In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
car_data = pd.read_csv("car_price.csv")
```

In [3]:

```
car_data.head()
```

Out[3]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engi
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	

5 rows × 26 columns

In [4]:

```
car_data.tail()
```

Out[4]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engine
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	

5 rows × 26 columns

In [5]:

```
car_data.shape
```

Out[5]:

```
(205, 26)
```

In [6]:

```
car_data.columns
```

Out[6]:

```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbas
e',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'strok
e',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaymp
g',
       'price'],
      dtype='object')
```

In [7]:

```
car_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [8]:

```
car_data.describe()
```

Out[8]:

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000

In [9]:

```
car_data.isnull().sum()
```

Out[9]:

```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [10]:

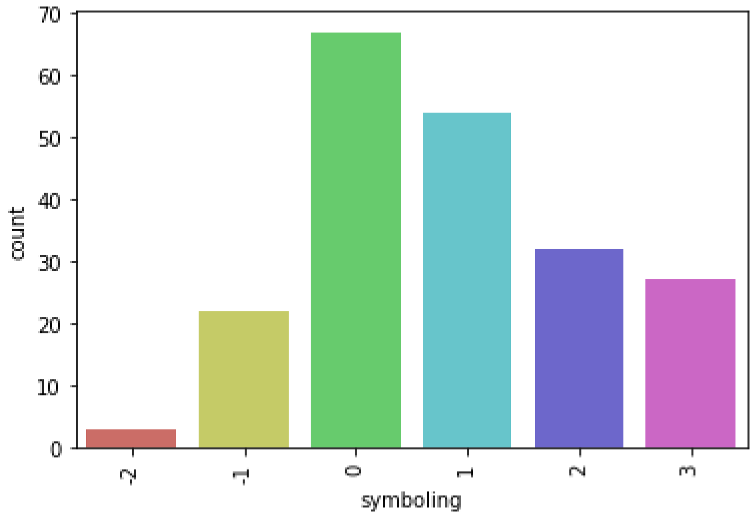
```
car_data.nunique()
```

Out[10]:

```
car_ID              205
symboling             6
CarName             147
fueltype              2
aspiration            2
doornumber            2
carbody               5
drivewheel            3
enginelocation        2
wheelbase            53
carlength            75
carwidth             44
carheight            49
curbweight          171
enginetype            7
cylindernumber        7
enginesize           44
fuelsystem            8
boreratio            38
stroke               37
compressionratio     32
horsepower           59
peakrpm              23
citympg              29
highwaympg           30
price               189
dtype: int64
```

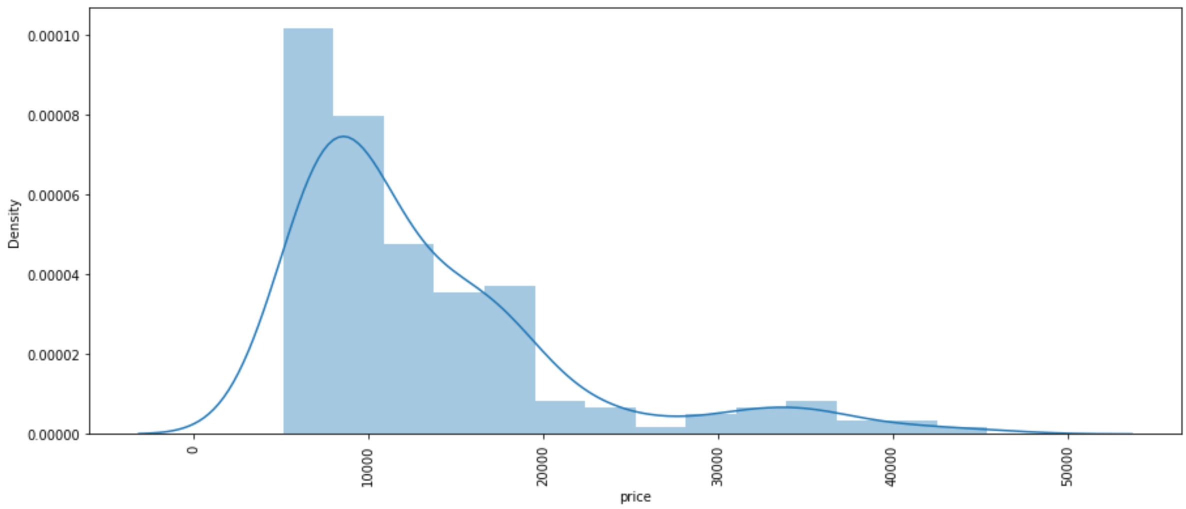
In [11]:

```
for i in car_data[['symboling', 'fueltype', 'aspiration', 'doornumber', 'carbody', 'driv
                  'enginelocation', 'enginetype', 'cylindernumber', 'fuelsystem']]:
    sns.countplot(car_data[i], data = car_data, palette='hls')
    plt.xticks(rotation = 90)
    plt.show()
```



In [12]:

```
plt.figure(figsize=(15,6))
sns.distplot(car_data['price'])
plt.xticks(rotation = 90)
plt.show()
```

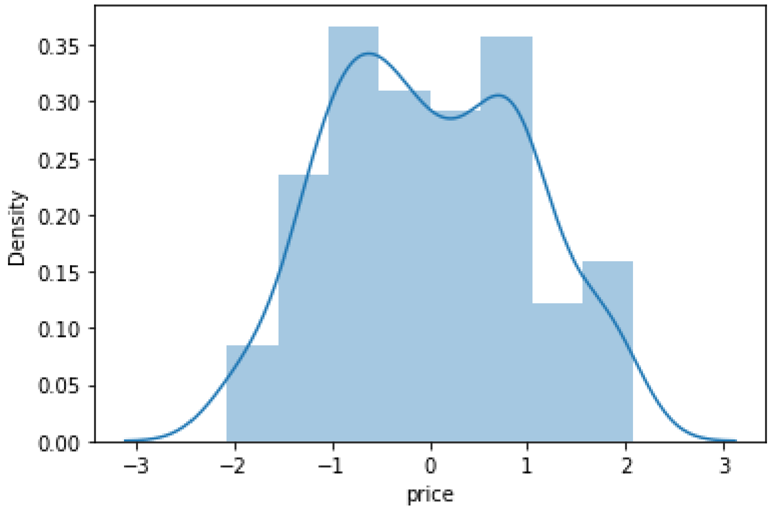


In [13]:

```
from sklearn.preprocessing import PowerTransformer
p = PowerTransformer(method = 'box-cox')
car_data['price'] = p.fit_transform(car_data[['price']])
sns.distplot(car_data.price)
```

Out[13]:

```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



In [14]:

```
car_data.drop(columns = ['car_ID'],inplace = True)
```

In [15]:

```
car = car_data.CarName.str.split(expand = True)
Brand = car[0]
car_data['Brand'] = Brand
car_data
```

Out[15]:

	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engineloca
0	3	alfa-romero giulia	gas	std	two	convertible	rwd	f
1	3	alfa-romero stelvio	gas	std	two	convertible	rwd	f
2	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	f
3	2	audi 100 ls	gas	std	four	sedan	fwd	f
4	2	audi 100ls	gas	std	four	sedan	4wd	f
...	...	...	...	...	...	...	...	
200	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	f
201	-1	volvo 144ea	gas	turbo	four	sedan	rwd	f
202	-1	volvo 244dl	gas	std	four	sedan	rwd	f
203	-1	volvo 246	diesel	turbo	four	sedan	rwd	f
204	-1	volvo 264gl	gas	turbo	four	sedan	rwd	f

205 rows × 26 columns

In [16]:

```
car_data.drop(columns = ['CarName'],inplace = True)
```

In [18]:

```
car_data.head(10)
```

Out[18]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase
0	3	gas	std	two	convertible	rwd	front	88.6
1	3	gas	std	two	convertible	rwd	front	88.6
2	1	gas	std	two	hatchback	rwd	front	94.5
3	2	gas	std	four	sedan	fwd	front	99.8
4	2	gas	std	four	sedan	4wd	front	99.4
5	2	gas	std	two	sedan	fwd	front	99.8
6	1	gas	std	four	sedan	fwd	front	105.8
7	1	gas	std	four	wagon	fwd	front	105.8
8	1	gas	turbo	four	sedan	fwd	front	105.8
9	0	gas	turbo	two	hatchback	4wd	front	99.5

10 rows × 25 columns

In [19]:

```
car_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   symboling         205 non-null    int64  
 1   fueltype          205 non-null    object 
 2   aspiration        205 non-null    object 
 3   doornumber        205 non-null    object 
 4   carbody           205 non-null    object 
 5   drivewheel        205 non-null    object 
 6   enginelocation    205 non-null    object 
 7   wheelbase         205 non-null    float64
 8   carlength         205 non-null    float64
 9   carwidth          205 non-null    float64
 10  carheight         205 non-null    float64
 11  curbweight        205 non-null    int64  
 12  enginetype        205 non-null    object 
 13  cylindernumber    205 non-null    object 
 14  enginesize        205 non-null    int64  
 15  fuelsystem        205 non-null    object 
 16  boreratio         205 non-null    float64
 17  stroke            205 non-null    float64
 18  compressionratio  205 non-null    float64
 19  horsepower        205 non-null    int64  
 20  peakrpm           205 non-null    int64  
 21  citympg           205 non-null    int64  
 22  highwaympg        205 non-null    int64  
 23  price             205 non-null    float64
 24  Brand             205 non-null    object 
dtypes: float64(8), int64(7), object(10)
memory usage: 40.2+ KB
```

In [20]:

```
car_data.corr()
```

Out[20]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize
symboling	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790
wheelbase	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329
carlength	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360
carwidth	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433
carheight	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149
curbweight	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594
enginesize	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000
boreratio	-0.130051	0.488750	0.606454	0.559150	0.171071	0.648480	0.583774
stroke	-0.008735	0.160959	0.129533	0.182942	-0.055307	0.168790	0.203129
compressionratio	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	0.028971
horsepower	0.070873	0.353294	0.552623	0.640732	-0.108802	0.750739	0.809769
peakrpm	0.273606	-0.360469	-0.287242	-0.220012	-0.320411	-0.266243	-0.244660
citympg	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	-0.653658
highwaympg	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	-0.677470
price	-0.091310	0.633710	0.791501	0.795125	0.181206	0.888370	0.779846

In [21]:

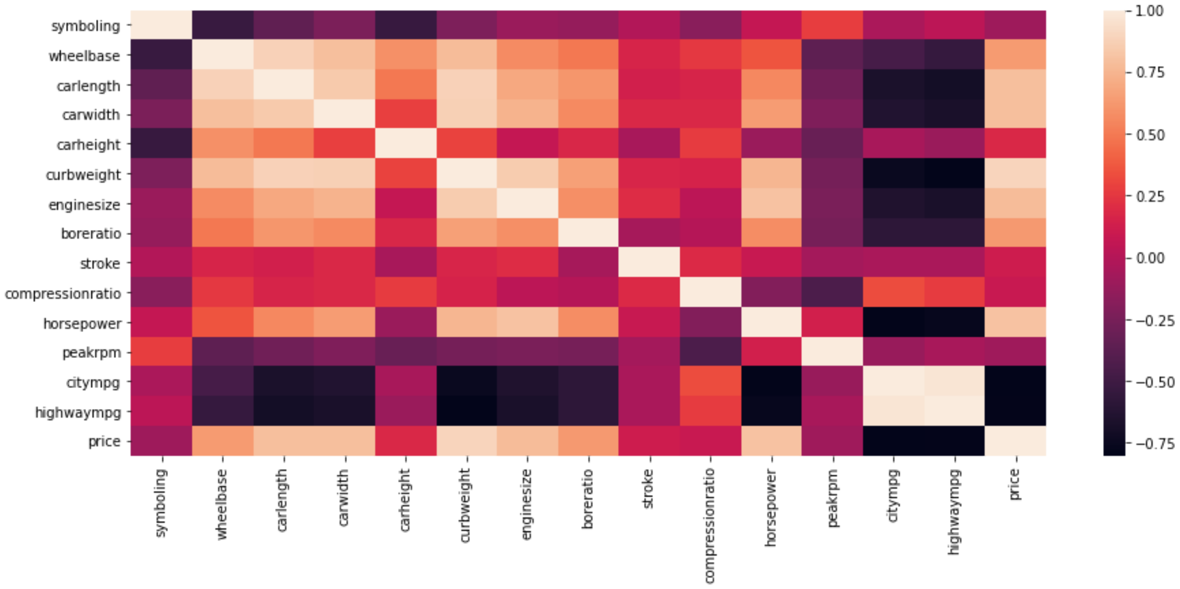
```
car_data.corr().style.background_gradient(cmap = 'coolwarm')
```

Out[21]:

	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize
symboling	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	-0.227691	-0.105790
wheelbase	-0.531954	1.000000	0.874587	0.795144	0.589435	0.776386	0.569329
carlength	-0.357612	0.874587	1.000000	0.841118	0.491029	0.877728	0.683360
carwidth	-0.232919	0.795144	0.841118	1.000000	0.279210	0.867032	0.735433
carheight	-0.541038	0.589435	0.491029	0.279210	1.000000	0.295572	0.067149
curbweight	-0.227691	0.776386	0.877728	0.867032	0.295572	1.000000	0.850594
enginesize	-0.105790	0.569329	0.683360	0.735433	0.067149	0.850594	1.000000
boreratio	-0.130051	0.488750	0.606454	0.559150	0.171071	0.648480	0.583774
stroke	-0.008735	0.160959	0.129533	0.182942	-0.055307	0.168790	0.203129
compressionratio	-0.178515	0.249786	0.158414	0.181129	0.261214	0.151362	0.028971
horsepower	0.070873	0.353294	0.552623	0.640732	-0.108802	0.750739	0.809769
peakrpm	0.273606	-0.360469	-0.287242	-0.220012	-0.320411	-0.266243	-0.244660
citympg	-0.035823	-0.470414	-0.670909	-0.642704	-0.048640	-0.757414	-0.653658
highwaympg	0.034606	-0.544082	-0.704662	-0.677218	-0.107358	-0.797465	-0.677470
price	-0.091310	0.633710	0.791501	0.795125	0.181206	0.888370	0.779846

In [22]:

```
plt.figure(figsize=(15,6))
sns.heatmap(car_data.corr())
plt.show()
```



In [23]:

```
car_data = car_data[car_data[("cylindernumber")].map(car_data["cylindernumber"].value_co
car_data = car_data[car_data[("enginetype")].map(car_data["enginetype"].value_counts()):
car_data = car_data[car_data[("fueltype")].map(car_data["fueltype"].value_counts())>3]
car_data = car_data[car_data[("Brand")].map(car_data["Brand"].value_counts())>9]
car_data = car_data[car_data.carbody!='convertable']
car_data = car_data[car_data.fueltype!='mfi']
```

In [24]:

```
car_data.shape
```

Out[24]:

```
(119, 25)
```

In [25]:

```
car_data.head(10)
```

Out[25]:

	symboling	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbas
30	2	gas	std	two	hatchback	fwd	front	86.
31	2	gas	std	two	hatchback	fwd	front	86.
32	1	gas	std	two	hatchback	fwd	front	93.
33	1	gas	std	two	hatchback	fwd	front	93.
34	1	gas	std	two	hatchback	fwd	front	93.
35	0	gas	std	four	sedan	fwd	front	96.
36	0	gas	std	four	wagon	fwd	front	96.
37	0	gas	std	two	hatchback	fwd	front	96.
38	0	gas	std	two	hatchback	fwd	front	96.
39	0	gas	std	four	sedan	fwd	front	96.

10 rows × 25 columns

In [26]:

```
x = car_data.iloc[:,0:23]
y = car_data[['price']]
```

In [27]:

```
x = car_data.drop(columns = ['price'])
```

In [28]:

```
x.shape
```

Out[28]:

```
(119, 24)
```

In [29]:

```
y.shape
```

Out[29]:

```
(119, 1)
```

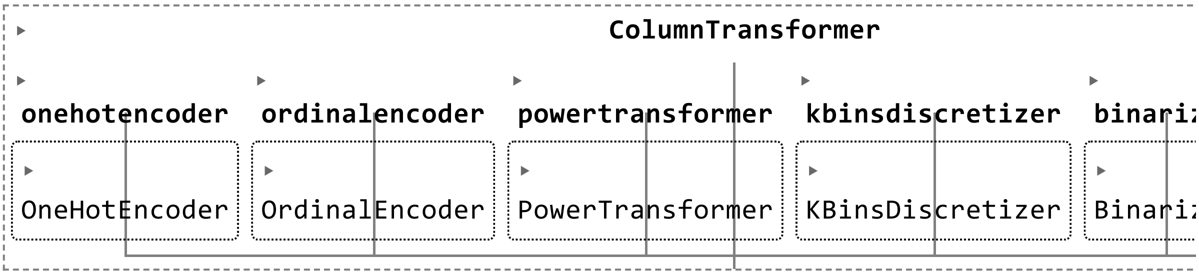
In [31]:

```
nomi_col = [4,5,12,15,23,13]
ordinal_col = [1,2,3,6]
numeric_col = [0,9,10,14,16,17,18,20,21,22]
KBin_col = [11,19]
Bina_col = [7,8]
```

In [32]:

```
from sklearn.preprocessing import KBinsDiscretizer,Binarizer,PowerTransformer
from sklearn.preprocessing import OneHotEncoder,OrdinalEncoder
from sklearn.compose import make_column_transformer
from sklearn import set_config
trans = make_column_transformer((OneHotEncoder(sparse = False),nomi_col),
                               (OrdinalEncoder(),ordinal_col),
                                 (PowerTransformer(),numeric_col),
                                 (KBinsDiscretizer(),KBin_col),
                                  (Binarizer(threshold = 50),Bina_col),
                                  remainder = 'passthrough')
set_config(display ='diagram')
trans
```

Out[32]:



In [33]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3)
```

In [34]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
from sklearn.pipeline import make_pipeline
pipe = make_pipeline(trans,model)
```

In [35]:

```
model
```

Out[35]:

```
▾ LinearRegression
LinearRegression()
```

In [36]:

```
pipe
```

Out[36]:

```
▸ Pipeline
▸ columntransformer: ColumnTransformer
▸ onehotencoder | ▸ ordinalencoder | ▸ powertransformer | ▸ kbinsdiscretizer | ▸ binar:
▸ OneHotEncoder | ▸ OrdinalEncoder | ▸ PowerTransformer | ▸ KBinsDiscretizer | ▸ Binar:
▸ LinearRegression
```

In [37]:

```
pipe.fit(x_train,y_train)
```

Out[37]:

```
▸ Pipeline
▸ columntransformer: ColumnTransformer
▸ onehotencoder | ▸ ordinalencoder | ▸ powertransformer | ▸ kbinsdiscretizer | ▸ binar:
▸ OneHotEncoder | ▸ OrdinalEncoder | ▸ PowerTransformer | ▸ KBinsDiscretizer | ▸ Binar:
▸ LinearRegression
```

In [38]:

```
pred = pipe.predict(x_test)
```

In [40]:

```
print("Training Accuracy :", pipe.score(x_train, y_train))
print("Testing Accuracy :", pipe.score(x_test, y_test))
```

```
Training Accuracy : 0.9592467871600668
Testing Accuracy : 0.8779740982655544
```

In [41]:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(pred,y_test)
```

Out[41]:

```
0.085096556437029
```

In [42]:

```
model.coef_
```

Out[42]:

```
array([[-1.04085006e+12, -1.04085006e+12, -1.04085006e+12,
        -1.04085006e+12, -1.04085006e+12,  3.89404297e-02,
        -1.42700195e-01,  1.07421875e-01,  3.82934570e-01,
         2.44140625e-03, -3.72314453e-03, -2.66235352e-01,
        -1.15112305e-01, -8.17565918e-01, -6.60400391e-02,
         1.65466309e-01,  3.99810791e-01,  3.17840576e-01,
         5.78292847e-01,  1.18896484e-01, -4.76409912e-01,
         5.32531738e-02,  2.47955322e-03, -2.66387939e-01,
        -4.01832581e-01,  3.87405396e-01, -6.98776245e-02,
         6.98471069e-02, -1.65409088e-01,  3.31193924e-01,
        -3.09400558e-01,  0.00000000e+00,  8.09755325e-02,
        -4.23507690e-02, -6.19957447e-02,  1.91562653e-01,
        -7.85109997e-02,  4.25249338e-02, -4.28781509e-02,
         0.00000000e+00, -2.18933821e-01,  2.55770683e-02,
        -4.57489014e-01, -6.58786297e-02,  3.97436619e-02,
         3.20430756e-01,  1.61612034e-01,  1.67842627e-01,
         3.00986767e-01,  1.89422965e-01, -1.94146633e-01,
        -4.59057689e-01,  0.00000000e+00,  0.00000000e+00]])
```