Code

Notebook_Regresi_B_Ridge_VS_SupportVektor_Bernard

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

df_dataset = pd.read_csv(r'D:\1 tugas kuliah\sem5\ML\project uts\Dataset UTS_Gasal 2425.csv')
df_dataset
```

[56] Python

	squaremeters	numberofrooms	hasyard	haspool	floors	citycode	citypartrange	numprevowners	made	isnewbuilt	hasstormprotector	basement	attic	garage	hasstorageroom	hasguestr
0	75523	3	no	yes	63	9373	3	8	2005	old	yes	4313	9005	956	no	
1	55712	58	no	yes	19	34457	6	8	2021	old	no	2937	8852	135	yes	
2	86929	100	yes	no	11	98155	3	4	2003	new	no	6326	4748	654	no	
3	51522	3	no	no	61	9047	8	3	2012	new	yes	632	5792	807	yes	
4	96470	74	yes	no	21	92029	4	2	2011	new	yes	5414	1172	716	yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	341	83	no	no	8	1960	4	4	1993	new	yes	2366	4016	229	yes	
9996	21514	5	no	yes	11	91373	1	1	1999	old	no	2584	5266	787	no	
9997	1726	89	no	yes	5	73133	7	6	2009	old	yes	9311	1698	218	no	
9998	44403	29	yes	yes	12	34606	9	4	1990	old	yes	9061	1742	230	no	
9999	1440	84	no	no	49	18412	6	10	1994	new	no	8485	2024	278	yes	

10000 rows × 18 columns

```python
df_dataset2 = df_dataset.drop(['category'], axis=1)
df_dataset2
```

[57] Python

	squaremeters	numberofrooms	hasyard	haspool	floors	citycode	citypartrange	numprevowners	made	isnewbuilt	hasstormprotector	basement	attic	garage	hasstorageroom	hasguestr
0	75523	3	no	yes	63	9373	3	8	2005	old	yes	4313	9005	956	no	
1	55712	58	no	yes	19	34457	6	8	2021	old	no	2937	8852	135	yes	
2	86929	100	yes	no	11	98155	3	4	2003	new	no	6326	4748	654	no	
3	51522	3	no	no	61	9047	8	3	2012	new	yes	632	5792	807	yes	
4	96470	74	yes	no	21	92029	4	2	2011	new	yes	5414	1172	716	yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	341	83	no	no	8	1960	4	4	1993	new	yes	2366	4016	229	yes	
9996	21514	5	no	yes	11	91373	1	1	1999	old	no	2584	5266	787	no	
9997	1726	89	no	yes	5	73133	7	6	2009	old	yes	9311	1698	218	no	
9998	44403	29	yes	yes	12	34606	9	4	1990	old	yes	9061	1742	230	no	
9999	1440	84	no	no	49	18412	6	10	1994	new	no	8485	2024	278	yes	

10000 rows × 17 columns

```python
df_dataset2.info()
```

[58] Python

```
<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
```

```python
df_dataset2.describe()
```

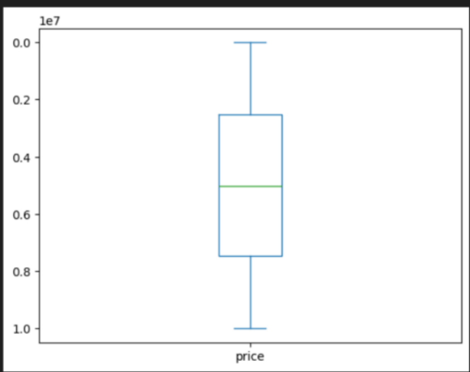
[59] Python

	squaremeters	numberofrooms	floors	citycode	citypartrange	numprevowners	made	basement	attic	garage	hasguestroom	price
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.00000	10000.00000	10000.00000	1.000000e+04
mean	49870.13120	50.358400	50.276300	50225.486100	5.510100	5.521700	2005.48850	5033.103900	5028.01060	553.12120	4.99460	4.993448e+06
std	28774.37535	28.816696	28.889171	29006.675799	2.872024	2.856667	9.30809	2876.729545	2894.33221	262.05017	3.17641	2.877424e+06
min	89.00000	1.000000	1.000000	3.000000	1.000000	1.000000	1990.00000	0.000000	1.00000	100.00000	0.00000	1.031350e+04
25%	25098.50000	25.000000	25.000000	24693.750000	3.000000	3.000000	1997.00000	2559.750000	2512.00000	327.75000	2.00000	2.516402e+06
50%	50105.50000	50.000000	50.000000	50693.000000	5.000000	5.000000	2005.50000	5092.500000	5045.00000	554.00000	5.00000	5.016180e+06
75%	74609.75000	75.000000	76.000000	75683.250000	8.000000	8.000000	2014.00000	7511.250000	7540.50000	777.25000	8.00000	7.469092e+06
max	99999.00000	100.000000	100.000000	99953.000000	10.000000	10.000000	2021.00000	10000.000000	10000.00000	1000.00000	10.00000	1.000677e+07

```python
print("Data null \n", df_dataset2.isnull().sum())
print("\nData kosong \n", df_dataset2.empty)
print("\nData NaN \n", df_dataset2.isna().sum())
```

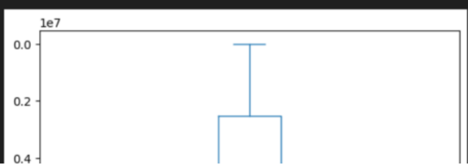
```
Data null
 squaremeters         0
numberofrooms        0
hasyard              0
haspool              0
floors               0
citycode             0
citypartrange        0
numprevowners        0
made                 0
isnewbuilt           0
hasstormprotector    0
basement             0
attic    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

Data kosong
 False

Data NaN
 squaremeters         0
...
hasstorageroom       0
hasguestroom         0
price                0
```

```python
import matplotlib.pyplot as plt

df_dataset2.price.plot(kind='box')
plt.gca().invert_yaxis()
plt.show()
```




```python
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_dataset_clean = remove_outlier(df_dataset2)
print("jumlah baris Dataframe sebelum dibuang outliner",df_dataset2.shape[0])
print("jumlah baris Dataframe setelah dibuang outliner",df_dataset_clean.shape[0])
df_dataset_clean.price.plot(kind='box',vert=True)

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

```
jumlah baris Dataframe sebelum dibuang outliner 10000
jumlah baris Dataframe setelah dibuang outliner 10000
```



```python
from sklearn.model_selection import train_test_split
X_regress = df_dataset_clean.drop('price',axis=1)
Y_regress = df_dataset_clean.price

x_train, x_test, y_train, y_test = train_test_split(X_regress, Y_regress, \
                                                    test_size = 0.25,
                                                    random_state=20)
```

```python
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'
)
```

```python
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
df_test_enc
```

	onehotencoder__hasyard_no	onehotencoder__hasyard_yes	onehotencoder__haspool_no	onehotencoder__haspool_yes	onehotencoder__isnewbuilt_new	onehotencoder__isnewbuilt_old	onehotencode
0	0.0	1.0	0.0	1.0	0.0	1.0	
1	1.0	0.0	1.0	0.0	0.0	1.0	
2	1.0	0.0	0.0	1.0	1.0	0.0	
3	0.0	1.0	1.0	0.0	0.0	1.0	
4	1.0	0.0	0.0	1.0	1.0	0.0	
...	...	...	...	...	...	...	
2495	1.0	0.0	0.0	1.0	1.0	0.0	
2496	0.0	1.0	0.0	1.0	1.0	0.0	
2497	0.0	1.0	0.0	1.0	0.0	1.0	
2498	1.0	0.0	1.0	0.0	0.0	1.0	
2499	1.0	0.0	1.0	0.0	0.0	1.0	

2500 rows × 21 columns

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

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

```python
param_grid_Ridge = {
    'reg__alpha':[0.01,0.1,1,10,100,1000],
    '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/biar:{}".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 Error: {}".format(np.sqrt(mse_Ridge)))
```

```
best Model:Pipeline(steps=[('scale', StandardScaler()),
                ('feature_selection',
                 SelectKBest(k=19,
                             score_func=<function f_regression at 0x000002797E4A7E20>)),
                ('reg', Ridge(alpha=0.01))])
Ridge best parameters:{'feature_selection__k': 19, 'reg__alpha': 0.01}
```

```python
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
```

	price	Ridge prediction	Selisih_IPK_RR
9957	3055190.2	3.051666e+06	-3524.193541
1687	7719488.8	7.719265e+06	-224.272822
2116	9265884.1	9.266680e+06	796.068579
231	6126371.1	6.126390e+06	18.889707
2780	9720521.5	9.722524e+06	2002.449403
...	...	...	...
8514	1349726.5	1.347881e+06	-1845.591504
5190	7578283.3	7.577291e+06	-992.486931
6766	2437565.3	2.435221e+06	-2344.283798
347	9086961.1	9.086326e+06	-634.693933
6961	3933732.6	3.934429e+06	696.320979

2500 rows × 3 columns

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

	price	Ridge prediction	Selisih_IPK_RR

```python
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 parameters:{}".format(GSCV_SVR.best_params_))

print("Koefisien/bobot:{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/biar:{}".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(k=1,
                             score_func=<function f_regression at 0x000002797E4A7E20>)),
                ('reg', SVR(C=100, kernel='linear'))])
SVR best parameters:{'feature_selection__k': 1, 'reg__C': 100, 'reg__epsilon': 0.1}
Koefisien/bobot:[[648847.92251728]]
Intercept/biar:[5021559.59331346]
SVR mean Squared Error (MSE): 4920673080502.885
SVR Mean Absolute Error (MAE): 1909730.1031783433
SVR Root Mean Squared Error: 2218259.020155871
```

```python
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()
```

	price	SVR prediction	Selisih_price_SVR
9957	3055190.2	4.582547e+06	1.527356e+06
1687	7719488.8	5.635386e+06	-2.084103e+06
2116	9265884.1	5.982655e+06	-3.283229e+06
231	6126371.1	5.276162e+06	-8.502087e+05
2780	9720521.5	6.085625e+06	-3.634897e+06

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

	price	SVR prediction	Selisih_price_SVR

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

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


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

df_results.head()
```

[96] Python

	price	Ridge Prediction	Selisih_price_RR	SVR Prediction	Selisih_price_SVR
9957	3055190.2	3.051666e+06	3524.193541	4.582547e+06	-1.527356e+06
1687	7719488.8	7.719265e+06	224.272822	5.635386e+06	2.084103e+06
2116	9265884.1	9.266680e+06	-796.068579	5.982655e+06	3.283229e+06
231	6126371.1	6.126390e+06	-18.889707	5.276162e+06	8.502087e+05
2780	9720521.5	9.722524e+06	-2002.449403	6.085625e+06	3.634897e+06

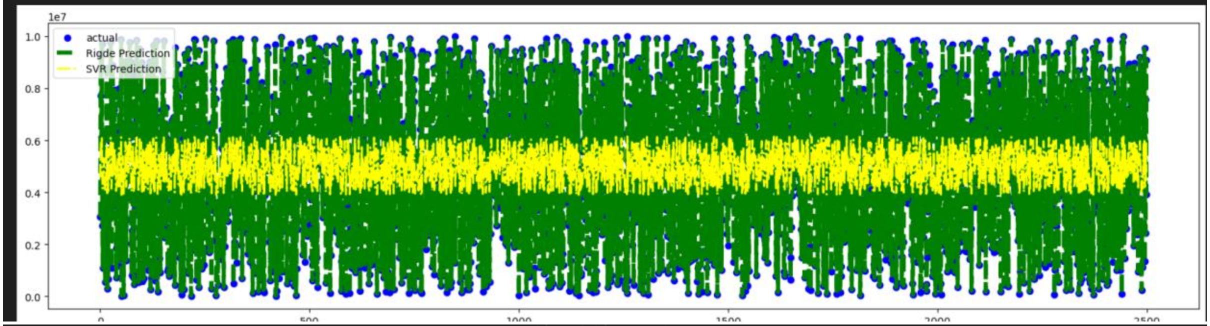
```python
df_results.describe()
```

[97] Python

	price	Ridge Prediction	Selisih_price_RR	SVR Prediction	Selisih_price_SVR
count	2.500000e+03	2.500000e+03	2500.000000	2.500000e+03	2.500000e+03
mean	4.984921e+06	4.984926e+06	-5.190951	5.019020e+06	-3.409985e+04
std	2.862950e+06	2.862962e+06	1881.424864	6.445100e+05	2.218441e+06
min	1.548800e+04	1.666434e+04	-6265.641829	3.901132e+06	-3.885644e+06
25%	2.531105e+06	2.533338e+06	-1178.857910	4.466953e+06	-1.935848e+06
50%	4.986667e+06	4.987591e+06	-2.898140	5.019965e+06	-3.354598e+04

```python
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['Ridge Prediction'], label="Rigde 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
```

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




```python
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}")
```

[100] Python

```
Ridge MAE: 1466.0025565480937, Ridge RMSE: 1881.0557034890273, Ridge Feature Count: 19
SVR MAE: 1909730.1031783433, SVR RMSE: 2218259.020155871, SVR Feature Count: 1
```

```python
import pickle

best_model = GSCV_SVR.best_estimator_

with open('BestModel_REG_SVR_SciPy.pkl', 'wb') as f:
    pickle.dump(best_model, f)

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