       object
hasguestroom          int64
```

```
category               object
dtype: object
```

Drop data yang kosong (Missing value)

```
#drop data yang memiliki missing value
print("Sebelum drop missing value",df_kategori2.shape)
df_kategori2 = df_kategori2.dropna(how="any",inplace=False)
print("sesudah drop missing value", df_kategori2.shape)
```

```
Sebelum drop missing value (10000, 17)
sesudah drop missing value (10000, 17)
```

pengecekan data duplikat

```
#pengecekan data duplikat
print("Sebelum Penegcekan data duplikat, ", df_kategori2.shape)
df_kategori3=df_kategori2.drop_duplicates(keep='last')
print("Setelah Pengecekan data duplikat, ", df_kategori3.shape)
```

```
Sebelum Penegcekan data duplikat,  (10000, 17)
Setelah Pengecekan data duplikat,  (10000, 17)
```

```
from sklearn.model_selection import train_test_split
x=df_kategori3.drop(columns=['category'],axis=1)
y= df_kategori3['category']

x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.25, random_state=78)

print(x_train.shape)
print(x_test.shape)
```

```
(7500, 16)
(2500, 16)
```

data encoding

```
#import oneHotEncoder dan make_column_transformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

#tentukan kolom kategorik yang akan diubah
kolom_kategori=['hasyard','haspool','isnewbuilt','hasstormprotector',
'hasstorageroom' ]

#buat objek transformer yang berisi OneHotEncoder
transform = make_column_transformer(
    (OneHotEncoder(), kolom_kategori), remainder='passthrough'
)
```

```python
#buat varibel baru untk menampung hadil transformasi kolom
x_train_enc=transform. fit_transform(x_train)
#khusus untuk train set gunakan fungsi fit_transform, untuk test set
gunakan transform saja
x_test_enc=transform. fit_transform(x_test)

#jika ingin melihat hasil dari transformasi, muat dalam dataframe
df_train_enc=pd.DataFrame(x_train_enc,
columns=transform.get_feature_names_out())
df_test_enc=pd.DataFrame(x_test_enc,
columns=transform.get_feature_names_out())

df_train_enc.head(10)
df_test_enc.head(10)
```

```
   onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                        1.0                         0.0
1                        0.0                         1.0
2                        1.0                         0.0
3                        1.0                         0.0
4                        0.0                         1.0
5                        0.0                         1.0
6                        0.0                         1.0
7                        0.0                         1.0
8                        0.0                         1.0
9                        1.0                         0.0

   onehotencoder__haspool_no  onehotencoder__haspool_yes  \
0                        0.0                         1.0
1                        0.0                         1.0
2                        1.0                         0.0
3                        0.0                         1.0
4                        1.0                         0.0
5                        0.0                         1.0
6                        0.0                         1.0
7                        0.0                         1.0
8                        1.0                         0.0
9                        1.0                         0.0

   onehotencoder__isnewbuilt_new  onehotencoder__isnewbuilt_old  \
0                            0.0                            1.0
1                            0.0                            1.0
2                            0.0                            1.0
3                            1.0                            0.0
4                            1.0                            0.0
5                            0.0                            1.0
6                            0.0                            1.0
7                            0.0                            1.0
8                            1.0                            0.0
9                            1.0                            0.0
```

```
   onehotencoder__hasstormprotector_no
onehotencoder__hasstormprotector_yes  \
0                                    1.0
0.0
1                                    0.0
1.0
2                                    1.0
0.0
3                                    1.0
0.0
4                                    1.0
0.0
5                                    0.0
1.0
6                                    0.0
1.0
7                                    0.0
1.0
8                                    0.0
1.0
9                                    1.0
0.0

   onehotencoder__hasstorageroom_no  onehotencoder__hasstorageroom_yes
...  \
0                                 1.0                                  0.0
...
1                                 0.0                                  1.0
...
2                                 0.0                                  1.0
...
3                                 0.0                                  1.0
...
4                                 1.0                                  0.0
...
5                                 0.0                                  1.0
...
6                                 0.0                                  1.0
...
7                                 0.0                                  1.0
...
8                                 0.0                                  1.0
...
9                                 0.0                                  1.0
...

   remainder__numberofrooms  remainder__floors  remainder__citycode  \
0                      73.0               13.0              42855.0
1                      95.0                3.0              75381.0
```

```
2                        39.0                  8.0                 91674.0
3                        47.0                 63.0                 58471.0
4                        64.0                 83.0                 30779.0
5                        91.0                 67.0                 65183.0
6                        48.0                 70.0                 21012.0
7                        93.0                 35.0                 12062.0
8                        74.0                 14.0                 76662.0
9                        72.0                 76.0                 87732.0

   remainder__citypartrange  remainder__numprevowners  remainder__made
\
0                       9.0                       6.0           2015.0

1                       5.0                       6.0           2003.0

2                       2.0                       2.0           2009.0

3                      10.0                       1.0           1990.0

4                       6.0                       4.0           1992.0

5                       1.0                      10.0           2019.0

6                       5.0                       3.0           2007.0

7                      10.0                       9.0           1998.0

8                       9.0                       6.0           2004.0

9                       2.0                       8.0           2017.0


   remainder__basement  remainder__attic  remainder__garage  \
0               2560.0            6823.0              239.0
1               6810.0            1391.0              556.0
2               1477.0            3153.0              952.0
3               1730.0            7967.0              722.0
4                594.0            8310.0              898.0
5               6824.0            7141.0              956.0
6               5861.0            2750.0              652.0
7               2064.0            2720.0              315.0
8               9501.0            9579.0              768.0
9               6414.0            6111.0              613.0

   remainder__hasguestroom
0                     10.0
1                      3.0
2                      2.0
3                      0.0
4                      5.0
```

```
5                       6.0
6                      10.0
7                       5.0
8                       1.0
9                       7.0

[10 rows x 21 columns]
```

Pipeline

```
#import Library yang dibutuhkan untuk pipeline, GSCV, dan metrik
evaluasi
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectPercentile, SelectKBest
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay

#buat rancangan pipeline mulai dari data scaling hingga classifier
pipe_svm = Pipeline(steps=[
    ('scale', MinMaxScaler()),
    ('feat_select', SelectKBest()),
    ('clf', SVC(class_weight='balanced'))
])

#buat parameter grid untuk step feature selection dan classifier
params_grid_svm = [
    {
    'scale': [MinMaxScaler()],
    'feat_select__k':np.arange(2,6),
    'clf__kernel': ['poly', 'rbf' ],
    'clf__C':[0.1,1],
    'clf__gamma':[0.1, 1]
    },
    {

    'scale': [MinMaxScaler()],
    'feat_select': [SelectPercentile()],
    'feat_select__percentile':np.arange(20,50),
    'clf__kernel': ['poly','rbf' ],
    'clf__C':[ 0.1, 1],
    'clf__gamma': [0.1, 1]
    },
    {
    'scale': [StandardScaler() ],
    'feat_select__k':np.arange(2,6),
    'clf__kernel': ['poly','rbf'],
```

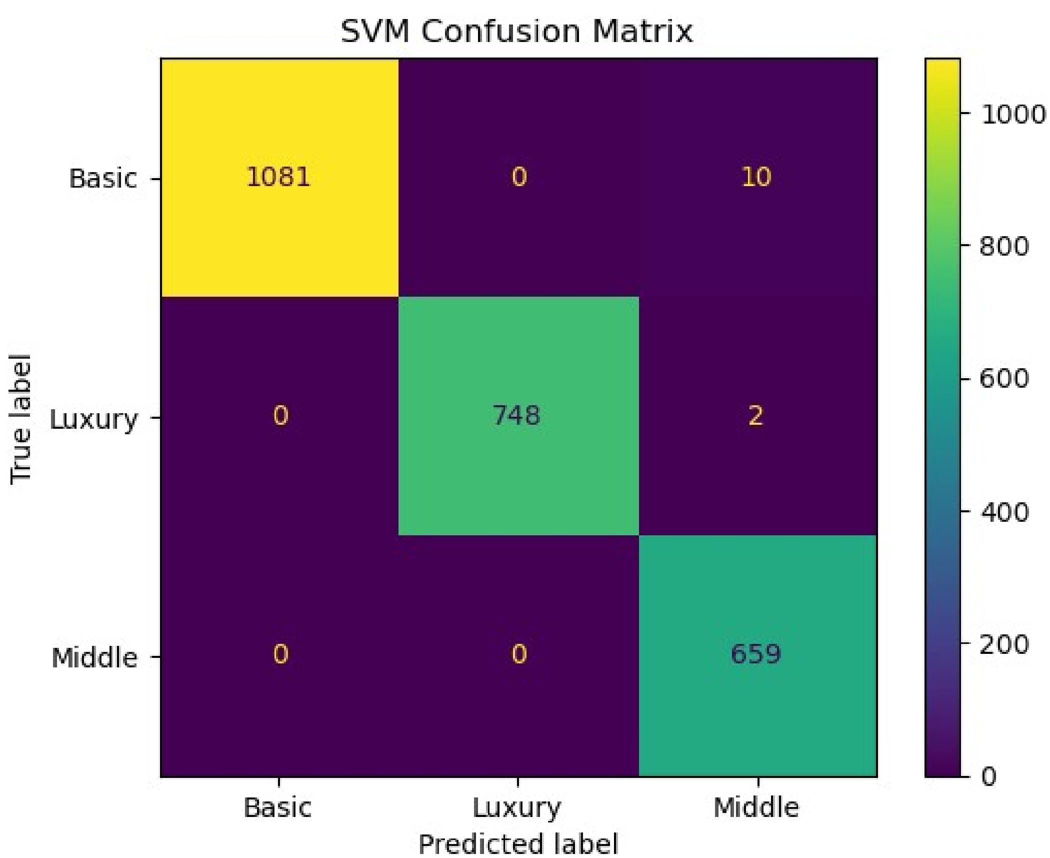
```
    'clf__C':[0.1, 1],
    'clf__gamma':[0.1, 1]

    },
    {
    'scale': [StandardScaler() ],
    'feat_select':[SelectPercentile()],
    'feat_select__percentile': np.arange(20,50),
    'clf__kernel':['poly','rbf'],
    'clf__C':[0.1, 1],
    'clf__gamma':[0.1, 1]

    }
]

#muat rancangan pipeline ke dalam objek pipeline
estimator_svm = Pipeline(pipe_svm)
#muat pipeline dan parameter gri ke dalam objek GSCV dengan Stratified
5-fold CV
SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=78)

GSCV_SVM = GridSearchCV(pipe_svm, params_grid_svm, cv=SKF)
#jalankan objek GSC untuk melatih model dengan train set menggunakan
fungsi fit
GSCV_SVM.fit(x_train_enc, y_train)
print("GSCV training finished")
#64min
```

```
GSCV training finished
```

```
#tampilkan skor cross-validation
print("CV Score : {}".format(GSCV_SVM.best_score_))
#tampilkan skor model terbaik GSCV pada test set
print("Test Score:
{}".format(GSCV_SVM.best_estimator_.score(x_test_enc, y_test)))
#tampilkan best model dan best features
print("Best model:", GSCV_SVM.best_estimator_)
mask =
GSCV_SVM.best_estimator_.named_steps['feat_select' ].get_support()
print("Best features:", df_train_enc.columns[mask])

#buat prediksi dari test set
SVM_pred = GSCV_SVM.predict(x_test_enc)

import matplotlib.pyplot as plt
#buat confusion matrix
cm = confusion_matrix(y_test, SVM_pred, labels=GSCV_SVM.classes_)
#buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_SVM.classes_)
```

```
disp.plot()
plt.title("SVM Confusion Matrix")
plt.show()

#tampilkan classification report
print("Classification report SVM:\n", classification_report(y_test,
SVM_pred))
```

```
CV Score : 0.9941333333333333
Test Score: 0.9952
Best model: Pipeline(steps=[('scale', StandardScaler()),
                ('feat_select', SelectPercentile(percentile=31)),
                ('clf',
                 SVC(C=1, class_weight='balanced', gamma=1,
kernel='poly'))])
Best features: Index(['onehotencoder__hasyard_no',
'onehotencoder__hasyard_yes',
       'onehotencoder__haspool_no', 'onehotencoder__haspool_yes',
       'onehotencoder__isnewbuilt_new',
'onehotencoder__isnewbuilt_old',
       'remainder__squaremeters'],
      dtype='object')
```



```
Classification report SVM:
               precision    recall  f1-score   support

       Basic       1.00      0.99      1.00      1091
      Luxury       1.00      1.00      1.00       750
      Middle       0.98      1.00      0.99       659

    accuracy                           1.00      2500
   macro avg       0.99      1.00      1.00      2500
weighted avg       1.00      1.00      1.00      2500
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn. feature_selection import SelectFromModel
from sklearn. tree import DecisionTreeClassifier

pipe_GBT = Pipeline(steps=[
    ('feat_select', SelectKBest()),
    ('clf', GradientBoostingClassifier(random_state=78))])##random
state isi dengan 2

params_grid_GBT = [
                    {
                        'feat_select__k': np.arange(2,6),
                        'clf__max_depth': [*np.arange(4,5)],
                        'clf__n_estimators': [100,150],
                        'clf__learning_rate': [0.01,0.1,1]
                    },
                    {
                        'feat_select': [SelectPercentile()],
                        'feat_select__percentile':np.arange(20,50),
                        'clf__max_depth': [*np.arange(4,5)],
                        'clf__n_estimators': [100,150],
                        'clf__learning_rate': [0.01,0.1,1]

                    },
                    {

                        'feat_select__k': np.arange(2,6),
                        'clf__max_depth': [*np.arange(4,5)],
                        'clf__n_estimators': [100,150],
                        'clf__learning_rate': [0.01,0.1,1]
                    },

                    {

                        'feat_select':[SelectPercentile()],
                        'feat_select__percentile':np.arange(20,50),
                        'clf__max_depth': [*np.arange(4,5)],
                        'clf__n_estimators': [100,150],
```

```
                        'clf__learning_rate':[0.01,0.1,1]

                    }

]

GSCV_GBT =
GridSearchCV(pipe_GBT,params_grid_GBT,cv=StratifiedKFold(n_splits=5))
GSCV_GBT.fit(x_train_enc,y_train)

print("GSCV Finished")

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[4], line 5
      2 from sklearn. feature_selection import SelectFromModel
      3 from sklearn. tree import DecisionTreeClassifier
----> 5 pipe_GBT = Pipeline(steps=[
      6     ('feat_select', SelectKBest()),
      7     ('clf',
GradientBoostingClassifier(random_state=78))])##random state isi
dengan 2
      9 params_grid_GBT = [
     10                     {
     11                         'feat_select__k': np.arange(2,6),
   (...)
     41
     42 ]
     44 GSCV_GBT =
GridSearchCV(pipe_GBT,params_grid_GBT,cv=StratifiedKFold(n_splits=5))

NameError: name 'Pipeline' is not defined

#tampilkan skor cross-validation
print("CV Score: {}".format(GSCV_GBT.best_score_))
#tampilkan skor model terbaik GSCV pada test set
print("Test Score:
{}".format(GSCV_GBT.best_estimator_.score(x_test_enc, y_test)))
#tampilkan best model dan best features
print("Best model:", GSCV_GBT.best_estimator_)

mask =
GSCV_GBT.best_estimator_.named_steps['feat_select' ].get_support()
print("Best features:", df_train_enc.columns[mask])

#buat prediksi dari test set
GBT_pred = GSCV_GBT.predict(x_test_enc)
```

```
import matplotlib.pyplot as plt
#buat confusion matrix
cm = confusion_matrix(y_test, GBT_pred, labels=GSCV_GBT.classes_)
#buat confusion matrix display
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=GSCV_GBT.classes_)
disp.plot()

plt.title("GBT Confusion Matrix")
plt.show()
#tampilkan Classification report
print("Classification report GBT: \n", classification_report(y_test,
GBT_pred))

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[3], line 2
      1 #tampilkan skor cross-validation
----> 2 print("CV Score: {}".format(GSCV_GBT.best_score_))
      3 #tampilkan skor model terbaik GSCV pada test set
      4 print("Test Score:
{}".format(GSCV_GBT.best_estimator_.score(x_test_enc, y_test)))

NameError: name 'GSCV_GBT' is not defined

import pickle

#simpan model menggunakan library Pickle
with open('BestModel_CLF_GBT_pytorch.pkl','wb') as r:
    pickle.dump((GSCV_GBT),r)

#File pickle akan tersimpan di folder yang sama dengan file notebook
print("Model GBT berhasil disimpan")

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[2], line 5
      3 #simpan model menggunakan library Pickle
      4 with open('BestModel_CLF_GBT_pytorch.pkl','wb') as r:
----> 5     pickle.dump((GSCV_GBT),r)
      7 #File pickle akan tersimpan di folder yang sama dengan file
notebook
      8 print("Model GBT berhasil disimpan")

NameError: name 'GSCV_GBT' is not defined
```

```python
import pandas as pd
import numpy as np

df_harga =pd.read_csv(r'D:\semester 5\Mesin Learning\UTS\Dataset
UTS_Gasal 2425.csv')
df_harga.head(20)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
2          86929            100     yes      no      11     98155
3          51522              3      no      no      61      9047
4          96470             74     yes      no      21     92029
5          79770              3      no     yes      69     54812
6          75985             60     yes      no      67      6517
7          64169             88      no     yes       6     61711
8          92383             12      no      no      78     71982
9          95121             46      no     yes       3      9382
10         76485             47     yes      no       9     90254
11         87060             27      no     yes      91     51803
12         66683             19     yes     yes       6     50801
13         84559             29      no     yes      69     53057
14         76091             38     yes      no      32     59451
15         92696             49     yes      no      38     74381
16         59800             47      no     yes      27     44815
17         54836             25      no     yes      53     64601
18         70021             52     yes      no      28     95678
19         54368             11     yes     yes      20     55761

    citypartrange  numprevowners  made isnewbuilt hasstormprotector
basement  \
0               3              8  2005        old               yes
4313
1               6              8  2021        old                no
2937
2               3              4  2003        new                no
6326
3               8              3  2012        new               yes
632
4               4              2  2011        new               yes
5414
5              10              5  2018        old               yes
8871
6               6              9  2009        new               yes
4878
7               3              9  2011        new               yes
3054
8               3              7  2000        old                no
7507
9               7              9  1994        old                no
```

```
615
10              2              9  2008        new                  no
2860
11              8             10  2000        old                  no
6629
12              6              2  2001        old                  no
7473
13              7              7  2000        new                  no
3573
14              5              8  2016        new                  no
8150
15              9              2  2021        old                  no
1559
16              6              9  2021        old                  no
5075
17             10              5  2020        new                  no
5278
18              4              6  1992        old                 yes
4480
19              3              7  2021        old                  no
231

    attic  garage hasstorageroom  hasguestroom      price category
0    9005     956             no             7  7559081.5   Luxury
1    8852     135            yes             9  5574642.1   Middle
2    4748     654             no            10  8696869.3   Luxury
3    5792     807            yes             5  5154055.2   Middle
4    1172     716            yes             9  9652258.1   Luxury
5    7117     240             no             7  7986665.8   Luxury
6     281     384            yes             5  7607322.9   Luxury
7     129     726             no             9  6420823.1   Middle
8    9056     892            yes             1  9244344.0   Luxury
9    1221     328             no            10  9515440.4   Luxury
10   3129     982             no             1  7653300.8   Luxury
11    435     512             no             7  8711426.0   Luxury
12    796     237            yes             3  6677649.1   Middle
13   9556     918            yes             8  8460604.0   Luxury
14   6037     930             no             7  7614076.6   Luxury
15   5111     957            yes             2  9272740.1   Luxury
16   3104     864             no             4  5984462.1   Middle
17   1059     313            yes             6  5492532.0   Middle
18   6919     680            yes             1  7005572.2   Luxury
19   1939     223             no             8  5446398.1   Middle
```

```
df_harga2 = df_harga.drop(['category'], axis=1)
df_harga2.head()
```

```
   squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0         75523              3      no     yes      63      9373
1         55712             58      no     yes      19     34457
```

```
2         86929            100     yes      no      11     98155
3         51522              3      no      no      61      9047
4         96470             74     yes      no      21     92029

   citypartrange  numprevowners  made isnewbuilt hasstormprotector
basement  \
0              3              8  2005        old               yes
4313
1              6              8  2021        old                no
2937
2              3              4  2003        new                no
6326
3              8              3  2012        new               yes
632
4              4              2  2011        new               yes
5414

   attic  garage hasstorageroom  hasguestroom      price
0   9005     956             no             7  7559081.5
1   8852     135            yes             9  5574642.1
2   4748     654             no            10  8696869.3
3   5792     807            yes             5  5154055.2
4   1172     716            yes             9  9652258.1
```

```
df_harga2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   squaremeters       10000 non-null  int64
 1   numberofrooms      10000 non-null  int64
 2   hasyard            10000 non-null  object
 3   haspool            10000 non-null  object
 4   floors             10000 non-null  int64
 5   citycode           10000 non-null  int64
 6   citypartrange      10000 non-null  int64
 7   numprevowners      10000 non-null  int64
 8   made               10000 non-null  int64
 9   isnewbuilt         10000 non-null  object
 10  hasstormprotector  10000 non-null  object
 11  basement           10000 non-null  int64
 12  attic              10000 non-null  int64
 13  garage             10000 non-null  int64
 14  hasstorageroom     10000 non-null  object
 15  hasguestroom       10000 non-null  int64
 16  price              10000 non-null  float64
dtypes: float64(1), int64(11), object(5)
memory usage: 1.3+ MB
```

```
df_harga2.describe()
```

```
       squaremeters  numberofrooms        floors      citycode  
citypartrange  \
count   10000.00000   10000.000000  10000.000000  10000.000000  
10000.000000   
mean    49870.13120      50.358400     50.276300  50225.486100  
5.510100   
std     28774.37535      28.816696     28.889171  29006.675799  
2.872024   
min        89.00000       1.000000      1.000000      3.000000  
1.000000   
25%     25098.50000      25.000000     25.000000  24693.750000  
3.000000   
50%     50105.50000      50.000000     50.000000  50693.000000  
5.000000   
75%     74609.75000      75.000000     76.000000  75683.250000  
8.000000   
max     99999.00000     100.000000    100.000000  99953.000000  
10.000000   

       numprevowners          made      basement         attic  
garage  \
count   10000.000000  10000.00000  10000.000000  10000.00000  
10000.00000   
mean        5.521700   2005.48850   5033.103900   5028.01060  
553.12120   
std         2.856667      9.30809   2876.729545   2894.33221  
262.05017   
min         1.000000   1990.00000      0.000000      1.00000  
100.00000   
25%         3.000000   1997.00000   2559.750000   2512.00000  
327.75000   
50%         5.000000   2005.50000   5092.500000   5045.00000  
554.00000   
75%         8.000000   2014.00000   7511.250000   7540.50000  
777.25000   
max        10.000000   2021.00000  10000.000000  10000.00000  
1000.00000   

       hasguestroom         price  
count   10000.00000  1.000000e+04  
mean        4.99460  4.993448e+06  
std         3.17641  2.877424e+06  
min         0.00000  1.031350e+04  
25%         2.00000  2.516402e+06  
50%         5.00000  5.016180e+06  
75%         8.00000  7.469092e+06  
max        10.00000  1.000677e+07  
```

```python
print(df_harga2['price'].value_counts())
```

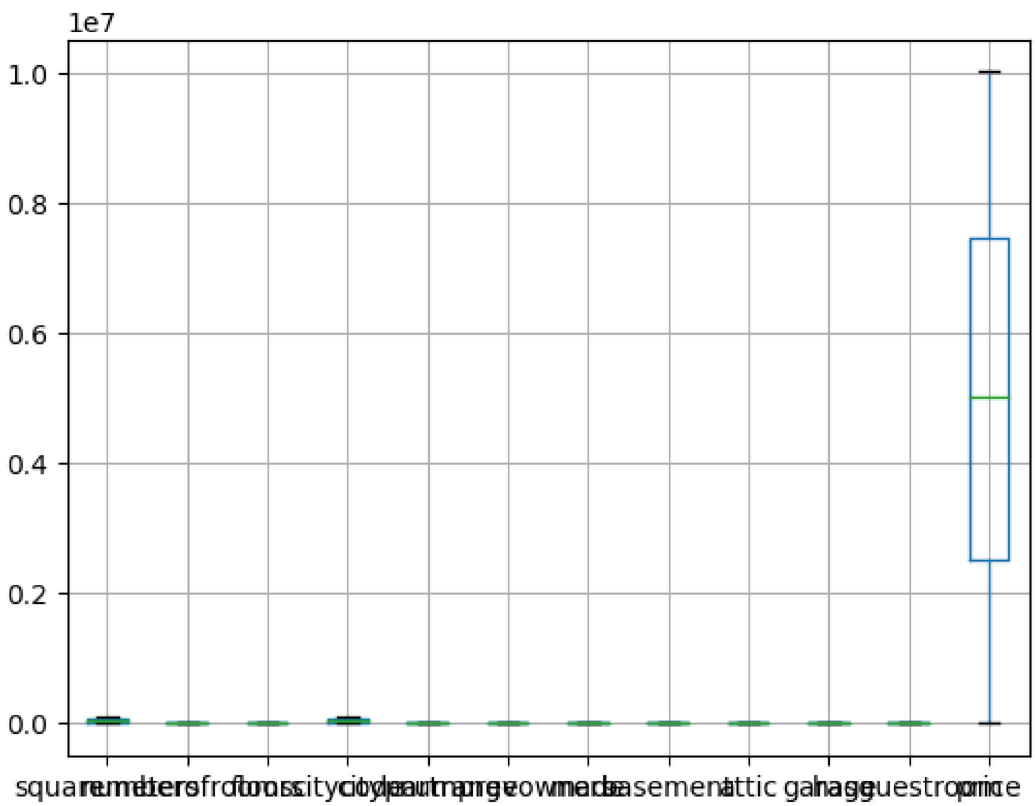
```
price
7559081.5    1
2600292.1    1
3804577.4    1
3658559.7    1
2316639.4    1
            ..
5555606.6    1
5501007.5    1
9986201.2    1
9104801.8    1
146708.4     1
Name: count, Length: 10000, dtype: int64
```

```python
print("data null \n", df_harga2.isnull().sum())
print("data kosong \n", df_harga2.empty)
print("data nan \n", df_harga2.isna().sum())
```

```
data null 
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
price                0
dtype: int64
data kosong 
 False
data nan 
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
```

```
made                     0
isnewbuilt               0
hasstormprotector        0
basement                 0
attic                    0
garage                   0
hasstorageroom           0
hasguestroom             0
price                    0
dtype: int64
```

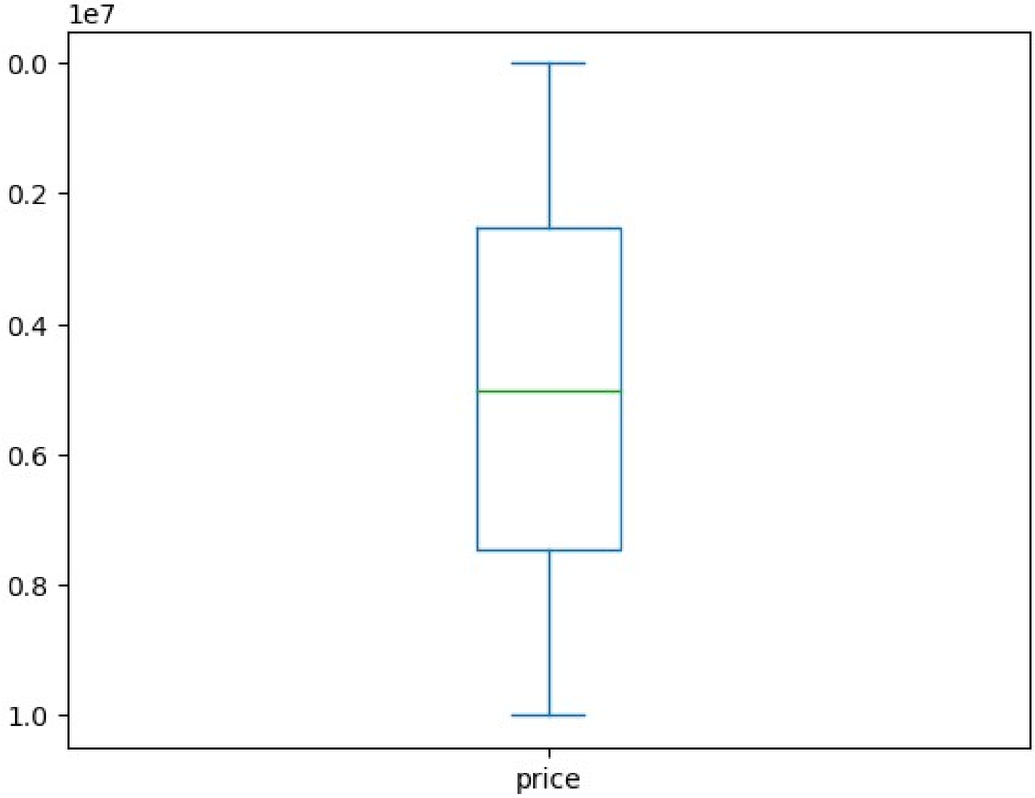
```python
#cek data outlier
import matplotlib.pyplot as plt

df_harga2.boxplot()
plt.show()
```




```python
print("Sebelum pengecekan data duplikat, ", df_harga.shape)
df_harga3 = df_harga2.drop_duplicates(keep = 'last')
print("Setelah pengecekan data duplikat, ", df_harga2.shape)
```

```
Sebelum pengecekan data duplikat,  (10000, 18)
Setelah pengecekan data duplikat,  (10000, 17)
```

```python
import matplotlib.pyplot as plt

df_harga3.price.plot(kind='box')
plt.gca().invert_yaxis()
plt.show()
```




```python
from sklearn.model_selection import train_test_split

x_regress = df_harga3.drop(columns=['price'], axis = 1)
y_regress = df_harga3['price']

x_train_price, x_test_price, y_train_price, y_test_price =
train_test_split(x_regress, y_regress, test_size= 0.25, random_state=
78)

print(x_train_price.shape)
print(x_test_price.shape)
```

```
(7500, 16)
(2500, 16)
```

```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

category_column = ['hasyard', 'haspool', 'isnewbuilt',
```

```
'hasstormprotector', 'hasstorageroom']

transform = make_column_transformer((OneHotEncoder(),
category_column), remainder = 'passthrough')

x_train_price_enc = transform.fit_transform(x_train_price)
x_test_price_enc = transform.fit_transform(x_test_price)

df_train_enc = pd.DataFrame(x_train_price_enc, columns =
transform.get_feature_names_out())
df_test_enc = pd.DataFrame(x_test_price_enc, columns =
transform.get_feature_names_out())

df_train_enc.head(20)
df_test_enc.head(20)
```

```
    onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                         1.0                         0.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         1.0                         0.0
4                         0.0                         1.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         0.0                         1.0
9                         1.0                         0.0
10                        0.0                         1.0
11                        0.0                         1.0
12                        1.0                         0.0
13                        0.0                         1.0
14                        1.0                         0.0
15                        0.0                         1.0
16                        1.0                         0.0
17                        0.0                         1.0
18                        0.0                         1.0
19                        1.0                         0.0

    onehotencoder__haspool_no  onehotencoder__haspool_yes  \
0                         0.0                         1.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         0.0                         1.0
4                         1.0                         0.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         1.0                         0.0
9                         1.0                         0.0
10                        0.0                         1.0
```

```
11                         1.0                           0.0
12                         1.0                           0.0
13                         1.0                           0.0
14                         1.0                           0.0
15                         1.0                           0.0
16                         1.0                           0.0
17                         0.0                           1.0
18                         1.0                           0.0
19                         1.0                           0.0

    onehotencoder__isnewbuilt_new  onehotencoder__isnewbuilt_old  \
0                             0.0                            1.0
1                             0.0                            1.0
2                             0.0                            1.0
3                             1.0                            0.0
4                             1.0                            0.0
5                             0.0                            1.0
6                             0.0                            1.0
7                             0.0                            1.0
8                             1.0                            0.0
9                             1.0                            0.0
10                            0.0                            1.0
11                            0.0                            1.0
12                            1.0                            0.0
13                            1.0                            0.0
14                            0.0                            1.0
15                            1.0                            0.0
16                            1.0                            0.0
17                            1.0                            0.0
18                            1.0                            0.0
19                            1.0                            0.0

    onehotencoder__hasstormprotector_no
onehotencoder__hasstormprotector_yes  \
0                                   1.0
0.0
1                                   0.0
1.0
2                                   1.0
0.0
3                                   1.0
0.0
4                                   1.0
0.0
5                                   0.0
1.0
6                                   0.0
1.0
7                                   0.0
```

```
1.0
8                                    0.0
1.0
9                                    1.0
0.0
10                                   1.0
0.0
11                                   0.0
1.0
12                                   0.0
1.0
13                                   1.0
0.0
14                                   1.0
0.0
15                                   1.0
0.0
16                                   1.0
0.0
17                                   1.0
0.0
18                                   1.0
0.0
19                                   1.0
0.0

    onehotencoder__hasstorageroom_no
onehotencoder__hasstorageroom_yes  ...  \
0                                 1.0
0.0  ...
1                                 0.0
1.0  ...
2                                 0.0
1.0  ...
3                                 0.0
1.0  ...
4                                 1.0
0.0  ...
5                                 0.0
1.0  ...
6                                 0.0
1.0  ...
7                                 0.0
1.0  ...
8                                 0.0
1.0  ...
9                                 0.0
1.0  ...
10                                0.0
```

```
1.0  ...
11                                 1.0
0.0  ...
12                                 0.0
1.0  ...
13                                 0.0
1.0  ...
14                                 1.0
0.0  ...
15                                 0.0
1.0  ...
16                                 0.0
1.0  ...
17                                 1.0
0.0  ...
18                                 0.0
1.0  ...
19                                 0.0
1.0  ...

    remainder__numberofrooms  remainder__floors
remainder__citycode  \
0                       73.0               13.0                42855.0

1                       95.0                3.0                75381.0

2                       39.0                8.0                91674.0

3                       47.0               63.0                58471.0

4                       64.0               83.0                30779.0

5                       91.0               67.0                65183.0

6                       48.0               70.0                21012.0

7                       93.0               35.0                12062.0

8                       74.0               14.0                76662.0

9                       72.0               76.0                87732.0

10                      18.0               66.0                38920.0

11                      37.0               26.0                96016.0

12                      76.0               95.0                76985.0

13                      26.0               99.0                38185.0

14                      69.0               45.0                88591.0
```

```
15                          9.0                 25.0                 33740.0

16                         29.0                 58.0                 13202.0

17                         91.0                 43.0                 93072.0

18                         68.0                 62.0                 97608.0

19                         48.0                 53.0                 34588.0


    remainder__citypartrange  remainder__numprevowners
remainder__made  \
0                        9.0                       6.0
2015.0
1                        5.0                       6.0
2003.0
2                        2.0                       2.0
2009.0
3                       10.0                       1.0
1990.0
4                        6.0                       4.0
1992.0
5                        1.0                      10.0
2019.0
6                        5.0                       3.0
2007.0
7                       10.0                       9.0
1998.0
8                        9.0                       6.0
2004.0
9                        2.0                       8.0
2017.0
10                       7.0                       2.0
2007.0
11                       3.0                       6.0
2000.0
12                       4.0                       7.0
2009.0
13                       7.0                       7.0
1994.0
14                       1.0                       8.0
1995.0
15                       2.0                       1.0
2009.0
16                       6.0                       5.0
1993.0
17                       1.0                       7.0
2012.0
```

```
18                       1.0                        9.0
2001.0
19                       9.0                        6.0
2003.0

    remainder__basement  remainder__attic  remainder__garage  \
0                2560.0            6823.0              239.0
1                6810.0            1391.0              556.0
2                1477.0            3153.0              952.0
3                1730.0            7967.0              722.0
4                 594.0            8310.0              898.0
5                6824.0            7141.0              956.0
6                5861.0            2750.0              652.0
7                2064.0            2720.0              315.0
8                9501.0            9579.0              768.0
9                6414.0            6111.0              613.0
10               4949.0            5811.0              185.0
11               1261.0            6205.0              850.0
12               3586.0            6017.0              472.0
13               7053.0            1109.0              827.0
14               3429.0            8363.0              876.0
15               7818.0            1494.0              567.0
16               6740.0            5829.0              484.0
17               1528.0            4961.0              893.0
18                131.0             336.0              656.0
19                907.0            3418.0              162.0

    remainder__hasguestroom
0                      10.0
1                       3.0
2                       2.0
3                       0.0
4                       5.0
5                       6.0
6                      10.0
7                       5.0
8                       1.0
9                       7.0
10                      6.0
11                     10.0
12                      7.0
13                      3.0
14                      3.0
15                      0.0
16                      5.0
17                      8.0
18                     10.0
19                      5.0

[20 rows x 21 columns]
```

```python
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_Ridge = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)),
    ('reg', Ridge())
])

param_grid_Ridge = {
    'reg__alpha': [0.01, 0.1, 1, 10, 100],
    'feature_selection__k': np.arange(1, 20)
}

GSCV_RR = GridSearchCV(pipe_Ridge, param_grid_Ridge, cv=5,
scoring='neg_mean_squared_error', error_score='raise')

GSCV_RR.fit(x_train_price_enc, y_train_price)

print("Best model: {}".format(GSCV_RR.best_estimator_))
print("Ridge best parameters: {}".format(GSCV_RR.best_params_))
print("Koefisien/bobot:
{}".format(GSCV_RR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:
{}".format(GSCV_RR.best_estimator_.named_steps['reg'].intercept_))

Ridge_predict = GSCV_RR.predict(x_test_price_enc)
mse_Ridge = mean_squared_error(y_test_price, Ridge_predict)
mae_Ridge = mean_absolute_error(y_test_price, Ridge_predict)

print("Ridge Mean Squared Error (MSE): {}".format(mse_Ridge))
print("Ridge Mean Absolute Error (MAE): {}".format(mae_Ridge))
print("Ridge Root Mean Squared Error: {}".format(np.sqrt(mse_Ridge)))
```

```
Best model: Pipeline(steps=[('scale', StandardScaler()),
                ('feature_selection',
                 SelectKBest(k=19,
                             score_func=<function f_regression at
0x000001B4370873A0>)),
                ('reg', Ridge(alpha=0.01))])
Ridge best parameters: {'feature_selection__k': 19, 'reg__alpha':
0.01}
Koefisien/bobot: [-7.56045182e+02  7.56045183e+02 -7.41944533e+02
7.41944533e+02
  4.65683060e+01 -4.65683083e+01 -4.05487999e+01  4.05488010e+01
 -5.34405351e+00  5.34405341e+00  2.88881677e+06  7.40651732e+00
```

```
  1.55973632e+03  1.41411464e+02 -8.98869834e+00 -1.35881746e+01
 -1.85257372e+01  3.70581728e+01 -1.76438214e+01]
Intercept/bias: 5003139.741906667
Ridge Mean Squared Error (MSE): 3463319.5471278434
Ridge Mean Absolute Error (MAE): 1436.9858096538003
Ridge Root Mean Squared Error: 1860.9996096527918
```

```
df_results = pd.DataFrame(y_test_price, columns=['price'])
df_results = pd.DataFrame(y_test_price)
df_results['Ridge Prediction'] = Ridge_predict

df_results['Selisih_price_RR'] = df_results['Ridge Prediction'] -
df_results['price']

df_results.head()
```

	price	Ridge Prediction	Selisih_price_RR
4208	7639752.5	7.639003e+06	-749.746498
3619	9873512.3	9.874681e+06	1169.094552
5826	1397748.9	1.398107e+06	357.831109
6538	1620485.0	1.622079e+06	1594.441132
8787	4872012.2	4.872000e+06	-12.548969

```
df_results.describe()
```

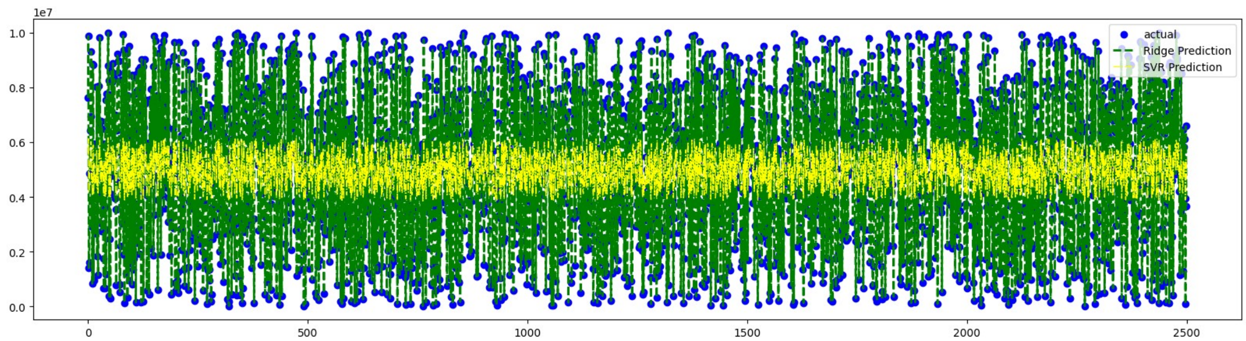
	price	Ridge Prediction	Selisih_price_RR
count	2.500000e+03	2.500000e+03	2500.000000
mean	4.964371e+06	4.964383e+06	12.219481
std	2.842791e+06	2.842819e+06	1861.331796
min	1.443130e+04	1.647641e+04	-6523.351317
25%	2.567703e+06	2.567287e+06	-1146.123872
50%	4.998880e+06	4.999651e+06	42.904055
75%	7.391681e+06	7.392256e+06	1169.838143
max	1.000294e+07	1.000120e+07	7046.231520

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_SVR = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)),
    ('reg', SVR(kernel='linear'))
])

param_grid_SVR = {
    'reg__C': [0.01, 0.1, 1, 10, 100],
    'reg__epsilon': [0.1, 0.2, 0.5, 1],
```

```
    'feature_selection__k': np.arange(1, 20)
}

GSCV_SVR = GridSearchCV(pipe_SVR, param_grid_SVR, cv=5,
scoring='neg_mean_squared_error')

GSCV_SVR.fit(x_train_price_enc, y_train_price)

print("Best model: {}".format(GSCV_SVR.best_estimator_))
print("SVR best parameters: {}".format(GSCV_SVR.best_params_))
print("Koefisien/bobot:
{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:
{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].intercept_))

SVR_predict = GSCV_SVR.predict(x_test_price_enc)

mse_SVR = mean_squared_error(y_test_price, SVR_predict)
mae_SVR = mean_absolute_error(y_test_price, SVR_predict)

print("SVR Mean Squared Error (MSE): {}".format(mse_SVR))
print("SVR Mean Absolute Error (MAE): {}".format(mae_SVR))
print("SVR Root Mean Squared Error: {}".format(np.sqrt(mse_SVR)))

Best model: Pipeline(steps=[('scale', StandardScaler()),
                ('feature_selection',
                 SelectKBest(k=4,
                             score_func=<function f_regression at
0x000001B4370873A0>)),
                ('reg', SVR(C=100, kernel='linear'))])
SVR best parameters: {'feature_selection__k': 4, 'reg__C': 100,
'reg__epsilon': 0.1}
Koefisien/bobot: [[649047.75506463  12391.30466663  14718.77900439  -
7693.23612297]]
Intercept/bias: [5010394.44877383]
SVR Mean Squared Error (MSE): 4858447421454.491
SVR Mean Absolute Error (MAE): 1895594.4965093078
SVR Root Mean Squared Error: 2204188.6084122863

df_results['SVR Prediction'] = SVR_predict
df_results = pd.DataFrame(y_test_price)
df_results['SVR Prediction'] = SVR_predict

df_results['Selisih_price_SVR'] = df_results['SVR Prediction'] -
df_results['price']
df_results.head()

          price  SVR Prediction  Selisih_price_SVR
4208  7639752.5    5.624434e+06      -2.015318e+06
3619  9873512.3    6.126296e+06      -3.747217e+06
5826  1397748.9    4.167064e+06       2.769315e+06
```

```
6538  1620485.0    4.220983e+06       2.600498e+06
8787  4872012.2    4.968442e+06       9.642940e+04

df_results.describe()

              price  SVR Prediction  Selisih_price_SVR
count  2.500000e+03    2.500000e+03       2.500000e+03
mean   4.964371e+06    5.001619e+06       3.724797e+04
std    2.842791e+06    6.388973e+05       2.204315e+06
min    1.443130e+04    3.852418e+06      -3.888487e+06
25%    2.567703e+06    4.459952e+06      -1.836367e+06
50%    4.998880e+06    5.006322e+06       1.100832e+04
75%    7.391681e+06    5.544842e+06       1.896727e+06
max    1.000294e+07    6.163456e+06       3.877589e+06

print(df_results.columns)

Index(['price', 'Ridge Prediction'], dtype='object')

print(len(y_test_price), len(Ridge_predict), len(x_test_price_enc)),

2500 2500 2500

(None,)

print(Ridge_predict)

[7639002.75350174 9874681.39455215 1398106.7311088  ...
87279.62484006
 6585554.73333456 3660613.32301272]

# Pastikan Anda tidak membuat ulang df_results setiap kali
df_results = pd.DataFrame(y_test_price, columns=['price'])

# Tambahkan prediksi SVR terlebih dahulu
df_results['SVR Prediction'] = SVR_predict

# Tambahkan prediksi Ridge Regression setelahnya
df_results['Ridge Prediction'] = Ridge_predict

# Pastikan kedua kolom ada
print(df_results.columns)

# Buat plot setelah memastikan semua kolom ada
plt.figure(figsize=(20, 5))
data_len = range(len(y_test_price))

plt.scatter(data_len, df_results['price'], label="actual",
color="blue")
plt.plot(data_len, df_results['Ridge Prediction'], label="Ridge
Prediction", color="green", linewidth=2, linestyle="dashed")
plt.plot(data_len, df_results['SVR Prediction'], label="SVR
```

```
Prediction", color="yellow", linewidth=1, linestyle="-.")

plt.legend()
plt.show()

Index(['price', 'SVR Prediction', 'Ridge Prediction'], dtype='object')
```




```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

mae_ridge = mean_absolute_error(df_results['price'], df_results['Ridge
Prediction'])
rmse_ridge = np.sqrt(mean_squared_error(df_results['price'],
df_results['Ridge Prediction']))
ridge_feature_count = GSCV_RR.best_params_['feature_selection__k']

mae_svr = mean_absolute_error(df_results['price'], df_results['SVR
Prediction'])
rmse_svr = np.sqrt(mean_squared_error(df_results['price'],
df_results['SVR Prediction']))
svr_feature_count = GSCV_SVR.best_params_['feature_selection__k']

print(f"Ridge MAE: {mae_ridge}, Ridge RMSE: {rmse_ridge}, Ridge
Feature Count: {ridge_feature_count}")
print(f"SVR MAE: {mae_svr}, SVR RMSE: {rmse_svr}, SVR Feature Count:
{svr_feature_count}")

Ridge MAE: 1436.9858096538003, Ridge RMSE: 1860.9996096527918, Ridge
Feature Count: 19
SVR MAE: 1895594.4965093078, SVR RMSE: 2204188.6084122863, SVR Feature
Count: 4
```

```python
import pandas as pd
import numpy as np

df_harga = pd.read_csv(r'D:\Funiversity\Sem5\ML\UTS\Dataset UTS_Gasal 2425.csv')
df_harga.head(20)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
2          86929            100     yes      no      11     98155
3          51522              3      no      no      61      9047
4          96470             74     yes      no      21     92029
5          79770              3      no     yes      69     54812
6          75985             60     yes      no      67      6517
7          64169             88      no     yes       6     61711
8          92383             12      no      no      78     71982
9          95121             46      no     yes       3      9382
10         76485             47     yes      no       9     90254
11         87060             27      no     yes      91     51803
12         66683             19     yes     yes       6     50801
13         84559             29      no     yes      69     53057
14         76091             38     yes      no      32     59451
15         92696             49     yes      no      38     74381
16         59800             47      no     yes      27     44815
17         54836             25      no     yes      53     64601
18         70021             52     yes      no      28     95678
19         54368             11     yes     yes      20     55761

    citypartrange  numprevowners  made isnewbuilt hasstormprotector
basement  \
0               3              8  2005        old               yes
4313
1               6              8  2021        old                no
2937
2               3              4  2003        new                no
6326
3               8              3  2012        new               yes
632
4               4              2  2011        new               yes
5414
5              10              5  2018        old               yes
8871
6               6              9  2009        new               yes
4878
7               3              9  2011        new               yes
3054
8               3              7  2000        old                no
7507
9               7              9  1994        old                no
```

```
615
10            2            9  2008       new              no
2860
11            8           10  2000       old              no
6629
12            6            2  2001       old              no
7473
13            7            7  2000       new              no
3573
14            5            8  2016       new              no
8150
15            9            2  2021       old              no
1559
16            6            9  2021       old              no
5075
17           10            5  2020       new              no
5278
18            4            6  1992       old             yes
4480
19            3            7  2021       old              no
231

    attic  garage hasstorageroom  hasguestroom      price category
0    9005     956             no             7  7559081.5   Luxury
1    8852     135            yes             9  5574642.1   Middle
2    4748     654             no            10  8696869.3   Luxury
3    5792     807            yes             5  5154055.2   Middle
4    1172     716            yes             9  9652258.1   Luxury
5    7117     240             no             7  7986665.8   Luxury
6     281     384            yes             5  7607322.9   Luxury
7     129     726             no             9  6420823.1   Middle
8    9056     892            yes             1  9244344.0   Luxury
9    1221     328             no            10  9515440.4   Luxury
10   3129     982             no             1  7653300.8   Luxury
11    435     512             no             7  8711426.0   Luxury
12    796     237            yes             3  6677649.1   Middle
13   9556     918            yes             8  8460604.0   Luxury
14   6037     930             no             7  7614076.6   Luxury
15   5111     957            yes             2  9272740.1   Luxury
16   3104     864             no             4  5984462.1   Middle
17   1059     313            yes             6  5492532.0   Middle
18   6919     680            yes             1  7005572.2   Luxury
19   1939     223             no             8  5446398.1   Middle
```

```
df_harga2 = df_harga.drop(['category'], axis = 1)
df_harga2.head(20)
```

```
    squaremeters  numberofrooms hasyard haspool  floors  citycode  \
0          75523              3      no     yes      63      9373
1          55712             58      no     yes      19     34457
```

```
2          86929              100     yes      no      11     98155
3          51522                3      no      no      61      9047
4          96470               74     yes      no      21     92029
5          79770                3      no     yes      69     54812
6          75985               60     yes      no      67      6517
7          64169               88      no     yes       6     61711
8          92383               12      no      no      78     71982
9          95121               46      no     yes       3      9382
10         76485               47     yes      no       9     90254
11         87060               27      no     yes      91     51803
12         66683               19     yes     yes       6     50801
13         84559               29      no     yes      69     53057
14         76091               38     yes      no      32     59451
15         92696               49     yes      no      38     74381
16         59800               47      no     yes      27     44815
17         54836               25      no     yes      53     64601
18         70021               52     yes      no      28     95678
19         54368               11     yes     yes      20     55761

    citypartrange  numprevowners  made isnewbuilt hasstormprotector
basement  \
0               3              8  2005        old               yes
4313
1               6              8  2021        old                no
2937
2               3              4  2003        new                no
6326
3               8              3  2012        new               yes
632
4               4              2  2011        new               yes
5414
5              10              5  2018        old               yes
8871
6               6              9  2009        new               yes
4878
7               3              9  2011        new               yes
3054
8               3              7  2000        old                no
7507
9               7              9  1994        old                no
615
10              2              9  2008        new                no
2860
11              8             10  2000        old                no
6629
12              6              2  2001        old                no
7473
13              7              7  2000        new                no
3573
```

```
14             5             8  2016       new           no
8150
15             9             2  2021       old           no
1559
16             6             9  2021       old           no
5075
17            10             5  2020       new           no
5278
18             4             6  1992       old          yes
4480
19             3             7  2021       old           no
231

    attic  garage hasstorageroom  hasguestroom      price
0    9005     956             no             7  7559081.5
1    8852     135            yes             9  5574642.1
2    4748     654             no            10  8696869.3
3    5792     807            yes             5  5154055.2
4    1172     716            yes             9  9652258.1
5    7117     240             no             7  7986665.8
6     281     384            yes             5  7607322.9
7     129     726             no             9  6420823.1
8    9056     892            yes             1  9244344.0
9    1221     328             no            10  9515440.4
10   3129     982             no             1  7653300.8
11    435     512             no             7  8711426.0
12    796     237            yes             3  6677649.1
13   9556     918            yes             8  8460604.0
14   6037     930             no             7  7614076.6
15   5111     957            yes             2  9272740.1
16   3104     864             no             4  5984462.1
17   1059     313            yes             6  5492532.0
18   6919     680            yes             1  7005572.2
19   1939     223             no             8  5446398.1

df_harga2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   squaremeters       10000 non-null  int64
 1   numberofrooms      10000 non-null  int64
 2   hasyard            10000 non-null  object
 3   haspool            10000 non-null  object
 4   floors             10000 non-null  int64
 5   citycode           10000 non-null  int64
 6   citypartrange      10000 non-null  int64
 7   numprevowners      10000 non-null  int64
```

```
 8   made               10000 non-null  int64
 9   isnewbuilt         10000 non-null  object
 10  hasstormprotector  10000 non-null  object
 11  basement           10000 non-null  int64
 12  attic              10000 non-null  int64
 13  garage             10000 non-null  int64
 14  hasstorageroom     10000 non-null  object
 15  hasguestroom       10000 non-null  int64
 16  price              10000 non-null  float64
dtypes: float64(1), int64(11), object(5)
memory usage: 1.3+ MB
```

```
df_harga2.describe()
```

```
       squaremeters  numberofrooms        floors      citycode
citypartrange  \
count   10000.00000   10000.000000  10000.000000  10000.000000
10000.000000
mean    49870.13120      50.358400     50.276300  50225.486100
5.510100
std     28774.37535      28.816696     28.889171  29006.675799
2.872024
min        89.00000       1.000000      1.000000      3.000000
1.000000
25%     25098.50000      25.000000     25.000000  24693.750000
3.000000
50%     50105.50000      50.000000     50.000000  50693.000000
5.000000
75%     74609.75000      75.000000     76.000000  75683.250000
8.000000
max     99999.00000     100.000000    100.000000  99953.000000
10.000000

       numprevowners         made      basement         attic
garage  \
count   10000.000000  10000.00000  10000.000000  10000.00000
10000.00000
mean        5.521700   2005.48850   5033.103900   5028.01060
553.12120
std         2.856667      9.30809   2876.729545   2894.33221
262.05017
min         1.000000   1990.00000      0.000000      1.00000
100.00000
25%         3.000000   1997.00000   2559.750000   2512.00000
327.75000
50%         5.000000   2005.50000   5092.500000   5045.00000
554.00000
75%         8.000000   2014.00000   7511.250000   7540.50000
777.25000
max        10.000000   2021.00000  10000.000000  10000.00000
```

```
1000.00000

       hasguestroom         price
count   10000.00000  1.000000e+04
mean        4.99460  4.993448e+06
std         3.17641  2.877424e+06
min         0.00000  1.031350e+04
25%         2.00000  2.516402e+06
50%         5.00000  5.016180e+06
75%         8.00000  7.469092e+06
max        10.00000  1.000677e+07
```

```python
print(df_harga2['price'].value_counts())
```

```
price
7559081.5    1
2600292.1    1
3804577.4    1
3658559.7    1
2316639.4    1
            ..
5555606.6    1
5501007.5    1
9986201.2    1
9104801.8    1
146708.4     1
Name: count, Length: 10000, dtype: int64
```

```python
print("\tdata null\n", df_harga2.isnull().sum())
print("\n\tdata kosong\n", df_harga2.empty)
print("\n\tdata nan\n", df_harga2.isna().sum())
```

```
     data null
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
price                0
```

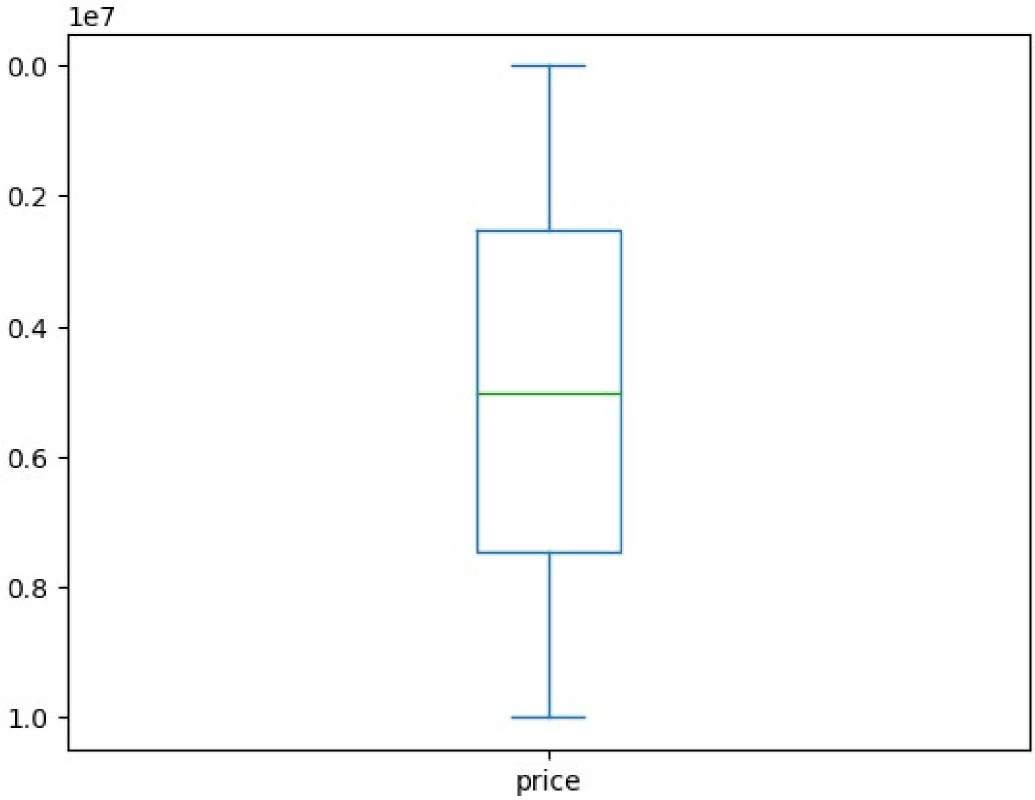
```
dtype: int64

     data kosong
 False

     data nan
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic                0
garage               0
hasstorageroom       0
hasguestroom         0
price                0
dtype: int64
```

```
print("Sebelum Pengecekan data duplikat, ",df_harga2.shape)
df_harga3 = df_harga2.drop_duplicates(keep = 'last')
print("Setelah Pengecekan data duplikat, ",df_harga2.shape)
```

```
Sebelum Pengecekan data duplikat,  (10000, 17)
Setelah Pengecekan data duplikat,  (10000, 17)
```

```
import matplotlib.pyplot as plt

df_harga3.price.plot(kind='box')
plt.gca().invert_yaxis()
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
from sklearn.model_selection import train_test_split

x_regress = df_harga3.drop(columns=['price'], axis = 1)
y_regress = df_harga3['price']

x_train_price, x_test_price, y_train_price, y_test_price =
train_test_split(x_regress, y_regress, test_size = 0.25, random_state
= 78)

print(x_train_price.shape)
print(x_test_price.shape)

(7500, 16)
(2500, 16)

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

category_column = ['hasyard', 'haspool', 'isnewbuilt',
'hasstormprotector', 'hasstorageroom']

transform = make_column_transformer((OneHotEncoder(),
category_column), remainder = 'passthrough')
```

```
x_train_price_enc = transform.fit_transform(x_train_price)
x_test_price_enc = transform.fit_transform(x_test_price)

df_train_enc = pd.DataFrame(x_train_price_enc, columns =
transform.get_feature_names_out())
df_test_enc = pd.DataFrame(x_test_price_enc, columns =
transform.get_feature_names_out())

df_train_enc.head(20)
df_test_enc.head(20)
```

```
    onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                         1.0                         0.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         1.0                         0.0
4                         0.0                         1.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         0.0                         1.0
9                         1.0                         0.0
10                        0.0                         1.0
11                        0.0                         1.0
12                        1.0                         0.0
13                        0.0                         1.0
14                        1.0                         0.0
15                        0.0                         1.0
16                        1.0                         0.0
17                        0.0                         1.0
18                        0.0                         1.0
19                        1.0                         0.0

    onehotencoder__haspool_no  onehotencoder__haspool_yes  \
0                         0.0                         1.0
1                         0.0                         1.0
2                         1.0                         0.0
3                         0.0                         1.0
4                         1.0                         0.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         0.0                         1.0
8                         1.0                         0.0
9                         1.0                         0.0
10                        0.0                         1.0
11                        1.0                         0.0
12                        1.0                         0.0
13                        1.0                         0.0
14                        1.0                         0.0
15                        1.0                         0.0
```

```
16                         1.0                          0.0
17                         0.0                          1.0
18                         1.0                          0.0
19                         1.0                          0.0

    onehotencoder__isnewbuilt_new  onehotencoder__isnewbuilt_old  \
0                             0.0                            1.0
1                             0.0                            1.0
2                             0.0                            1.0
3                             1.0                            0.0
4                             1.0                            0.0
5                             0.0                            1.0
6                             0.0                            1.0
7                             0.0                            1.0
8                             1.0                            0.0
9                             1.0                            0.0
10                            0.0                            1.0
11                            0.0                            1.0
12                            1.0                            0.0
13                            1.0                            0.0
14                            0.0                            1.0
15                            1.0                            0.0
16                            1.0                            0.0
17                            1.0                            0.0
18                            1.0                            0.0
19                            1.0                            0.0

    onehotencoder__hasstormprotector_no
onehotencoder__hasstormprotector_yes  \
0                                   1.0
0.0
1                                   0.0
1.0
2                                   1.0
0.0
3                                   1.0
0.0
4                                   1.0
0.0
5                                   0.0
1.0
6                                   0.0
1.0
7                                   0.0
1.0
8                                   0.0
1.0
9                                   1.0
0.0
```

```
10                                    1.0
0.0
11                                    0.0
1.0
12                                    0.0
1.0
13                                    1.0
0.0
14                                    1.0
0.0
15                                    1.0
0.0
16                                    1.0
0.0
17                                    1.0
0.0
18                                    1.0
0.0
19                                    1.0
0.0

    onehotencoder__hasstorageroom_no
onehotencoder__hasstorageroom_yes  ...  \
0                                 1.0
0.0  ...
1                                 0.0
1.0  ...
2                                 0.0
1.0  ...
3                                 0.0
1.0  ...
4                                 1.0
0.0  ...
5                                 0.0
1.0  ...
6                                 0.0
1.0  ...
7                                 0.0
1.0  ...
8                                 0.0
1.0  ...
9                                 0.0
1.0  ...
10                                0.0
1.0  ...
11                                1.0
0.0  ...
12                                0.0
1.0  ...
```

```
13                                 0.0
1.0  ...
14                                 1.0
0.0  ...
15                                 0.0
1.0  ...
16                                 0.0
1.0  ...
17                                 1.0
0.0  ...
18                                 0.0
1.0  ...
19                                 0.0
1.0  ...

    remainder__numberofrooms  remainder__floors
remainder__citycode  \
0                        73.0               13.0               42855.0

1                        95.0                3.0               75381.0

2                        39.0                8.0               91674.0

3                        47.0               63.0               58471.0

4                        64.0               83.0               30779.0

5                        91.0               67.0               65183.0

6                        48.0               70.0               21012.0

7                        93.0               35.0               12062.0

8                        74.0               14.0               76662.0

9                        72.0               76.0               87732.0

10                       18.0               66.0               38920.0

11                       37.0               26.0               96016.0

12                       76.0               95.0               76985.0

13                       26.0               99.0               38185.0

14                       69.0               45.0               88591.0

15                        9.0               25.0               33740.0

16                       29.0               58.0               13202.0
```

```
17                       91.0               43.0              93072.0

18                       68.0               62.0              97608.0

19                       48.0               53.0              34588.0


    remainder__citypartrange  remainder__numprevowners
remainder__made  \
0                        9.0                       6.0
2015.0
1                        5.0                       6.0
2003.0
2                        2.0                       2.0
2009.0
3                       10.0                       1.0
1990.0
4                        6.0                       4.0
1992.0
5                        1.0                      10.0
2019.0
6                        5.0                       3.0
2007.0
7                       10.0                       9.0
1998.0
8                        9.0                       6.0
2004.0
9                        2.0                       8.0
2017.0
10                       7.0                       2.0
2007.0
11                       3.0                       6.0
2000.0
12                       4.0                       7.0
2009.0
13                       7.0                       7.0
1994.0
14                       1.0                       8.0
1995.0
15                       2.0                       1.0
2009.0
16                       6.0                       5.0
1993.0
17                       1.0                       7.0
2012.0
18                       1.0                       9.0
2001.0
19                       9.0                       6.0
2003.0
```

```
    remainder__basement  remainder__attic  remainder__garage  \
0                2560.0            6823.0              239.0
1                6810.0            1391.0              556.0
2                1477.0            3153.0              952.0
3                1730.0            7967.0              722.0
4                 594.0            8310.0              898.0
5                6824.0            7141.0              956.0
6                5861.0            2750.0              652.0
7                2064.0            2720.0              315.0
8                9501.0            9579.0              768.0
9                6414.0            6111.0              613.0
10               4949.0            5811.0              185.0
11               1261.0            6205.0              850.0
12               3586.0            6017.0              472.0
13               7053.0            1109.0              827.0
14               3429.0            8363.0              876.0
15               7818.0            1494.0              567.0
16               6740.0            5829.0              484.0
17               1528.0            4961.0              893.0
18                131.0             336.0              656.0
19                907.0            3418.0              162.0

    remainder__hasguestroom
0                      10.0
1                       3.0
2                       2.0
3                       0.0
4                       5.0
5                       6.0
6                      10.0
7                       5.0
8                       1.0
9                       7.0
10                      6.0
11                     10.0
12                      7.0
13                      3.0
14                      3.0
15                      0.0
16                      5.0
17                      8.0
18                     10.0
19                      5.0

[20 rows x 21 columns]
```

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_regression,
```

```
SelectPercentile

from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_Lasso = Pipeline(steps = [
    ('scale', StandardScaler()),
    ('feature', SelectKBest(score_func = f_regression)),
    ('reg', Lasso(max_iter = 1000))
])

param_grid_Lasso = [
    {
        'scale' : [StandardScaler()],
        'feature' : [SelectKBest(f_regression)],
        'feature__k' : np.arange(1, 20),
        'reg__alpha' : [0.01, 0.1, 1, 10, 100]
    },

    {
        'scale' : [StandardScaler()],
        'feature' : [SelectPercentile(f_regression)],
        'feature__percentile' : np.arange(10, 100, 10),
        'reg__alpha' : [0.01, 0.1, 1, 10, 100]
    },

    {
        'scale' : [MinMaxScaler()],
        'feature' : [SelectKBest(f_regression)],
        'feature__k' : np.arange(1, 20),
        'reg__alpha' : [0.01, 0.1, 1, 10, 100]
    },

    {
        'scale' : [MinMaxScaler()],
        'feature' : [SelectPercentile(f_regression)],
        'feature__percentile' : np.arange(10, 100, 10),
        'reg__alpha' : [0.01, 0.1, 1, 10, 100]
    }
]

KF = KFold(n_splits = 5, shuffle = True, random_state = 78)

GSCV_Lasso = GridSearchCV(pipe_Lasso, param_grid_Lasso, cv = KF,
scoring = 'neg_mean_squared_error')
GSCV_Lasso.fit(x_train_price_enc, y_train_price)

print("Best Model: {}".format(GSCV_Lasso.best_estimator_))
mask = GSCV_Lasso.best_estimator_.named_steps['feature'].get_support()
print("Best features:",df_train_enc.columns[mask])
```

```
print("Koefisien/Bobot:
{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'].coef_))
print("Intercept/Bias:
{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'].intercept_))

Lasso_predict = GSCV_Lasso.predict(x_test_price_enc)

mae_Lasso = mean_absolute_error(y_test_price, Lasso_predict)
mse_Lasso = mean_squared_error(y_test_price, Lasso_predict)

print("Lasso Regression MAE: {}".format(mae_Lasso))
print("Lasso Regression MSE: {}".format(mse_Lasso))
print("Lasso Regression RMSE: {}".format(np.sqrt(mse_Lasso)))
```

```
Best Model: Pipeline(steps=[('scale', StandardScaler()),
                ('feature',
                 SelectKBest(k=19,
                             score_func=<function f_regression at
0x000001E948B663A0>)),
                ('reg', Lasso(alpha=10))])
Best features: Index(['onehotencoder__hasyard_no',
'onehotencoder__hasyard_yes',
       'onehotencoder__haspool_no', 'onehotencoder__haspool_yes',
       'onehotencoder__isnewbuilt_new',
'onehotencoder__isnewbuilt_old',
       'onehotencoder__hasstormprotector_no',
       'onehotencoder__hasstormprotector_yes',
       'onehotencoder__hasstorageroom_no',
'onehotencoder__hasstorageroom_yes',
       'remainder__squaremeters', 'remainder__numberofrooms',
       'remainder__floors', 'remainder__citypartrange',
       'remainder__numprevowners', 'remainder__basement',
'remainder__attic',
       'remainder__garage', 'remainder__hasguestroom'],
      dtype='object')
Koefisien/Bobot: [-1.50189031e+03  4.77302819e-13 -1.47466707e+03
1.14475066e-13
  8.29146159e+01 -0.00000000e+00 -7.13309651e+01  5.42883451e-12
 -7.55801132e-01  5.86364573e-11  2.88881051e+06  0.00000000e+00
  1.54979947e+03  1.31187184e+02 -0.00000000e+00 -3.74045759e+00
 -8.09905705e+00  2.77422673e+01 -7.45876850e+00]
Intercept/Bias: 5003139.741906667
Lasso Regression MAE: 1436.6510138328122
Lasso Regression MSE: 3460594.830991738
Lasso Regression RMSE: 1860.2674084635623
```

```
df_results = pd.DataFrame(y_test_price, columns=['price'])
df_results = pd.DataFrame(y_test_price)
```

```python
df_results['Lasso Regression Prediction'] = Lasso_predict

df_results['Selisih_Harga_LR'] = df_results['Lasso Regression
Prediction'] - df_results['price']

df_results.head()
```

	price	Lasso Regression Prediction	Selisih_Harga_LR
4208	7639752.5	7.639049e+06	-703.338378
3619	9873512.3	9.874637e+06	1124.307238
5826	1397748.9	1.398120e+06	371.343414
6538	1620485.0	1.622021e+06	1535.868119
8787	4872012.2	4.871971e+06	-40.732719

```python
df_results.describe()
```

	price	Lasso Regression Prediction	Selisih_Harga_LR
count	2.500000e+03	2.500000e+03	2500.000000
mean	4.964371e+06	4.964382e+06	11.600983
std	2.842791e+06	2.842814e+06	1860.603393
min	1.443130e+04	1.647411e+04	-6534.806896
25%	2.567703e+06	2.567290e+06	-1158.542087
50%	4.998880e+06	4.999656e+06	45.301221
75%	7.391681e+06	7.392305e+06	1161.233831
max	1.000294e+07	1.000122e+07	6976.939177

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_regression,
SelectPercentile

from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_RF = Pipeline(steps = [
    ('scale', StandardScaler()),
    ('feature', SelectKBest(score_func = f_regression)),
    ('reg', RandomForestRegressor(random_state = 78))
])

param_grid_RF = [
    {
        'scale' : [StandardScaler()],
        'feature' : [SelectKBest(f_regression)],
        'feature__k' : np.arange(1, 20),
        'reg__n_estimators' : [100, 150],
        'reg__max_depth' : [4, 5, 6]
    },

    {
```

```
        'scale' : [StandardScaler()],
        'feature' : [SelectPercentile(f_regression)],
        'feature__percentile' : np.arange(10, 100, 10),
        'reg__n_estimators' : [100, 150],
        'reg__max_depth' : [4, 5, 6]
    },

    {
        'scale' : [MinMaxScaler()],
        'feature' : [SelectKBest(f_regression)],
        'feature__k' : np.arange(1, 20),
        'reg__n_estimators' : [100, 150],
        'reg__max_depth' : [4, 5, 6]
    },

    {
        'scale' : [MinMaxScaler()],
        'feature' : [SelectPercentile(f_regression)],
        'feature__percentile' : np.arange(10, 100, 10),
        'reg__n_estimators' : [100, 150],
        'reg__max_depth' : [4, 5, 6]
    }
]

KF = KFold(n_splits = 5, shuffle = True, random_state = 78)

GSCV_RF = GridSearchCV(pipe_RF, param_grid_RF, cv = KF, scoring =
'neg_mean_squared_error')

GSCV_RF.fit(x_train_price_enc, y_train_price)

print("Best Model: {}".format(GSCV_RF.best_estimator_))
mask = GSCV_RF.best_estimator_.named_steps['feature'].get_support()
print("Best features:",df_train_enc.columns[mask])

RF_predict = GSCV_RF.predict(x_test_price_enc)

mae_RF = mean_absolute_error(y_test_price, RF_predict)
mse_RF = mean_squared_error(y_test_price, RF_predict)

print("Random Forest Regression MAE: {}".format(mae_RF))
print("Random Forest Regression MSE: {}".format(mse_RF))
print("Random Forest Regression RMSE: {}".format(np.sqrt(mse_RF)))
```

```
Best Model: Pipeline(steps=[('scale', StandardScaler()),
                ('feature',
                 SelectKBest(k=6,
                             score_func=<function f_regression at
0x000001E948B663A0>)),
                ('reg',
```

```
                 RandomForestRegressor(max_depth=6, n_estimators=150,
                                       random_state=78))])
Best features: Index(['onehotencoder__hasstormprotector_yes',
'remainder__squaremeters',
       'remainder__numberofrooms', 'remainder__numprevowners',
       'remainder__attic', 'remainder__garage'],
      dtype='object')
Random Forest Regression MAE: 16083.918675581288
Random Forest Regression MSE: 412641335.1781159
Random Forest Regression RMSE: 20313.575145161325
```

```
df_results['RF Regression Prediction'] = RF_predict
df_results = pd.DataFrame(y_test_price)
df_results['RF Regression Prediction'] = RF_predict

df_results['Selisih_Harga_RFR'] = df_results['RF Regression
Prediction'] - df_results['price']

df_results.head()
```

```
          price  RF Regression Prediction  Selisih_Harga_RFR
4208  7639752.5              7.638896e+06        -856.207487
3619  9873512.3              9.921015e+06       47502.930129
5826  1397748.9              1.447161e+06       49411.957540
6538  1620485.0              1.631266e+06       10780.865479
8787  4872012.2              4.886406e+06       14393.606394
```

```
df_results.describe()
```

```
              price  RF Regression Prediction  Selisih_Harga_RFR
count  2.500000e+03              2.500000e+03        2500.000000
mean   4.964371e+06              4.964135e+06        -235.718277
std    2.842791e+06              2.842884e+06       20316.271125
min    1.443130e+04              8.363800e+04      -81929.769871
25%    2.567703e+06              2.554733e+06      -14193.902928
50%    4.998880e+06              5.022442e+06          20.626003
75%    7.391681e+06              7.407247e+06       12956.223520
max    1.000294e+07              9.921015e+06       69206.702595
```

```
df_results = pd.DataFrame({'price': y_test_price})

df_results['Lasso Regression Prediction'] = Lasso_predict
df_results['Selisih_Harga_LR'] = df_results['price'] -
df_results['Lasso Regression Prediction']

df_results['RF Regression Prediction'] = RF_predict
df_results['Selisih_Harga_RFR'] = df_results['price'] - df_results['RF
Regression Prediction']

df_results.head()
```

```
           price  Lasso Regression Prediction  Selisih_Harga_LR  \
4208  7639752.5                 7.639049e+06        703.338378
3619  9873512.3                 9.874637e+06      -1124.307238
5826  1397748.9                 1.398120e+06       -371.343414
6538  1620485.0                 1.622021e+06      -1535.868119
8787  4872012.2                 4.871971e+06         40.732719

      RF Regression Prediction  Selisih_Harga_RFR
4208              7.638896e+06         856.207487
3619              9.921015e+06      -47502.930129
5826              1.447161e+06      -49411.957540
6538              1.631266e+06      -10780.865479
8787              4.886406e+06      -14393.606394
```

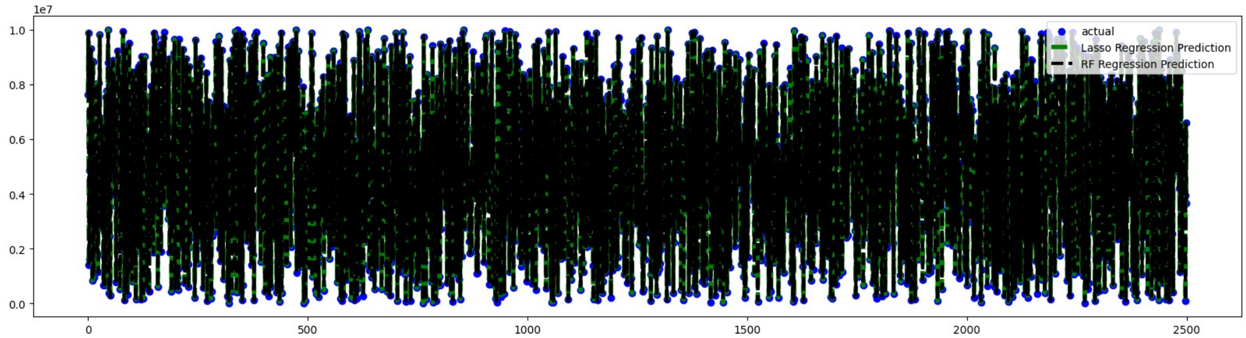
```python
df_results.describe()
```

```
              price  Lasso Regression Prediction  Selisih_Harga_LR  \
count  2.500000e+03                 2.500000e+03       2500.000000
mean   4.964371e+06                 4.964382e+06        -11.600983
std    2.842791e+06                 2.842814e+06       1860.603393
min    1.443130e+04                 1.647411e+04      -6976.939177
25%    2.567703e+06                 2.567290e+06      -1161.233831
50%    4.998880e+06                 4.999656e+06        -45.301221
75%    7.391681e+06                 7.392305e+06       1158.542087
max    1.000294e+07                 1.000122e+07       6534.806896

       RF Regression Prediction  Selisih_Harga_RFR
count              2.500000e+03        2500.000000
mean               4.964135e+06         235.718277
std                2.842884e+06       20316.271125
min                8.363800e+04      -69206.702595
25%                2.554733e+06      -12956.223520
50%                5.022442e+06         -20.626003
75%                7.407247e+06       14193.902928
max                9.921015e+06       81929.769871
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize =(20,5))
data_len = range(len(y_test_price))
plt.scatter(data_len, df_results.price, label = "actual", color =
"blue")

plt.plot(data_len, df_results['Lasso Regression Prediction'], label =
"Lasso Regression Prediction", color = "green", linewidth = 4,
linestyle = "dashed")
plt.plot(data_len, df_results['RF Regression Prediction'], label = "RF
Regression Prediction", color = "black", linewidth = 3, linestyle =
"--")
```

```
plt.legend()
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

mae_lasso = mean_absolute_error(df_results['price'], df_results['Lasso
Regression Prediction'])
rmse_lasso = np.sqrt(mean_squared_error(df_results['price'],
df_results['Lasso Regression Prediction']))
lasso_feature_count = GSCV_Lasso.best_params_['feature__k']

mae_RFR = mean_absolute_error(df_results['price'], df_results['RF
Regression Prediction'])
rmse_RFR = np.sqrt(mean_squared_error(df_results['price'],
df_results['RF Regression Prediction']))
RFR_feature_count = GSCV_RF.best_params_['feature__k']


print(f"Lasso MAE: {mae_lasso}, Lasso RMSE: {rmse_lasso}, Lasso
Feature Count: {lasso_feature_count}")
print(f"RFR MAE: {mae_RFR}, RFR RMSE: {rmse_RFR}, RFR Feature Count:
{RFR_feature_count}")
```

```
Lasso MAE: 1436.6510138328122, Lasso RMSE: 1860.2674084635623, Lasso
Feature Count: 19
RFR MAE: 16083.918675581288, RFR RMSE: 20313.575145161325, RFR Feature
Count: 6
```