In [87]:
```
# importin libraries

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

In [88]:
```
#loading booston dataset from sklearn library

from sklearn.datasets import load_boston
boston=load_boston()
```

EDA

In [3]:
```
df=pd.DataFrame(boston.data)
df.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [4]:
```
df.shape
```

Out[4]:
```
(506, 13)
```

In [5]:
```
#adding features name to thr dataframe
df.columns=boston.feature_names
```

In [6]:
```
df.head()
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [7]:
```
df["price"]=boston.target
```

In [8]:
```
df.shape
```

Out[8]:
```
(506, 14)
```

In [9]:
```
df.head()
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [10]:
```
df.columns
```

Out[10]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'price'],
      dtype='object')
```

In [11]:
```
df.isnull().value_counts()
```

Out[11]:
```
CRIM   ZN     INDUS  CHAS   NOX    RM     AGE    DIS    RAD    TAX    PTRATIO  B      LS
TAT  price
False  False  False  False  False  False  False  False  False  False  False    False  Fa
lse  False    506
dtype: int64
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
price      0
dtype: int64
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  price    506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [14]:
```
df.describe()
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000

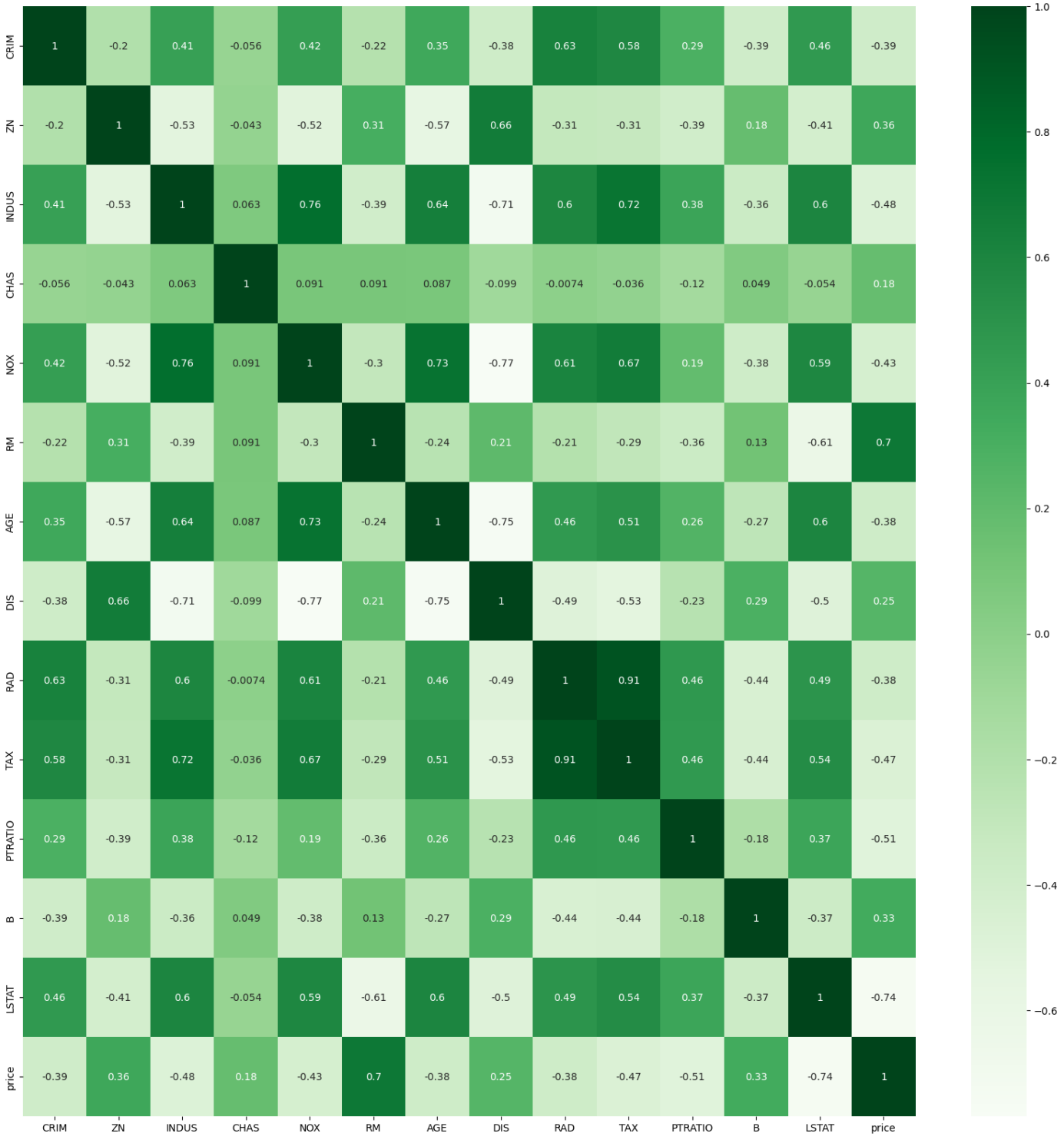
In [15]:
```
corr=df.corr()
corr
```

Out[15]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543
price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468

In [16]:
```
plt.figure(figsize=(20,20))
sns.heatmap(corr,annot=True,cmap="Greens" )
```

Out[16]:
```
<AxesSubplot:>
```



Modeling

In [17]:
```
#splitting data into target variable and features

x=df.drop(["price"], axis=1)
y=df["price"]
```

In [18]:
```
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, random_state=4)
```

Linear Regression

In [19]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[19]:
```
LinearRegression()
```

In [20]:
```
reg.intercept_
```

Out[20]:
```
36.35704137659525
```

In [21]:
```
reg.coef_
```

Out[21]:
```
array([-1.22569795e-01,  5.56776996e-02, -8.83428230e-03,  4.69344849e+00,
       -1.44357828e+01,  3.28008033e+00, -3.44778157e-03, -1.55214419e+00,
        3.26249618e-01, -1.40665500e-02, -8.03274915e-01,  9.35368715e-03,
       -5.23477529e-01])
```

In [22]:
```
coefficient =pd.DataFrame([x_train.columns,reg.coef_]).T
coefficient =coefficient.rename(columns={0:"Attribute",1:"coefficient"})
coefficient
```

Out[22]:

	Attribute	coefficient
0	CRIM	-0.12257
1	ZN	0.055678
2	INDUS	-0.008834
3	CHAS	4.693448
4	NOX	-14.435783
5	RM	3.28008
6	AGE	-0.003448
7	DIS	-1.552144
8	RAD	0.32625
9	TAX	-0.014067
10	PTRATIO	-0.803275
11	B	0.009354
12	LSTAT	-0.523478

In [23]:
```
y_pred=reg.predict(x_train)

y_pred
```

Out[23]:
```
array([24.52247959, 15.19750953, 25.5772058 , 13.93939959, 39.46651291,
       17.45959949, 39.71029914, 16.51748069, 20.19733317, 40.7977555 ,
       33.57245034, 14.50420619, 11.44514488, 23.06563951, 24.39734404,
       25.01096096, 14.36116466, 28.28341539, 25.04931481, 22.42825155,
       21.81588517, 18.85208726, 13.3562124 , 13.65792701, 23.64766018,
       18.06876331, 16.12957228, 41.12414874, 19.43391814, 13.17980896,
       12.88401778, 24.48360463, 24.47855835, 22.1443201 , 23.67882714,
       16.25760322, 18.73996794,  6.65661308, 13.32321855, 15.97333187,
       18.13637439, 20.60454402, 30.72884443,  7.75848746, 13.9207036 ,
       -3.80684116, 30.22337446,  9.24045737, 11.75825936, 25.0286616 ,
       18.84078418, 11.88476435, 28.69569373, 23.23374551, 28.59506092,
       20.30414612, 20.27099511, 25.21759304, 29.00813278, 17.29931495,
       10.61483731, 28.61476335, 28.34559327, 30.02547802, 17.11529423,
       20.57024793, 18.28474144, 13.03972891, 23.08527124, 22.90261359,
       20.38074908, 36.37414868, 20.26621151, 21.57778471, 14.51678385,
       19.8953567 , 21.51131918, 32.08090183, 35.12834332, 31.36704913,
       23.32859519, 22.59388715,  4.2838954 , 13.58551483, 17.84671145,
       22.39664149, 20.39218149, 21.10186967, 16.48607917, 20.89919772,
       19.24881766, 21.06506621, 33.14030608, 32.78970133, 14.78281   ,
       21.74354375, 27.16260271,  6.88304858, 20.34936977, 21.11588523,
       22.81012375, 25.39687827, 30.23966814, 14.62421558, 25.45042561,
       42.4032797 , 13.00311155, 17.23560015, 13.03879388, 16.43177226,
       16.78793009, 18.97978513, 22.65365079, 20.56436157, 29.30683492,
       21.77542094, 22.35785257, 24.66179729, 27.39184783, 33.36778289,
       18.60406   , 24.84058236, 21.87737973, 19.94977795, 20.2320981 ,
       28.80459631, 29.86452531, 19.0848428 , 23.73428397, 18.95219368,
        8.97390019, 26.75912306, 17.27253903, 14.55622874, 23.08500269,
       25.02655397, 25.16822738, 26.69701146, 19.54226068,  9.02486456,
       19.83161456, 24.7979826 , 25.91255945, 15.88668039, 25.06758329,
       23.43543477, 20.22343755, 36.60456605, 35.87394807, 14.16498026,
       35.7639938 , 26.89548837, 19.53419979, 24.0319941 , 18.32092746,
       19.34979091, 19.98984953, 30.84128072, 26.64756547, 32.78247806,
       14.78316127, 22.45545196, 15.81526628, 23.56128263, 21.7242162 ,
       27.32126354, 31.01013789, 22.81223613, 18.13416079, 45.69032277,
       17.1939182 , 24.10802533, 26.59321259, 15.60279544, 21.02771101,
        7.44005131, 22.17053479, 23.66051183, 31.01185063, 14.55358282,
       31.67018192, 21.97556068, 29.68281335, 22.63575956, 20.8096207 ,
       24.64566263, 24.99554971, 25.01653794, 33.51995131,  9.17715777,
       20.22700039, 14.11833626, 11.89907312, 17.32844649,  7.8211504 ,
       16.4015338 , 25.03819927, 34.52082211, 18.69844879, 12.54415505,
       27.52362409, 18.14569097, 32.62591972, 18.81336522, 25.34819514,
       10.30630161, 20.97762312, 16.75364295, 24.79811585, 34.51812328,
       17.5962357 , 16.81995806, 22.52681599, 34.49248022, 24.37976859,
       26.3891315 , 28.2701436 , 20.64370006, 22.02979521, 24.40052032,
       21.55933518, 20.76505653, 21.35398659, 21.63454359,  4.66800164,
       19.78493866, 15.48836079, 22.14765613, 23.52375126, 17.59029302,
       25.13686178, 14.95557981, 36.75987631, 33.2559292 , 31.61517802,
       20.90157211, 10.03744596, 20.9447537 , 37.08766373, 21.01939321,
       34.32376634, 13.85900709, 34.22456075, 19.32066433, 17.65498064,
        5.24826916, 34.88064127, 23.84714723, 22.47313566, 18.27631394,
       20.56812098, 17.77668675, 26.16651954, 18.259761  ,  7.23026096,
       23.31459663, 15.86666632,  7.22490521, 28.20893505, 16.76862409,
       15.76240974, 18.24652915, 23.96087459, 20.78378432, 24.50433793,
       35.21593709, 23.53739443, 20.95954548, 21.34702919, 32.24233771,
       21.03836719, 27.38910962, 11.90350999, 27.91947755, 29.75232698,
       31.85739443, 32.40263439, 33.45022214, 21.98205792, 25.13087325,
       34.68748318, 19.88892304, 13.34808093, 29.86667097, 16.05575864,
       33.58539138, 24.82396668, 18.73520405, 25.53501563, 32.74769528,
       23.70442095, 27.64551933, 26.49800198, 12.09728846, 21.00340178,
       21.58838992, 17.42189106, 28.17731487, 34.11882749, 17.78837317,
       24.85777712, 36.14387743, 28.09586722,  2.2097601 , 27.67952556,
       26.81582736, 35.21436795, 12.99108512, 16.35766119, 27.00345932,
       25.42702449, 13.11240734, 27.51069857, 19.01462202, 22.97871867,
       17.06885216, 16.05089171, 11.88063208,  6.86396864, 26.14095608,
```

```
31.12264996, 24.76013181, 18.63369868, 29.02792362, 15.15549492,
17.9944167 , 30.76172229, 29.54307499,  6.25936797, 27.17935245,
14.80595579, 23.59443828, 22.6686058 , 16.02071242, 24.05810667,
20.66133508, 25.37935258, 27.55369062, 26.95070997, 26.7556681 ,
19.8699349 , 19.69025622, 24.33259862, 21.92486901, 20.3544687 ,
35.33844969, 13.00764099, 25.81335033, 22.95996791,  8.60836873,
31.51107779, 13.64719071, 26.5010622 , 20.54096512])
```

In [24]:
```
from sklearn import metrics
r_sqrd=metrics.r2_score(y_train,y_pred)
r_sqrd
```

Out[24]:
```
0.7465991966746853
```

In [25]:
```
MAE=metrics.mean_absolute_error(y_train,y_pred)
MAE
```

Out[25]:
```
3.0898610949711274
```

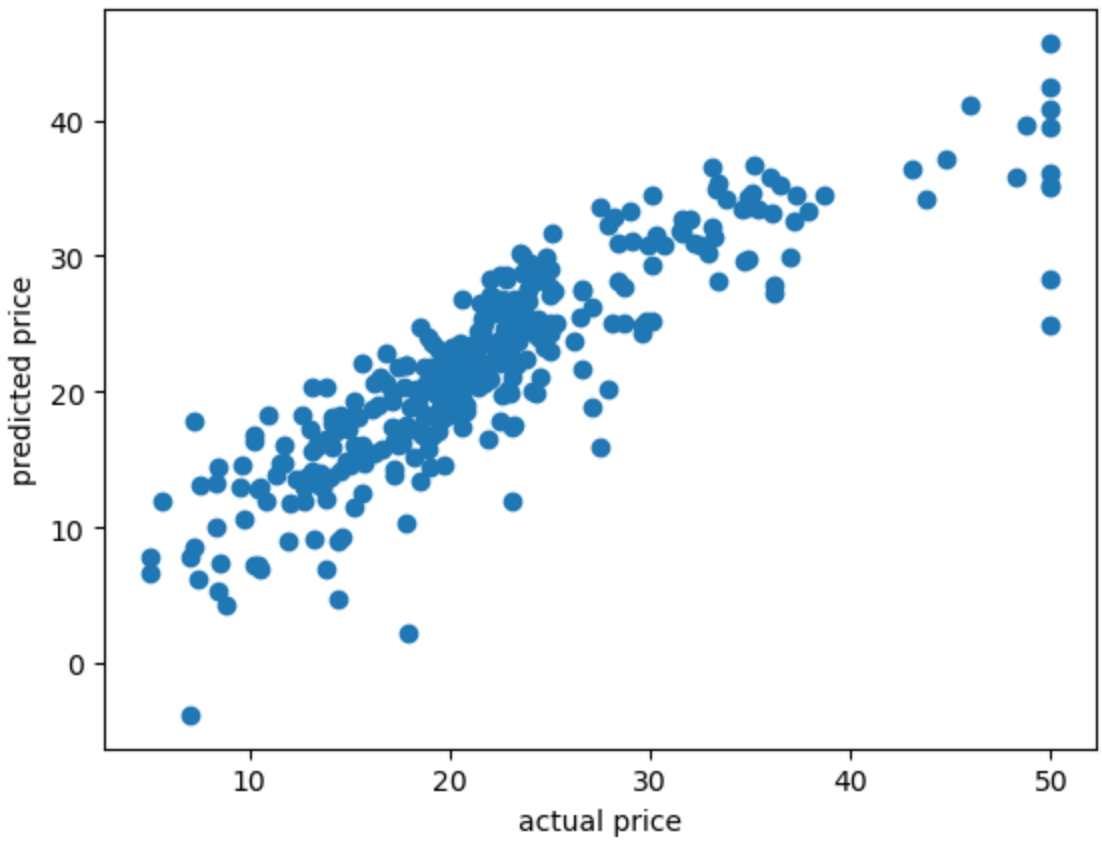
In [26]:
```
MSE=metrics.mean_squared_error(y_train,y_pred)
MSE
```

Out[26]:
```
19.07368870346904
```

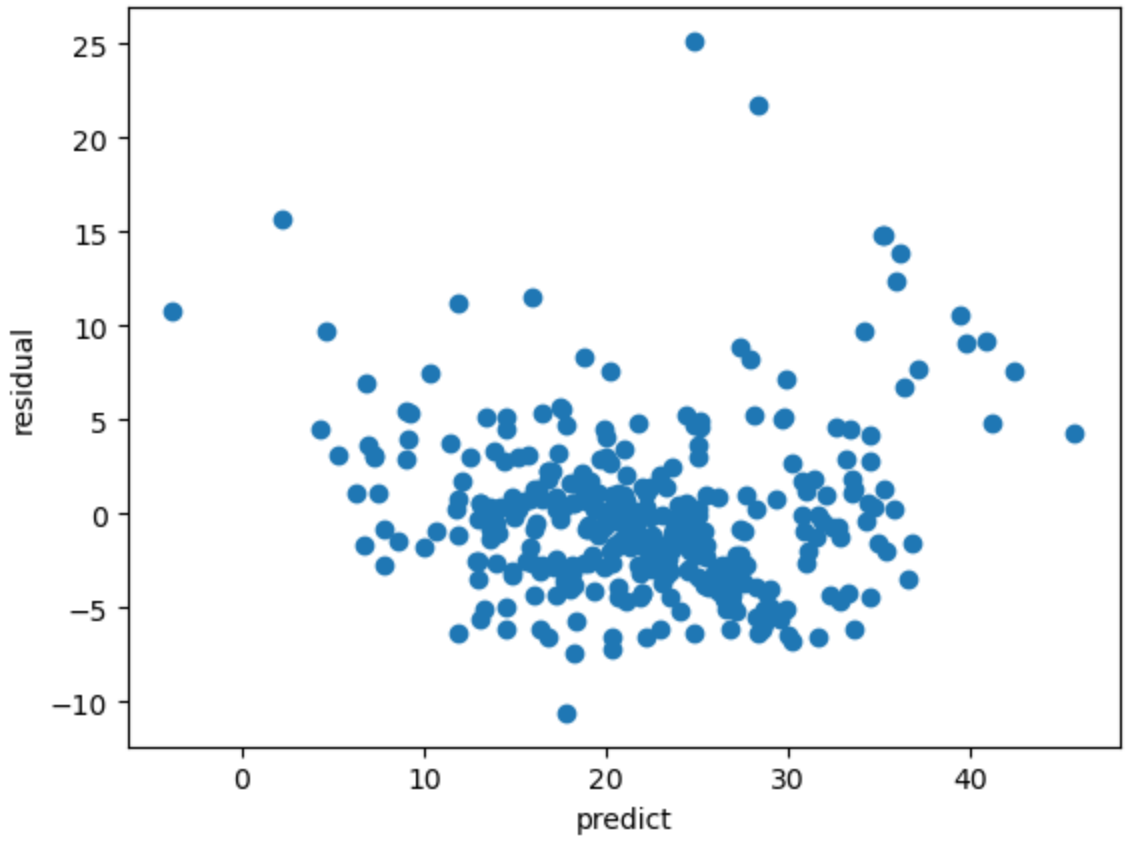
In [27]:
```
RMSE=np.sqrt(metrics.mean_squared_error(y_train,y_pred))
RMSE
```

Out[27]:
```
4.367343437774163
```

In [28]:
```
plt.scatter(y_train,y_pred)
plt.xlabel("actual price")
plt.ylabel("predicted price")
plt.show()
```

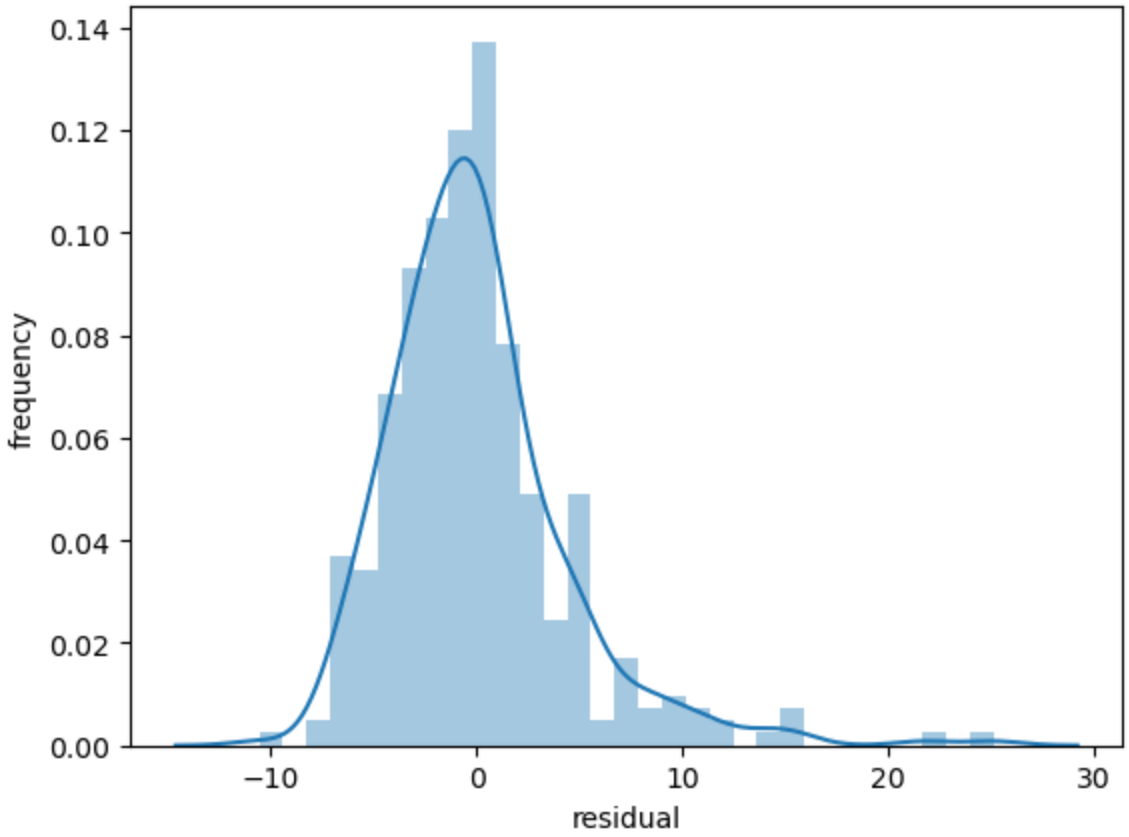


In [29]:
```
#checking residual
plt.scatter(y_pred, y_train-y_pred)
plt.xlabel("predict")
plt.ylabel("residual")
plt.show()
```



In [30]:
```
#checking normality of error ------ie-- normal distribution of residual data
sns.distplot(y_train-y_pred)
plt.ylabel("frequency")
plt.xlabel("residual")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [31]:
```
y_test_pred = reg.predict(x_test)
y_test_pred
```

Out[31]:
```
array([11.07380893, 26.47910329, 17.34489869, 19.1948608 , 36.36170735,
       24.77095832, 31.00051311, 19.94226881, 19.22375105, 24.42998435,
       28.31512637, 28.40796034, 19.27427968, 33.82295207, 21.28596487,
       15.11171444, 20.97688767, 11.28556596, 11.8611348 , 13.88444387,
        5.37422679, 17.55278177, 20.58171204, 22.59849951, 16.07544265,
       20.45924503, 19.1068775 , 14.37832191, 21.23235601, 17.52186564,
       14.40725559, 23.68483414, 33.7410661 , 22.02733357, 17.62139147,
       19.97241153, 30.24069397, 34.69718954, 23.85821534, 24.30715093,
       36.13378112, 31.97532293, 19.626175  , 31.61097971, 34.58127809,
       25.62718797, 39.95041812, 17.60880538, 19.90319708, 23.40417501,
       33.70182396, 25.62491083, 18.25559302, 27.27317174, 13.46377871,
       23.43470656, 24.43721849, 33.52056736, 16.99896935, 37.94464404,
       15.94567818, 19.32528916, 31.84088262, 15.25081303, 38.40344789,
       27.45372884, 34.36154312,  9.37353936, 19.42580066, 21.99218459,
       22.79983394, 22.50810313, 22.30918714, 27.84395887, 16.40818345,
       22.55507669, 16.5147332 , 25.11106836, 13.76991927, 19.78656399,
       22.10247463, 20.26663237, 28.15165586, 19.52050766, 30.33254364,
       22.79109999, 29.2663436 , 19.43113706, 24.7968264 , 37.46275648,
       31.05503576, 41.3372879 , 18.46365381, 36.67964528, 19.40842405,
       23.61810063, 27.93475362, 24.41825213,  9.4599059 , 20.68088677,
        8.99426788, 28.4492398 , 31.88237066, 14.04302958, 24.8347909 ,
       19.94124425, 36.90271393, 31.06556982, 33.91883403, 28.64591536,
       31.1007263 , 22.82363163, 11.58125942, 29.46902405, 37.06066106,
       23.01945872, 41.79865192, 18.44334162,  3.433324  , 18.57485663,
       22.21257489, 16.71192648, 28.00473344, 28.42374739, 19.6417452 ,
       18.76090758, 35.37631447, 13.12349548, 14.73923539, 18.16202333,
       38.26604753, 15.97821613, 41.91544265, 30.44631625, 28.65848089,
       24.19590457, 12.06559683, 26.01408744, 23.25012698, 18.92506857,
       17.05016777, 17.50245392, 20.89247338, 24.62630514,  1.82167558,
       23.03969555, 19.35693345, 17.89193065, 38.43943954, 19.7075262 ,
       31.67181183, 19.0130913 ])
```

In [32]:
```
reg.intercept_
```

Out[32]:
```
36.35704137659525
```

In [33]:
```
reg.coef_
```

Out[33]:
```
array([-1.22569795e-01,  5.56776996e-02, -8.83428230e-03,  4.69344849e+00,
       -1.44357828e+01,  3.28008033e+00, -3.44778157e-03, -1.55214419e+00,
        3.26249618e-01, -1.40665500e-02, -8.03274915e-01,  9.35368715e-03,
       -5.23477529e-01])
```

In [34]:
```
r2=metrics.r2_score(y_test,y_test_pred)
r2
```

Out[34]:
```
0.7121818377409185
```

In [35]:
```
MAE=metrics.mean_absolute_error(y_test,y_test_pred)
MAE
```

Out[35]:
```
3.859005592370742
```

In [36]:
```
MSE=metrics.mean_squared_error(y_test,y_test_pred)
MSE
```

Out[36]:
```
30.05399330712424
```

In [37]:
```
RMSE= np.sqrt(MSE)
RMSE
```

Out[37]:
```
5.482152251362985
```

#RandomForestRegressor

In [38]:
```
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
reg.fit(x_train,y_train)
```

Out[38]:
```
RandomForestRegressor()
```

In [39]:
```
#model evaluation

y_pred=reg.predict(x_train)
y_pred
```

Out[39]:
```
array([23.266, 18.774, 20.154, 13.795, 47.795, 21.818, 47.272, 14.058,
       20.316, 49.491, 34.641, 10.21 , 15.616, 22.354, 24.7  , 25.037,
       18.232, 44.907, 22.806, 20.798, 18.565, 20.108, 19.537, 14.601,
       22.68 , 15.997, 15.174, 46.355, 20.885, 13.947, 16.227, 21.88 ,
       21.876, 21.838, 23.501, 18.315, 16.117,  5.793,  8.739, 26.152,
       18.787, 21.158, 29.351,  6.53 , 11.514,  7.393, 33.215, 14.865,
       12.699, 25.867, 17.558,  6.789, 23.046, 23.591, 22.818, 18.442,
       13.326, 23.447, 24.88 , 14.997,  9.994, 22.245, 23.311, 27.955,
       14.972, 19.094, 20.239, 15.111, 20.257, 17.809, 20.208, 42.732,
       30.878, 20.395, 17.27 , 19.651, 21.72 , 32.933, 47.68 , 32.668,
       20.206, 21.183,  8.883, 12.242, 14.376, 22.77 , 18.769, 21.266,
       21.786, 21.519, 17.925, 22.48 , 35.295, 29.117, 12.066, 19.372,
       22.172,  9.447, 21.512, 19.492, 21.041, 23.637, 23.704, 18.783,
       25.534, 49.358, 10.229, 17.948, 14.015, 15.912, 16.951, 16.022,
       20.77 , 20.881, 28.101, 21.497, 22.402, 24.469, 25.99 , 36.574,
       18.54 , 28.039, 19.115, 21.848, 23.162, 24.113, 27.321, 20.893,
       22.596, 18.945, 16.588, 23.26 , 13.526, 19.815, 21.171, 21.828,
       23.54 , 21.655, 20.838, 11.909, 23.131, 23.675, 23.806, 13.856,
       24.428, 21.077, 19.66 , 31.714, 47.122, 14.669, 35.255, 22.577,
       19.013, 19.948, 13.821, 15.144, 24.965, 29.31 , 24.851, 31.327,
       13.281, 21.603, 17.07 , 21.827, 27.66 , 35.468, 27.717, 21.05 ,
       15.051, 48.388, 17.74 , 22.797, 20.769, 11.68 , 21.009,  7.853,
       15.911, 26.39 , 32.081, 10.13 , 31.513, 16.686, 34.188, 19.998,
       22.222, 22.808, 27.079, 23.174, 34.185, 14.287, 18.392, 13.97 ,
       22.168, 21.124, 10.625, 13.833, 24.045, 30.042, 20.407, 15.113,
       28.041, 17.39 , 35.581, 23.475, 22.858, 16.352, 20.566, 10.429,
       23.334, 36.866, 20.454, 16.838, 20.24 , 41.203, 24.573, 24.534,
       24.221, 18.261, 20.743, 23.03 , 21.319, 19.902, 20.883, 21.541,
       14.442, 20.183, 16.232, 21.81 , 20.196, 18.664, 30.221, 15.213,
       38.427, 27.769, 25.66 , 21.434,  8.944, 21.316, 45.96 , 23.132,
       35.186, 15.56 , 33.728, 19.874, 14.254,  8.759, 33.426, 22.419,
       20.791, 19.049, 20.973,  8.631, 25.621, 16.584,  9.153, 22.016,
       14.465,  9.388, 24.327, 19.06 , 19.125, 10.459, 24.316, 19.83 ,
       24.041, 35.633, 21.14 , 20.362, 19.899, 27.504, 20.724, 26.886,
       10.705, 32.172, 33.471, 31.956, 32.019, 33.793, 21.858, 28.095,
       33.828, 18.012, 12.673, 34.559, 16.338, 26.904, 19.565, 19.864,
       22.642, 30.6  , 23.438, 27.036, 24.194, 14.144, 19.896, 20.554,
       15.89 , 27.69 , 43.519, 21.124, 44.124, 40.277, 34.028, 15.814,
       27.625, 22.62 , 47.488, 10.862, 12.292, 23.625, 23.095,  8.864,
       24.906, 17.127, 20.863, 19.702, 17.149, 13.537, 12.759, 23.919,
       28.275, 24.291, 21.239, 23.732, 15.32 , 19.419, 31.618, 24.466,
        7.911, 24.666, 15.418, 21.729, 21.369, 12.483, 21.901, 18.654,
       22.167, 25.489, 23.119, 21.078, 18.93 , 21.937, 30.721, 23.363,
       14.169, 34.1  , 13.913, 22.275, 24.116,  8.198, 31.077, 12.847,
       21.987, 19.704])
```

In [40]:
```
r2=metrics.r2_score(y_train,y_pred)
r2
```

Out[40]:
```
0.9772451453529503
```

In [41]:
```
MAE=metrics.mean_absolute_error(y_train,y_pred)
MAE
```

Out[41]:
```
0.8565338983050853
```

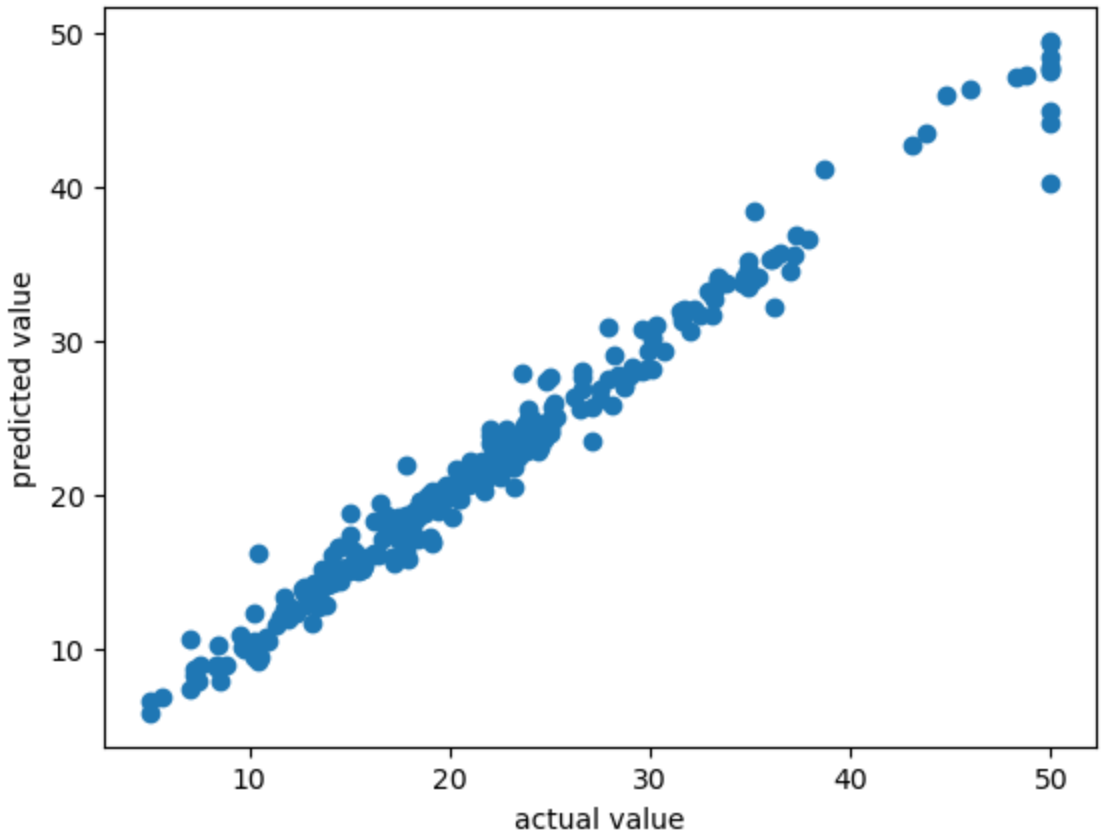
In [42]:
```
MSE=metrics.mean_squared_error(y_train,y_pred)
MSE
```

Out[42]:
```
1.7127767881355935
```

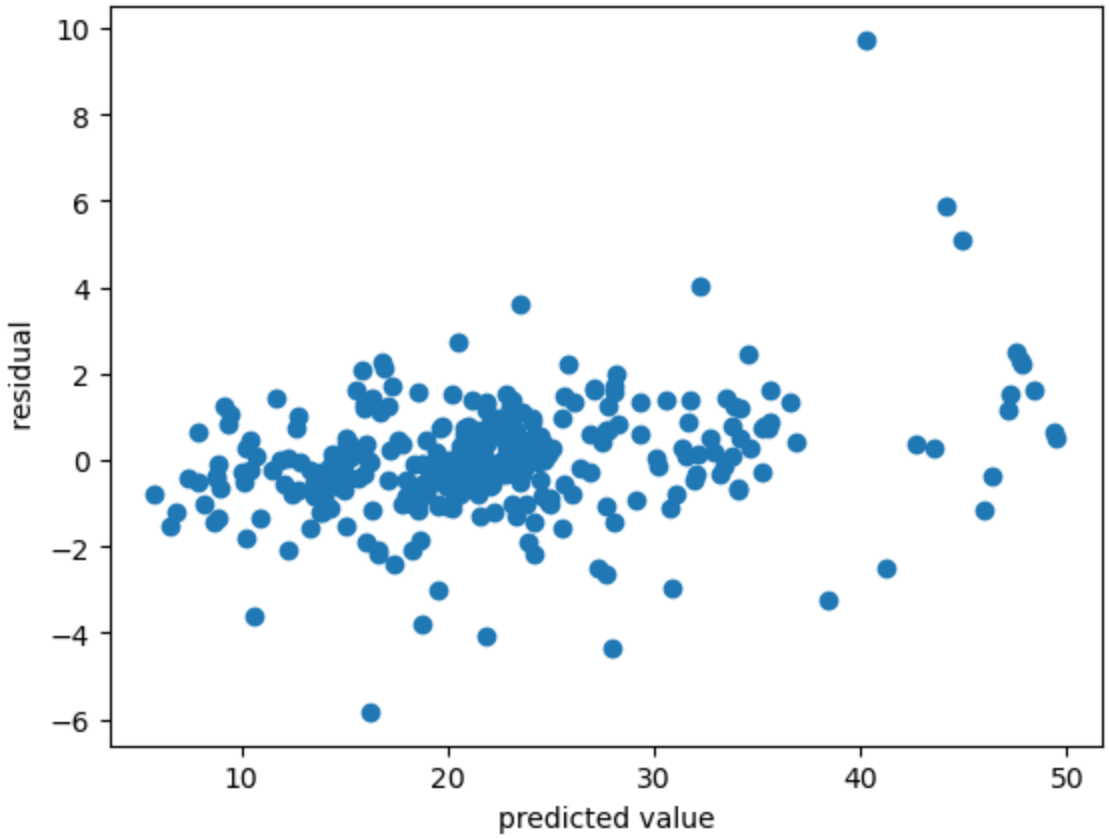
In [43]:
```
RMSE=np.sqrt(MSE)
```

Out[43]: 1.3087309838678052

In [44]:
```
plt.scatter(y_train,y_pred)
plt.xlabel("actual value")
plt.ylabel("predicted value")
plt.show()
```

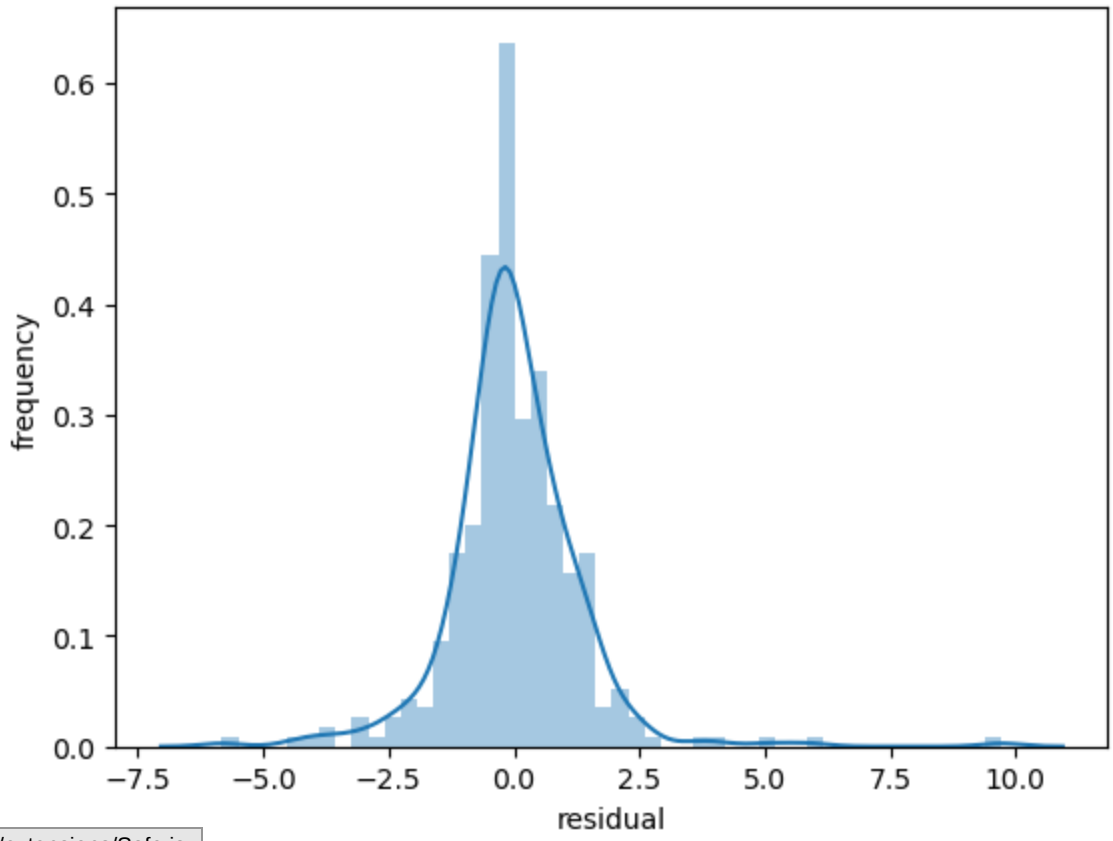


In [45]:
```
plt.scatter(y_pred,y_train-y_pred)
plt.xlabel("predicted value")
plt.ylabel(" residual")
plt.show()
```



In [46]:
```
sns.distplot(y_train-y_pred)
plt.xlabel("residual")
plt.ylabel("frequency")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [47]:
```
#now working with test data
y_test_pred=reg.predict(x_test)
y_test_pred
```

Out[47]:
```
array([17.825, 23.993, 19.304, 18.075, 46.862, 23.76 , 34.621, 18.595,
       17.43 , 16.28 , 30.172, 24.219, 20.697, 25.055, 21.643, 13.3  ,
       20.095, 12.515, 13.638, 16.031,  7.877, 15.502, 19.939, 20.44 ,
       20.431, 20.636, 17.579, 15.64 , 22.211, 19.13 , 14.541, 23.229,
       32.773, 21.905, 15.166, 13.514, 30.926, 44.691, 23.673, 23.477,
       46.391, 30.044, 12.982, 29.86 , 28.593, 20.104, 48.572, 19.705,
       21.398, 23.046, 30.653, 22.79 , 12.936, 27.286, 14.909, 20.813,
       25.035, 32.054, 20.165, 29.589, 17.475, 21.131, 26.444, 20.565,
       45.027, 27.375, 29.216,  8.307, 19.115, 21.656, 21.854, 20.485,
       26.062, 26.613, 16.939, 33.696, 16.061, 23.351, 14.719, 21.878,
       19.827, 16.85 , 26.564, 20.671, 24.589, 20.531, 33.259, 20.555,
       20.836, 47.16 , 27.075, 48.4  , 19.652, 47.338, 21.02 , 21.394,
       21.304, 30.541, 14.901, 19.665,  8.874, 21.143, 35.068, 15.335,
       24.386, 20.307, 38.239, 32.226, 45.611, 22.657, 22.564, 22.255,
       18.73 , 29.795, 35.977, 24.972, 48.421, 15.023, 11.82 , 19.022,
       21.357, 13.124, 23.445, 22.51 , 19.188, 18.338, 48.266, 13.776,
       14.751, 10.453, 46.584, 17.213, 47.276, 26.051, 41.253, 19.917,
       10.815, 23.304, 21.013, 19.977, 14.971, 11.515, 22.303, 29.516,
       11.824, 19.646, 14.94 , 15.076, 40.41 , 19.593, 26.743, 14.469])
```

In [48]:
```
r2=metrics.r2_score(y_test,y_test_pred)
r2
```

Out[48]:
```
0.8390713230537836
```

In [49]:
```
MAE=metrics.mean_absolute_error(y_test,y_test_pred)
MAE
```

Out[49]:
```
2.4851710526315776
```

In [50]:
```
MSE=metrics.mean_squared_error(y_test,y_test_pred)
MSE
```

Out[50]:
```
16.804184078947358
```

In [51]:
```
RMSE=np.sqrt(MAE)
RMSE
```

Out[51]:
```
1.5764425307100725
```

Data preprocessing

In [52]:
```
# creating scaled set to be used in model to improve our results
from sklearn.preprocessing  import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [53]:
```
x_test
```

Out[53]:
```
array([[-3.68998988e-01, -1.38677536e-04, -5.09555940e-01, ...,
        -1.48812077e+00,  3.26142192e-01,  2.42391983e+00],
       [-3.90301961e-01,  1.68618016e+00, -8.84134205e-01, ...,
        -8.32891927e-01,  1.62149582e-01, -3.96062600e-01],
       [-3.78582604e-01, -1.38677536e-04, -7.73964127e-01, ...,
         2.43555458e-01,  4.39100214e-01,  9.83319688e-02],
       ...,
       [-3.91892056e-01, -5.27113314e-01, -9.03230352e-01, ...,
         8.51982241e-01,  4.16112652e-01, -2.51058606e-01],
       [-3.92244004e-01,  2.63473451e+00, -1.23227165e+00, ...,
        -3.72569032e-02,  4.25131695e-01, -1.11279663e+00],
       [ 8.99755257e-01, -5.27113314e-01,  9.93163927e-01, ...,
         8.51982241e-01,  2.49920275e-01,  7.80541234e-01]])
```

In [54]:
```
x_train
```

Out[54]:
```
array([[-0.42546852, -0.47076769, -0.95468627, ...,  0.00545961,
         0.44188904, -0.44481854],
       [-0.42632345,  2.99257588, -1.3301574 , ...,  1.61604587,
         0.28749838, -0.66643827],
       [-0.3851898 , -0.47076769, -0.7058275 , ..., -0.50072464,
         0.42371255,  1.22650481],
       ...,
       [ 0.62797244, -0.47076769,  1.02599668, ...,  0.78774436,
         0.44188904,  1.42695069],
       [-0.42164728, -0.47076769, -1.01872011, ..., -0.86885864,
         0.40170367, -0.44199536],
       [-0.42068632, -0.47076769,  2.12475908, ...,  0.28156011,
         0.23592534,  0.72821331]])
```

Support Vector Machine(SVM)

In [55]:
```
from sklearn import svm
reg=svm.SVR()
```

In [56]:
```
reg.fit(x_train,y_train)
```

Out[56]:
```
SVR()
```

In [57]:
```
y_pred=reg.predict(x_train)
y_pred
```

Out[57]:
```
array([23.86494587, 19.91150003, 21.79984564, 15.53703846, 34.02546519,
       19.73274461, 32.59747715, 15.05468902, 21.60482169, 25.19238108,
       31.48859251, 13.08228021, 16.41729745, 21.64028355, 23.76190034,
       23.77252256, 18.59385973, 22.01383969, 22.88860472, 20.13662466,
       20.64184005, 17.09053974, 16.90642061, 15.33794008, 24.16241418,
       14.28372759, 17.65782368, 27.48317144, 20.10211034, 13.68149878,
       19.46967903, 22.23091192, 23.26439445, 20.13515571, 23.66286943,
       17.50018909, 16.54403764, 11.89917835, 12.14399337, 20.76080675,
       18.18637823, 21.11277622, 29.23114073, 17.56937427, 12.376143  ,
       16.46012426, 27.23678311, 16.75508625, 11.35753045, 24.48843701,
       18.19126147, 10.87392193, 25.61938169, 22.1395693 , 24.69142731,
       18.30433557, 16.29196466, 24.21367071, 24.90000428, 15.57141036,
       11.77427961, 23.41515601, 26.654883  , 28.54717974, 14.20061989,
       18.59325751, 19.41360915, 16.63762815, 22.4242584 , 20.84941386,
       20.29570168, 32.63524812, 16.50696207, 19.36248949, 14.25167524,
       19.10795583, 21.72777363, 29.88673115, 33.37798433, 31.27621416,
       21.23854887, 20.80083115, 11.83584582, 12.36257318, 14.21441524,
       21.09718856, 18.81566315, 20.09265809, 16.97601944, 20.82736078,
       18.49819229, 21.48736174, 31.38816722, 28.57110316, 13.47547731,
       20.89180465, 24.83248813, 11.12386744, 19.60307062, 23.98761441,
       20.50668307, 23.40044248, 25.03858398, 17.94966259, 24.25338601,
       30.29024084, 11.63392286, 17.52007529, 14.85943594, 13.43271633,
       15.95458316, 16.30013802, 21.8080845 , 20.64692992, 26.59193485,
       22.47585589, 22.71995114, 24.6391301 , 24.99037878, 29.53124203,
       18.45942393, 22.3381924 , 20.36756896, 19.30520298, 22.85290451,
       25.6454602 , 26.93461363, 18.70097213, 22.50029345, 18.31553376,
       16.24834858, 23.58405913, 14.31109092, 15.82737673, 21.85812671,
       21.5996653 , 22.66144077, 24.89836223, 20.82509193, 13.43175934,
       21.13991481, 23.89966464, 25.16733485, 13.50947972, 24.36229341,
       21.41937549, 19.07227658, 26.55184229, 33.61439507, 15.44822093,
       31.52651574, 25.03748178, 19.52599336, 20.88619418, 15.77641173,
       16.487946  , 22.97284293, 29.33920264, 24.88405844, 26.39704423,
       14.77167745, 20.19994982, 16.50003885, 22.93504283, 25.16849214,
       28.22474797, 27.16987872, 22.11508679, 15.10494153, 25.10387082,
       17.34821321, 23.319561  , 21.79540899, 12.57185771, 21.72732621,
       12.30749673, 19.84792763, 25.14844399, 27.7390757 , 12.87762247,
       30.21109028, 20.43575875, 29.84847797, 20.18867826, 20.95938896,
       21.75564026, 23.91080311, 23.51439051, 30.6674109 , 14.85745293,
       18.58806113, 17.46272502, 18.13824118, 20.70017674, 14.81192738,
       14.17810526, 24.13838739, 29.85312269, 19.52103169, 16.93715326,
       26.74669992, 20.35284121, 30.02208671, 19.94247452, 23.22547195,
       16.54220018, 22.17980247, 12.81414793, 23.61234116, 28.62518572,
       15.84660484, 14.79975619, 19.868356  , 32.66998401, 23.14320565,
       23.3048527 , 25.9370605 , 18.89637235, 21.329798  , 21.58624355,
       20.4517428 , 19.59608889, 20.99879281, 17.80606628, 14.49976665,
       19.62971535, 16.29966523, 21.24149796, 21.69452598, 17.4915544 ,
       23.66472941, 16.27504779, 27.33343721, 25.84861288, 25.19997688,
       20.85855682, 12.11774105, 21.12682134, 33.41397923, 21.01708139,
       29.01112439, 12.25228269, 31.59289807, 18.85377603, 15.09797243,
       12.07066822, 28.58926286, 22.34813058, 19.98689063, 19.57055787,
       21.21623477, 13.89255242, 25.25071798, 17.61892075, 11.20342646,
       22.96829038, 13.84967518, 10.90870356, 24.11407821, 20.41115752,
       19.47412239, 14.15726925, 24.44555409, 21.21994354, 22.31560735,
       31.53054937, 21.71716553, 18.98563999, 20.4094258 , 26.46175156,
       21.33540113, 26.13625917, 10.69968962, 23.88627933, 26.74837361,
       31.45201864, 31.60010261, 28.33129008, 18.68115506, 22.22195013,
       30.42893591, 19.88856809, 12.07276735, 28.70348311, 16.77772934,
       25.70394121, 22.55687413, 19.50006928, 23.33747601, 29.43809955,
       23.50033421, 26.03497443, 26.9052406 , 16.50705934, 20.48887967,
       21.23551067, 14.76790069, 26.6581981 , 32.81395789, 17.56476564,
       19.40458592, 23.64325535, 27.9469287 , 15.32011701, 24.02303847,
       23.81405951, 29.6008712 , 12.10350945, 14.93042436, 25.08939579,
       23.32021226, 12.93986893, 26.50287392, 16.32119191, 21.6279404 ,
       17.76703205, 16.63593183, 14.70742973, 12.02183633, 23.30365569,
```

```
27.31955115, 23.60581758, 15.85779496, 24.1999247 , 17.72635677,
17.52205129, 25.86067246, 28.29416261, 10.64569552, 25.88578233,
16.815693  , 22.28670257, 21.96083215, 13.27805994, 22.03767202,
19.24911363, 22.89711826, 24.20874421, 24.02628705, 22.57570002,
18.46001093, 18.76319913, 25.70896754, 22.075379  , 16.90614765,
31.67199473, 13.54878801, 22.86931247, 20.71947117, 10.40041783,
27.49341131, 12.69969014, 22.49975134, 18.99701009])
```

In [58]:
```
#claculation of metrics
r2=metrics.r2_score(y_train,y_pred)
r2
```

Out[58]:
```
0.6419097248941197
```

In [59]:
```
MSE=metrics.mean_squared_error(y_train,y_pred)
MSE
```

Out[59]:
```
26.95375210133292
```

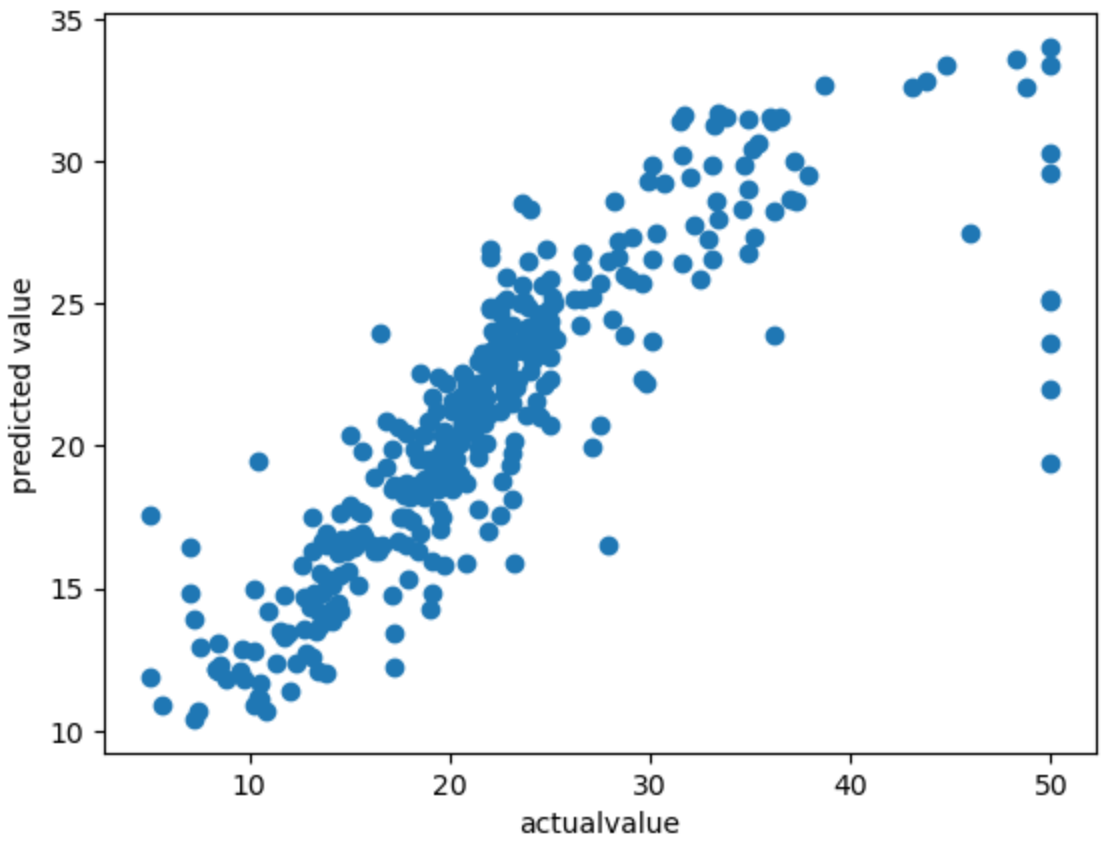
In [60]:
```
MAE=metrics.mean_absolute_error(y_train,y_pred)
MAE
```

Out[60]:
```
2.9361501059460284
```

In [61]:
```
RMSE=np.sqrt(MSE)
RMSE
```

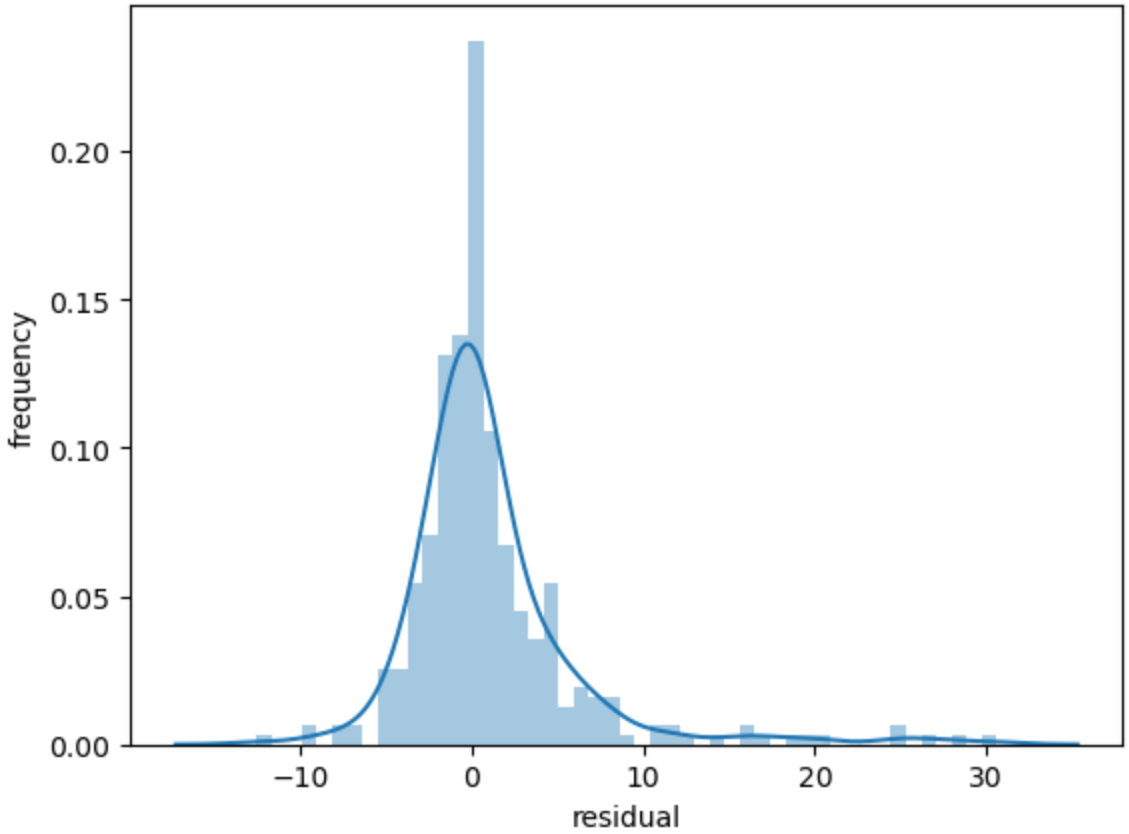
Out[61]:
```
5.191700309275654
```

In [62]:
```
plt.scatter(y_train,y_pred)
plt.xlabel("actualvalue")
plt.ylabel("predicted value")
plt.show()
```

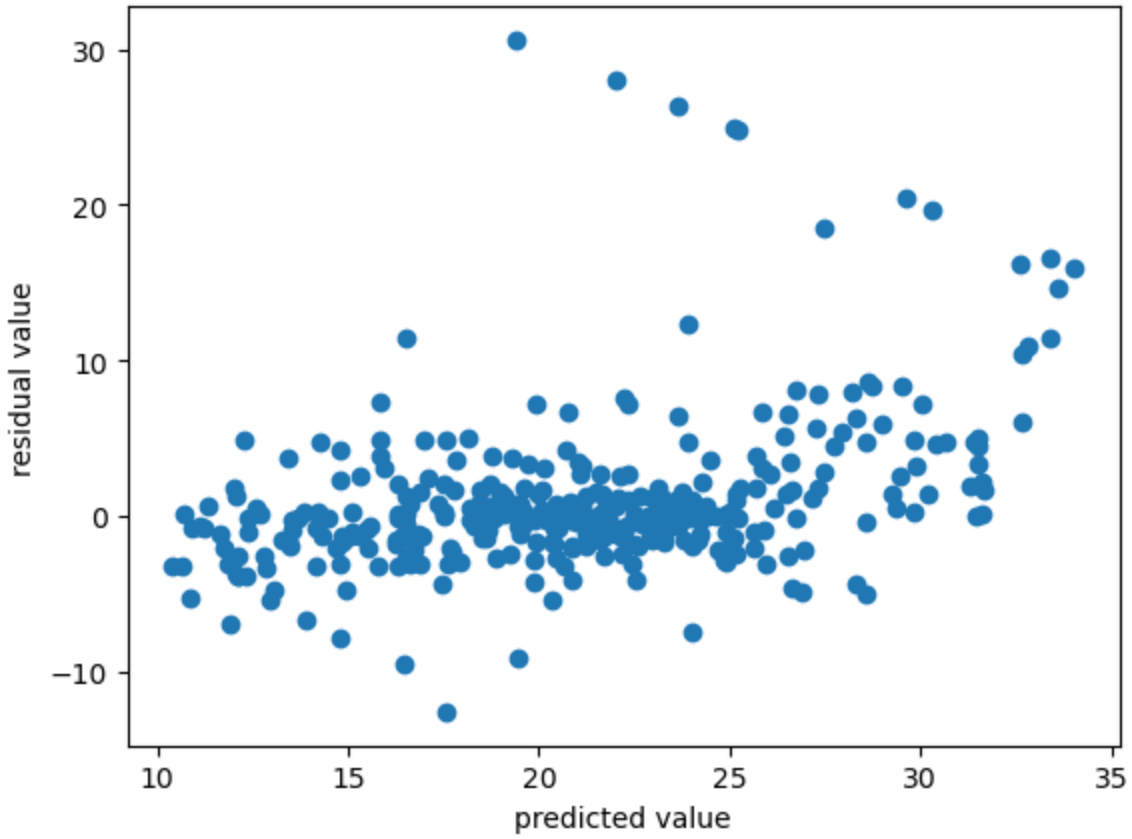


In [63]:
```
sns.distplot(y_train-y_pred)
plt.xlabel("residual")
plt.ylabel("frequency")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [64]:
```
plt.scatter(y_pred,y_train-y_pred)
plt.xlabel("predicted value")
plt.ylabel("residual value")
plt.show()
```



In [65]:
```
#now working with test data
y_test_pred=reg.predict(x_test)
y_test_pred
```

Out[65]:
```
array([17.04500847, 23.87993003, 18.38002707, 18.70352441, 32.0934929 ,
       23.51108167, 26.10873071, 17.92058434, 13.56439277, 20.30356687,
       26.48894957, 24.58189505, 18.88581113, 23.8535099 , 20.36155774,
       13.31471007, 20.14084903, 11.63356988, 12.76122624, 15.42825813,
       20.08902677, 18.35170198, 18.62145241, 19.55215349, 19.10010228,
       18.36876361, 18.53639126, 15.67384465, 16.22827102, 16.41954528,
       12.57790157, 21.66994193, 29.91257329, 20.05550542, 15.22871987,
       14.86918608, 28.16736813, 31.16202218, 21.70042003, 22.43828736,
       29.57320624, 26.85682272, 15.17823935, 26.93297506, 25.82814616,
       21.12533168, 31.06856969, 17.70267494, 17.85812323, 21.90123504,
       26.58700712, 23.0358111 , 13.97073784, 25.07125876, 15.16383393,
       20.64951089, 21.91556124, 28.53772131, 18.52860006, 27.42064307,
       16.68530578, 17.21503994, 26.68058558, 18.21757828, 32.72847644,
       24.29683303, 23.07803063, 12.38108139, 18.33064062, 21.62294699,
       21.5453404 , 21.07518346, 22.75167466, 24.99837571, 17.04764231,
       23.52418709, 16.24665646, 22.57125794, 16.16224666, 18.34614795,
       20.15448683, 16.62691162, 25.71088171, 18.36986183, 25.90533028,
       19.75915946, 28.02423458, 18.31020625, 21.47493903, 33.28966302,
       26.63316141, 30.26357206, 17.76291011, 33.06745163, 19.81233108,
       20.45368848, 22.26478752, 21.02526106, 16.71429568, 19.64886383,
       12.28099107, 22.64623544, 28.53617435, 15.12515154, 23.48748208,
       19.28763813, 29.08057038, 26.51507296, 30.49444034, 24.13477671,
       23.63831395, 20.14467401, 18.88795474, 26.15790323, 28.41616187,
       18.99200536, 27.12948213, 15.28351112, 14.4982336 , 17.71976763,
       20.27434483, 13.93455142, 24.8461797 , 25.09391927, 17.8682372 ,
       17.1574836 , 29.17115349, 13.27545853, 15.29374243, 14.68835765,
       32.85726433, 16.21714326, 32.83107012, 26.79098839, 27.44727485,
       20.37775246, 12.07236702, 23.25405788, 19.79626816, 17.63679427,
       18.69765438, 12.98538202, 21.2238682 , 17.82014373, 13.12359452,
       19.34684282, 15.41481305, 14.79201666, 23.17541267, 18.88952861,
       25.89254973, 15.14006361])
```

In [66]:
```
r2=metrics.r2_score(y_test,y_test_pred)
r2
```

Out[66]:
```
0.5730701114054623
```

In [67]:
```
MAE=metrics.mean_absolute_error(y_test,y_test_pred)
MAE
```

Out[67]:
```
3.937604384114756
```

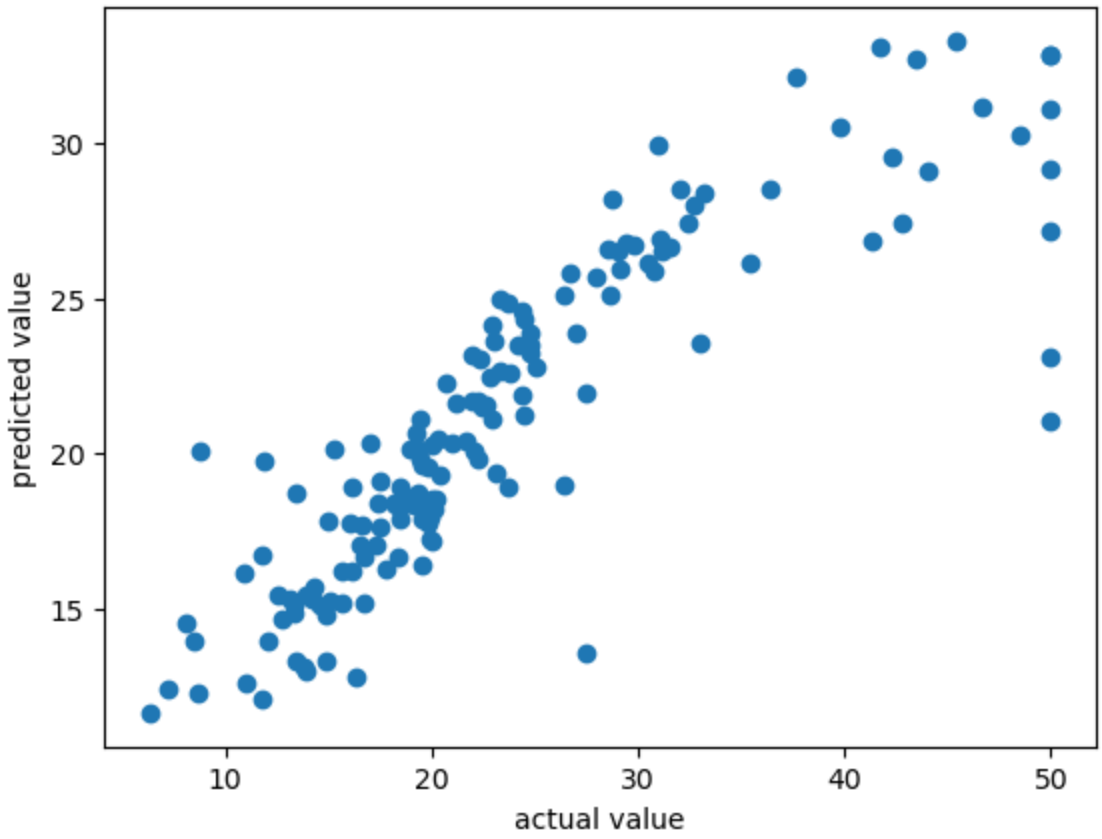
In [68]:
```
MSE=metrics.mean_squared_error(y_test,y_test_pred)
MSE
```

Out[68]:
```
44.580049826326345
```

In [69]:
```
RMSE=np.sqrt(MSE)
RMSE
```

Out[69]:
```
6.67682932433699
```

In [70]:
```
plt.scatter(y_test,y_test_pred)
plt.xlabel("actual value")
plt.ylabel("predicted value")
plt.show()
```



XGBoost Regressor

In [71]:
```
#XGBoost regressor
from xgboost import XGBRegressor
reg=XGBRegressor()
reg.fit(x_train,y_train)
```

Out[71]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

In [72]:
```
y_pred=reg.predict(x_train)
y_pred
```

Out[72]:
```
array([23.898668 , 18.202156 , 21.698345 , 13.493743 , 49.992077 ,
       23.10208  , 48.794533 , 13.817799 , 20.102333 , 50.00149  ,
       34.924053 ,  8.39999  , 15.187474 , 22.976278 , 24.694382 ,
       25.28971  , 17.211199 , 50.002148 , 22.878271 , 20.199478 ,
       17.412027 , 19.499348 , 18.501293 , 14.001987 , 22.610323 ,
       14.109741 , 15.607283 , 46.006325 , 20.496004 , 13.502242 ,
       10.402703 , 21.431185 , 21.598164 , 23.185183 , 23.025291 ,
       17.60395  , 16.11072  ,  5.00106  ,  8.298136 , 27.49878  ,
       18.691305 , 21.697563 , 30.692205 ,  5.008037 , 11.308817 ,
        7.000945 , 32.904667 , 14.601417 , 11.990232 , 28.090174 ,
       17.998945 ,  5.5982914, 23.606403 , 24.698915 , 22.4925   ,
       17.7013   , 13.096569 , 23.114618 , 25.0129   , 14.903409 ,
        9.713751 , 22.810575 , 22.009037 , 23.61215  , 14.306128 ,
       18.795181 , 19.897982 , 13.618455 , 19.405636 , 16.81838  ,
       20.00098  , 43.11957  , 27.88593  , 20.115192 , 18.981453 ,
       19.216972 , 21.709469 , 33.10091  , 49.99492  , 33.200787 ,
       20.118906 , 21.10423  ,  8.802685 , 12.2546835, 14.4984255,
       23.788445 , 18.701365 , 21.803246 , 21.895859 , 21.698557 ,
       17.102743 , 23.09697  , 36.097076 , 28.195421 , 11.524114 ,
       19.017113 , 22.012339 , 10.486594 , 21.41922  , 16.49911  ,
       20.59537  , 23.30626  , 23.50774  , 15.000558 , 26.490255 ,
       50.000847 , 10.492561 , 17.519302 , 13.59443  , 17.19163  ,
       19.098984 , 16.397259 , 20.590748 , 20.906923 , 30.069551 ,
       20.705257 , 22.199791 , 24.594826 , 25.205332 , 37.89925  ,
       20.085173 , 29.596458 , 18.694838 , 22.986837 , 22.894016 ,
       24.586397 , 24.80737  , 20.80593  , 22.403542 , 18.20351  ,
       14.401529 , 23.185047 , 13.000709 , 19.696535 , 21.17987  ,
       21.703268 , 23.983181 , 22.002363 , 20.60496  , 11.898764 ,
       24.276354 , 23.779114 , 22.80477  , 13.328387 , 24.9945   ,
       20.989483 , 20.401754 , 33.09654  , 48.301083 , 14.49227  ,
       36.00746  , 22.592964 , 18.392752 , 18.927902 , 12.6190405,
       15.213818 , 24.094107 , 29.901505 , 23.910872 , 31.594862 ,
       11.701875 , 20.29879  , 16.601126 , 22.176989 , 26.601843 ,
       36.191845 , 28.413687 , 20.82073  , 15.398618 , 49.999863 ,
       18.09718  , 23.073902 , 21.493528 , 13.083476 , 21.795036 ,
        8.502242 , 15.604793 , 26.208536 , 32.198586 ,  9.606724 ,
       31.602957 , 17.798496 , 34.697876 , 19.993362 , 21.002974 ,
       22.694841 , 28.680223 , 23.875227 , 35.41479  , 13.195639 ,
       18.289051 , 13.100803 , 23.106459 , 20.59922  ,  7.000341 ,
       13.384457 , 24.104977 , 30.103985 , 20.304886 , 15.612457 ,
       26.613838 , 15.00681  , 37.20695  , 27.093132 , 24.397884 ,
       17.802233 , 19.806568 , 10.210543 , 23.113098 , 37.294025 ,
       23.170357 , 19.073616 , 19.654306 , 38.7006   , 25.031815 ,
       23.711258 , 22.794123 , 16.200487 , 20.325508 , 24.296154 ,
       21.201744 , 19.326782 , 20.595406 , 21.384968 , 14.406331 ,
       19.91124  , 16.19812  , 22.480814 , 19.128166 , 17.818674 ,
       30.0999   , 14.803343 , 35.199852 , 29.001917 , 25.095686 ,
       21.505173 ,  8.2950325, 21.97281  , 44.80051  , 24.48276  ,
       34.89165  , 17.199    , 33.80461  , 19.603312 , 14.0736685,
        8.423438 , 33.316612 , 23.398811 , 21.405186 , 18.891907 ,
       21.190443 ,  7.2160635, 27.092018 , 14.507033 , 10.389338 ,
       21.398642 , 14.091925 , 10.196625 , 24.285624 , 18.603714 ,
       18.90239  , 10.909892 , 24.393059 , 19.293148 , 24.998196 ,
       36.489086 , 20.514498 , 20.396557 , 19.599285 , 27.896692 ,
       21.098457 , 26.594328 , 10.784633 , 36.180676 , 34.88086  ,
       31.495077 , 31.708479 , 34.577793 , 17.793695 , 29.80721  ,
       35.100258 , 17.095772 , 13.396441 , 36.98654  , 15.213398 ,
       27.510576 , 18.50712  , 19.58253  , 23.203495 , 31.976273 ,
       23.401077 , 28.692957 , 21.999323 , 13.794538 , 19.694435 ,
       20.905571 , 17.09005  , 28.394444 , 43.800564 , 22.482336 ,
       50.00451  , 49.99332  , 33.421295 , 17.89427  , 25.002327 ,
       22.3129   , 50.00089  ,  9.503665 , 10.206892 , 23.712202 ,
       23.793251 ,  7.500893 , 23.905233 , 18.388336 , 20.41871  ,
       19.397469 , 17.395752 , 12.699183 , 13.792526 , 22.00778  ,
```

```
29.095022 , 24.695938 , 20.797657 , 24.094175 , 15.395751 ,
19.554085 , 32.495182 , 24.008078 ,  7.401533 , 25.019457 ,
15.699164 , 21.704689 , 21.206905 , 11.691455 , 22.68644  ,
16.846464 , 21.598713 , 23.889149 , 22.111568 , 20.584314 ,
19.391888 , 22.591293 , 29.591736 , 23.302452 , 13.795169 ,
33.40695  , 12.712214 , 22.213974 , 25.005445 ,  7.2029343,
30.30505  , 12.809058 , 22.581894 , 20.50012  ], dtype=float32)
```

In [73]:
```
r2=metrics.r2_score(y_train,y_pred)
r2
```

Out[73]:
```
0.9999980912185324
```

In [74]:
```
MAE=metrics.mean_absolute_error(y_train,y_pred)
MAE
```

Out[74]:
```
0.008653184923075066
```

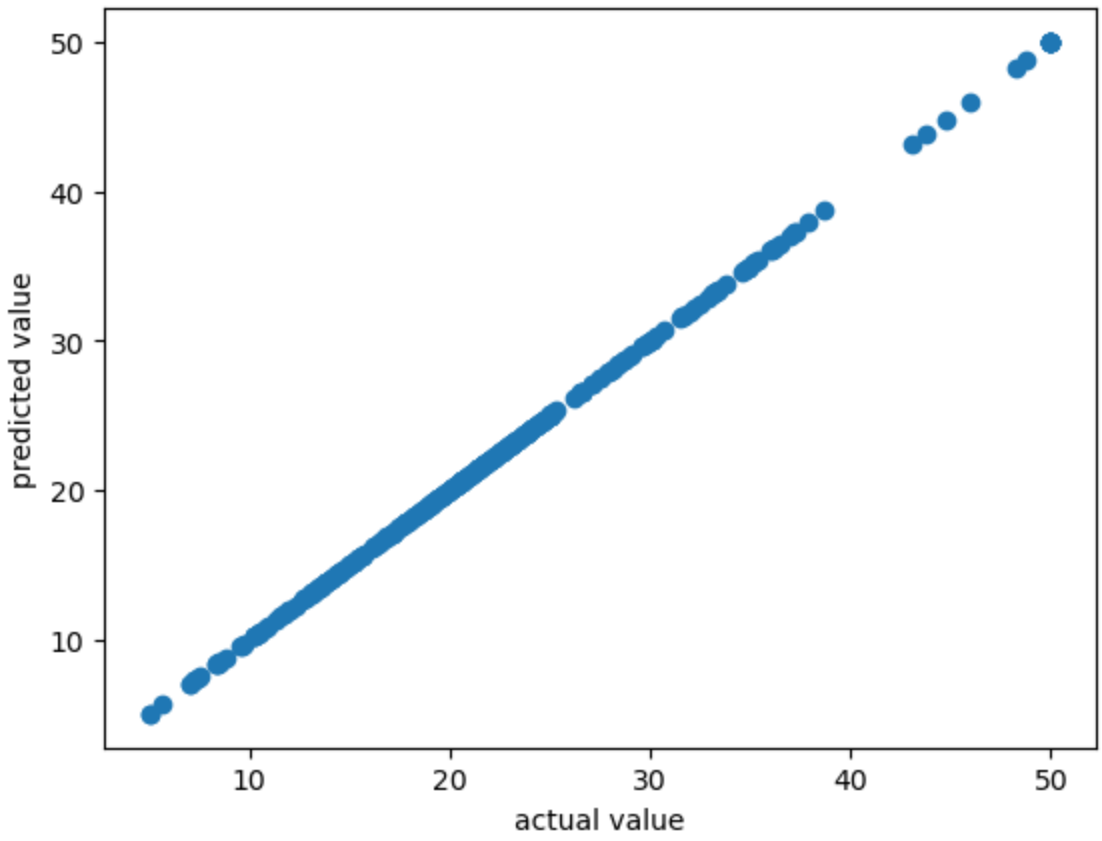
In [75]:
```
MSE=metrics.mean_squared_error(y_train,y_pred)
MSE
```

Out[75]:
```
0.00014367556470779537
```

In [76]:
```
RMSE=np.sqrt(MSE)
RMSE
```

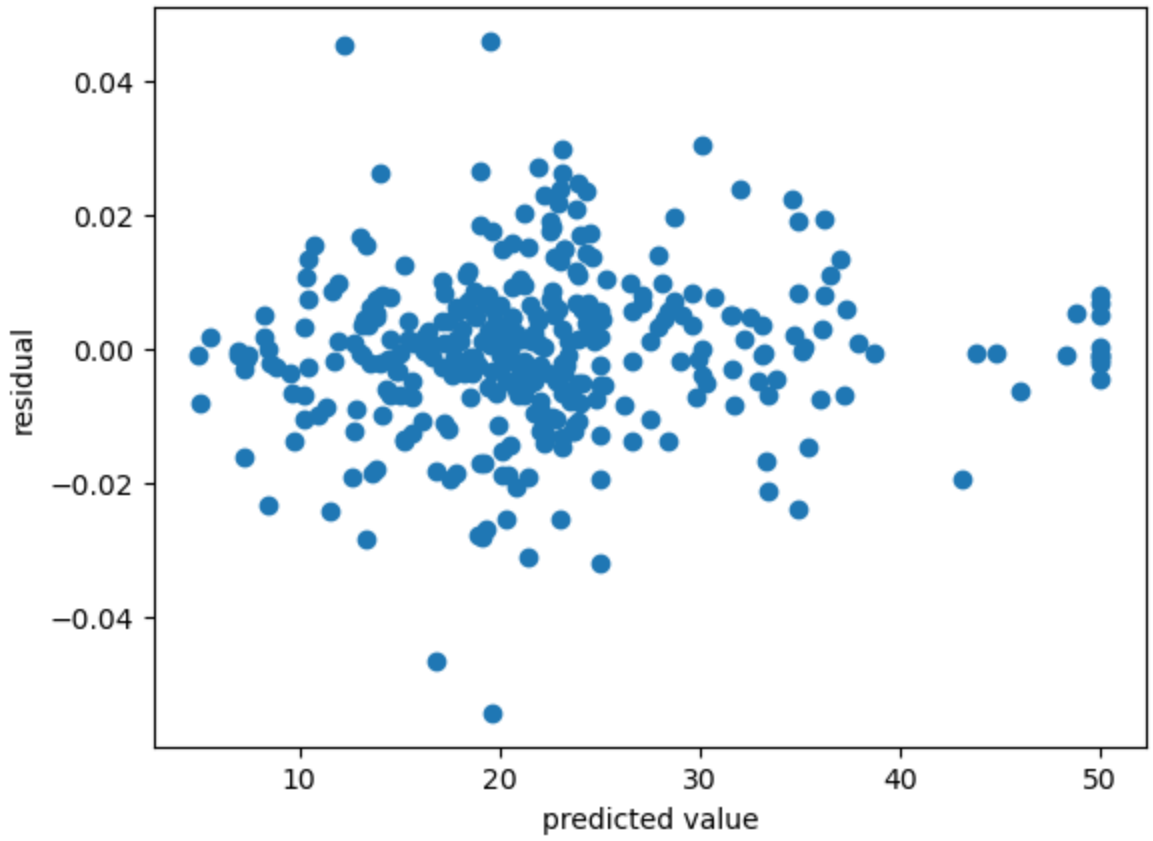
Out[76]:
```
0.011986474240067234
```

In [77]:
```
plt.scatter(y_train,y_pred)
plt.xlabel("actual value")
plt.ylabel("predicted value")
plt.show()
```



In [78]:
```
plt.scatter(y_pred,y_train-y_pred)
plt.ylabel("predicted value")
```

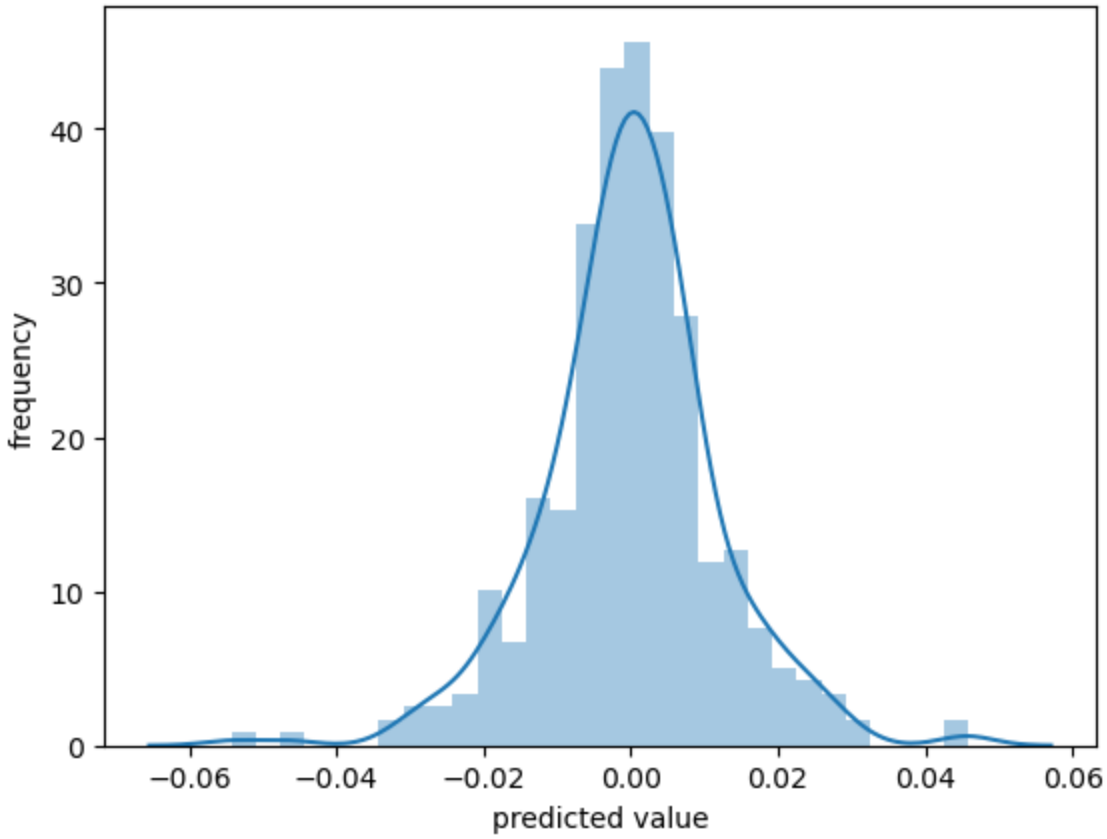
```
plt.ylabel("residual")
plt.show()
```



In [79]:

```
sns.distplot(y_train-y_pred)
plt.xlabel("predicted value")
plt.ylabel("frequency")
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [80]:
```
#working with test data
y_test_pred=reg.predict(x_test)
y_test_pred
```

Out[80]:
```
array([15.863819 , 21.67604  , 19.235802 , 14.842502 , 46.311783 ,
       23.694664 , 34.513798 , 20.381325 , 13.39046  , 17.792131 ,
       25.545677 , 24.869177 , 22.354187 , 27.517454 , 21.106268 ,
       11.673415 , 19.347652 , 10.006418 ,  8.781274 , 14.948037 ,
        5.567042 , 16.759457 , 18.944693 , 20.557404 , 20.38494  ,
       20.373217 , 13.453175 , 16.241825 , 18.17131  , 18.36038  ,
       11.087232 , 22.981165 , 30.13889  , 21.993643 , 14.323056 ,
       13.959554 , 27.066145 , 34.77947  , 22.86305  , 24.035765 ,
       34.68355  , 26.407751 , 12.918008 , 26.142246 , 24.85836  ,
       19.334013 , 46.63722  , 21.441216 , 20.02646  , 21.434977 ,
       26.892422 , 21.971079 , 10.27197  , 24.847075 , 16.250822 ,
       19.755371 , 23.41665  , 25.596136 , 19.34093  , 28.298368 ,
       17.698952 , 19.234072 , 23.762753 , 20.439041 , 38.327583 ,
       23.759893 , 41.25209  ,  7.2632017, 18.498402 , 21.273682 ,
       21.099424 , 20.54484  , 21.03961  , 22.556599 , 15.1354475,
       26.437487 , 16.240515 , 22.300533 , 12.743763 , 18.149197 ,
       19.730423 , 17.835678 , 25.767576 , 18.449648 , 23.652304 ,
       20.77714  , 31.997715 , 19.742804 , 19.642832 , 45.90525  ,
       25.223652 , 46.486732 , 19.290497 , 46.114693 , 18.810596 ,
       18.287647 , 20.58872  , 45.27108  , 23.171047 , 18.274876 ,
        8.7964945, 20.891    , 30.115185 , 15.223908 , 21.697859 ,
       19.962133 , 37.167316 , 26.625395 , 34.688057 , 22.81598  ,
       22.997597 , 20.473232 , 16.35227  , 24.265656 , 29.662632 ,
       24.888407 , 50.687588 , 15.847447 , 12.302824 , 18.323452 ,
       22.155785 , 12.620952 , 24.211836 , 24.101448 , 18.256004 ,
       16.321823 , 48.29596  , 14.282765 , 15.701176 , 12.916605 ,
       45.72868  , 15.729728 , 47.767303 , 25.6033   , 35.49262  ,
       18.046848 , 10.012141 , 24.141258 , 19.84744  , 20.45557  ,
       24.725069 , 12.077423 , 20.80813  , 19.162853 , 14.369612 ,
       20.908848 , 13.432678 , 17.826756 , 36.41659  , 18.235542 ,
       24.54224  , 16.305426 ], dtype=float32)
```

In [81]:
```
#now working with metrics of test data
r2=metrics.r2_score(y_test,y_test_pred)
r2
```

Out[81]:
```
0.8342668340646416
```

In [82]:
```
MAE=metrics.mean_absolute_error(y_test,y_test_pred)
MAE
```

Out[82]:
```
2.880474171513005
```

In [83]:
```
MSE=metrics.mean_squared_error(y_test,y_test_pred)
MSE
```

Out[83]:
```
17.305869166470945
```

In [84]:
```
RMSE=np.sqrt(MSE)
RMSE
```

Out[84]:
```
4.160032351613499
```

In [85]:
```
# evaluation and comparison of all models

models=pd.DataFrame({"model": ["Lnear regression","Random forest regressor","XGBoost","S
models.sort_values(by="score",ascending=False)
```

Out[85]:

	model	score
2	XGBoost	0.83
1	Random forest regressor	0.82
0	Lnear regression	0.71
3	SVM	0.57

Hence XGBoost Regression works the best for this dataset.