Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, SelectPercentile,
f_regression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error



df_p = pd.read_csv("car_price_prediction.csv")

print(df_p.columns.tolist())
columns = df_p.columns.tolist()
df_t = df_p
df_t.head()
```

```
['ID', 'Price', 'Levy', 'Manufacturer', 'Model', 'Prod. year',
'Category', 'Leather interior', 'Fuel type', 'Engine volume',
'Mileage', 'Cylinders', 'Gear box type', 'Drive wheels', 'Doors',
'Wheel', 'Color', 'Airbags']
```

```
         ID  Price  Levy Manufacturer    Model  Prod. year   Category  \
0  45654403  13328  1399        LEXUS   RX 450        2010       Jeep

1  44731507  16621  1018    CHEVROLET  Equinox        2011       Jeep

2  45774419   8467     -        HONDA      FIT        2006  Hatchback

3  45769185   3607   862         FORD   Escape        2011       Jeep

4  45809263  11726   446        HONDA      FIT        2014  Hatchback


  Leather interior Fuel type Engine volume    Mileage  Cylinders  \
0              Yes    Hybrid           3.5  186005 km        6.0
1               No    Petrol             3  192000 km        6.0
2               No    Petrol           1.3  200000 km        4.0
3              Yes    Hybrid           2.5  168966 km        4.0
4              Yes    Petrol           1.3   91901 km        4.0
```

```
  Gear box type Drive wheels   Doors             Wheel   Color
Airbags
0     Automatic          4x4  04-May        Left wheel  Silver
12
1     Tiptronic          4x4  04-May        Left wheel   Black
8
2      Variator        Front  04-May  Right-hand drive   Black
2
3     Automatic          4x4  04-May        Left wheel   White
0
4     Automatic        Front  04-May        Left wheel  Silver
4
```

Data Checking

```
print("Jumlah baris, kolom:", df_t.shape)
print("\nTipe data:")

df_t.describe()
df_t.info()

Jumlah baris, kolom: (19237, 18)

Tipe data:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19237 entries, 0 to 19236
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                19237 non-null  int64
 1   Price             19237 non-null  int64
 2   Levy              19237 non-null  object
 3   Manufacturer      19237 non-null  object
 4   Model             19237 non-null  object
 5   Prod. year        19237 non-null  int64
 6   Category          19237 non-null  object
 7   Leather interior  19237 non-null  object
 8   Fuel type         19237 non-null  object
 9   Engine volume     19237 non-null  object
 10  Mileage           19237 non-null  object
 11  Cylinders         19237 non-null  float64
 12  Gear box type     19237 non-null  object
 13  Drive wheels      19237 non-null  object
 14  Doors             19237 non-null  object
 15  Wheel             19237 non-null  object
 16  Color             19237 non-null  object
 17  Airbags           19237 non-null  int64
dtypes: float64(1), int64(4), object(13)
memory usage: 2.6+ MB
```

```
print("Jumlah nilai kosong per kolom:\n", df_t.isnull().sum())
print("Jumlah data duplikat:", df_t.duplicated().sum())
print("Jumlah data Empty String: ", (df_t == '').sum().sum())
```

```
Jumlah nilai kosong per kolom:
 ID                   0
Price                0
Levy                 0
Manufacturer         0
Model                0
Prod. year           0
Category             0
Leather interior     0
Fuel type            0
Engine volume        0
Mileage              0
Cylinders            0
Gear box type        0
Drive wheels         0
Doors                0
Wheel                0
Color                0
Airbags              0
dtype: int64
Jumlah data duplikat: 313
Jumlah data Empty String:  0
```

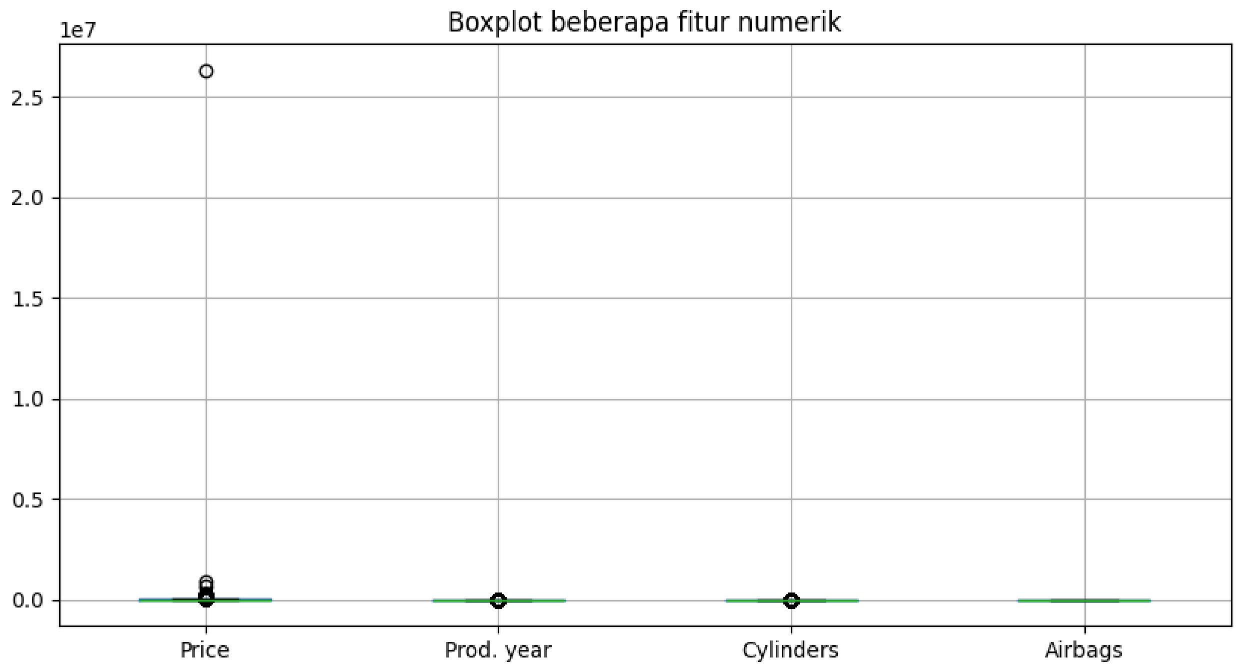
```
df_t['Color'].unique()
```

```
array(['Silver', 'Black', 'White', 'Grey', 'Blue', 'Green', 'Red',
       'Sky blue', 'Orange', 'Yellow', 'Brown', 'Golden', 'Beige',
       'Carnelian red', 'Purple', 'Pink'], dtype=object)
```

Outlier check

Identifikasi outlier dilakukan pada seluruh fitur numerik menggunakan visualisasi boxplot, yang menunjukkan adanya nilai ekstrem pada fitur seperti Price. Boxplot dipilih karena mampu menampilkan sebaran data, median, serta nilai ekstrem secara visual.

```
numeric_cols = [
    'Price', 'Levy', 'Prod. year', 'Engine volume',
    'Mileage', 'Cylinders', 'Doors', 'Airbags'
]
numeric_cols = [c for c in numeric_cols
                if c in df_t.columns]

plt.figure(figsize=(10, 5))
df_t[numeric_cols].boxplot()
plt.title("Boxplot beberapa fitur numerik")
plt.show()
```



Data Preparation

Penangan Null dan Duplikat

```
df_t['Doors'] = df_t['Doors'].fillna(df_t['Doors'].mode()[0])
df_t['Color'] = df_t['Color'].fillna('Unknown')

df_t = df_t.drop_duplicates()

print(df_t.isnull().sum())
print("Duplikat:", df_t.duplicated().sum())
```

```
ID                  0
Price               0
Levy                0
Manufacturer        0
Model               0
Prod. year          0
Category            0
Leather interior    0
Fuel type           0
Engine volume       0
Mileage             0
Cylinders           0
Gear box type       0
Drive wheels        0
Doors               0
Wheel               0
Color               0
```

```
Airbags              0
dtype: int64
Duplikat: 0
```

Data Outlier Handling

Penanganan outlier dilakukan untuk mengurangi pengaruh nilai ekstrem pada fitur numerik yang berpotensi menyebabkan model regresi menjadi bias dan meningkatkan error prediksi, khususnya pada algoritma regresi linear seperti Lasso dan Ridge.

Metode Penanganan: Interquartile Range (IQR)

Penanganan outlier dilakukan menggunakan metode Interquartile Range (IQR) dengan langkah sebagai berikut:

Seluruh kolom numerik diidentifikasi secara otomatis menggunakan:

df.select_dtypes(include=np.number)

Untuk setiap fitur numerik dihitung:

Kuartil bawah (Q1, 25%)

Kuartil atas (Q3, 75%)

IQR = Q3 − Q1

Batas outlier ditentukan sebagai:

Lower bound = Q1 − 1.5 × IQR

Upper bound = Q3 + 1.5 × IQR

Baris data yang memiliki setidaknya satu fitur numerik berada di luar rentang tersebut dihapus dari dataset.

Pendekatan ini memastikan bahwa data yang dipertahankan berada dalam rentang distribusi yang wajar untuk setiap fitur numerik.

```python
def handle_outlier_iqr(df, numeric_cols):
    df = df.copy()
    numeric_cols = df.select_dtypes(include=np.number).columns

    Q1 = df[numeric_cols].quantile(0.25)
    Q3 = df[numeric_cols].quantile(0.75)
    IQR = Q3 - Q1

    df = df[~((df[numeric_cols] < (Q1 - 1.5 * IQR)) |
            (df[numeric_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]

    numeric_cols = [c for c in numeric_cols if c in df.columns]
    return df
```

```
df_t = handle_outlier_iqr(df_t, numeric_cols)
```

Setelah proses outlier removal, dilakukan kembali visualisasi boxplot untuk memastikan bahwa:

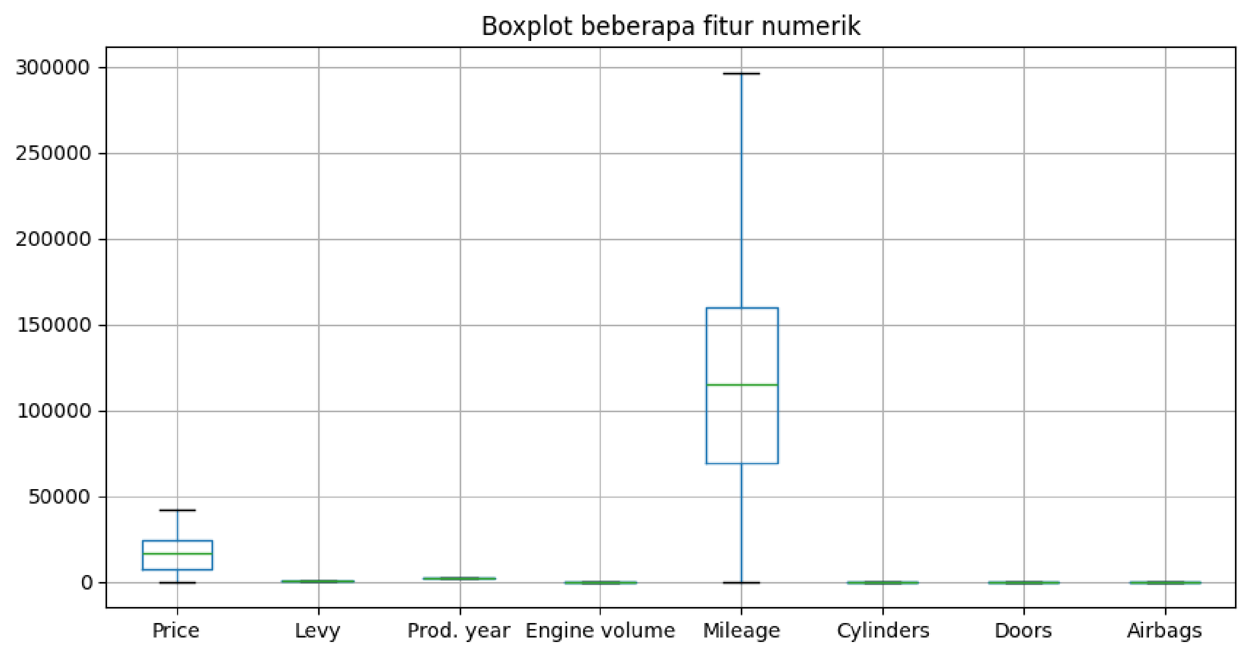
Sebaran data menjadi lebih terpusat

Nilai ekstrem berkurang secara signifikan

Distribusi fitur numerik menjadi lebih stabil untuk pemodelan

Hasil boxplot setelah penanganan menunjukkan distribusi yang lebih seimbang dibandingkan sebelum outlier removal.

```
plt.figure(figsize=(10, 5))
df_t[numeric_cols].boxplot()
plt.title("Boxplot beberapa fitur numerik")
plt.show()
```



Encoding

Data Encoding dilakukan dengan mengonversi seluruh fitur kategorik menjadi bentuk numerik agar dapat diproses oleh model regresi. Sebelum encoding, dilakukan pembersihan dan transformasi data seperti konversi string numerik, imputasi nilai hilang menggunakan median atau modus, serta reduksi kardinalitas pada fitur dengan kategori sangat banyak (misalnya Manufacturer dan Model) dengan mempertahankan kategori terbanyak dan mengelompokkan sisanya sebagai Other. Fitur kategorik multi-kelas kemudian dienkode menggunakan LabelEncoder, sedangkan fitur biner dimapping secara langsung ke nilai 0 dan 1. Proses ini menghasilkan dataset yang seluruh fiturnya bersifat numerik dan siap digunakan pada tahap pemodelan serta pipeline regresi.

```
from sklearn.preprocessing import LabelEncoder

df_t = df_t.drop(columns=['ID'])

df_t['Levy'] = df_t['Levy'].replace('-', np.nan).astype(float)
df_t['Levy'] = df_t['Levy'].fillna(df_t['Levy'].median())

df_t['Mileage'] = df_t['Mileage'].str.replace(' km', '',
regex=False).astype(int)

df_t['Engine volume'] = (
    df_t['Engine volume']
    .str.replace(' Turbo', '', regex=False)
    .astype(float)
)

df_t['Doors'] = df_t['Doors'].map({
    '02-Mar': 2,
    '04-May': 4
})

df_t["Doors"] = df_t["Doors"].fillna(df_t["Doors"].mode()[0])

df_t['Leather interior'] = df_t['Leather interior'].map({'Yes': 1,
'No': 0})

top_manufacturer =
df_t['Manufacturer'].value_counts().nlargest(10).index
df_t['Manufacturer'] = df_t['Manufacturer'].where(
    df_t['Manufacturer'].isin(top_manufacturer),
    'Other'
)

top_model = df_t['Model'].value_counts().nlargest(20).index
df_t['Model'] = df_t['Model'].where(
    df_t['Model'].isin(top_model),
    'Other'
)

label_cols = [
    'Manufacturer', 'Model', 'Category',
    'Fuel type', 'Gear box type',
    'Drive wheels', 'Wheel'
]

le = LabelEncoder()
for col in label_cols:
    df_t[col] = le.fit_transform(df_t[col])

color_map = {
```

```
    'Black': 0,
    'White': 1,
    'Silver': 2,
    'Grey': 3,
    'Red': 4,
    'Blue': 5,
    'Green': 6,
    'Yellow': 7,
    'Sky blue': 8,
    'Orange': 9,
    'Brown': 10,
    'Golden': 11,
    'Beige': 12,
    'Carnelian red': 13,
    'Purple': 14,
    'Pink': 15
}

df_t['Color'] = df_t['Color'].map(color_map)
```

```
print(df_t.shape)
print(df_t.dtypes)
df_t.head()
```

```
(11139, 17)
Price                 int64
Levy                float64
Manufacturer          int64
Model                 int64
Prod. year            int64
Category              int64
Leather interior      int64
Fuel type             int64
Engine volume       float64
Mileage               int64
Cylinders           float64
Gear box type         int64
Drive wheels          int64
Doors               float64
Wheel                 int64
Color                 int64
Airbags               int64
dtype: object
```

```
   Price   Levy  Manufacturer  Model  Prod. year  Category  Leather
interior  \
```

```
2   8467  749.0             2      7         2006         3
0
3   3607  862.0             1     14         2011         4
1
4  11726  446.0             2      7         2014         3
1
5  39493  891.0             3     17         2016         4
1
6   1803  761.0             9     15         2010         3
1

   Fuel type  Engine volume  Mileage  Cylinders  Gear box type  Drive
wheels  \
2          4            1.3   200000        4.0              3
1
3          2            2.5   168966        4.0              0
0
4          4            1.3    91901        4.0              0
1
5          1            2.0   160931        4.0              0
1
6          2            1.8   258909        4.0              0
1

   Doors  Wheel  Color  Airbags
2    4.0      1      0        2
3    4.0      0      1        0
4    4.0      0      2        4
5    4.0      0      1        4
6    4.0      0      1       12
```

```
print("Jumlah nilai kosong per kolom:\n", df_t.isnull().sum())
print("Jumlah data duplikat:", df_t.duplicated().sum())
```

```
Jumlah nilai kosong per kolom:
 Price               0
Levy                0
Manufacturer        0
Model               0
Prod. year          0
Category            0
Leather interior    0
Fuel type           0
Engine volume       0
Mileage             0
Cylinders           0
Gear box type       0
Drive wheels        0
Doors               0
Wheel               0
```

```
Color                  0
Airbags                0
dtype: int64
Jumlah data duplikat: 1810
```

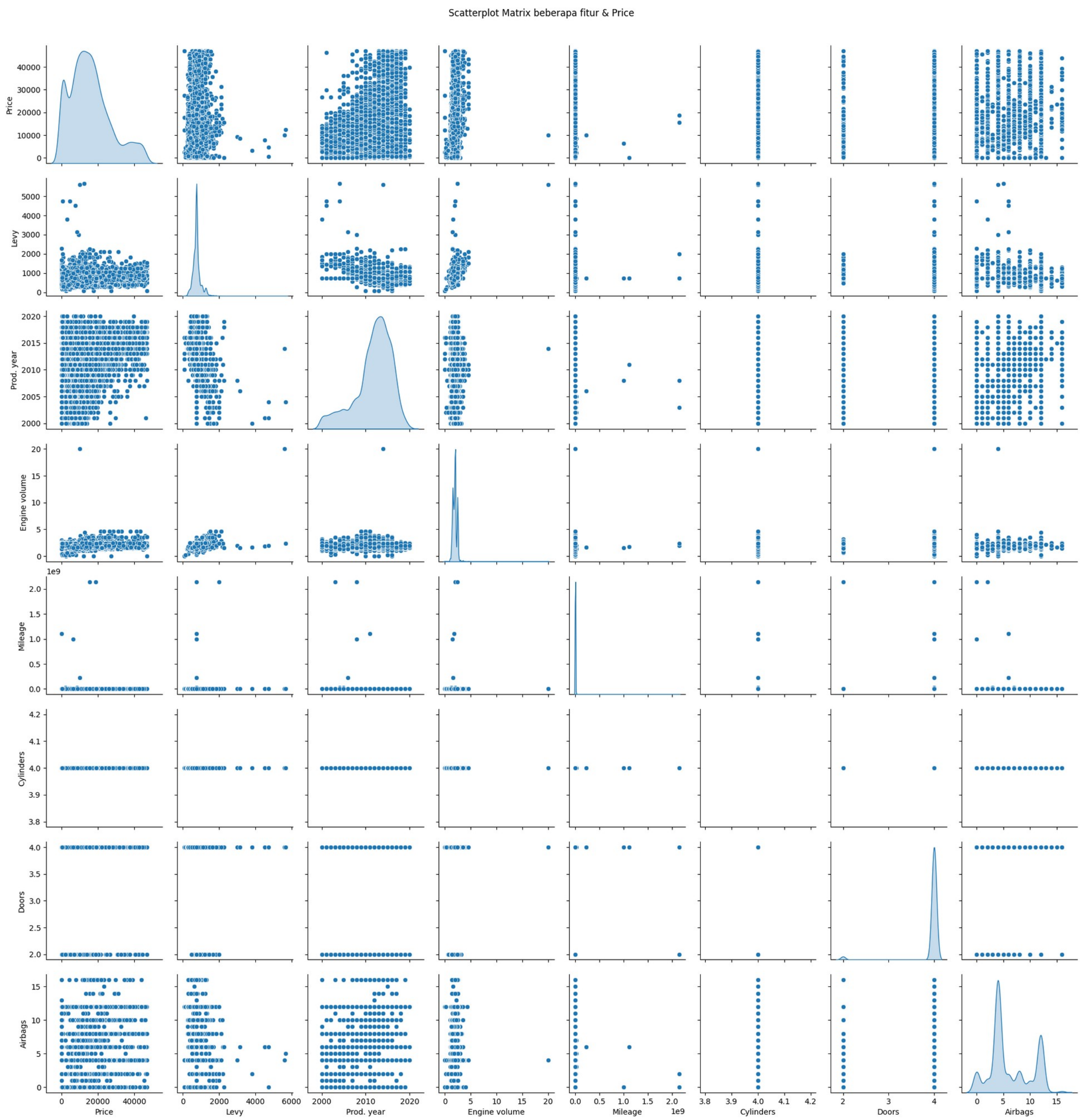
Drop Data Duplicate

```python
df_t = df_t.drop_duplicates()

print("Duplikat:", df_t.duplicated().sum())
print("Jumlah nilai kosong per kolom:\n", df_t.isnull().sum())
print("Jumlah data duplikat:", df_t.duplicated().sum())
df = df_t
```

```
Duplikat: 0
Jumlah nilai kosong per kolom:
 Price                  0
Levy                   0
Manufacturer           0
Model                  0
Prod. year             0
Category               0
Leather interior       0
Fuel type              0
Engine volume          0
Mileage                0
Cylinders              0
Gear box type          0
Drive wheels           0
Doors                  0
Wheel                  0
Color                  0
Airbags                0
dtype: int64
Jumlah data duplikat: 0
```

```python
from sklearn.model_selection import train_test_split

X = df.drop(columns=['Price'])
y = df['Price']

rs = 86

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=rs
)
```

```
sns.pairplot(df[numeric_cols], diag_kind="kde")
plt.suptitle("Scatterplot Matrix beberapa fitur & Price", y=1.02)
plt.show()
```



Pipeline Lasso

Pipeline Lasso dibangun untuk memastikan seluruh tahapan praproses dan pemodelan dilakukan secara terintegrasi dan konsisten selama proses validasi silang. Pipeline terdiri dari tahap penskalaan fitur menggunakan StandardScaler atau MinMaxScaler, seleksi fitur menggunakan SelectKBest atau SelectPercentile dengan fungsi skor f_regression, serta pemodelan menggunakan algoritma Lasso Regression. Untuk memperoleh kombinasi parameter terbaik, dilakukan GridSearchCV dengan 5-fold cross validation yang mengeksplorasi variasi metode penskalaan, jumlah fitur terpilih, dan nilai regularisasi (alpha). Evaluasi performa model dilakukan menggunakan metrik R², MSE, MAE, dan RMSE, sementara fitur-fitur yang paling relevan diidentifikasi berdasarkan hasil seleksi fitur dari estimator terbaik.

```
lasso = Pipeline([
    ('scaler', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)),
    ('model', Lasso(max_iter=10000))
])


param_grid_lasso = [

    {
        'scaler': [StandardScaler(), MinMaxScaler()],
        'feature_selection': [SelectKBest(score_func=f_regression)],
        'feature_selection__k': [5, 10, 15],
        'model__alpha': [0.01, 0.1, 1, 10]
    },

    {
        'scaler': [StandardScaler(), MinMaxScaler()],
        'feature_selection':
[SelectPercentile(score_func=f_regression)],
        'feature_selection__percentile': [20, 40, 60],
        'model__alpha': [0.01, 0.1, 1, 10]
    }
]

from sklearn.model_selection import GridSearchCV

grid_lasso = GridSearchCV(
    lasso,
    param_grid_lasso,
    cv=5,
    scoring='r2',
    n_jobs=-1
)


grid_lasso.fit(X_train, y_train)

print("Best score:", grid_lasso.best_score_)
print("Best params:", grid_lasso.best_params_)

Best score: 0.2840245262375081
Best params: {'feature_selection': SelectKBest(score_func=<function
```

```
f_regression at 0x000002C470C9CA60>), 'feature_selection__k': 15,
'model__alpha': 1, 'scaler': MinMaxScaler()}
```

```
best_selector =
grid_lasso.best_estimator_.named_steps['feature_selection']
selected_features = X_train.columns[best_selector.get_support()]

print("Selected features:")
print(selected_features)
```

```
Selected features:
Index(['Levy', 'Manufacturer', 'Model', 'Prod. year', 'Category',
       'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Gear box type', 'Drive wheels', 'Doors', 'Wheel', 'Color',
'Airbags'],
      dtype='object')
```

```
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
import numpy as np
import pandas as pd

yl_pred = grid_lasso.best_estimator_.predict(X_test) #y_prednya Lasso

error_matrix = pd.DataFrame({
    'Metric': ['R2', 'MSE', 'MAE', 'RMSE'],
    'Value': [
        r2_score(y_test, yl_pred),
        mean_squared_error(y_test, yl_pred),
        mean_absolute_error(y_test, yl_pred),
        np.sqrt(mean_squared_error(y_test, yl_pred))
    ]
})
```

Pipeline Ridge

Pipeline Ridge Regression dibangun dengan pendekatan yang sama seperti Lasso untuk memastikan proses praproses, seleksi fitur, dan pelatihan model dilakukan secara terintegrasi. Pipeline terdiri dari tahap penskalaan fitur menggunakan StandardScaler atau MinMaxScaler, seleksi fitur menggunakan SelectKBest atau SelectPercentile dengan fungsi skor f_regression, serta pemodelan menggunakan Ridge Regression. Optimasi hiperparameter dilakukan menggunakan GridSearchCV dengan 5-fold cross validation untuk mencari kombinasi terbaik dari metode penskalaan, jumlah fitur terpilih, dan nilai regularisasi (alpha). Kinerja model dievaluasi menggunakan metrik R^2, MSE, MAE, dan RMSE, sehingga performa Ridge dapat dibandingkan secara langsung dengan Lasso pada kondisi data dan pipeline yang sama.

```
ridge = Pipeline([
    ('scaler', StandardScaler()),
```

```
    ('feature_selection', SelectKBest(score_func=f_regression)),
    ('model', Ridge())
])


param_grid_ridge = [
    {
        'scaler': [StandardScaler(), MinMaxScaler()],
        'feature_selection': [SelectKBest(score_func=f_regression)],
        'feature_selection__k': [5, 10, 15],
        'model__alpha': [0.1, 1, 10, 100]
    },
    {
        'scaler': [StandardScaler(), MinMaxScaler()],
        'feature_selection':
[SelectPercentile(score_func=f_regression)],
        'feature_selection__percentile': [20, 40, 60],
        'model__alpha': [0.1, 1, 10, 100]
    }
]


grid_ridge = GridSearchCV(
    ridge,
    param_grid_ridge,
    cv=5,
    scoring='r2',
    n_jobs=-1
)

grid_ridge.fit(X_train, y_train)

print("Best Ridge R2:", grid_ridge.best_score_)
print("Best Ridge Params:", grid_ridge.best_params_)
```

```
Best Ridge R2: 0.28362375255746
Best Ridge Params: {'feature_selection':
SelectKBest(score_func=<function f_regression at 0x000002C470C9CA60>),
'feature_selection__k': 15, 'model__alpha': 100, 'scaler':
StandardScaler()}
```

Evaluasi Lasso

```
error_matrix
```

	Metric	Value
0	R2	2.943406e-01
1	MSE	8.658600e+07
2	MAE	7.330100e+03
3	RMSE	9.305160e+03

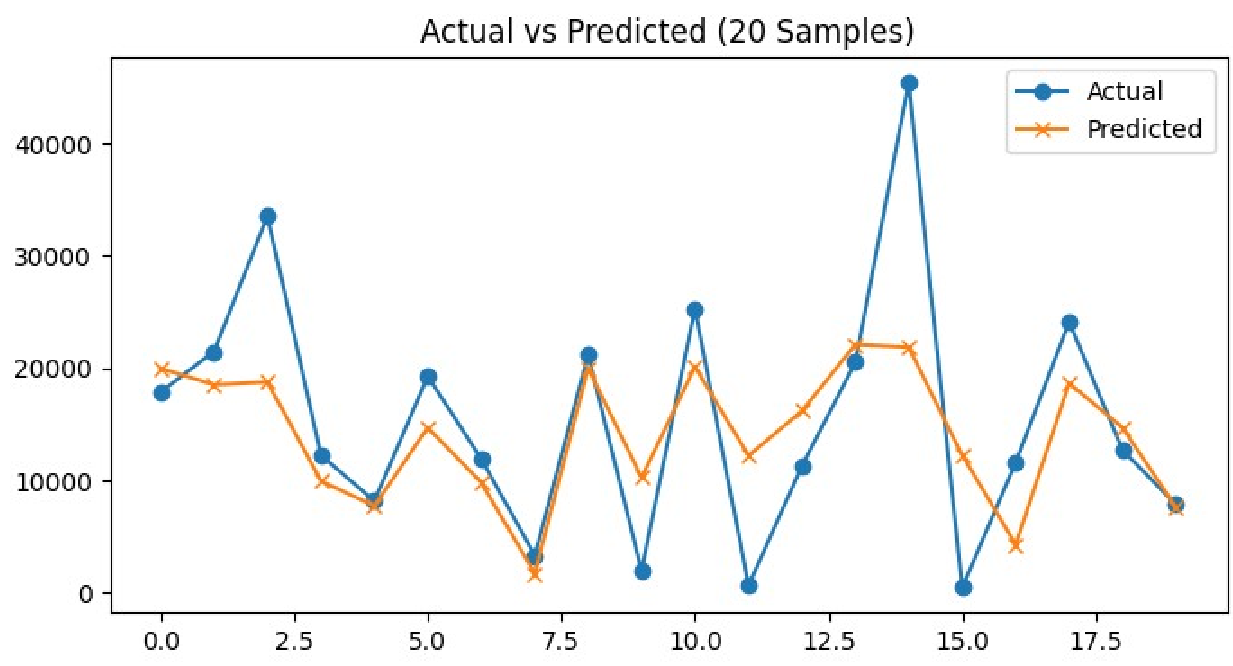
Feature Lasso

```
best_selector =
grid_lasso.best_estimator_.named_steps['feature_selection']
selected_features = X_train.columns[best_selector.get_support()]

print("Selected features:")
print(selected_features)
```

```
Selected features:
Index(['Levy', 'Manufacturer', 'Model', 'Prod. year', 'Category',
       'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Gear box type', 'Drive wheels', 'Doors', 'Wheel', 'Color',
'Airbags'],
      dtype='object')
```

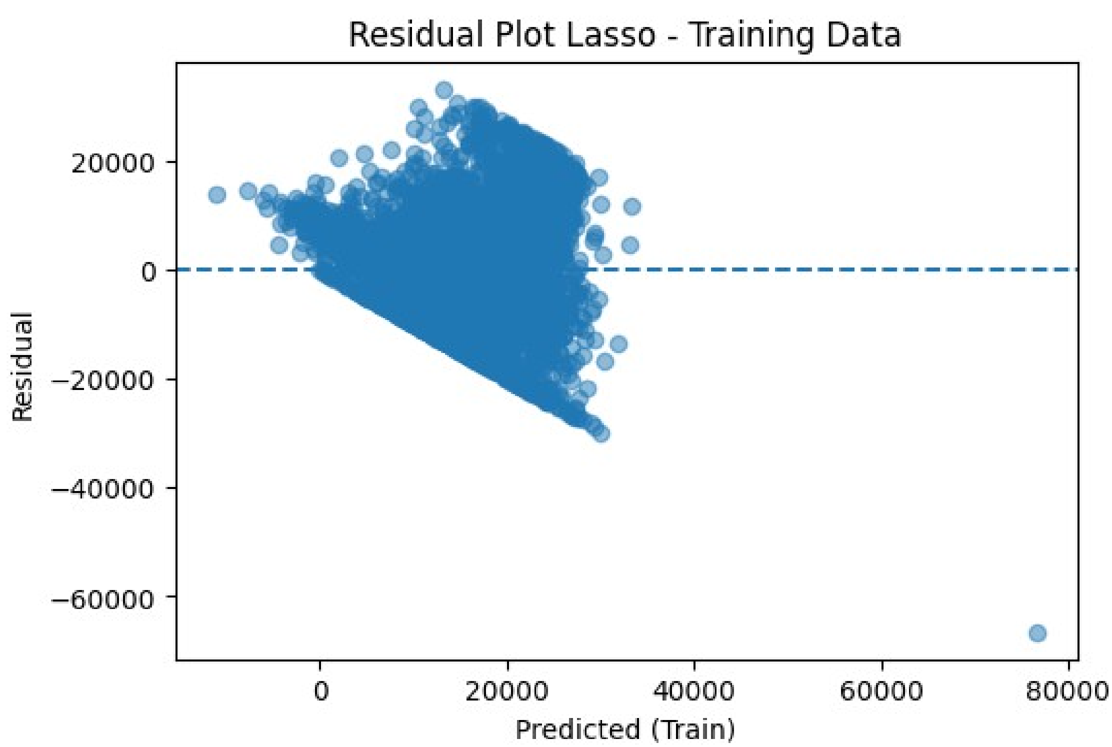
Data Visualization

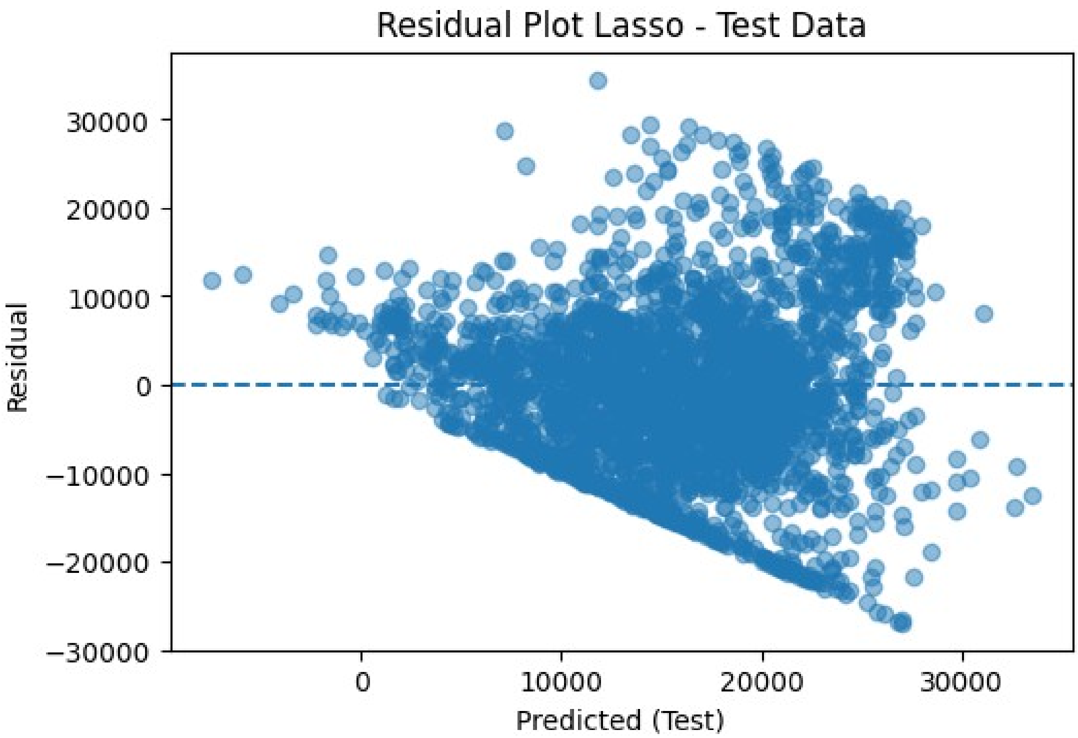
```
plt.figure(figsize=(8,4))
plt.plot(y_test.values[:20], label='Actual', marker='o')
plt.plot(yl_pred[:20], label='Predicted', marker='x')
plt.legend()
plt.title("Actual vs Predicted (20 Samples)")
plt.show()
```



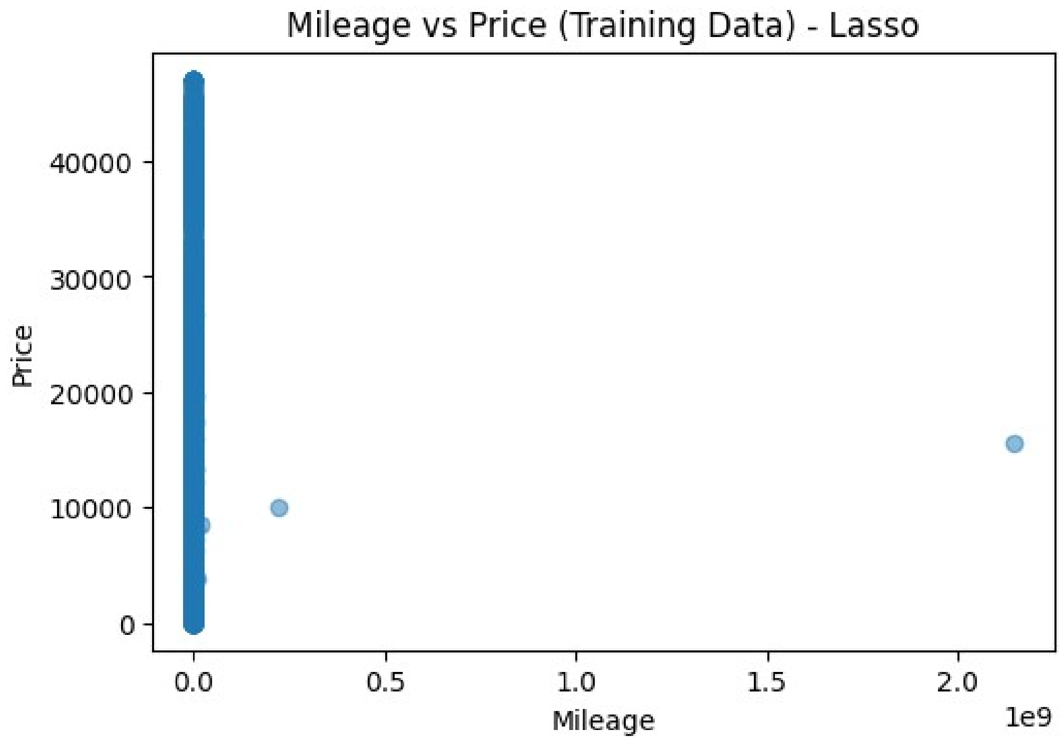

```
best_model = grid_lasso.best_estimator_


y_pred_train_lasso = best_model.predict(X_train)
```

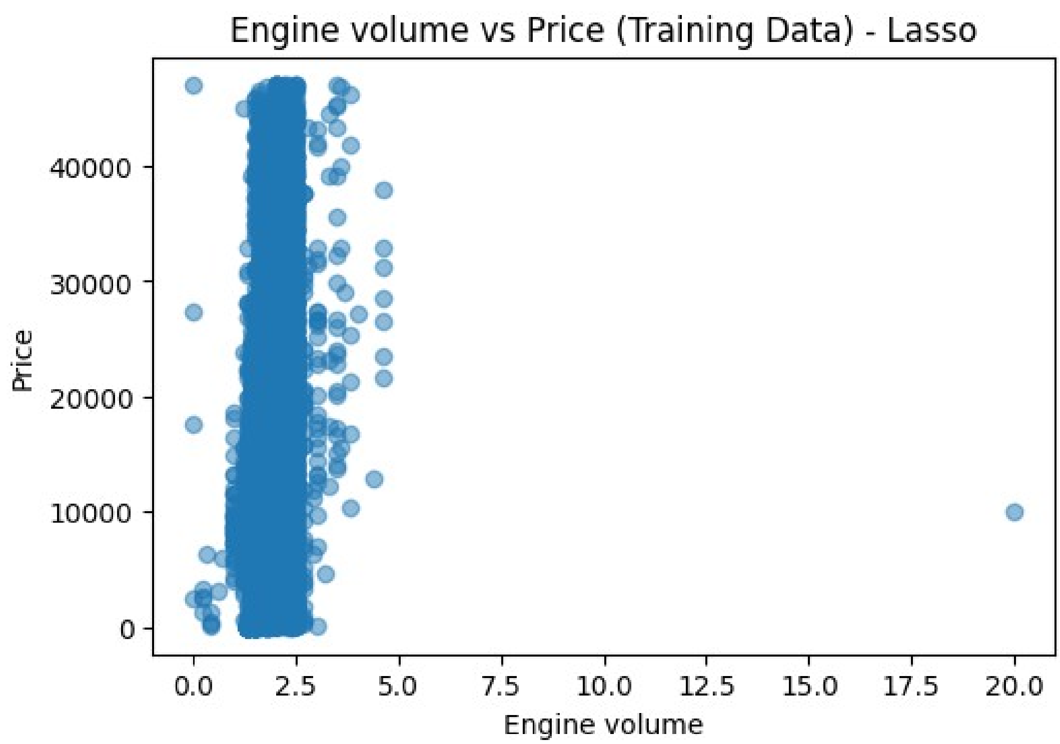
```
y_pred_test_lasso  = best_model.predict(X_test)

residual_train = y_train - y_pred_train_lasso
residual_test  = y_test - y_pred_test_lasso

plt.figure(figsize=(6,4))
plt.scatter(y_pred_train_lasso, residual_train, alpha=0.5)
plt.axhline(0, linestyle='--')
plt.xlabel("Predicted (Train)")
plt.ylabel("Residual")
plt.title("Residual Plot Lasso - Training Data")
plt.show()

plt.figure(figsize=(6,4))
plt.scatter(y_pred_test_lasso, residual_test, alpha=0.5)
plt.axhline(0, linestyle='--')
plt.xlabel("Predicted (Test)")
plt.ylabel("Residual")
plt.title("Residual Plot Lasso - Test Data")
plt.show()
```

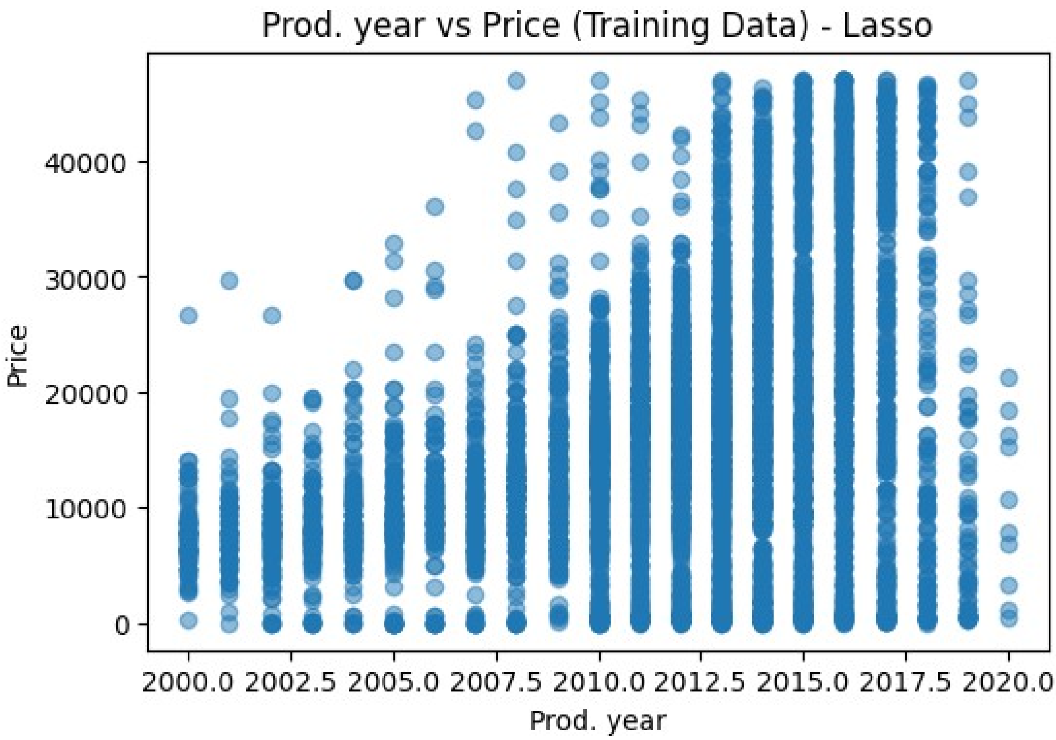




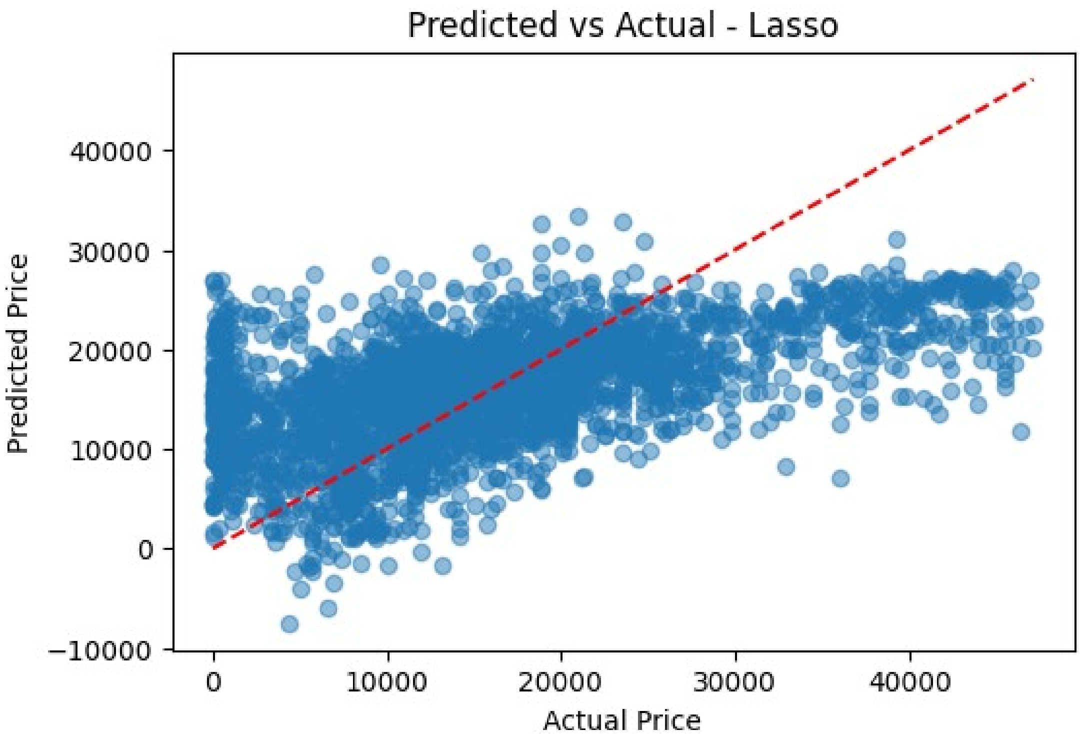

```python
features = ['Mileage', 'Engine volume', 'Prod. year']

for feature in features:
    plt.figure(figsize=(6,4))
    plt.scatter(X_train[feature], y_train, alpha=0.5)
    plt.xlabel(feature)
    plt.ylabel("Price")
    plt.title(f"{feature} vs Price (Training Data) - Lasso")
    plt.show()
```


Mileage vs Price (Training Data) - Lasso
Price
40000
30000
20000
10000
0
0.0
0.5
1.0
1.5
2.0
Mileage
1e9

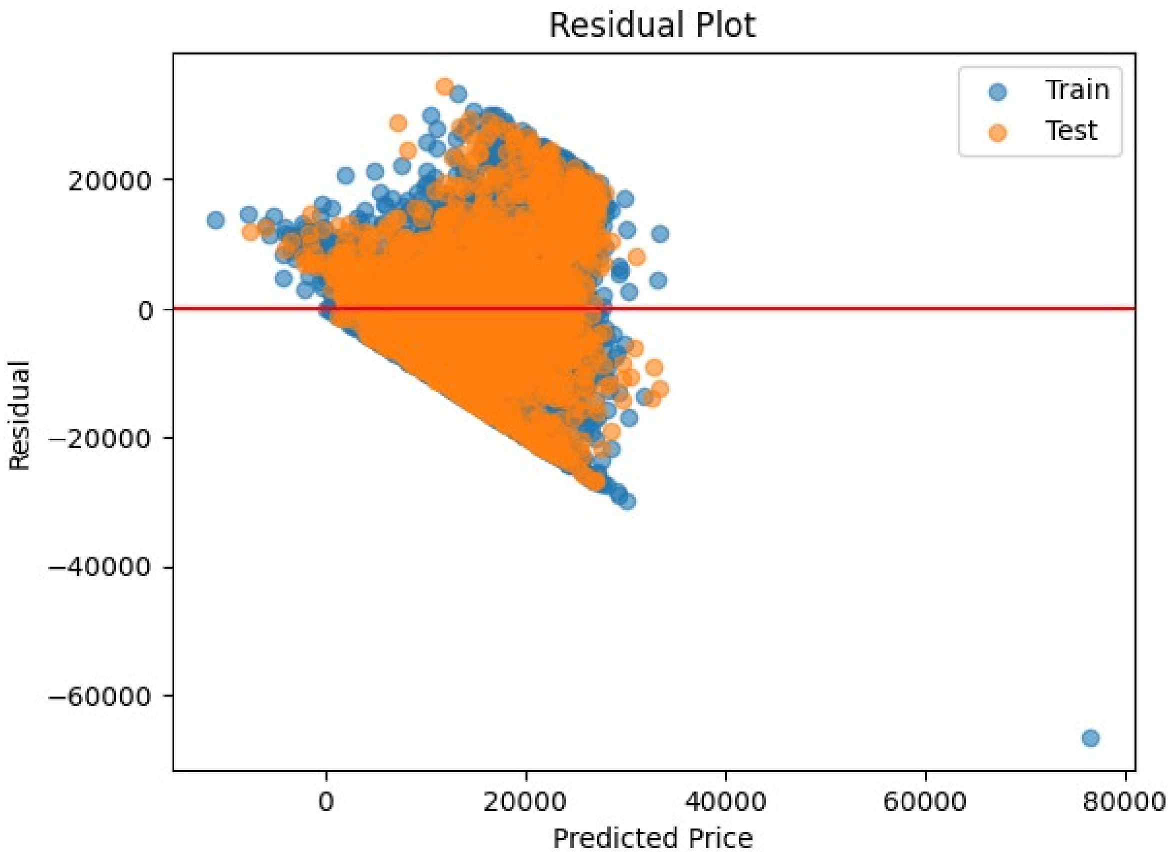

Engine volume vs Price (Training Data) - Lasso
Price
40000
30000
20000
10000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Engine volume




```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred_test_lasso, alpha=0.5)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual - Lasso")
plt.plot(
    [y_test.min(), y_test.max()],
    [y_test.min(), y_test.max()],
    color='red',
    linestyle='--'
)
plt.show()
```




```
y_train_pred = grid_lasso.predict(X_train)

plt.scatter(y_train_pred, y_train - y_train_pred, label="Train",
alpha=0.6)
plt.scatter(yl_pred, y_test - yl_pred, label="Test", alpha=0.6)
plt.axhline(0, color="red")
plt.legend()
plt.title("Residual Plot")
plt.xlabel("Predicted Price")
plt.ylabel("Residual")
plt.show()

error_matrix
```




```
  Metric         Value
0     R2  2.943406e-01
1    MSE  8.658600e+07
2    MAE  7.330100e+03
3   RMSE  9.305160e+03
```

Evaluasi Ridge

```
y_pred_ridge = grid_ridge.best_estimator_.predict(X_test)

error_matrix_ridge = pd.DataFrame({
    'Metric': ['R2', 'MSE', 'MAE', 'RMSE'],
    'Value': [
        r2_score(y_test, y_pred_ridge),
        mean_squared_error(y_test, y_pred_ridge),
        mean_absolute_error(y_test, y_pred_ridge),
        np.sqrt(mean_squared_error(y_test, y_pred_ridge))
    ]
})

error_matrix_ridge
```

```
  Metric          Value
0     R2  2.941729e-01
1    MSE  8.660658e+07
2    MAE  7.322765e+03
3   RMSE  9.306266e+03
```

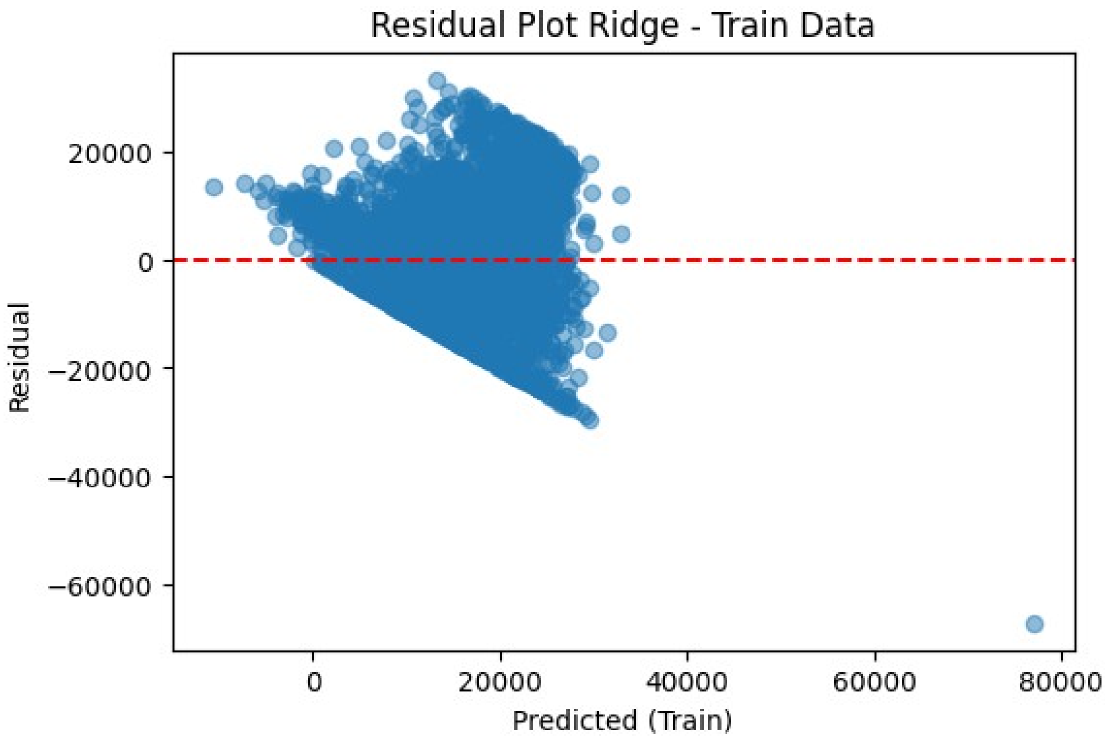
Feature Ridge

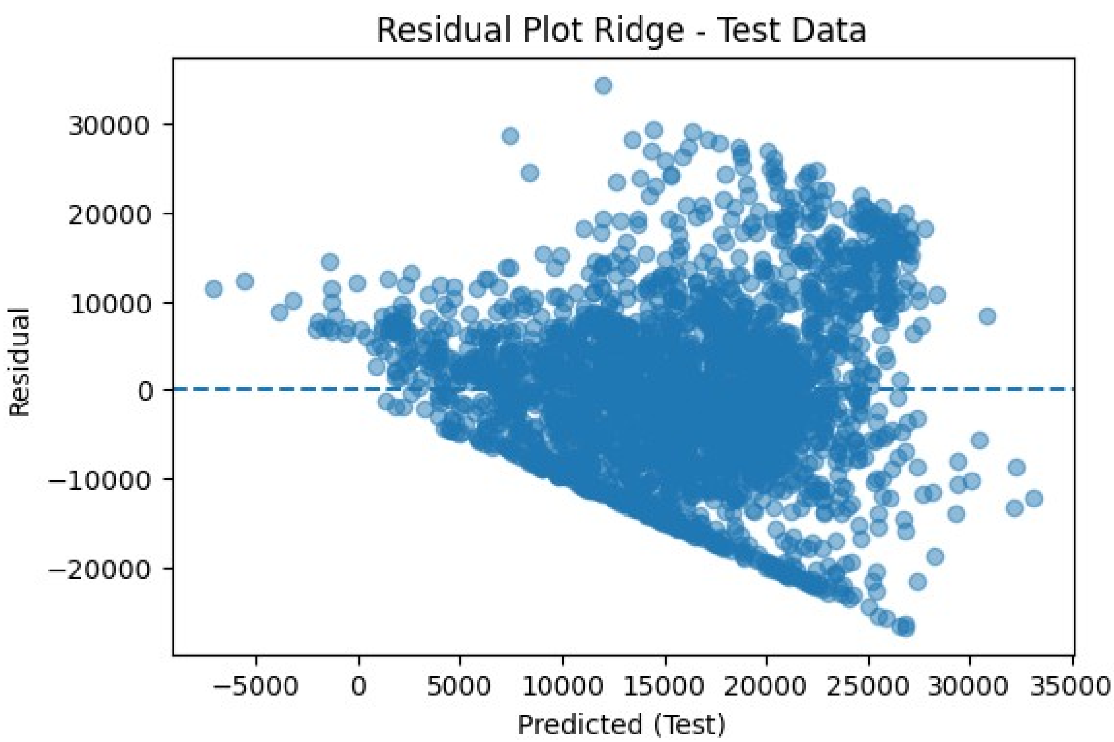
```
best_selector_ridge =
grid_ridge.best_estimator_.named_steps['feature_selection']
selected_features_ridge =
X_train.columns[best_selector_ridge.get_support()]

print("Selected features (Ridge):")
print(selected_features_ridge)
```

```
Selected features (Ridge):
Index(['Levy', 'Manufacturer', 'Model', 'Prod. year', 'Category',
       'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Gear box type', 'Drive wheels', 'Doors', 'Wheel', 'Color',
'Airbags'],
      dtype='object')
```

Data Visualization

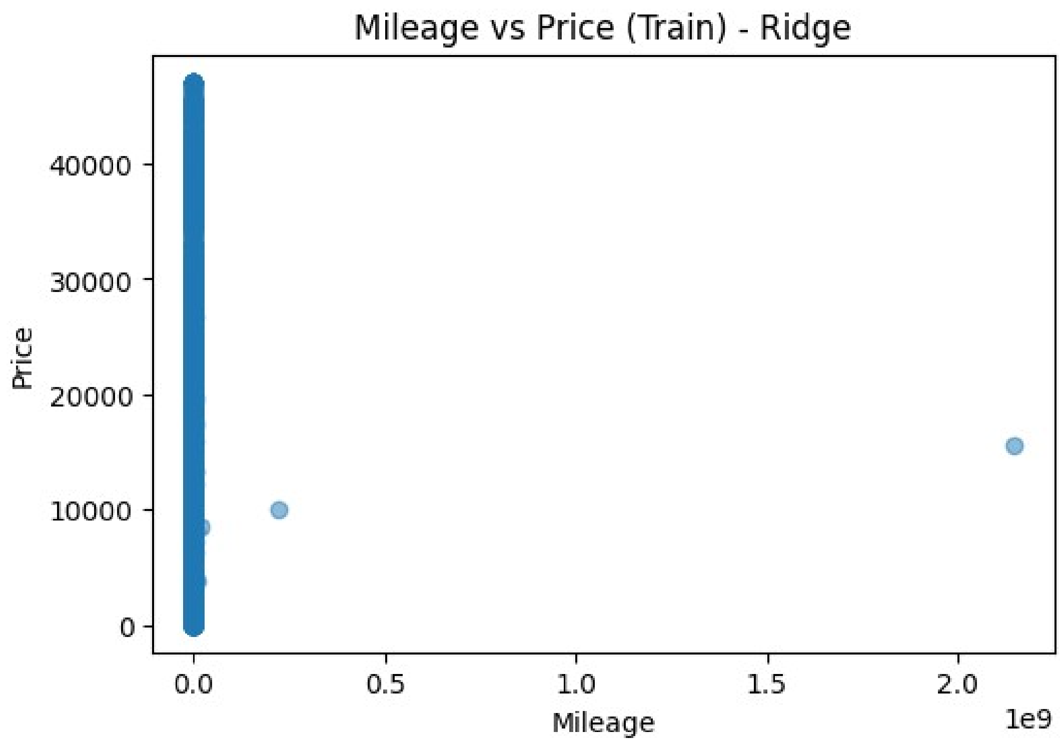
```
best_model_ridge = grid_ridge.best_estimator_

y_pred_train_ridge = best_model_ridge.predict(X_train)
y_pred_test_ridge  = best_model_ridge.predict(X_test)

plt.figure(figsize=(6,4))
plt.scatter(y_pred_train_ridge, y_train - y_pred_train_ridge,
alpha=0.5)
plt.axhline(0, linestyle='--',color ="red")
plt.xlabel("Predicted (Train)")
plt.ylabel("Residual")
plt.title("Residual Plot Ridge - Train Data")
plt.show()


plt.figure(figsize=(6,4))
plt.scatter(y_pred_test_ridge, y_test - y_pred_test_ridge, alpha=0.5)
plt.axhline(0, linestyle='--')
plt.xlabel("Predicted (Test)")
plt.ylabel("Residual")
plt.title("Residual Plot Ridge - Test Data")
plt.show()
```

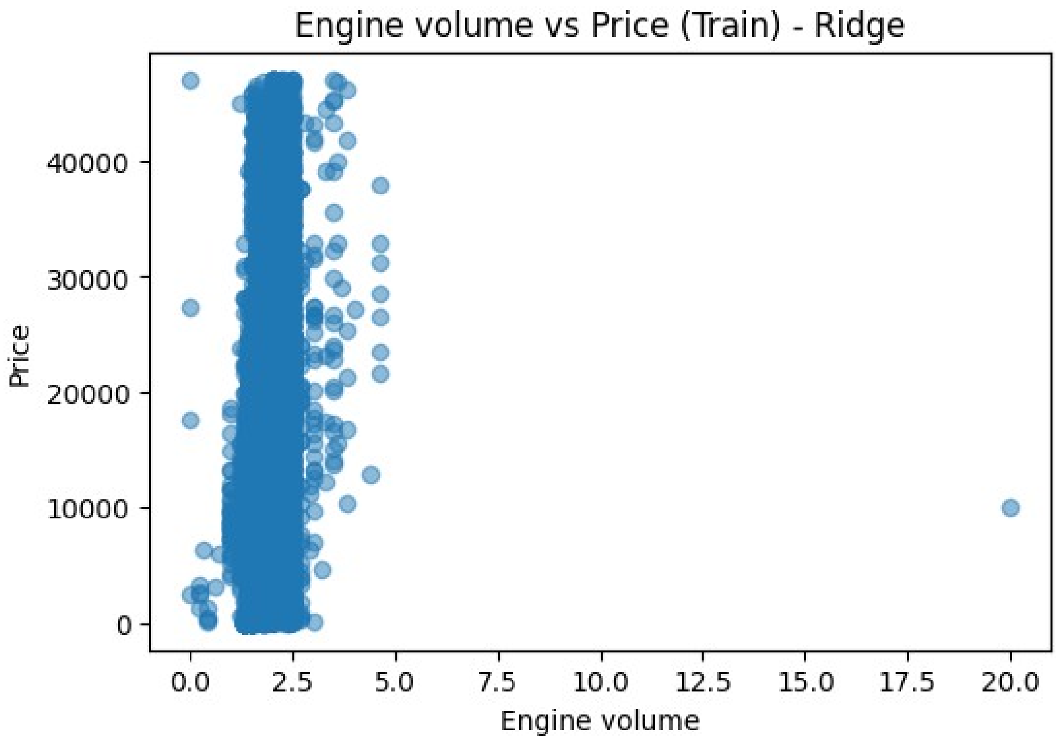


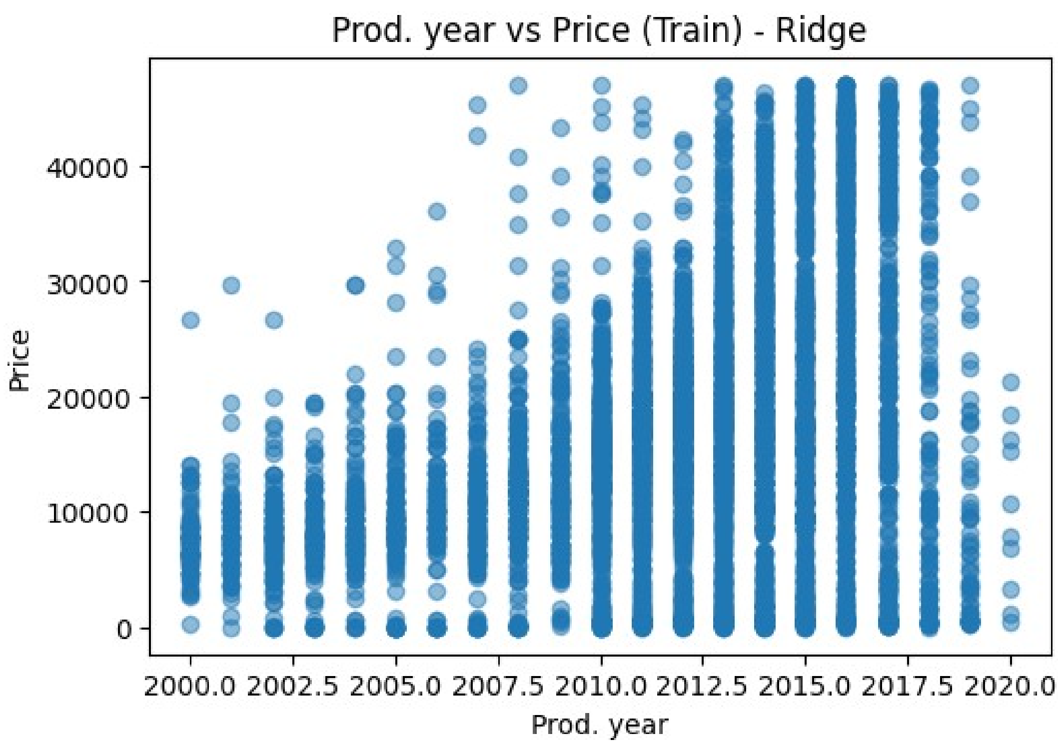



```
features = ['Mileage', 'Engine volume', 'Prod. year']

for feature in features:
```

```
    plt.figure(figsize=(6,4))
    plt.scatter(X_train[feature], y_train, alpha=0.5)
    plt.xlabel(feature)
    plt.ylabel("Price")
    plt.title(f"{feature} vs Price (Train) - Ridge")
    plt.show()
```

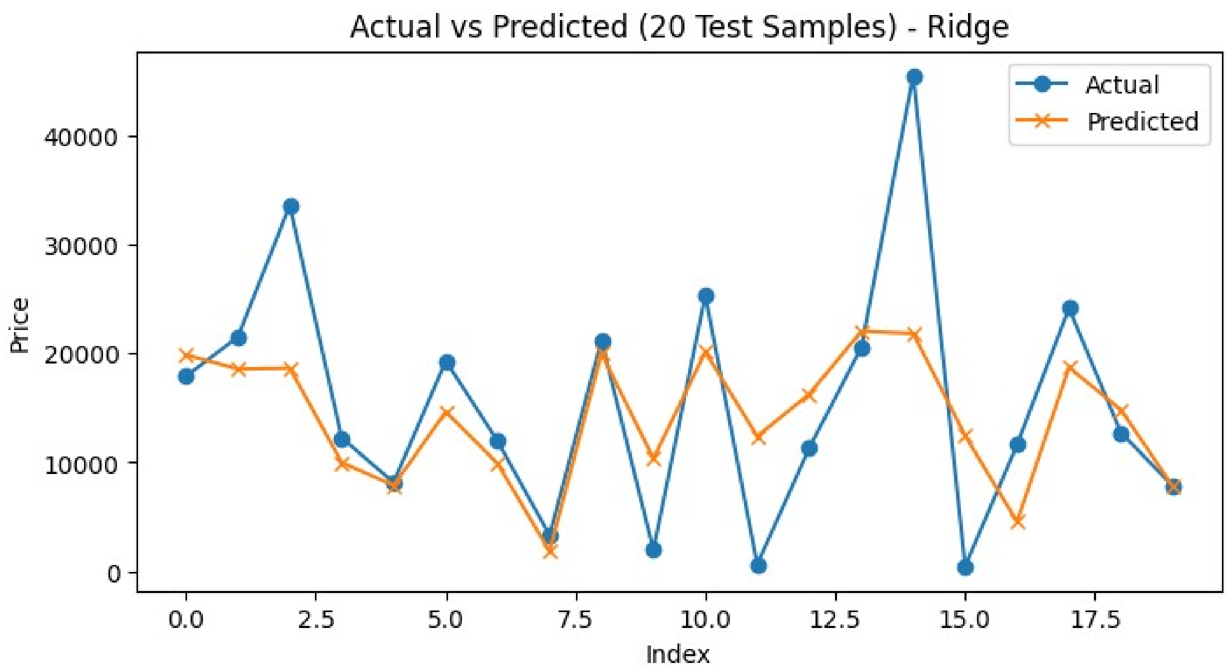




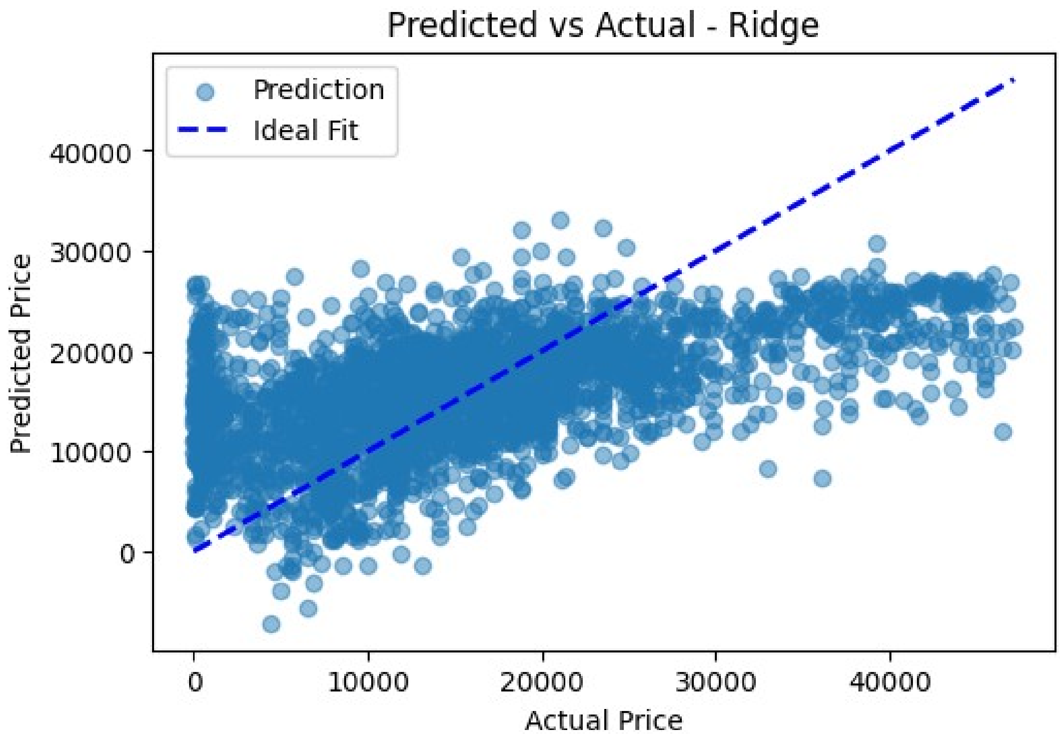



```
plt.figure(figsize=(8,4))
plt.plot(y_test.values[:20], label='Actual', marker='o')
plt.plot(y_pred_test_ridge[:20], label='Predicted', marker='x')
```

```
plt.legend()
plt.xlabel("Index")
plt.ylabel("Price")
plt.title("Actual vs Predicted (20 Test Samples) - Ridge")
plt.show()
```




```
plt.figure(figsize=(6,4))
plt.scatter(y_test, y_pred_test_ridge, alpha=0.5, label='Prediction')

plt.plot(
    [y_test.min(), y_test.max()],
    [y_test.min(), y_test.max()],
    color='blue',
    linestyle='--',
    linewidth=2,
    label='Ideal Fit'
)

plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted vs Actual - Ridge")
plt.legend()
plt.show()
```


Predicted vs Actual - Ridge
Prediction
Ideal Fit
Predicted Price
Actual Price
0
10000
20000
30000
40000

IMPOR LIBRARY

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.feature_selection import SelectKBest, SelectPercentile,
f_regression
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
```

DATA LOADING

```
df = pd.read_csv("car_price_prediction.csv")
df.head()
```

```
         ID  Price  Levy Manufacturer    Model  Prod. year   Category
\
0  45654403  13328  1399        LEXUS   RX 450        2010       Jeep

1  44731507  16621  1018    CHEVROLET  Equinox        2011       Jeep

2  45774419   8467     -        HONDA      FIT        2006  Hatchback

3  45769185   3607   862         FORD   Escape        2011       Jeep

4  45809263  11726   446        HONDA      FIT        2014  Hatchback


  Leather interior Fuel type Engine volume    Mileage  Cylinders  \
0              Yes    Hybrid           3.5  186005 km        6.0
1               No    Petrol             3  192000 km        6.0
2               No    Petrol           1.3  200000 km        4.0
3              Yes    Hybrid           2.5  168966 km        4.0
4              Yes    Petrol           1.3   91901 km        4.0

  Gear box type Drive wheels   Doors             Wheel   Color
Airbags
0     Automatic          4x4  04-May        Left wheel  Silver
12
1     Tiptronic          4x4  04-May        Left wheel   Black
8
2      Variator        Front  04-May  Right-hand drive   Black
```

```
2
3     Automatic          4x4  04-May        Left wheel   White
0
4     Automatic        Front  04-May        Left wheel  Silver
4
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19237 entries, 0 to 19236
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   ID                19237 non-null  int64
 1   Price             19237 non-null  int64
 2   Levy              19237 non-null  object
 3   Manufacturer      19237 non-null  object
 4   Model             19237 non-null  object
 5   Prod. year        19237 non-null  int64
 6   Category          19237 non-null  object
 7   Leather interior  19237 non-null  object
 8   Fuel type         19237 non-null  object
 9   Engine volume     19237 non-null  object
 10  Mileage           19237 non-null  object
 11  Cylinders         19237 non-null  float64
 12  Gear box type     19237 non-null  object
 13  Drive wheels      19237 non-null  object
 14  Doors             19237 non-null  object
 15  Wheel             19237 non-null  object
 16  Color             19237 non-null  object
 17  Airbags           19237 non-null  int64
dtypes: float64(1), int64(4), object(13)
memory usage: 2.6+ MB
```

```
df.describe()
```

```
                 ID         Price    Prod. year     Cylinders
Airbags
count  1.923700e+04  1.923700e+04  19237.000000  19237.000000
19237.000000
mean   4.557654e+07  1.855593e+04   2010.912824      4.582991
6.582627
std    9.365914e+05  1.905813e+05      5.668673      1.199933
4.320168
min    2.074688e+07  1.000000e+00   1939.000000      1.000000
0.000000
25%    4.569837e+07  5.331000e+03   2009.000000      4.000000
4.000000
50%    4.577231e+07  1.317200e+04   2012.000000      4.000000
6.000000
```

```
75%    4.580204e+07  2.207500e+04   2015.000000      4.000000
12.000000
max    4.581665e+07  2.630750e+07   2020.000000     16.000000
16.000000
```

EDA

```
df.isnull().sum()
```

```
ID                  0
Price               0
Levy                0
Manufacturer        0
Model               0
Prod. year          0
Category            0
Leather interior    0
Fuel type           0
Engine volume       0
Mileage             0
Cylinders           0
Gear box type       0
Drive wheels        0
Doors               0
Wheel               0
Color               0
Airbags             0
dtype: int64
```

```
df.duplicated().sum()
```

```
np.int64(313)
```

```
df = df.drop_duplicates()
df.duplicated().sum()
```

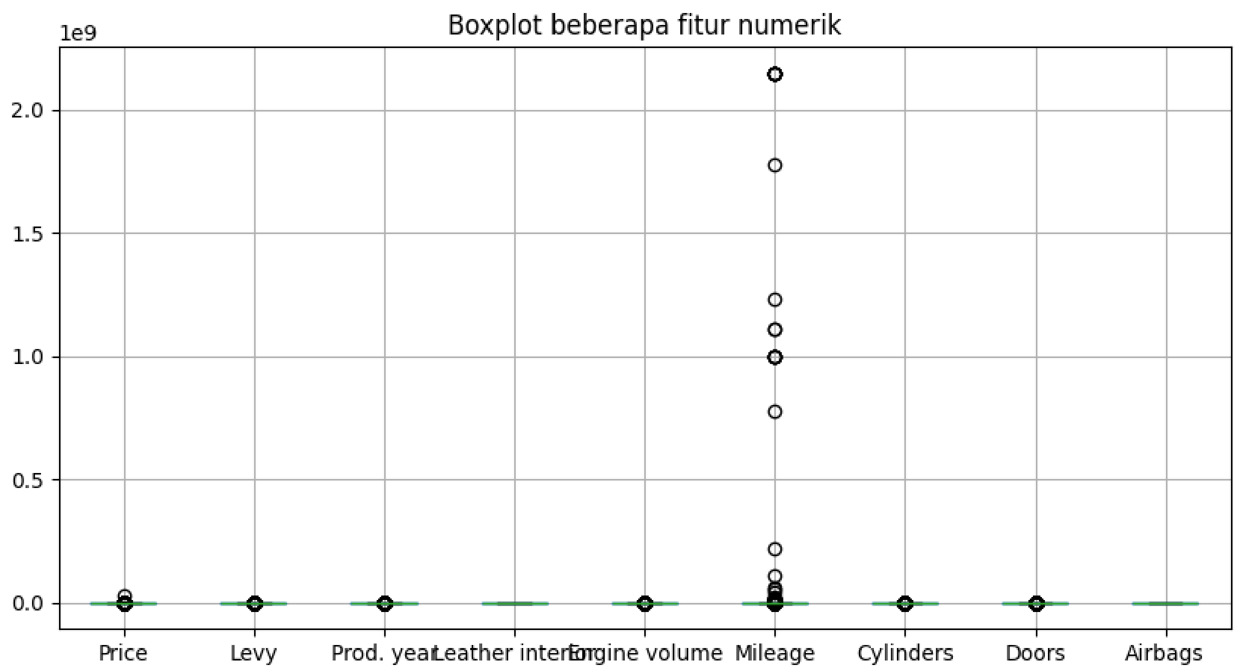
```
np.int64(0)
```

```
df = df.drop(columns=["ID"])

df["Levy"] = df["Levy"].replace("-", np.nan).astype(float)
df["Levy"] = df["Levy"].fillna(df["Levy"].median())

df["Mileage"] = df["Mileage"].str.replace(" km", "",
regex=False).astype(int)

df["Engine volume"] = (
    df["Engine volume"]
    .str.replace(" Turbo", "", regex=False)
    .astype(float)
)
```

```
df["Doors"] = df["Doors"].map({
    "02-Mar": 2,
    "04-May": 4
})
df["Doors"] = df["Doors"].fillna(df["Doors"].mode()[0])

df["Leather interior"] = df["Leather interior"].map({"Yes": 1, "No":
0})
```

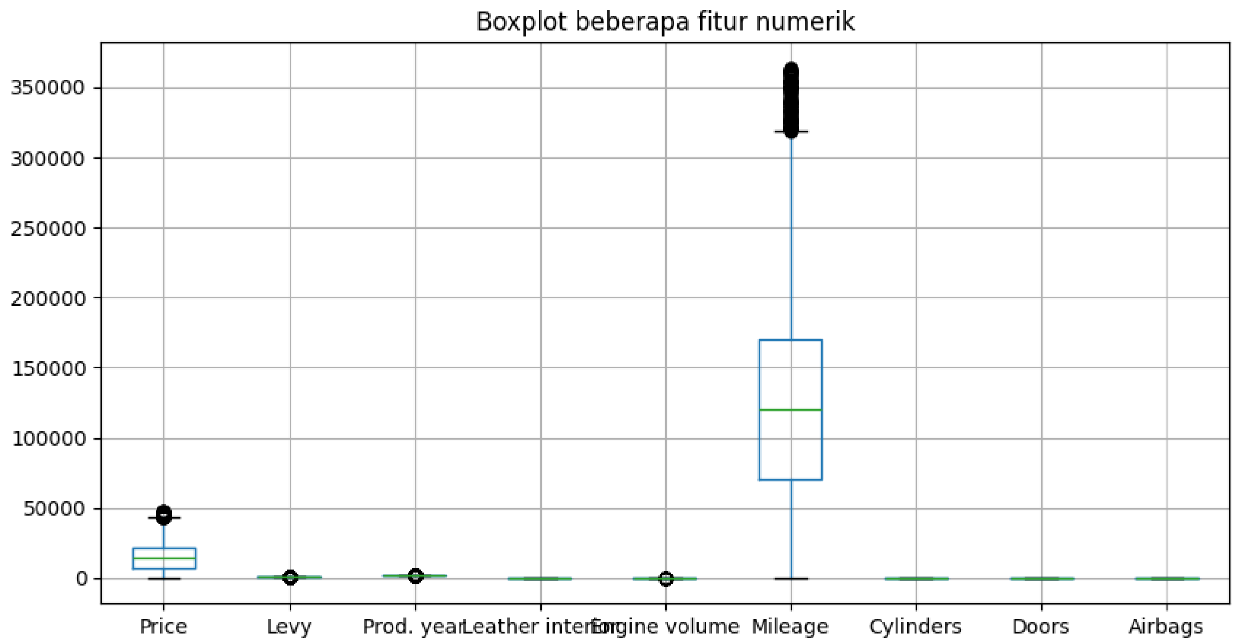
MEMERIKSA OUTLIER

```
numeric_cols = df.select_dtypes(include=np.number).columns
numeric_cols = [c for c in numeric_cols
                if c in df.columns]

plt.figure(figsize=(10, 5))
df[numeric_cols].boxplot()
plt.title("Boxplot beberapa fitur numerik")
plt.show()
```




```
#Hapus outlier
numeric_cols = df.select_dtypes(include=np.number).columns

Q1 = df[numeric_cols].quantile(0.25)
Q3 = df[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[numeric_cols] < (Q1 - 1.5 * IQR)) |
```

```
            (df[numeric_cols] > (Q3 + 1.5 * IQR))).any(axis=1)]

numeric_cols = [c for c in numeric_cols if c in df.columns]

plt.figure(figsize=(10, 5))
df[numeric_cols].boxplot()
plt.title("Boxplot beberapa fitur numerik")
plt.show()
```



DATA ENCODING

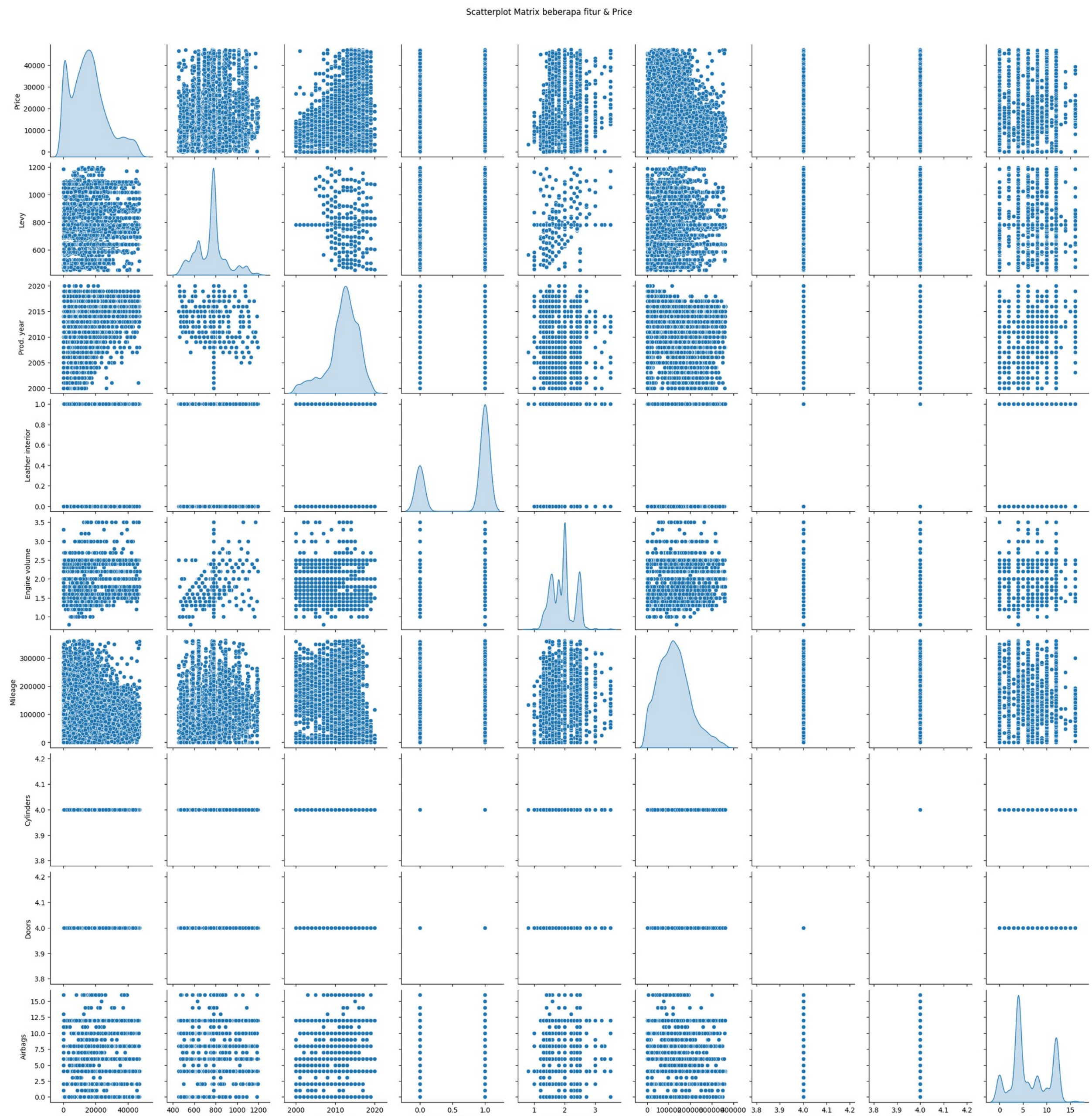
```
cat_cols = df.select_dtypes(include="object").columns
cat_cols

Index(['Manufacturer', 'Model', 'Category', 'Fuel type', 'Gear box
type',
        'Drive wheels', 'Wheel', 'Color'],
       dtype='object')

df_encoded = pd.get_dummies(
    df,
    columns=cat_cols,
    drop_first=True
)
df_encoded.head()

   Price   Levy  Prod. year  Leather interior  Engine volume  Mileage
\
2   8467  781.0        2006                 0            1.3   200000
```

```
3   3607  862.0        2011                 1            2.5   168966

5  39493  891.0        2016                 1            2.0   160931

6   1803  761.0        2010                 1            1.8   258909

7    549  751.0        2013                 1            2.4   216118


   Cylinders  Doors  Airbags  Manufacturer_ALFA ROMEO  ...
Color_Green  \
2        4.0    4.0        2                    False  ...
False
3        4.0    4.0        0                    False  ...
False
5        4.0    4.0        4                    False  ...
False
6        4.0    4.0       12                    False  ...
False
7        4.0    4.0       12                    False  ...
False

   Color_Grey  Color_Orange  Color_Pink  Color_Purple  Color_Red  \
2       False         False       False         False      False
3       False         False       False         False      False
5       False         False       False         False      False
6       False         False       False         False      False
7        True         False       False         False      False

   Color_Silver  Color_Sky blue  Color_White  Color_Yellow
2         False           False        False         False
3         False           False         True         False
5         False           False         True         False
6         False           False         True         False
7         False           False        False         False

[5 rows x 892 columns]
```

PRE TRAIN

```
X = df_encoded.drop("Price", axis=1)
y = df_encoded["Price"]

RANDOM_STATE = 86

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=RANDOM_STATE
)
```

```
sns.pairplot(df[numeric_cols], diag_kind="kde")
plt.suptitle("Scatterplot Matrix beberapa fitur & Price", y=1.02)
plt.show()
```



PIPELINE 1 - DECISION TREE REGRESSION

```
pipe_dt = Pipeline([
    ("scaler", StandardScaler()),
    ("selector", SelectKBest(score_func=f_regression)),
```

```
    ("model", DecisionTreeRegressor(random_state=RANDOM_STATE))
])

param_dt = [

    #Untuk SelectKBest
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [SelectKBest(score_func=f_regression)],
        "selector__k": [5, 8, 10],
        "model__max_depth": [None, 5, 10],
        "model__min_samples_split": [2, 5, 10]
    },

    #Untuk SelectPercentile
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [SelectPercentile(score_func=f_regression)],
        "selector__percentile": [20, 40, 60],
        "model__max_depth": [None, 5, 10],
        "model__min_samples_split": [2, 5, 10]
    }
]

gs_dt = GridSearchCV(
    pipe_dt,
    param_dt,
    cv=5,
    scoring="r2",
    n_jobs=-1
)

gs_dt.fit(X_train, y_train)
print("Best parameters (Decision Tree):", gs_dt.best_params_)
print("Best CV score DT (R2):", gs_dt.best_score_)
```

```
Best parameters (Decision Tree): {'model__max_depth': 10,
'model__min_samples_split': 10, 'scaler': StandardScaler(),
'selector': SelectPercentile(score_func=<function f_regression at
0x00000124EE688C20>), 'selector__percentile': 60}
Best CV score DT (R2): 0.6540533461527953
```

PIPELINE 2 RANDOM FOREST REGRESSION

```
pipe_rf = Pipeline([
    ("scaler", StandardScaler()),
    ("selector", SelectKBest(score_func=f_regression)),
    ("model", RandomForestRegressor(random_state=RANDOM_STATE,
n_jobs=-1))
])
```

```
param_rf = [
    # Untuk SelectKBest
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [SelectKBest(score_func=f_regression)],
        "selector__k": [5, 8, 10],
        "model__n_estimators": [100, 500],
        "model__max_depth": [None, 10, 20]
    },
    # Untuk SelectPercentile
    {
        "scaler": [StandardScaler(), MinMaxScaler()],
        "selector": [SelectPercentile(score_func=f_regression)],
        "selector__percentile": [20, 40, 60],
        "model__n_estimators": [100, 500],
        "model__max_depth": [None, 10, 20]
    }
]

gs_rf = GridSearchCV(
    pipe_rf,
    param_rf,
    cv=5,
    scoring="r2",
    n_jobs=-1
)

gs_rf.fit(X_train, y_train)
print("Best parameters (Random Forest):", gs_rf.best_params_)
print("Best CV score RF (R2):", gs_rf.best_score_)
```

```
Best parameters (Random Forest): {'model__max_depth': None,
'model__n_estimators': 500, 'scaler': MinMaxScaler(), 'selector':
SelectPercentile(score_func=<function f_regression at
0x00000124EE688C20>), 'selector__percentile': 60}
Best CV score RF (R2): 0.7734461251618352
```

EVALUASI MODEL

```
def regression_metrics(y_true, y_pred):
    return {
        "R2": r2_score(y_true, y_pred),
        "MSE": mean_squared_error(y_true, y_pred),
        "MAE": mean_absolute_error(y_true, y_pred),
        "RMSE": np.sqrt(mean_squared_error(y_true, y_pred))
    }

#DT
y_pred_dt = gs_dt.predict(X_test)
```

```python
metrics_dt = regression_metrics(y_test, y_pred_dt)
metrics_dt
```

```
{'R2': 0.6719078428847016,
 'MSE': np.float64(40342040.32178318),
 'MAE': np.float64(4374.2092977514385),
 'RMSE': np.float64(6351.538421656849)}
```

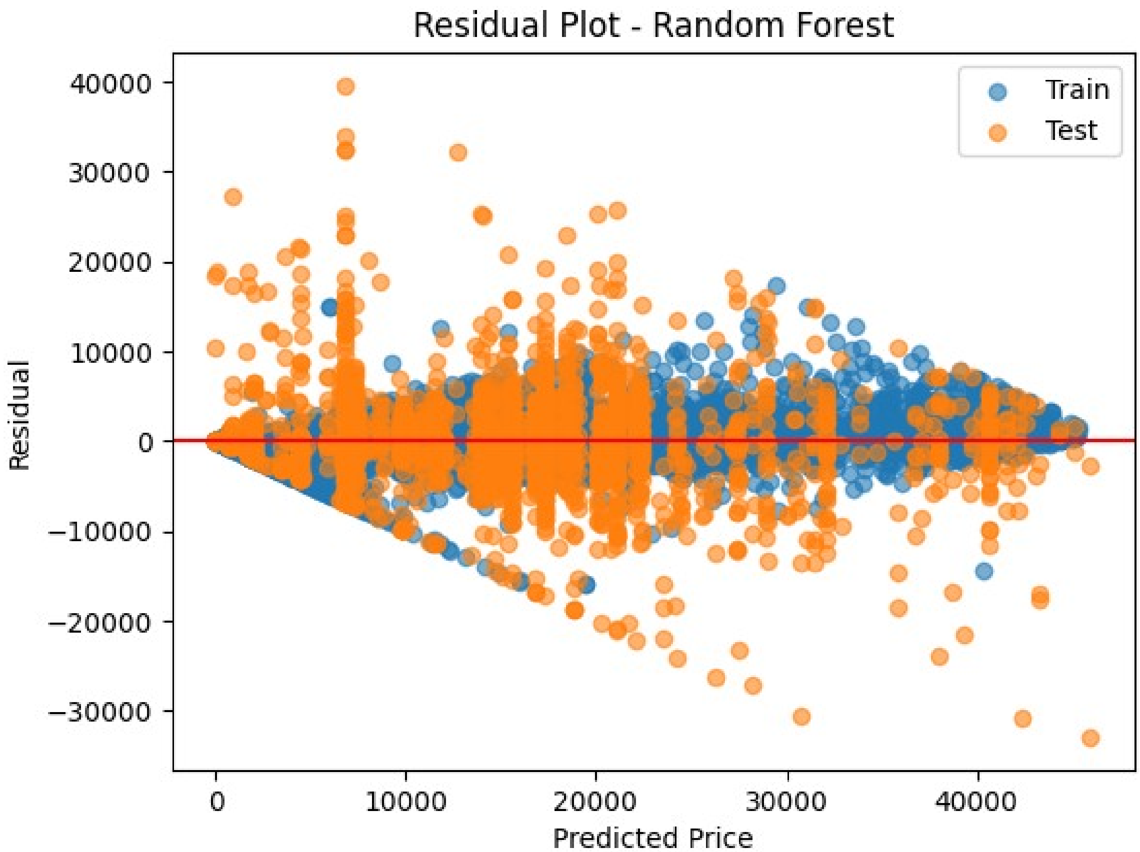
```python
# RF
y_pred_rf = gs_rf.predict(X_test)
metrics_rf = regression_metrics(y_test, y_pred_rf)
metrics_rf
```

```
{'R2': 0.7881148236365519,
 'MSE': np.float64(26053290.647354484),
 'MAE': np.float64(3224.9963253509172),
 'RMSE': np.float64(5104.242416593718)}
```

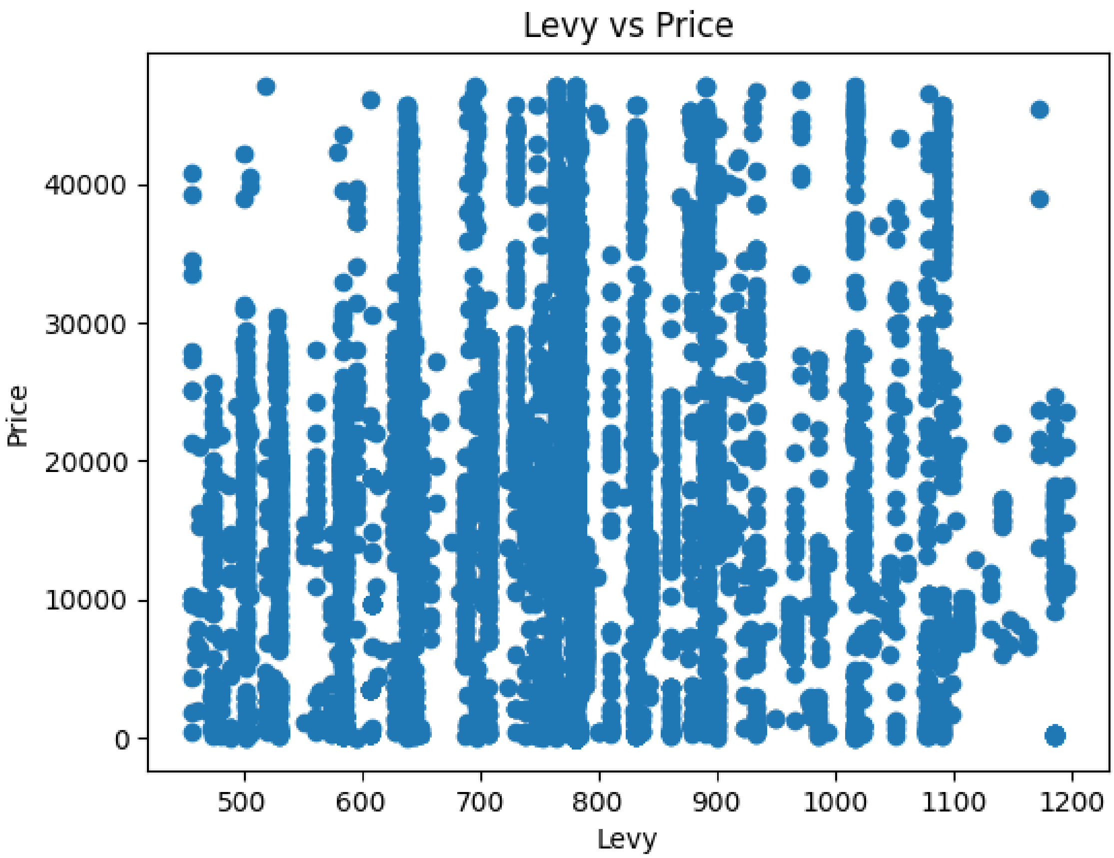
```python
# ganti best selector kalau R2 bagusan RF

best_selector = gs_rf.best_estimator_.named_steps["selector"]
selected_features = X.columns[best_selector.get_support()]
selected_features
```

```
Index(['Levy', 'Prod. year', 'Leather interior', 'Engine volume',
'Mileage',
       'Airbags', 'Manufacturer_AUDI', 'Manufacturer_BMW',
       'Manufacturer_BUICK', 'Manufacturer_CADILLAC',
       ...
       'Color_Carnelian red', 'Color_Golden', 'Color_Green',
'Color_Grey',
       'Color_Pink', 'Color_Red', 'Color_Silver', 'Color_Sky blue',
       'Color_White', 'Color_Yellow'],
      dtype='object', length=534)
```

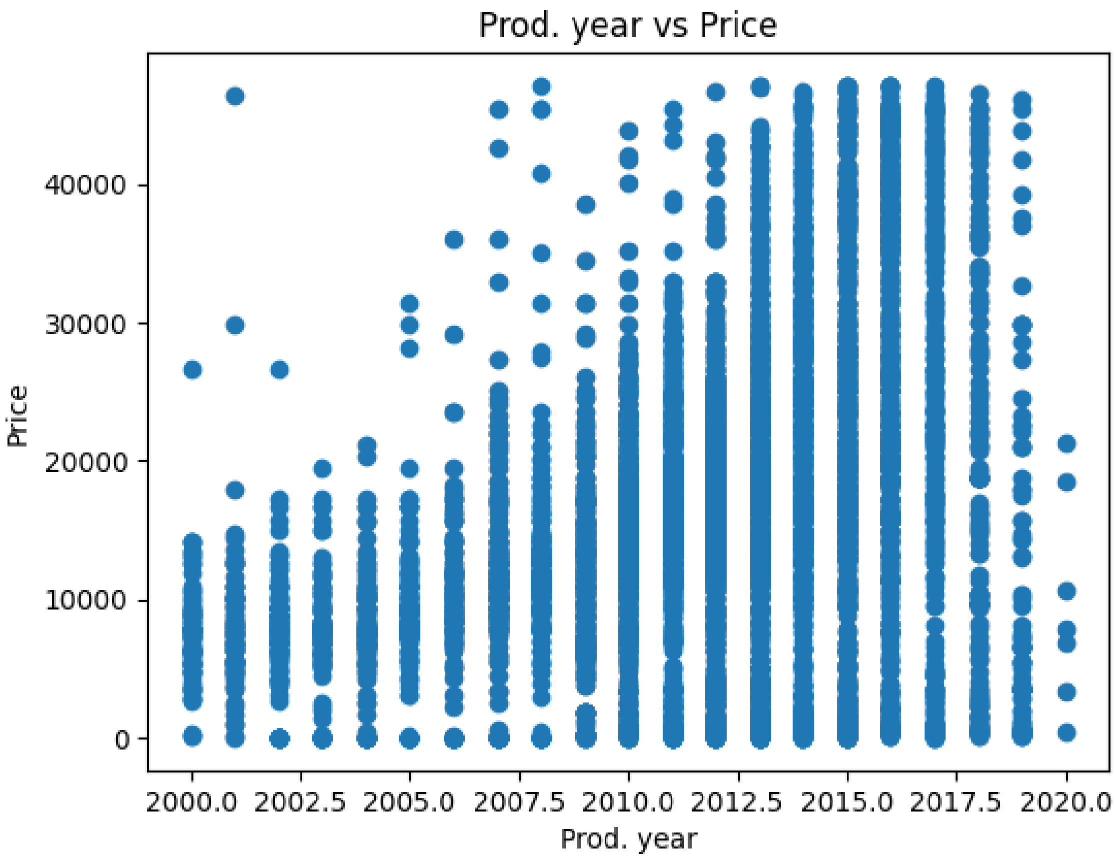
```python
# ganti gs.dt kalau R2 bagusan RF, jadi gs.rf [pokoknya yang ada dt
nya diganti rf kalau bagusan RF]

y_train_pred = gs_rf.predict(X_train)

plt.scatter(y_train_pred, y_train - y_train_pred, label="Train",
alpha=0.6)
plt.scatter(y_pred_dt, y_test - y_pred_dt, label="Test", alpha=0.6)
plt.axhline(0, color="red")
plt.legend()
plt.title("Residual Plot - Random Forest")
plt.xlabel("Predicted Price")
plt.ylabel("Residual")
plt.show()
```

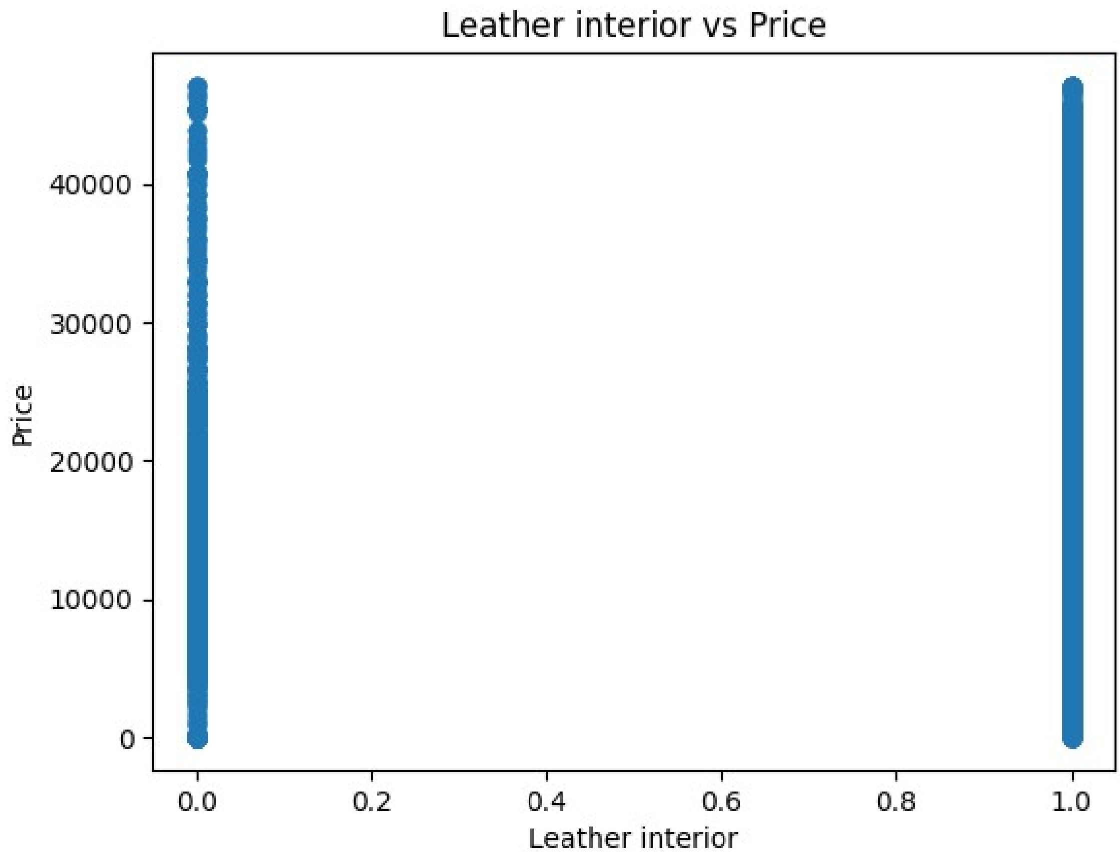



```
for feature in selected_features[:3]:
    plt.scatter(X_train[feature], y_train)
    plt.xlabel(feature)
    plt.ylabel("Price")
    plt.title(f"{feature} vs Price")
    plt.show()
```



Levy vs Price
Price
40000
30000
20000
10000
0
500
600
700
800
900
1000
1100
1200
Levy


Prod. year vs Price
Price
40000
30000
20000
10000
0
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
Prod. year

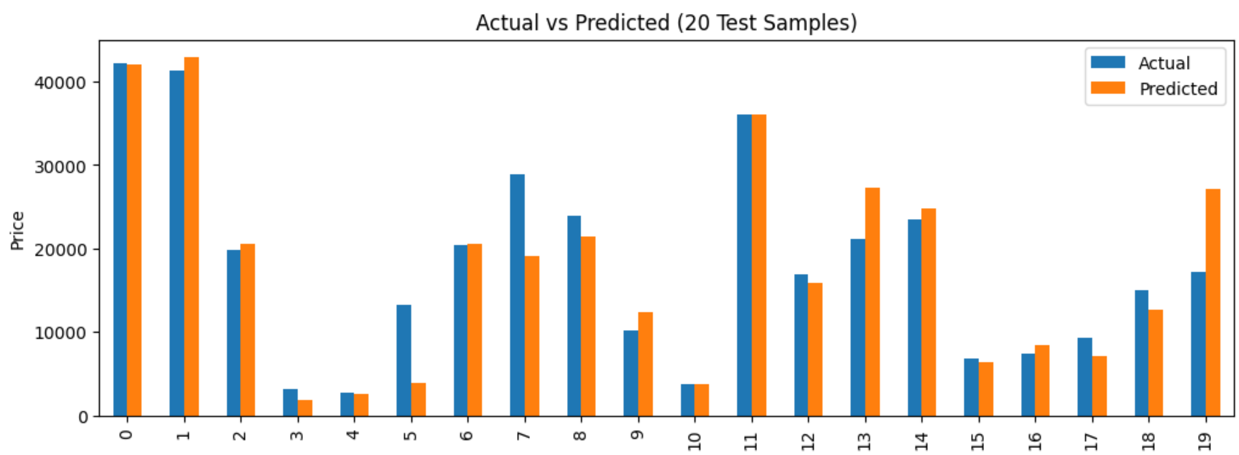



```
# ini juga kalau R2 RF bagus ganti aja DT ke RF
comparison = pd.DataFrame({
    "Actual": y_test.iloc[:20].values,
    "Predicted": y_pred_rf[:20]
})

comparison
```

	Actual	Predicted
0	42171	41968.258000
1	41281	42875.066000
2	19757	20603.652800
3	3136	1772.646000
4	2666	2592.547333
5	13172	3878.528000
6	20385	20604.684000
7	28830	19149.786000
8	23834	21411.036000
9	10192	12286.236000
10	3763	3675.486000
11	36065	36044.467967
12	16935	15870.412000
13	21109	27301.156000

```
14   23521  24736.182000
15    6743   6335.674000
16    7370   8375.472000
17    9252   7147.886000
18   15053  12695.848000
19   17249  27163.824000
```

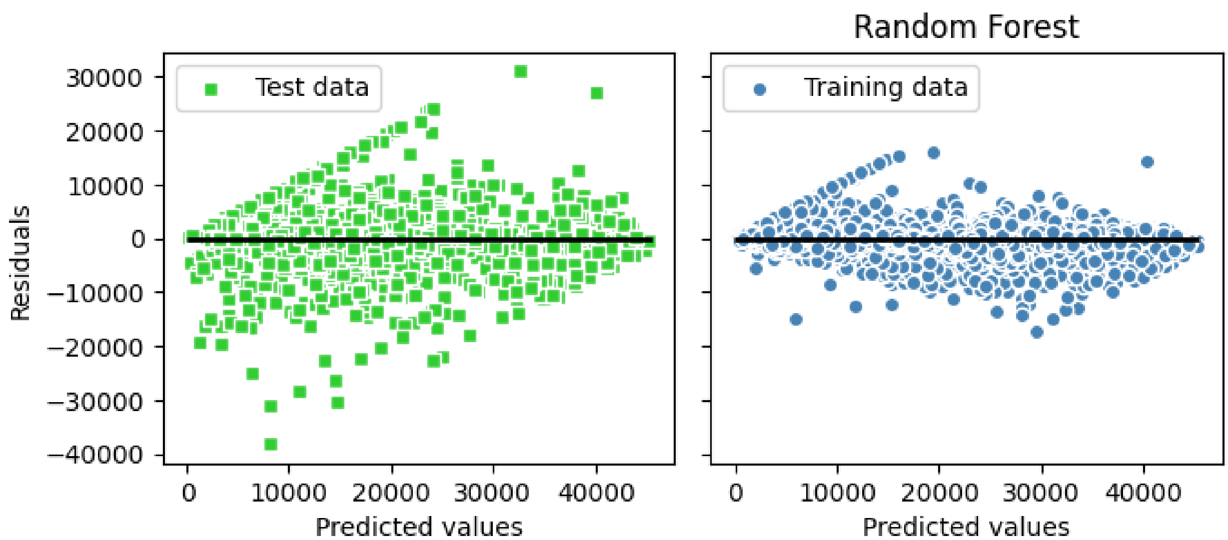
```
comparison.plot(kind="bar", figsize=(12,4))
plt.title("Actual vs Predicted (20 Test Samples)")
plt.ylabel("Price")
plt.show()
```



Residual Plot

```
if metrics_rf["R2"] > metrics_dt["R2"]:
    y_train_pred = gs_rf.predict(X_train)
    y_test_pred = gs_rf.predict(X_test)
    model_name = "Random Forest"
else:
    y_train_pred = gs_dt.predict(X_train)
    y_test_pred = gs_dt.predict(X_test)
    model_name = "Decision Tree"

x_max = np.max([np.max(y_train_pred), np.max(y_test_pred)])
x_min = np.min([np.min(y_train_pred), np.min(y_test_pred)])

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(7, 3), sharey=True)

ax1.scatter(y_test_pred, y_test_pred - y_test,
            c='limegreen', marker='s', edgecolor='white',
            label='Test data')

ax2.scatter(y_train_pred, y_train_pred - y_train,
            c='steelblue', marker='o', edgecolor='white',
            label='Training data')
```

```python
ax1.set_ylabel('Residuals')

for ax in (ax1, ax2):
    ax.set_xlabel('Predicted values')
    ax.legend(loc='upper left')
    ax.hlines(y=0, xmin=x_min-100, xmax=x_max+100, color='black',
lw=2)

plt.tight_layout()
plt.title(model_name)
plt.show()
```



Model Deployment

```python
import pickle

best_model = gs_rf.best_estimator_

with open("BestModel_RandomForest_CatBoost.pkl", "wb") as f:
    pickle.dump(best_model, f)

with open("columns.pkl", "wb") as f:
    pickle.dump(X.columns.tolist(), f)

print("□")

□
```