Kode Random Forest VS Logistic Regression
Veronica-Eliandani

[1]:
```
import pandas as pd
import numpy as np

df_prt=pd.read_csv('DatasetUTS.csv')
df_prt.head(20)
```

[1]:

	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

[2]:
```
df_prt2=df_prt.drop('price',axis=1)
df_prt2.head(20)
```

[2]:

	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
```

[3]:
```
df_prt2['category'].value_counts()
```

[3]:
```
category
Basic     4344
Luxury    3065
Middle    2591
Name: count, dtype: int64
```

[4]:
```
print("data null \n", df_prt2.isnull().sum())
print("\ndata kosong \n", df_prt2.empty)
print("\ndata nnan \n",df_prt2.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 nnan
 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
```

[5]:
```
print("Sebelum Pengecekan data duplikat,",df_prt2.shape)

df_prt3 = df_prt2.drop_duplicates(keep='last')
print("Setelah Pengecekan data duplikat,", df_prt2.shape)
```

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

[6]:
```
from sklearn.model_selection import train_test_split
x = df_prt3.drop(columns=['category'], axis=1)
y = df_prt3['category']

X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.25,␣
 ↪random_state=71)

print(X_train.shape)
print(X_test.shape)
```

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

[7]:
```
import pandas as pd
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_enc = transform.fit_transform(X_train)

X_test_enc = transform.transform(X_test)

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())

print("Encoded Training Data:\n", df_train_enc.head(10))
print("Encoded Testing Data:\n", df_test_enc.head(10))
```

```
Encoded Training Data:
    onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                         0.0                         1.0
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

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

   onehotencoder__isnewbuilt_new  onehotencoder__isnewbuilt_old  \
0                            0.0                            1.0
1                            1.0                            0.0
2                            0.0                            1.0
3                            0.0                            1.0
```

```
4                              1.0                              0.0
5                              0.0                              1.0
6                              1.0                              0.0
7                              1.0                              0.0
8                              0.0                              1.0
9                              0.0                              1.0

   onehotencoder__hasstormprotector_no  onehotencoder__hasstormprotector_yes  \
0                                  1.0                                   0.0
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                                  1.0                                   0.0
8                                  0.0                                   1.0
9                                  1.0                                   0.0

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

   remainder__numberofrooms  remainder__floors  remainder__citycode  \
0                      93.0               98.0              68872.0
1                      97.0               20.0              93104.0
2                      86.0               27.0              61694.0
3                      10.0               94.0              15304.0
4                      98.0               63.0               1173.0
5                      46.0               19.0              14625.0
6                      47.0               23.0              80519.0
7                      82.0               21.0              62528.0
8                      92.0               73.0              39608.0
9                      55.0               76.0              67738.0

   remainder__citypartrange  remainder__numprevowners  remainder__made  \
0                       6.0                      10.0           2009.0
1                       9.0                       3.0           1993.0
2                      10.0                       9.0           1995.0
3                      10.0                       8.0           2005.0
```

```
4                      8.0                      8.0          2016.0
5                     10.0                      4.0          2021.0
6                      6.0                      2.0          2013.0
7                      7.0                      6.0          1995.0
8                      6.0                      7.0          2000.0
9                      3.0                      4.0          2011.0

   remainder__basement  remainder__attic  remainder__garage  \
0               1016.0            2441.0              792.0
1               6768.0            7803.0              901.0
2               1588.0            2201.0              789.0
3               4171.0            4794.0              574.0
4               2287.0            8181.0              426.0
5               9802.0            4844.0              984.0
6               2975.0            5534.0              900.0
7               5336.0            8216.0              634.0
8               4147.0            7554.0              606.0
9               5746.0            3337.0              220.0

   remainder__hasguestroom
0                      7.0
1                      2.0
2                      0.0
3                     10.0
4                      6.0
5                      9.0
6                      2.0
7                      0.0
8                      3.0
9                      8.0

[10 rows x 21 columns]
Encoded Testing Data:
    onehotencoder__hasyard_no  onehotencoder__hasyard_yes  \
0                         1.0                         0.0
1                         0.0                         1.0
2                         0.0                         1.0
3                         1.0                         0.0
4                         0.0                         1.0
5                         0.0                         1.0
6                         0.0                         1.0
7                         1.0                         0.0
8                         1.0                         0.0
9                         1.0                         0.0

   onehotencoder__haspool_no  onehotencoder__haspool_yes  \
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                         1.0                         0.0
6                         1.0                         0.0
7                         0.0                         1.0
8                         1.0                         0.0
9                         0.0                         1.0

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

   onehotencoder__hasstormprotector_no  onehotencoder__hasstormprotector_yes  \
0                                  0.0                                   1.0
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__hasstorageroom_no  onehotencoder__hasstorageroom_yes  …  \
0                               0.0                                1.0  …
1                               0.0                                1.0  …
2                               0.0                                1.0  …
3                               1.0                                0.0  …
4                               0.0                                1.0  …
5                               0.0                                1.0  …
6                               1.0                                0.0  …
7                               0.0                                1.0  …
8                               0.0                                1.0  …
9                               1.0                                0.0  …

   remainder__numberofrooms  remainder__floors  remainder__citycode  \
0                      10.0               87.0              41867.0
1                      35.0                2.0              78416.0
```

```
2                    28.0                20.0              47125.0
3                    42.0                65.0              70699.0
4                    42.0                76.0              67080.0
5                    19.0                97.0              78485.0
6                     2.0                11.0               3545.0
7                     7.0                24.0              19018.0
8                    96.0                41.0               2434.0
9                    49.0                66.0              93489.0

   remainder__citypartrange  remainder__numprevowners  remainder__made  \
0                       6.0                      10.0           1997.0
1                       1.0                       6.0           2016.0
2                       1.0                       3.0           2015.0
3                       9.0                       2.0           2010.0
4                       8.0                       2.0           2015.0
5                      10.0                       6.0           2017.0
6                       2.0                       4.0           2020.0
7                       7.0                       2.0           1998.0
8                       5.0                       1.0           2007.0
9                       3.0                       8.0           2001.0

   remainder__basement  remainder__attic  remainder__garage  \
0               8945.0            1203.0              142.0
1               3311.0            1636.0              967.0
2               8462.0             890.0              523.0
3               2336.0            8483.0              319.0
4               5678.0            8046.0              135.0
5               6040.0            2747.0              150.0
6                339.0            1049.0              245.0
7                 21.0            9711.0              657.0
8               1816.0            4036.0              534.0
9               5294.0            9984.0              276.0

   remainder__hasguestroom
0                     10.0
1                      6.0
2                      7.0
3                      2.0
4                      4.0
5                      3.0
6                      1.0
7                      7.0
8                      1.0
9                      5.0

[10 rows x 21 columns]
```

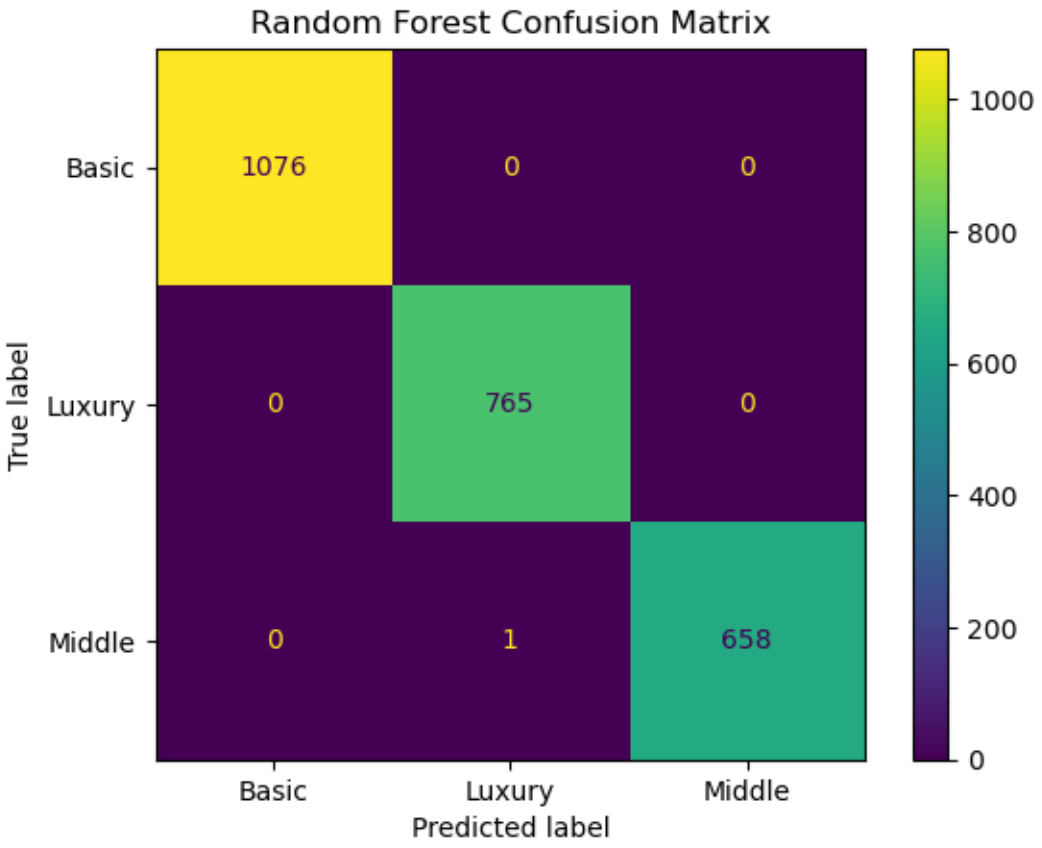
[8]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectPercentile, SelectKBest
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.metrics import␣
 ↪classification_report,confusion_matrix,ConfusionMatrixDisplay
import numpy as np

pipe_RF = Pipeline(steps=[
    ('data scaling', StandardScaler()),
    ('feature select', SelectKBest()),
    ('clf', RandomForestClassifier(random_state=71, 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]
    }
]

SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=71)

GSCV_RF = GridSearchCV(pipe_RF, params_grid_RF, cv=SKF)
```

```
GSCV_RF.fit(X_train_enc, y_train)

print("GSV Training Finished")
```

```
GSV Training Finished
```

[9]:
```
print("CV Score : {}".format(GSCV_RF.best_score_))
print("Test Score : {}".format(GSCV_RF.best_estimator_.score(X_test_enc,␣
 ↪y_test)))
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_enc)

import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, 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, RF_pred))
```

```
CV Score : 0.9994666666666667
Test Score : 0.9996
Best Model:  Pipeline(steps=[('data scaling', StandardScaler()),
                ('feature select', SelectPercentile(percentile=36)),
                ('clf',
                 RandomForestClassifier(class_weight='balanced', max_depth=4,
                                        n_estimators=150, random_state=71))])
Best Features: Index(['onehotencoder__hasyard_no', 'onehotencoder__hasyard_yes',
       'onehotencoder__haspool_no', 'onehotencoder__haspool_yes',
       'onehotencoder__isnewbuilt_new', 'onehotencoder__isnewbuilt_old',
       'remainder__squaremeters', 'remainder__garage'],
      dtype='object')
```




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

       Basic       1.00      1.00      1.00      1076
      Luxury       1.00      1.00      1.00       765
      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
```

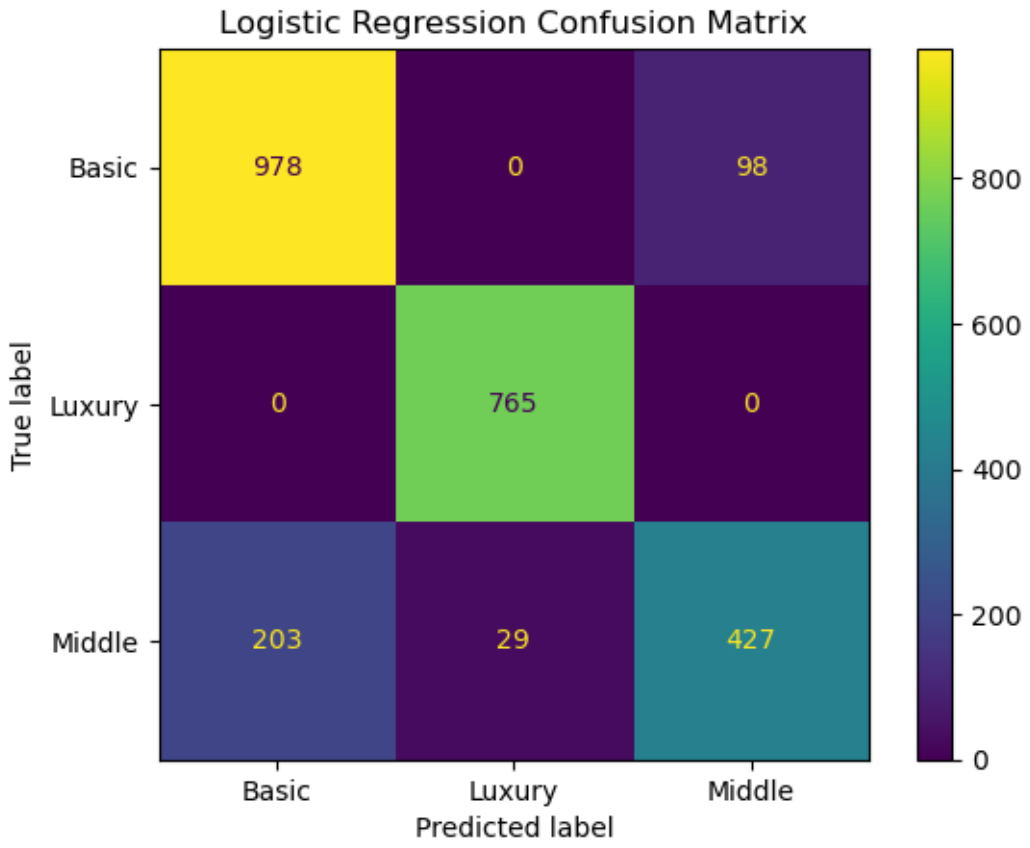
[10]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.feature_selection import SelectPercentile, SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.metrics import␣
 ↪classification_report,confusion_matrix,ConfusionMatrixDisplay
import numpy as np
```

```
pipe_logreg = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('feature_select', SelectKBest()),
    ('classifier', LogisticRegression(random_state=71, class_weight='balanced'))
])

params_grid_logreg = [
    {
        'scaler': [StandardScaler()],
        'feature_select__k': np.arange(2, 6),
        'classifier__penalty': ['l2'],
        'classifier__C': [0.1, 1, 10],
        'classifier__solver': ['liblinear']
    },
    {
        'scaler': [StandardScaler()],
        'feature_select': [SelectPercentile()],
        'feature_select__percentile': np.arange(20, 50),
        'classifier__penalty': ['l2'],
        'classifier__C': [0.1, 1, 10],
        'classifier__solver': ['liblinear']
    },
    {
        'scaler': [MinMaxScaler()],
        'feature_select__k': np.arange(2, 6),
        'classifier__penalty': ['l2'],
        'classifier__C': [0.1, 1, 10],
        'classifier__solver': ['liblinear']
    },
    {
        'scaler': [MinMaxScaler()],
        'feature_select': [SelectPercentile()],
        'feature_select__percentile': np.arange(20, 50),
        'classifier__penalty': ['l2'],
        'classifier__C': [0.1, 1, 10],
        'classifier__solver': ['liblinear']
    }
]

SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=71)

GSCV_logreg = GridSearchCV(pipe_logreg, params_grid_logreg, cv=SKF)
GSCV_logreg.fit(X_train_enc, y_train)

print("Logistic Regression Training Finished")
```

```
Logistic Regression Training Finished
```

[11]:
```
print("CV Score: {}".format(GSCV_logreg.best_score_))
print("Test Score: {}".format(GSCV_logreg.best_estimator_.score(X_test_enc,␣
 ↪y_test)))
print("Best Model: ", GSCV_logreg.best_estimator_)

mask = GSCV_logreg.best_estimator_.named_steps['feature_select'].get_support()
print("Best Features:", df_train_enc.columns[mask])

logreg_pred = GSCV_logreg.predict(X_test_enc)

import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
 ↪classification_report

cm = confusion_matrix(y_test, logreg_pred, labels=GSCV_logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=GSCV_logreg.
 ↪classes_)
disp.plot()
plt.title("Logistic Regression Confusion Matrix")
plt.show()

print("Classification Report:\n", classification_report(y_test, logreg_pred))
```

```
CV Score: 0.8712
Test Score: 0.868
Best Model:  Pipeline(steps=[('scaler', StandardScaler()),
                ('feature_select', SelectPercentile(percentile=36)),
                ('classifier',
                 LogisticRegression(C=10, class_weight='balanced',
                                    random_state=71, solver='liblinear'))])
Best Features: Index(['onehotencoder__hasyard_no', 'onehotencoder__hasyard_yes',
       'onehotencoder__haspool_no', 'onehotencoder__haspool_yes',
       'onehotencoder__isnewbuilt_new', 'onehotencoder__isnewbuilt_old',
       'remainder__squaremeters', 'remainder__garage'],
      dtype='object')
```




```
Classification Report:
              precision    recall  f1-score   support

       Basic       0.83      0.91      0.87      1076
      Luxury       0.96      1.00      0.98       765
      Middle       0.81      0.65      0.72       659

    accuracy                           0.87      2500
   macro avg       0.87      0.85      0.86      2500
weighted avg       0.87      0.87      0.86      2500
```

[12]:
```
import pickle
with open('BestModel_RF_LR_Seaborn.pkl', 'wb') as r:
    pickle.dump((GSCV_RF),r)

print("Model RF Disimpan")
```

```
Model RF Disimpan
```

Kode Gradient Boosting
Classifier VS Support Vector Machine
Szwagery

[4]:
```
import pandas as pd
import numpy as np

df_prt = pd.read_csv('DatasetUTS.csv')
df_prt.head(20)
```

[4]:

	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

[5]:
```
df_prt2 = df_prt.drop('price' ,axis=1)
df_prt2.head(50)
```

[5]:

	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
```

[6]:
```
df_prt2['category'].value_counts()
```

[6]:
```
category
Basic     4344
Luxury    3065
Middle    2591
Name: count, dtype: int64
```

[7]:
```
print("data null\n", df_prt2.isnull().sum())
print("\ndata kosong\n", df_prt2.empty)
print("\ndata nan\n", df_prt2.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
 <bound method DataFrame.sum of       squaremeters  numberofrooms  hasyard
haspool  floors  citycode  \
0            False          False    False    False   False     False
1            False          False    False    False   False     False
2            False          False    False    False   False     False
```

```
3            False          False    False    False   False     False
4            False          False    False    False   False     False
…            …            …      …        …       …       …
9995         False          False    False    False   False     False
9996         False          False    False    False   False     False
9997         False          False    False    False   False     False
9998         False          False    False    False   False     False
9999         False          False    False    False   False     False

      citypartrange  numprevowners   made  isnewbuilt  hasstormprotector  \
0             False          False  False       False              False
1             False          False  False       False              False
2             False          False  False       False              False
3             False          False  False       False              False
4             False          False  False       False              False
…             …            …    …         …                …
9995          False          False  False       False              False
9996          False          False  False       False              False
9997          False          False  False       False              False
9998          False          False  False       False              False
9999          False          False  False       False              False

      basement  attic  garage  hasstorageroom  hasguestroom  category
0        False  False   False           False         False     False
1        False  False   False           False         False     False
2        False  False   False           False         False     False
3        False  False   False           False         False     False
4        False  False   False           False         False     False
…        …    …     …             …           …       …
9995     False  False   False           False         False     False
9996     False  False   False           False         False     False
9997     False  False   False           False         False     False
9998     False  False   False           False         False     False
9999     False  False   False           False         False     False

[10000 rows x 17 columns]>
```

[8]:
```
print("Sebelum Pengecekan data duplikat, ", df_prt2.shape)
df_prt3 = df_prt2.drop_duplicates(keep='last')
print("Sebelum Pengecekan data duplikat, ", df_prt3.shape)
```

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

[9]:
```
from sklearn.model_selection import train_test_split
x = df_prt3.drop(columns=['category'],axis=1)
y = y=df_prt3['category']
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.
↪25,random_state=71)

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

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

[10]:
```
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_enc=transform.fit_transform(x_train)

x_test_enc=transform.fit_transform(x_test)

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

[10]:

	onehotencoder__hasyard_no	onehotencoder__hasyard_yes	\
0	0.0	1.0	
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	1.0	0.0	
11	1.0	0.0	
12	0.0	1.0	
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                         0.0                         1.0
19                         0.0                         1.0

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

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

```
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                                   0.0                                   1.0
4                                   1.0                                   0.0
5                                   0.0                                   1.0
6                                   0.0                                   1.0
7                                   1.0                                   0.0
8                                   0.0                                   1.0
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                                  0.0                                   1.0
16                                  1.0                                   0.0
17                                  0.0                                   1.0
18                                  0.0                                   1.0
19                                  0.0                                   1.0

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

    remainder__numberofrooms  remainder__floors  remainder__citycode  \
```

0	93.0	98.0	68872.0
1	97.0	20.0	93104.0
2	86.0	27.0	61694.0
3	10.0	94.0	15304.0
4	98.0	63.0	1173.0
5	46.0	19.0	14625.0
6	47.0	23.0	80519.0
7	82.0	21.0	62528.0
8	92.0	73.0	39608.0
9	55.0	76.0	67738.0
10	40.0	24.0	31204.0
11	87.0	32.0	99233.0
12	5.0	35.0	18130.0
13	56.0	100.0	93052.0
14	91.0	12.0	93863.0
15	11.0	83.0	37564.0
16	89.0	21.0	2882.0
17	4.0	7.0	36027.0
18	41.0	1.0	85611.0
19	44.0	89.0	2587.0

	remainder__citypartrange	remainder__numprevowners	remainder__made	\
0	6.0	10.0	2009.0	
1	9.0	3.0	1993.0	
2	10.0	9.0	1995.0	
3	10.0	8.0	2005.0	
4	8.0	8.0	2016.0	
5	10.0	4.0	2021.0	
6	6.0	2.0	2013.0	
7	7.0	6.0	1995.0	
8	6.0	7.0	2000.0	
9	3.0	4.0	2011.0	
10	3.0	4.0	2004.0	
11	2.0	6.0	2013.0	
12	1.0	3.0	1990.0	
13	3.0	4.0	2012.0	
14	6.0	4.0	2015.0	
15	1.0	9.0	1998.0	
16	3.0	7.0	2012.0	
17	10.0	7.0	1991.0	
18	4.0	6.0	1994.0	
19	5.0	3.0	2021.0	

	remainder__basement	remainder__attic	remainder__garage	\
0	1016.0	2441.0	792.0	
1	6768.0	7803.0	901.0	
2	1588.0	2201.0	789.0	

```
3                4171.0           4794.0              574.0
4                2287.0           8181.0              426.0
5                9802.0           4844.0              984.0
6                2975.0           5534.0              900.0
7                5336.0           8216.0              634.0
8                4147.0           7554.0              606.0
9                5746.0           3337.0              220.0
10               3190.0           4409.0              337.0
11               3694.0           7923.0              317.0
12                999.0           2259.0              974.0
13               6358.0           7861.0              304.0
14               6751.0           7894.0              158.0
15               6004.0           4723.0              803.0
16               1540.0           5378.0              318.0
17               5623.0           2451.0              394.0
18               7386.0           7266.0              805.0
19               5328.0           7061.0              826.0

    remainder__hasguestroom
0                       7.0
1                       2.0
2                       0.0
3                      10.0
4                       6.0
5                       9.0
6                       2.0
7                       0.0
8                       3.0
9                       8.0
10                      4.0
11                      1.0
12                      0.0
13                      3.0
14                      4.0
15                      5.0
16                      5.0
17                      0.0
18                      9.0
19                      2.0

[20 rows x 21 columns]
```

[11]:
```
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
```

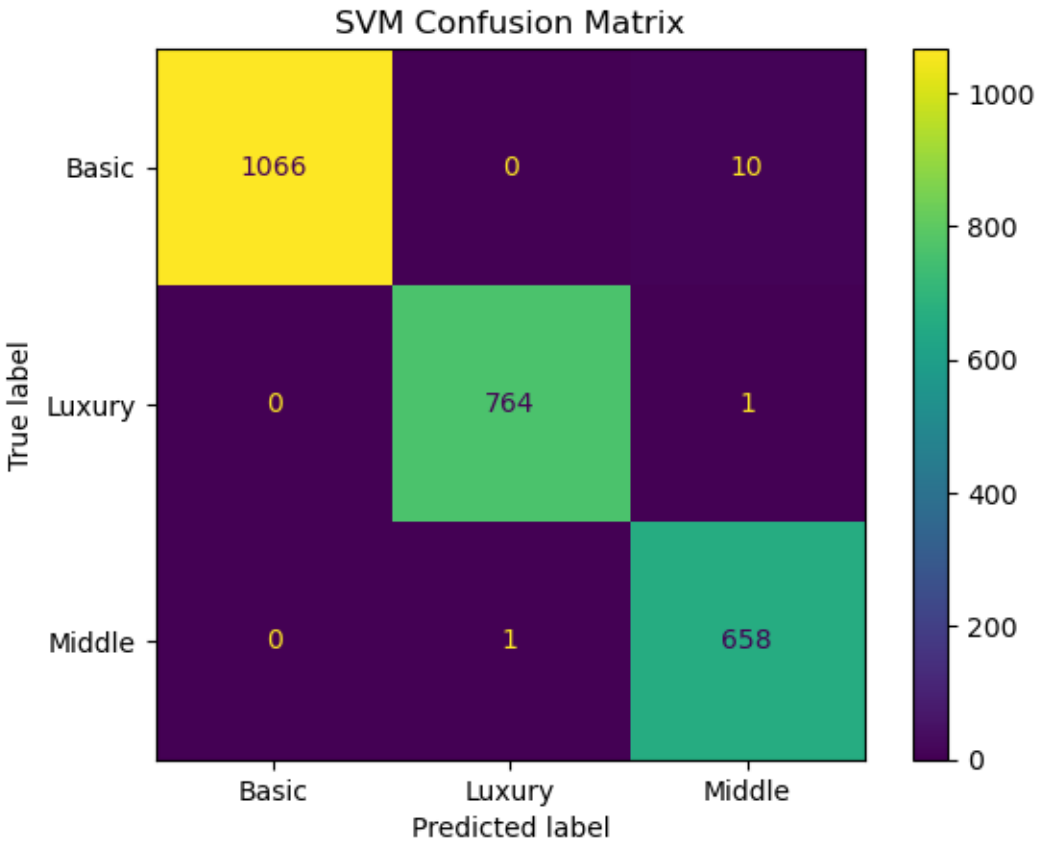
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪ConfusionMatrixDisplay

pipe_svm = Pipeline(steps=[
    ('scale', MinMaxScaler()),
    ('feat_select', SelectKBest()),
    ('clf', SVC(class_weight='balanced')),
])

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]
    }
]

estimator_svm = Pipeline(pipe_svm)

SKF = StratifiedKFold(n_splits=5, shuffle=True, random_state=71)
```

```
GSCV_SVM = GridSearchCV(pipe_svm, params_grid_svm, cv=SKF)

GSCV_SVM.fit(x_train_enc, y_train)
print("GSCV training finished")
```

```
GSCV training finished
```

[12]:
```
print("CV Score : {}".format(GSCV_SVM.best_score_))

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

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])

SVM_pred = GSCV_SVM.predict(x_test_enc)

import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, SVM_pred, labels=GSCV_SVM.classes_)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=GSCV_SVM.
 ↪classes_)
disp.plot()
plt.title("SVM Confusion Matrix")
plt.show()

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

```
CV Score : 0.9942666666666666
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      1076
      Luxury       1.00      1.00      1.00       765
      Middle       0.98      1.00      0.99       659

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

[13]:
```
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=71))])
```

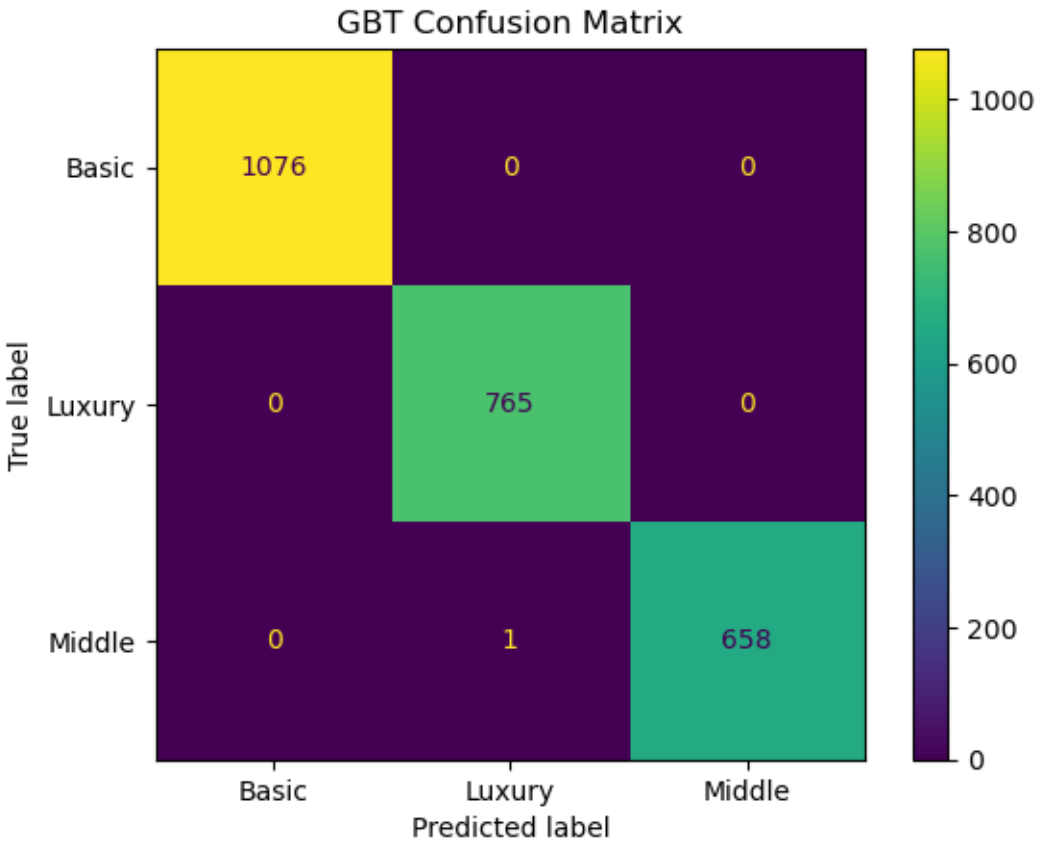
```
params_grid_GBT = [
    {
        'feat_select__k': np.arange(2,6),
     'clf__max_depth': [*np.arange(4, 5)],
     'clf__n_estimators': [100, 150],
     'clf__learning_rate': [0.01,0.1,1]
    },
    {
        'feat_select': [SelectPercentile()],
        'feat_select__percentile': np.arange(20, 50),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

```
GSCV training finished
```

[14]:
```
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_SVM.best_estimator_.named_steps['feat_select'].get_support()
print("Best features:", df_train_enc.columns[mask])
```

```
GBT_pred = GSCV_GBT.predict(x_test_enc)

import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, GBT_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, GBT_pred))
```

```
CV Score : 0.9989333333333335
Test Score: 0.9996
Best Model: {} Pipeline(steps=[('feat_select', SelectPercentile(percentile=31)),
                ('clf',
                 GradientBoostingClassifier(learning_rate=0.01, max_depth=4,
                                            random_state=71))])
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 GBT:
              precision    recall  f1-score   support

       Basic       1.00      1.00      1.00      1076
      Luxury       1.00      1.00      1.00       765
      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
```

[15]:
```
import pickle
with open('klasifikasi_model.pkl', 'wb') as r:
    pickle.dump((GSCV_GBT),r)

print("Model GBT Disimpan")
```

```
Model GBT Disimpan
```

Kode Ridge Regression VS Support Vector Regressor
I Made Ivan

[84]:
```
import pandas as  pd
import numpy as np

df_prt=pd.read_csv(r'D:\SEMESTER 5\Machine Learning\TESTUTS\Dataset UTS_Gasal␣
 ↪2425.csv')
df_prt.head()
```

[84]:

	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	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

[85]:
```
df_prt2=df_prt.drop(['category'],axis=1)
df_prt2.head()
```

[85]:

	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
```

[86]:
```
df_prt2.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
```

[87]:
```
df_prt.describe()
```

[87]:
```
       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
```

[88]:
```
print(df_prt2['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
```

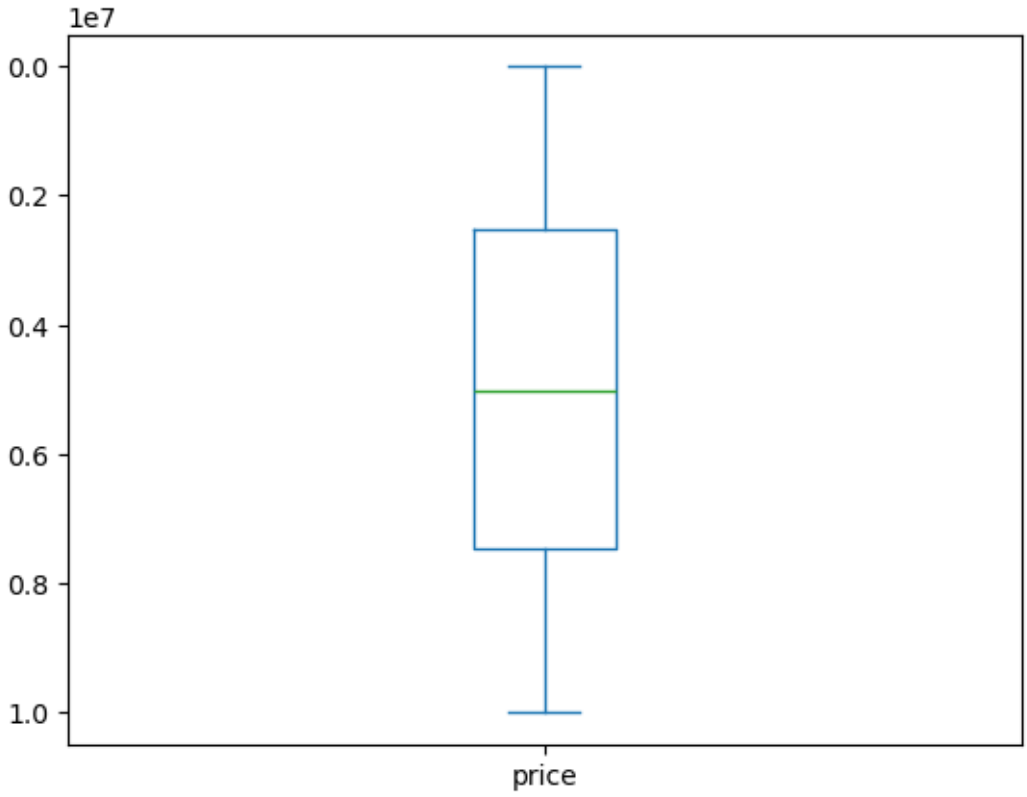
[89]:
```
print("data null \n", df_prt2.isnull().sum())
print("data kosong \n", df_prt2.empty)
print("data nan \n",df_prt2.isna().sum())
```

```
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
```

[90]:
```
import matplotlib.pyplot as plt
df_prt2.price.plot(kind='box')
plt.gca().invert_yaxis()
plt.show
```

[90]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



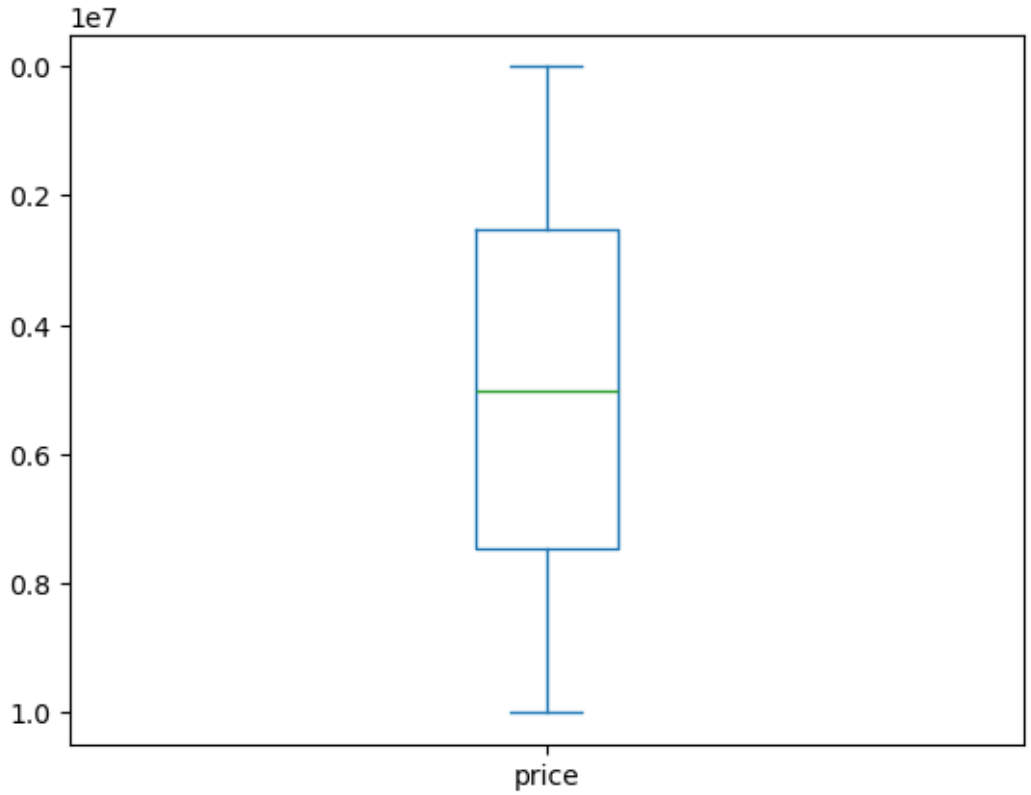
[91]:
```
print("Sebelum Pengecekan data duplikat, ",df_prt2.shape)
df_prt3=df_prt2.drop_duplicates(keep='last')
print("Setelah pengecekan data duplikat, ",df_prt3.shape)
```

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

[92]:
```
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_prt_clean= remove_outlier(df_prt3)
print("Jumlah baris DataFrame sebelum dibuang outlier", df_prt3.shape[0])
print("jumlah baris DataFrame sesudah dibuang outlier", df_prt_clean.shape[0])
df_prt_clean.price.plot(kind='box', vert=True)

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

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



[93]:
```
from sklearn.model_selection import train_test_split
x=df_prt_clean.drop('price', axis =1)
y=y=df_prt_clean['price']
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.30,␣
 ↪random_state=71)
print(x_train.shape)
print(x_test.shape)
```

```
(7000, 16)
(3000, 16)
```

[94]:
```
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_enc=transform.fit_transform(x_train)

x_test_enc=transform.fit_transform(x_test)

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()
df_test_enc.head()
```

[94]:

	onehotencoder__hasyard_no	onehotencoder__hasyard_yes	\
0	1.0	0.0	
1	0.0	1.0	
2	0.0	1.0	
3	1.0	0.0	
4	0.0	1.0	

	onehotencoder__haspool_no	onehotencoder__haspool_yes	\
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	

	onehotencoder__isnewbuilt_new	onehotencoder__isnewbuilt_old	\
0	0.0	1.0	
1	0.0	1.0	
2	0.0	1.0	
3	0.0	1.0	
4	1.0	0.0	

	onehotencoder__hasstormprotector_no	onehotencoder__hasstormprotector_yes	\
0	0.0	1.0	
1	0.0	1.0	
2	1.0	0.0	
3	1.0	0.0	
4	0.0	1.0	

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

   remainder__numberofrooms  remainder__floors  remainder__citycode  \
0                      10.0               87.0              41867.0
1                      35.0                2.0              78416.0
2                      28.0               20.0              47125.0
3                      42.0               65.0              70699.0
4                      42.0               76.0              67080.0

   remainder__citypartrange  remainder__numprevowners  remainder__made  \
0                       6.0                      10.0           1997.0
1                       1.0                       6.0           2016.0
2                       1.0                       3.0           2015.0
3                       9.0                       2.0           2010.0
4                       8.0                       2.0           2015.0

   remainder__basement  remainder__attic  remainder__garage  \
0               8945.0            1203.0              142.0
1               3311.0            1636.0              967.0
2               8462.0             890.0              523.0
3               2336.0            8483.0              319.0
4               5678.0            8046.0              135.0

   remainder__hasguestroom
0                     10.0
1                      6.0
2                      7.0
3                      2.0
4                      4.0

[5 rows x 21 columns]
```

[121]:
```
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_enc, y_train)

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_enc)
mse_Ridge = mean_squared_error(y_test, Ridge_predict)
mae_Ridge = mean_absolute_error (y_test, 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 Eror: {}".format(np.sqrt(mse_Ridge)))
```

```
Best model:Pipeline(steps=[('scale', StandardScaler()),
                ('feature_selection',
                 SelectKBest(k=15,
                             score_func=<function f_regression at
0x000001D0D3870A40>)),
                ('reg', Ridge(alpha=0.01))])
Ridge best parameters:{'feature_selection__k': 15, 'reg__alpha': 0.01}
Koefisien/bobot:[-7.47070712e+02  7.47070712e+02 -7.44570858e+02  7.44570880e+02
  3.96052910e+01 -3.96053005e+01 -2.43537338e+01  2.43537202e+01
  2.88294485e+06  1.83512203e+01  1.38622056e+02 -1.81546415e+01
 -6.55762786e+00  4.83577972e+01 -5.44061594e+01]
Intercept/bias:4985857.002414286
Ridge Mean Squared Error (MSE): 6160574.544214095
Ridge Mean Absolute Error (MAE): 1980.657218509307
Ridge Root Mean Squared Eror: 2482.0504717297945
```

[122]:
```
df_results= pd.DataFrame(y_test, columns=['price'])
df_results=pd.DataFrame(y_test)
```

```
df_results['Ridge Prediction']=Ridge_predict

df_results['Selisih_price_RR']=df_results['Ridge Prediction']-␣
 ↪df_results['price']
df_results.head()
```

[122]:

	price	Ridge Prediction	Selisih_price_RR
312	7694726.7	7.696913e+06	2185.822744
7926	1611775.6	1.610822e+06	-953.474572
5782	3253846.5	3.254829e+06	982.001773
8188	2359701.0	2.359300e+06	-401.472867
795	8007425.5	8.006832e+06	-593.390352

[123]:
```
df_results.describe()
```

[123]:

	price	Ridge Prediction	Selisih_price_RR
count	3.000000e+03	3.000000e+03	3000.000000
mean	5.011159e+06	5.011128e+06	-30.666039
std	2.864485e+06	2.864492e+06	2482.274770
min	1.679920e+04	1.831542e+04	-9598.444902
25%	2.526030e+06	2.526748e+06	-1537.662663
50%	5.077257e+06	5.077359e+06	434.464843
75%	7.480795e+06	7.482453e+06	1861.156444
max	1.000294e+07	1.000077e+07	6464.129853

[98]:
```
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_enc, y_train)
```

```
print("Best model:{}".format(GSCV_SVR.best_estimator_))
print("SVR best parrameter:{}".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_enc)
mse_SVR= mean_squared_error(y_test, SVR_predict)
mae_SVR = mean_absolute_error(y_test, 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(score_func=<function f_regression at
0x000001D0D3870A40>)),
                ('reg', SVR(C=100, epsilon=1, kernel='linear'))])
SVR best parrameter:{'feature_selection__k': 10, 'reg__C': 100, 'reg__epsilon':
1}
koefisien/bobot: [[  6400.01671843  -6400.01671843 -10200.00041633
10200.00041633
   -4000.14694687   4000.14694687 605098.56123321  15049.56098259
   -4828.01628375  -9910.67321994]]
Intercept/bias: [4973262.08457655]
SVR Mean Squared Error (MSE): 5124644869912.438
SVR Mean Absolute Error (MAE): 1954758.342532865
SVR Root Mean Squared Error: 2263767.848060494
```

[99]:
```
df_results['SVR Prediction']= SVR_predict

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

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

[99]:

	price	SVR Prediction	Selisih_price_SVR
312	7694726.7	5.603579e+06	-2.091148e+06
7926	1611775.6	4.272577e+06	2.660802e+06
5782	3253846.5	4.603777e+06	1.349931e+06
8188	2359701.0	4.432184e+06	2.072483e+06
795	8007425.5	5.599749e+06	-2.407677e+06

[100]:
```
df_results.describe()
```

[100]:
```
              price  SVR Prediction  Selisih_price_SVR
count  3.000000e+03    3.000000e+03       3.000000e+03
mean   5.011159e+06    4.979057e+06      -3.210212e+04
std    2.864485e+06    6.016515e+05       2.263918e+06
min    1.679920e+04    3.863237e+06      -4.007492e+06
25%    2.526030e+06    4.455131e+06      -1.977637e+06
50%    5.077257e+06    4.996517e+06      -9.487858e+04
75%    7.480795e+06    5.487022e+06       1.937160e+06
max    1.000294e+07    6.069531e+06       3.958099e+06
```

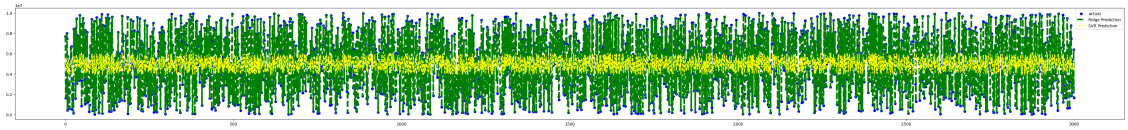
[124]:
```
df_results = pd.DataFrame({'price': y_test})

df_results['Ridge Prediction']=Ridge_predict

df_results['Selisih_IPK_RR'] = df_results['price']-df_results['Ridge␣
 ↪Prediction']

df_results['SVR Prediction']= SVR_predict
df_results['Selisih_IPK_SVR']= df_results['price']- df_results['SVR Prediction']


df_results.head()
```

[124]:
```
          price  Ridge Prediction  Selisih_IPK_RR  SVR Prediction  \
312   7694726.7      7.696913e+06    -2185.822744    5.603579e+06
7926  1611775.6      1.610822e+06      953.474572    4.272577e+06
5782  3253846.5      3.254829e+06     -982.001773    4.603777e+06
8188  2359701.0      2.359300e+06      401.472867    4.432184e+06
795   8007425.5      8.006832e+06      593.390352    5.599749e+06

      Selisih_IPK_SVR
312      2.091148e+06
7926    -2.660802e+06
5782    -1.349931e+06
8188    -2.072483e+06
795      2.407677e+06
```

[126]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(50,5))

data_len= range(len(y_test))

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=4,linestyle="dashed")
plt.plot(data_len, df_results['SVR Prediction'], label ="SVR Prediction",␣
 ↪color="yellow", linewidth=2,linestyle="--")

plt.legend()
plt.show
```

[126]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[129]:
```
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:1980.657218509307, Ridge RMSE: 2482.0504717297945,Ridge Feature Count
:15
SVR MAE:1954758.342532865, SVR RMSE: 2263767.848060494, SVR Feature Count :10
```

[1]:
```
import pickle
best_model = GSCV_SVR.best_estimator_

with open('BestModel_REG_SVR_Seaborn.pkl','wb') as f:
```

```
    pickle.dump(best_model, f)


print("Model terbaik berhasil disimpan ke 'BestModel_REG_SVR_Seaborn.pkl'")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 2
      1 import pickle
----> 2 best_model = GSCV_SVR.best_estimator_
      4 with open('BestModel_REG_SVR_Seaborn.pkl','wb') as f:
      5     pickle.dump(best_model, f)

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

Kode Lasso Regression VS Random Forest Regressor
Josua

[31]:
```
import pandas as  pd
import numpy as np

df_prt=pd.read_csv(r'C:\Users\Lenovo\Documents\Kuliah - UAJY\Semester V -␣
 ↪2024-2025\Pembelajaran mesin dan Pembelajaran Mendalam - A\Projek UTS Gasal␣
 ↪20242025-20241018\Dataset UTS_Gasal 2425.csv')
df_prt.head()
```

[31]:

	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	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

[32]:
```
df_prt2=df_prt.drop(['category'],axis=1)
df_prt2.head()
```

[32]:

	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
```

[33]:
```
df_prt2.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
```

[34]:
```
df_prt.describe()
```

[34]:
```
        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
```

[35]:
```
print(df_prt2['price'].value_counts())
```

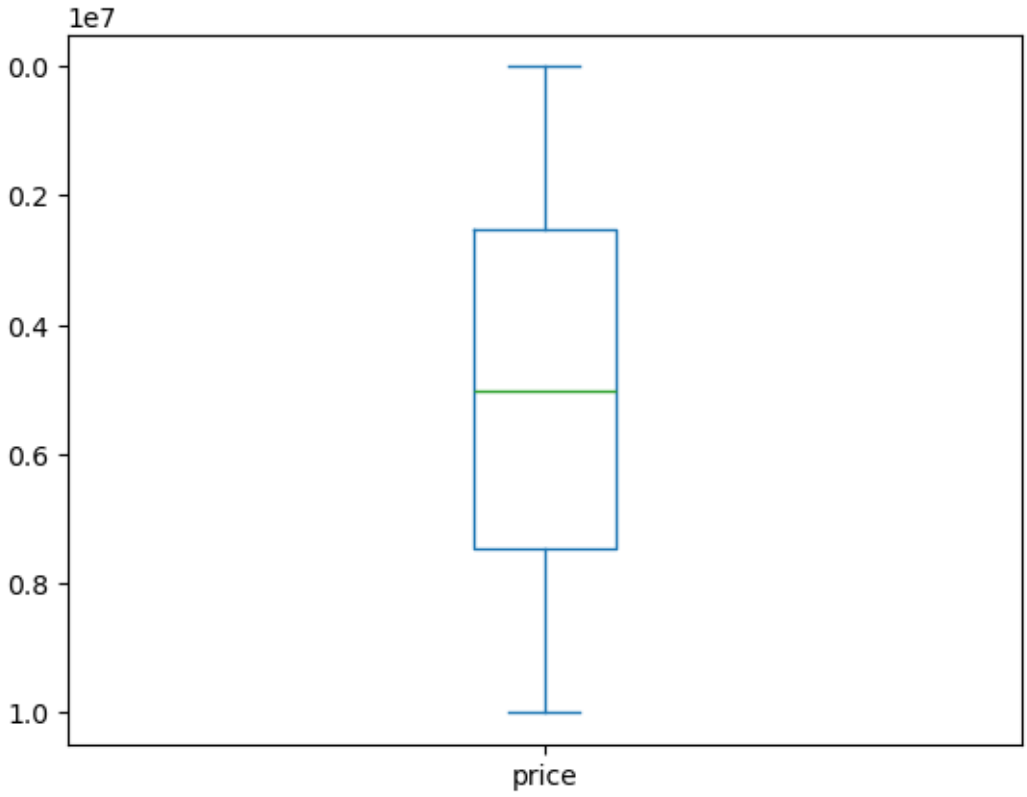
```
price
7559081.5    1
2600292.1    1
3804577.4    1
3658559.7    1
2316639.4    1
            ..
5555606.6    1
5501007.5    1
9986201.2    1
9104801.8    1
146708.4     1
Name: count, Length: 10000, dtype: int64
```

[36]:
```
print("data null \n", df_prt2.isnull().sum())
print("data kosong \n", df_prt2.empty)
print("data nan \n",df_prt2.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
```

[37]:
```
import matplotlib.pyplot as plt
df_prt2.price.plot(kind='box')
plt.gca().invert_yaxis()
plt.show
```

[37]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



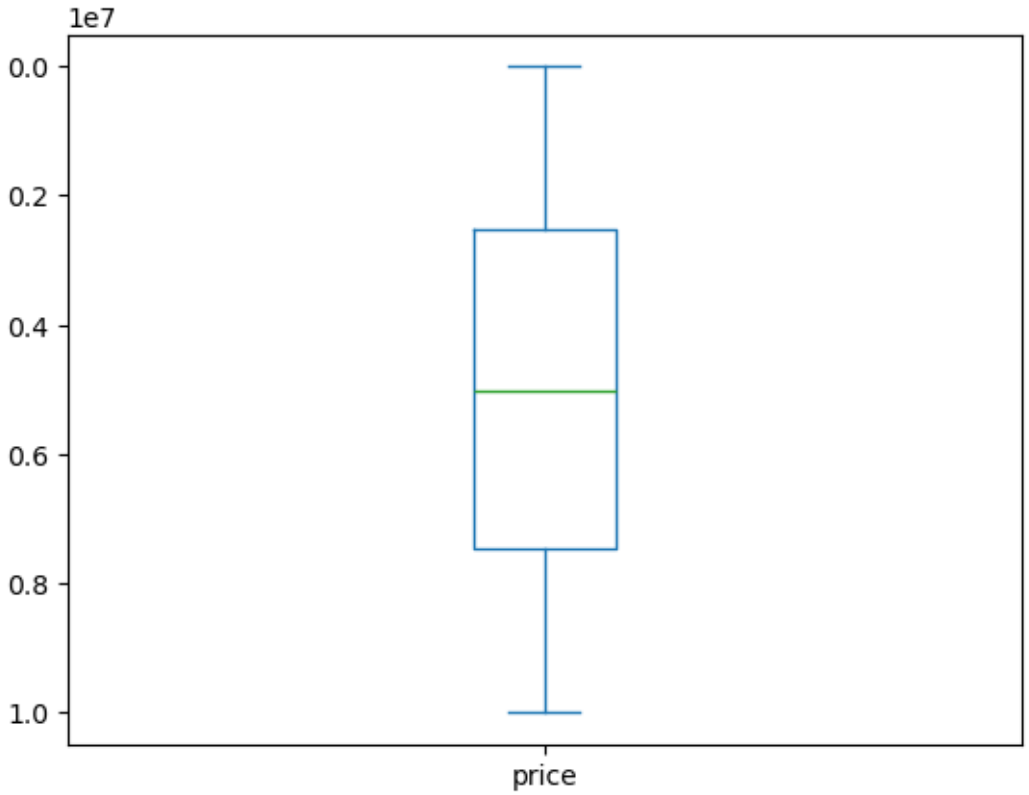
[38]:
```
print("Sebelum Pengecekan data duplikat, ",df_prt2.shape)
df_prt3=df_prt2.drop_duplicates(keep='last')
print("Setelah pengecekan data duplikat, ",df_prt3.shape)
```

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

[39]:
```
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_prt_clean= remove_outlier(df_prt3)
print("Jumlah baris DataFrame sebelum dibuang outlier", df_prt3.shape[0])
print("jumlah baris DataFrame sesudah dibuang outlier", df_prt_clean.shape[0])
df_prt_clean.price.plot(kind='box', vert=True)

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

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



[40]:
```
from sklearn.model_selection import train_test_split

x=df_prt_clean.drop('price', axis =1)
y=y=df_prt_clean['price']

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.30,␣
  ↪random_state=71)
print(x_train.shape)
print(x_test.shape)
```

```
(7000, 16)
```

```
(3000, 16)
```

[41]:
```
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_enc=transform.fit_transform(x_train)

x_test_enc=transform.fit_transform(x_test)

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()
df_test_enc.head()
```

[41]:

	onehotencoder__hasyard_no	onehotencoder__hasyard_yes	\
0	1.0	0.0	
1	0.0	1.0	
2	0.0	1.0	
3	1.0	0.0	
4	0.0	1.0	

	onehotencoder__haspool_no	onehotencoder__haspool_yes	\
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	

	onehotencoder__isnewbuilt_new	onehotencoder__isnewbuilt_old	\
0	0.0	1.0	
1	0.0	1.0	
2	0.0	1.0	
3	0.0	1.0	
4	1.0	0.0	

	onehotencoder__hasstormprotector_no	onehotencoder__hasstormprotector_yes	\
0	0.0	1.0	
1	0.0	1.0	
2	1.0	0.0	

```
3                                 1.0                                   0.0
4                                 0.0                                   1.0

   onehotencoder__hasstorageroom_no  onehotencoder__hasstorageroom_yes  …  \
0                               0.0                                1.0  …
1                               0.0                                1.0  …
2                               0.0                                1.0  …
3                               1.0                                0.0  …
4                               0.0                                1.0  …

   remainder__numberofrooms  remainder__floors  remainder__citycode  \
0                      10.0               87.0              41867.0
1                      35.0                2.0              78416.0
2                      28.0               20.0              47125.0
3                      42.0               65.0              70699.0
4                      42.0               76.0              67080.0

   remainder__citypartrange  remainder__numprevowners  remainder__made  \
0                       6.0                      10.0           1997.0
1                       1.0                       6.0           2016.0
2                       1.0                       3.0           2015.0
3                       9.0                       2.0           2010.0
4                       8.0                       2.0           2015.0

   remainder__basement  remainder__attic  remainder__garage  \
0               8945.0            1203.0              142.0
1               3311.0            1636.0              967.0
2               8462.0             890.0              523.0
3               2336.0            8483.0              319.0
4               5678.0            8046.0              135.0

   remainder__hasguestroom
0                     10.0
1                      6.0
2                      7.0
3                      2.0
4                      4.0

[5 rows x 21 columns]
```

[42]:
```
from sklearn.linear_model import Lasso
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_Lasso = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)),
    ('reg', Lasso(max_iter=1000))
])

param_grid_Lasso = {
    'reg__alpha': [0.01, 0.1, 1.0, 10, 100],
    'feature_selection__k': np.arange(1, x_train_enc.shape[1] + 1)
}

GSCV_Lasso = GridSearchCV(pipe_Lasso, param_grid_Lasso, cv=5,
                          scoring='neg_mean_squared_error')

GSCV_Lasso.fit(x_train_enc, y_train)

print("Best model: {}".format(GSCV_Lasso.best_estimator_))
print("Lasso best parameters: {}".format(GSCV_Lasso.best_params_))

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_enc)

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

print("Lasso Mean Squared Error (MSE): {}".format(mse_Lasso))
print("Lasso Mean Absolute Error (MAE): {}".format(mae_Lasso))
print("Lasso Root Mean Squared Error : {}".format(np.sqrt(mse_Lasso)))
```

```
Best model: Pipeline(steps=[('scale', StandardScaler()),
                ('feature_selection',
                 SelectKBest(k=21,
                             score_func=<function f_regression at
0x00000199BAD5CC20>)),
                ('reg', Lasso(alpha=10))])
Lasso best parameters: {'feature_selection__k': 21, 'reg__alpha': 10}
Koefisien/bobot: [-1.47311841e+03  1.10178787e-13 -1.47477770e+03
2.07884503e-15
  6.01957021e+01 -0.00000000e+00 -6.09633387e+01  2.07884503e-14
 -7.50646917e+00  5.82076609e-14  2.88293144e+06 -0.00000000e+00
  1.56711009e+03 -0.00000000e+00  1.25698947e+02 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00  1.84582788e+01
 -4.57599517e+00]
```

```
Intercept/bias: 4985857.002414286
Lasso Mean Squared Error (MSE): 3684095.8047116166
Lasso Mean Absolute Error (MAE): 1500.9684308209926
Lasso Root Mean Squared Error : 1919.3998553484412
```

[43]:
```
df_results= pd.DataFrame(y_test, columns=['price'])
df_results=pd.DataFrame(y_test)
df_results['Lasso Prediction']=Lasso_predict

df_results['Selisih_price_RR']=df_results['Lasso Prediction']-␣
 ↪df_results['price']
df_results.head()
```

[43]:

	price	Lasso Prediction	Selisih_price_RR
312	7694726.7	7.699083e+06	4356.273967
7926	1611775.6	1.608193e+06	-3582.826481
5782	3253846.5	3.253207e+06	-639.321545
8188	2359701.0	2.360080e+06	379.126360
795	8007425.5	8.008200e+06	774.391222

[44]:
```
df_results.describe()
```

[44]:

	price	Lasso Prediction	Selisih_price_RR
count	3.000000e+03	3.000000e+03	3000.000000
mean	5.011159e+06	5.011127e+06	-31.388900
std	2.864485e+06	2.864471e+06	1919.463116
min	1.679920e+04	1.632142e+04	-6989.863933
25%	2.526030e+06	2.525897e+06	-1298.239677
50%	5.077257e+06	5.075742e+06	3.999044
75%	7.480795e+06	7.482028e+06	1195.909671
max	1.000294e+07	1.000120e+07	6914.646163

[45]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error

pipe_rf = Pipeline(steps=[
            ('scale', StandardScaler()),
            ('feature_selection', SelectKBest(score_func=f_regression)),
            ('reg', RandomForestRegressor(random_state=0))
            ])

param_grid_rf = {
    'scale': [StandardScaler(), MinMaxScaler()],
```

```
    'feature_selection__k': np.arange(5, 20),
    'reg__max_depth': np.arange(4, 6),
    'reg__n_estimators': [100, 150]
}

GSCV_rf = GridSearchCV(pipe_rf, param_grid_rf, cv=5,
                       scoring='neg_mean_squared_error')

GSCV_rf.fit(x_train_enc, y_train)

print("Best model: {}".format(GSCV_rf.best_estimator_))
print("Random Forest best parameters: {}".format(GSCV_rf.best_params_))

rf_predict = GSCV_rf.predict(x_test_enc)

mse_rf = mean_squared_error(y_test, rf_predict)
mae_rf = mean_absolute_error(y_test, rf_predict)

print("RF Mean Squared Error (MSE): {}".format(mse_rf))
print("RF Mean Absolute Error (MAE): {}".format(mae_rf))
print("RF Root Mean Squared Error : {}".format(np.sqrt(mse_rf)))
```

```
Best model: Pipeline(steps=[('scale', MinMaxScaler()),
                ('feature_selection',
                 SelectKBest(k=17,
                             score_func=<function f_regression at
0x00000199BAD5CC20>)),
                ('reg', RandomForestRegressor(max_depth=5, random_state=0))])
Random Forest best parameters: {'feature_selection__k': 17, 'reg__max_depth': 5,
'reg__n_estimators': 100, 'scale': MinMaxScaler()}
RF Mean Squared Error (MSE): 4025392643.7151427
RF Mean Absolute Error (MAE): 55679.41427729769
RF Root Mean Squared Error : 63445.982092762526
```

[46]:
```
df_results['RF Prediction'] = rf_predict
df_results = pd.DataFrame(y_test)
df_results['RF Prediction'] = rf_predict

df_results['Selisih_price_RF'] = df_results['RF Prediction'] -␣
 ↪df_results['price']

df_results.head()
```

[46]:

	price	RF Prediction	Selisih_price_RF
312	7694726.7	7.662325e+06	-32401.618400
7926	1611775.6	1.714107e+06	102331.032568
5782	3253846.5	3.260558e+06	6711.562914

```
8188  2359701.0   2.340957e+06     -18743.632949
795   8007425.5   7.961173e+06     -46252.005161
```

[47]:
```
df_results.describe()
```

[47]:
```
              price  RF Prediction  Selisih_price_RF
count  3.000000e+03   3.000000e+03       3000.000000
mean   5.011159e+06   5.012454e+06       1295.560606
std    2.864485e+06   2.860337e+06      63443.327874
min    1.679920e+04   1.623675e+05    -145931.208126
25%    2.526030e+06   2.619698e+06     -55080.791351
50%    5.077257e+06   5.150637e+06        213.473997
75%    7.480795e+06   7.429236e+06      58800.239989
max    1.000294e+07   9.857014e+06     145568.337350
```

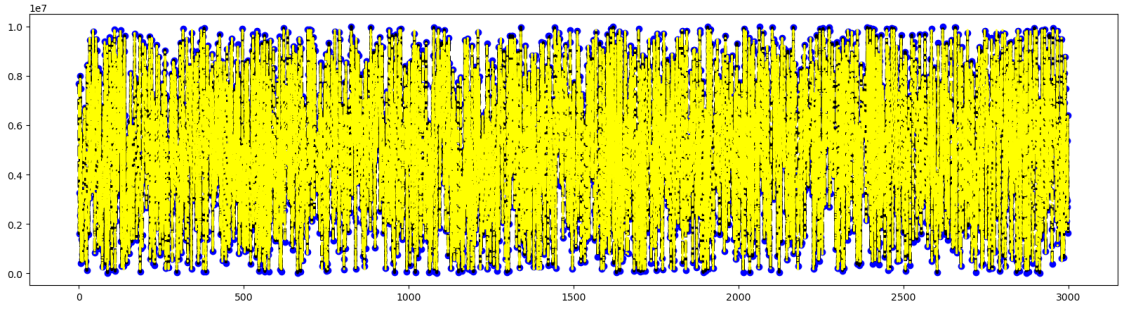
[49]:
```
df_results = pd.DataFrame({'price': y_test})

df_results['Lasso Prediction'] = Lasso_predict
df_results['Selisih_price_LR'] = df_results['price'] - df_results['Lasso␣
 ↪Prediction']

df_results['RF Prediction'] = rf_predict
df_results['Selisih_price_RF'] = df_results['price'] - df_results['RF␣
 ↪Prediction']

df_results.head()
```

[49]:
```
          price  Lasso Prediction  Selisih_price_LR  RF Prediction  \
312   7694726.7      7.699083e+06      -4356.273967   7.662325e+06
7926  1611775.6      1.608193e+06       3582.826481   1.714107e+06
5782  3253846.5      3.253207e+06        639.321545   3.260558e+06
8188  2359701.0      2.360080e+06       -379.126360   2.340957e+06
795   8007425.5      8.008200e+06       -774.391222   7.961173e+06

      Selisih_price_RF
312       32401.618400
7926    -102331.032568
5782      -6711.562914
8188      18743.632949
795       46252.005161
```

[50]:
```
import matplotlib.pyplot as plt

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

```
plt.plot(data_len, df_results['Lasso Prediction'], label='Lasso Prediction',␣
 ↪color="black", linewidth=3, linestyle="--")
plt.plot(data_len, df_results['RF Prediction'], label='RF Prediction',␣
 ↪color="yellow", linewidth=2, linestyle="-.")
```

[50]:
```
[<matplotlib.lines.Line2D at 0x199c7e0ce90>]
```



[51]:
```
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␣
 ↪Prediction'])
rmse_lasso = np.sqrt(mean_squared_error(df_results['price'], df_results['Lasso␣
 ↪Prediction']))
lasso_feature_count = GSCV_Lasso.best_params_['feature_selection__k']

mae_RF = mean_absolute_error(df_results['price'], df_results['RF Prediction'])
rmse_RF = np.sqrt(mean_squared_error(df_results['price'], df_results['RF␣
 ↪Prediction']))
RF_feature_count = GSCV_rf.best_params_['feature_selection__k']

print(f"Lasso MAE: {mae_lasso}, Lasso RMSE: {rmse_lasso}, Lasso Feature Count:␣
 ↪{lasso_feature_count}")
print(f"RF MAE: {mae_RF}, RF RMSE: {rmse_RF}, RF Feature Count:␣
 ↪{RF_feature_count}")
```

```
Lasso MAE: 1500.9684308209926, Lasso RMSE: 1919.3998553484412, Lasso Feature
Count: 21
RF MAE: 55679.41427729769, RF RMSE: 63445.982092762526, RF Feature Count: 17
```

[52]:
```
import pickle

best_model = GSCV_rf.best_estimator_

with open('BestModel_REG_RF_SeaBorn.pkl', 'wb') as f:
```

```
    pickle.dump(best_model, f)

print("Model terbaik berhasil disimpan ke 'BestModel_REG_RF_SeaBorn.pkl")
```

```
Model terbaik berhasil disimpan ke 'BestModel_REG_RF_SeaBorn.pkl
```