In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
df=pd.read_csv("boston.csv")
```

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

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.9

In [7]:
```
df.columns
```

Out[7]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

Attribute Information

Input features in order:

CRIM: per capita crime rate by town

ZN: proportion of residential land zoned for lots over 25,000 square feet.

INDUS: proportion of non-retail business acres per town

CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX: nitric oxides concentration (parts per 10 million) [parts/10M]

RM: average number of rooms per dwelling

AGE: proportion of owner-occupied units built prior to 1940

DIS: weighted distances to five Boston employment centers

RAD: index of accessibility to radial highways

TAX: full-value property-tax rate per 10,000[/10k]

PTRATIO: pupil-teacher ratio by town

B: The result of the equation B=1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT: % lower status of the population

Output variable:

MEDV: Median value of owner-occupied homes in $1000s

In [8]:
```
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    int64  
 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    int64  
 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  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
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
MEDV       0
dtype: int64
```

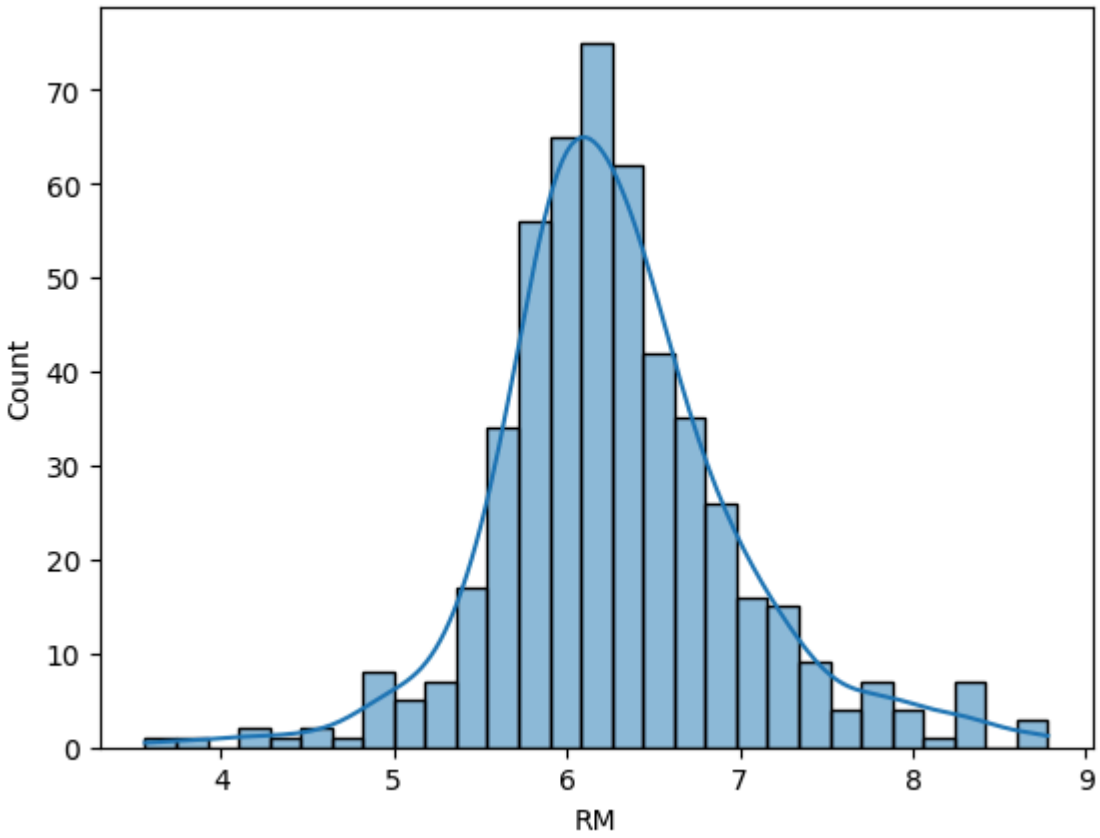
In [13]:
```
df.describe()
```

Out[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	A
count	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	68.574(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000(

In [17]:
```
sns.histplot(data=df,x='RM',kde=True)
```

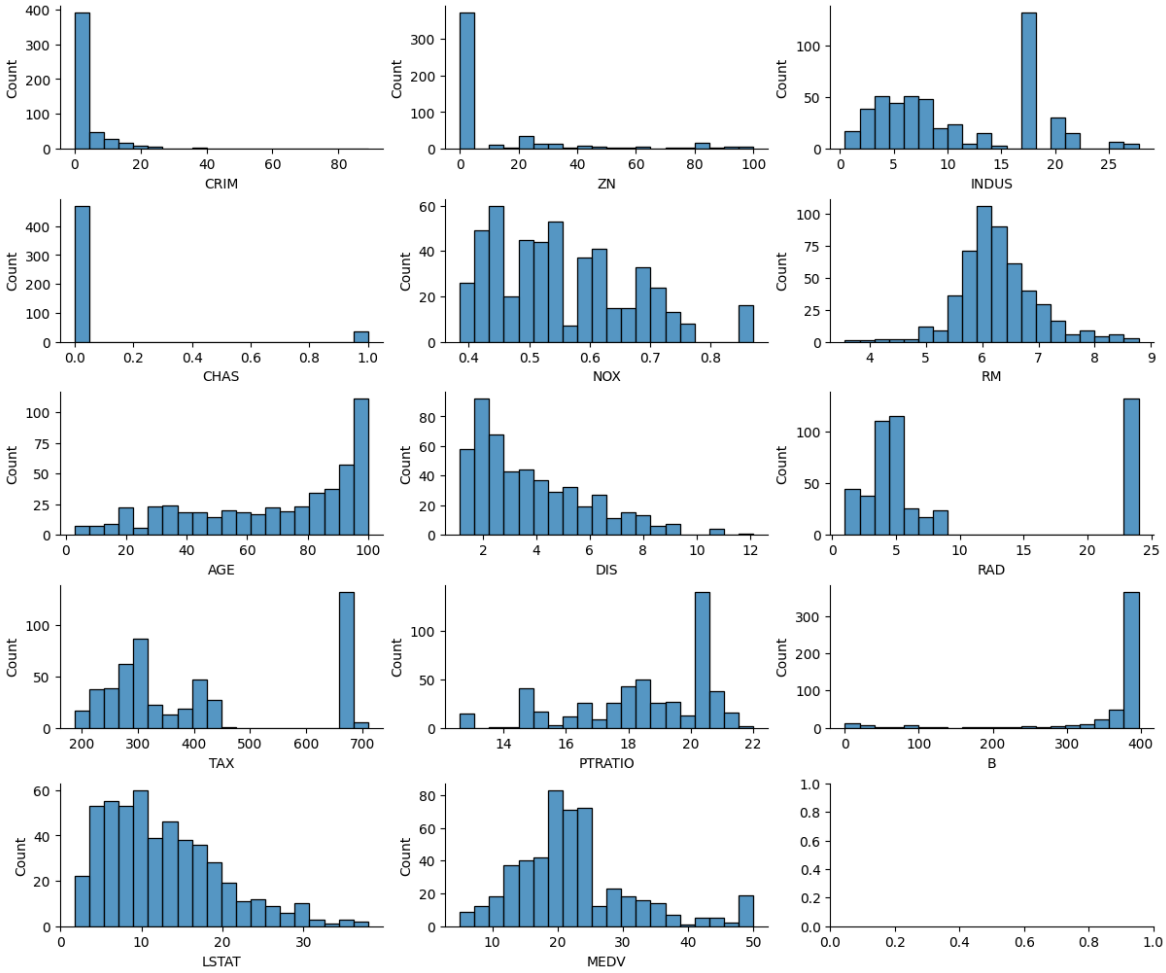
Out[17]:
```
<Axes: xlabel='RM', ylabel='Count'>
```



Histogram of Data Distribution

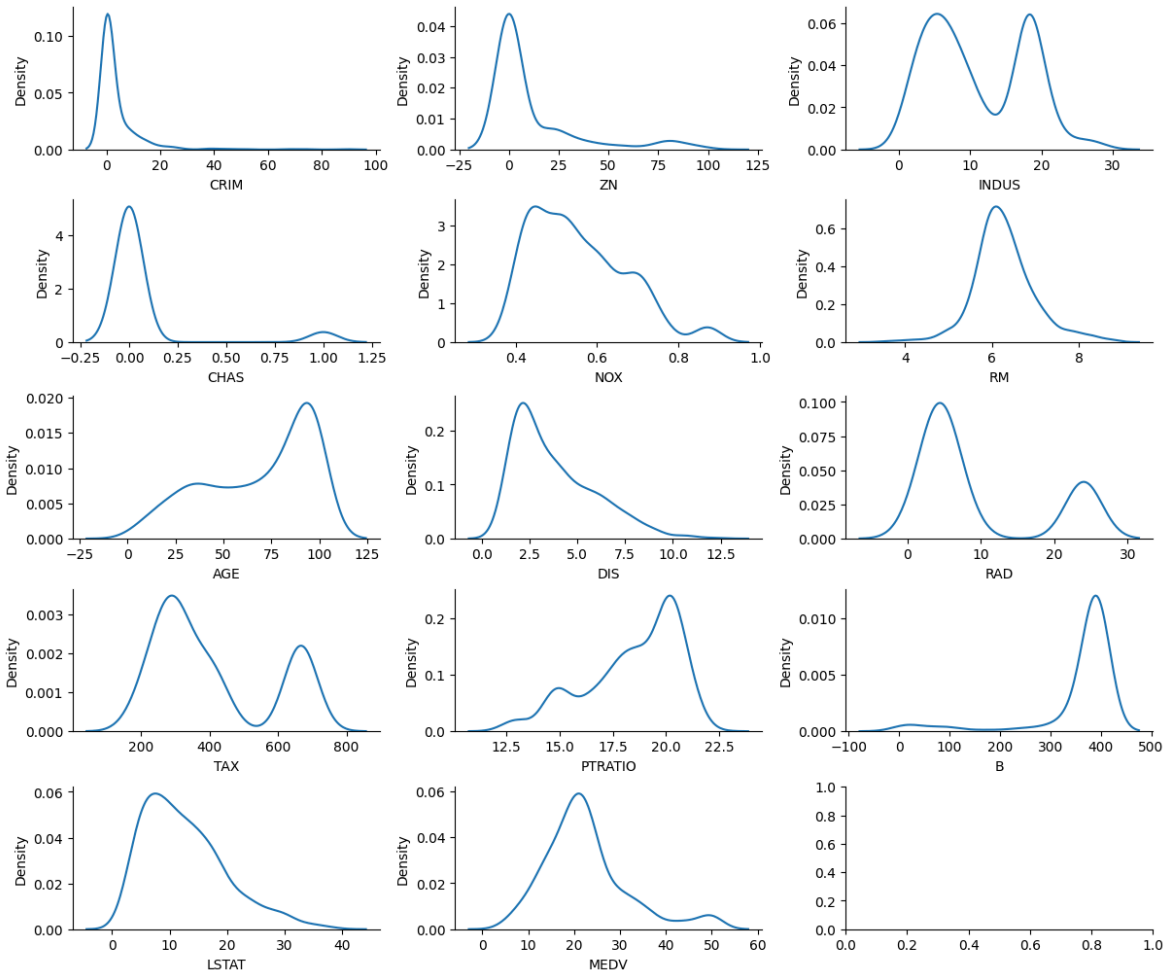
In [21]:
```
cols=list(df.columns)

fig,ax=plt.subplots(5,3,figsize=(12,10),constrained_layout=True)

axes_list = [axes for axes_row in ax for axes in axes_row]

for i, col_name in enumerate(cols):
```

```
    sns.histplot(data=df, x=col_name, ax=axes_list[i], bins=20)



# remove axes line on top-right
sns.despine()
plt.show()
```



Density of Data Distribution

In [22]:
```
cols=list(df.columns)

fig,ax=plt.subplots(5,3,figsize=(12,10),constrained_layout=True)

axes_list = [axes for axes_row in ax for axes in axes_row]

for i, col_name in enumerate(cols):
    sns.kdeplot(data=df, x=col_name, ax=axes_list[i])



# remove axes line on top-right
sns.despine()
plt.show()
```



Handling Skewed Data

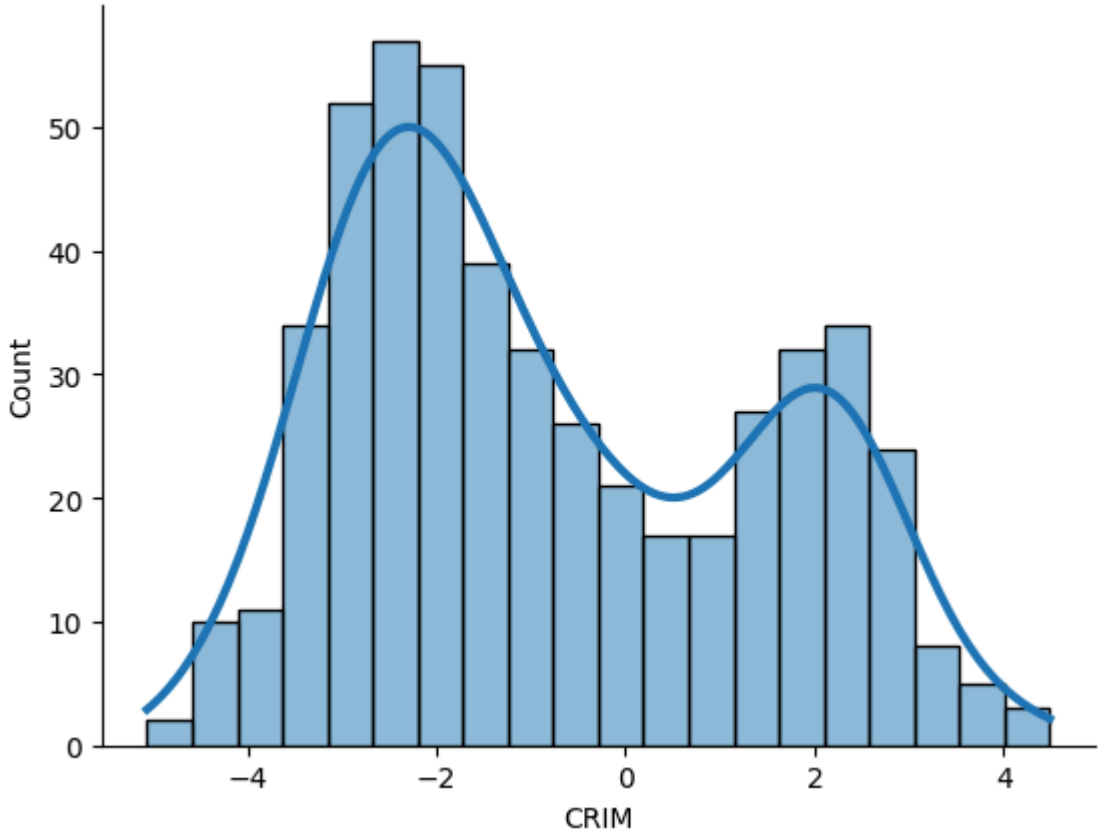
In [25]:
```
df.skew()
```

Out[25]:
```
CRIM       5.223149
ZN         2.225666
INDUS      0.295022
CHAS       3.405904
NOX        0.729308
RM         0.403612
AGE       -0.598963
DIS        1.011781
RAD        1.004815
TAX        0.669956
PTRATIO   -0.802325
B         -2.890374
LSTAT      0.906460
MEDV       1.108098
dtype: float64
```

In [26]:
```
crim_log = np.log(df['CRIM'])
crim_log.skew().round(2)
```

Out[26]:
```
np.float64(0.41)
```

In [27]:
```
sns.histplot(crim_log, bins=20, kde=True, line_kws={'linewidth':3})
sns.despine()
plt.show()
```



Correlation Metric

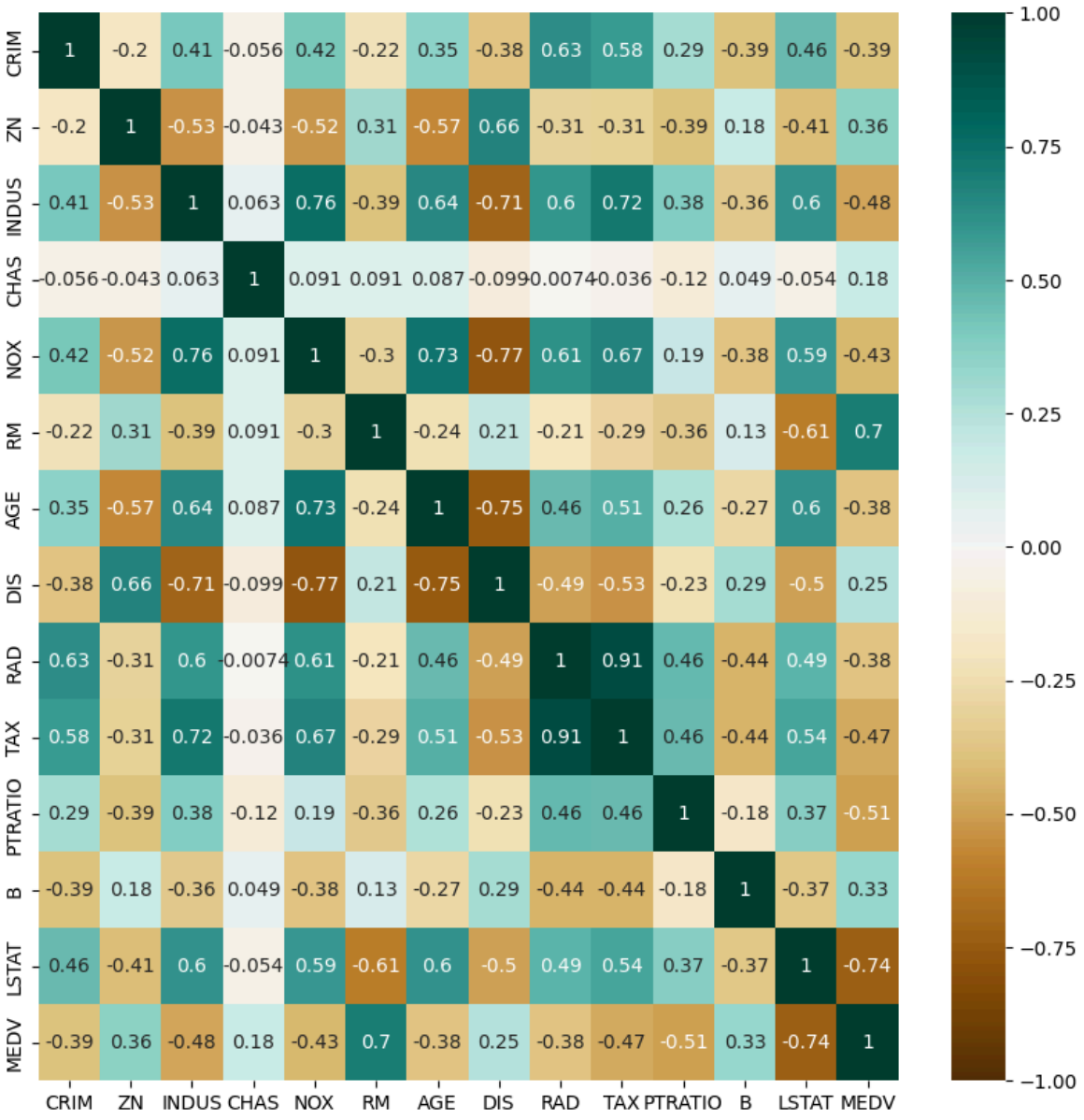
In [29]:
```
corr_matrix = df.corr()
corr_matrix
```

Out[29]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0
MEDV	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0

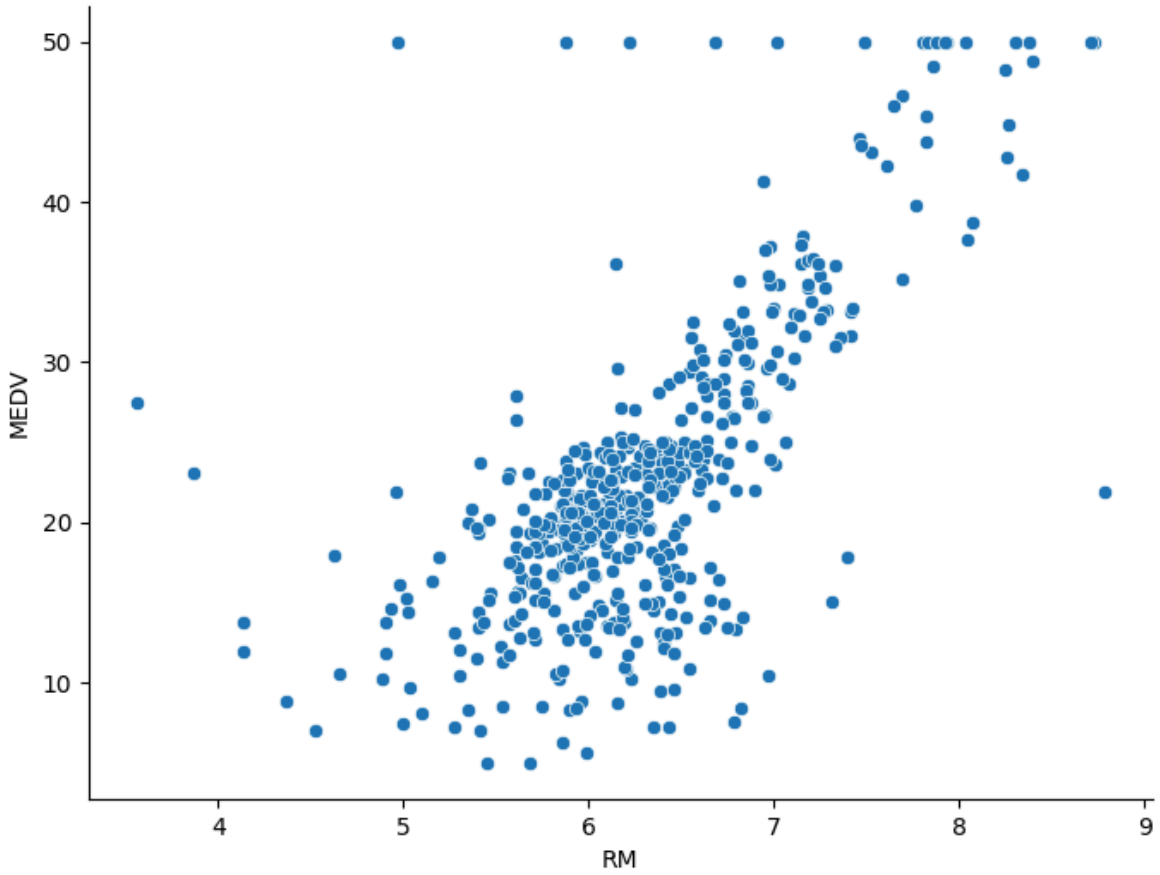
In [31]:

```
plt.figure(figsize=(10,10))
sns.heatmap(corr_matrix, center=0, vmin=-1, vmax=1, annot=True, cmap='BrBG')
plt.show()
```



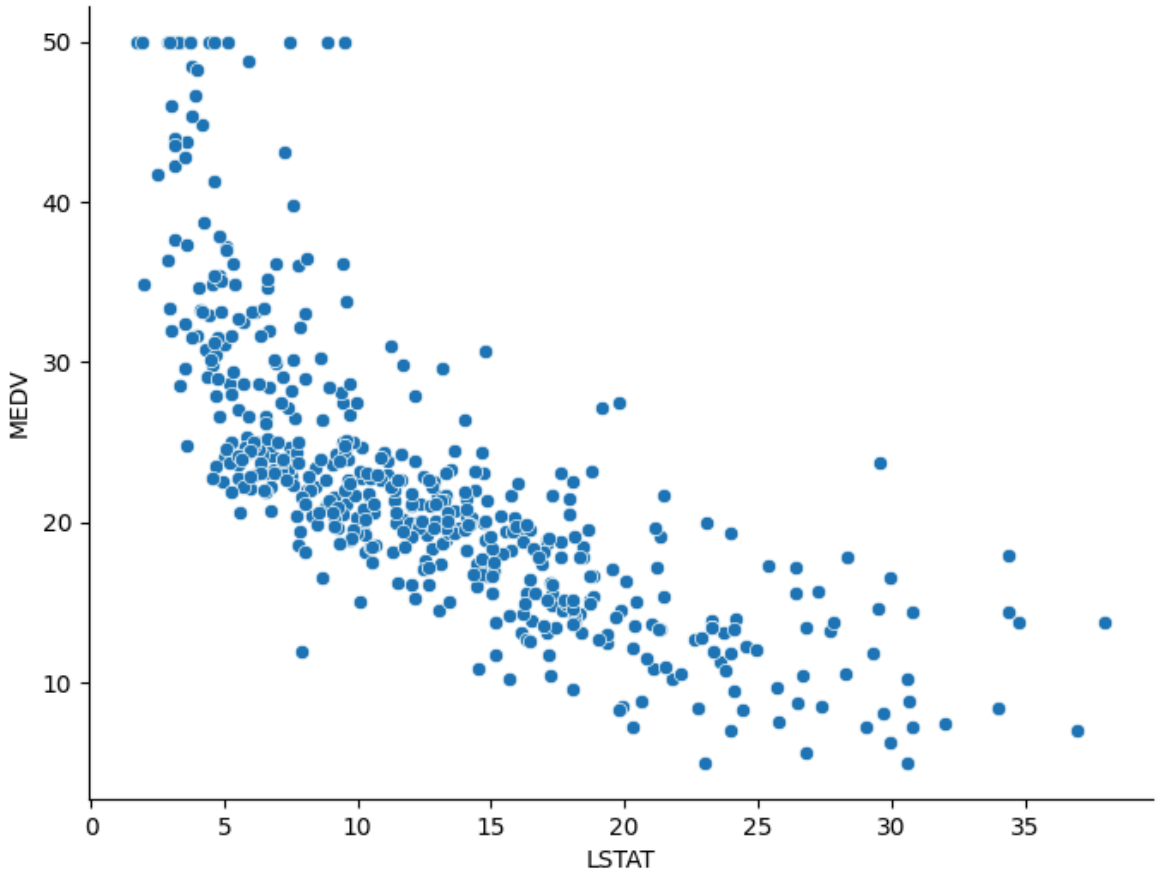
RM and MEDV

In [32]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='RM', y='MEDV')
sns.despine()
plt.show()
```



LSTAT and MEDV

In [34]:
```
plt.figure(figsize=(8,6))
sns.scatterplot(data=df, x='LSTAT', y='MEDV')
sns.despine()
plt.show()
```



Choosing Feature

In [68]:
```
X=df.drop('MEDV',axis=1)
Y=df['MEDV']
```

In [69]:
```
X.head()
```

Out[69]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.9
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.9
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.8
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.9

In [70]:
```
Y.head()
```

Out[70]:
```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: MEDV, dtype: float64
```

Train Test Split

In [71]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.7, random

print(X.shape)
print(Y.shape)
print()
print(X_train.shape)
print(Y_train.shape)
print()
print(X_test.shape)
print(Y_test.shape)
```

```
(506, 13)
(506,)

(354, 13)
(354,)

(152, 13)
(152,)
```

Instantiating the Model

In [72]:
```
from sklearn.linear_model import LinearRegression

model=LinearRegression()
```

Training the Model

In [73]:
```
model.fit(X_train,Y_train)
```

Out[73]:
```
▾ LinearRegression
LinearRegression()
```

In [74]:
```
m=model.coef_
c=model.intercept_
print(f'Slope: {m}')
print(f'Intercept: {c}')
```

```
Slope: [-9.85424717e-02  6.07841138e-02  5.91715401e-02  2.43955988e+00
 -2.14699650e+01  2.79581385e+00  3.57459778e-03 -1.51627218e+00
  3.07541745e-01 -1.12800166e-02 -1.00546640e+00  6.45018446e-03
 -5.68834539e-01]
Intercept: 46.39649387182381
```

Prediction

In [75]:
```
Y_pred=model.predict(X_test)
```

```
Y_pred
```

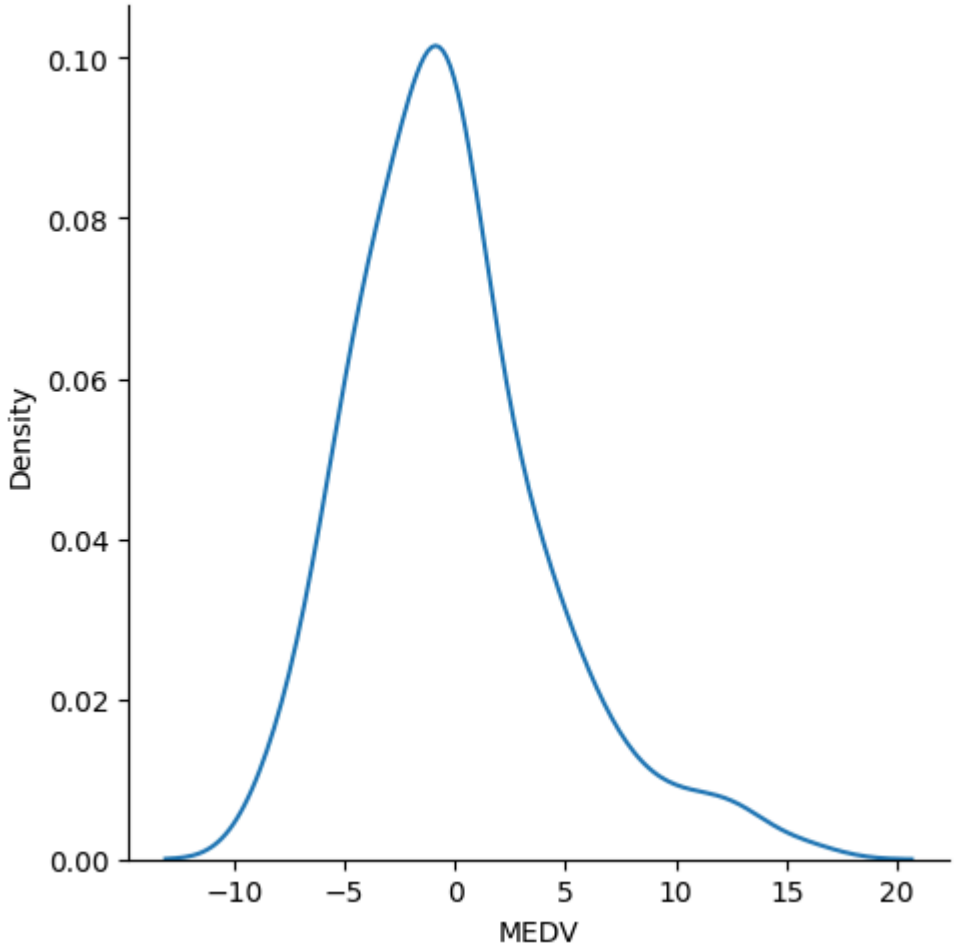
Out[75]:
```
array([32.3914647 , 27.94401279, 17.83762764, 21.66941388, 18.93639614,
       20.01995795, 32.63414961, 17.86179976, 24.73211676, 26.77346826,
       27.26138023, 28.63243014, 21.16103038, 27.15098589, 23.33198614,
       21.02894005, 17.11819755, 37.52455938, 30.69864427,  8.46104486,
       20.83966032, 16.26732451, 25.05030307, 24.8069032 , 31.6789299 ,
       10.98989363, 13.54111562, 16.54697099, 35.86699683, 14.69589266,
       21.05655107, 13.90301353, 43.01321262, 17.80335305, 21.68817023,
       20.45077825, 17.38095983, 26.81334904, 10.08706217, 19.96174586,
       24.1983249 , 21.00202622, 29.38682967, 16.33215467, 19.37439646,
       14.45525235, 39.52670798, 18.18535021, 26.38536303, 20.41420863,
       25.08511581, 24.40208165, 24.87902769, 26.97887956,  4.93921657,
       24.0001365 , 10.66711602, 26.98494775, 16.63507993, 35.31652989,
       19.49249915, 27.37577863, 16.53965715, 19.09746715, 11.06765336,
       32.03445261, 36.15847648, 21.53547868, 24.96434626, 25.50601154,
       23.41203509,  7.20513615, 16.8659937 , 20.06269471, 20.61979368,
       21.97238475, 34.25908683, 27.9998298 , 24.95261644, 34.74201458,
       18.63786454, 23.95355085, 34.50655345, 13.15979729, 20.53691051,
       30.29844955, 16.932953  , 24.29285934, 19.11800155, 16.9848188 ,
       26.80165303, 40.90027245, 14.43790215, 23.20589688, 15.39439519,
       21.77332361, 22.85660249, 29.45718412, 36.54373379, 20.36577747,
       18.01790611, 17.41615519, 25.23381893, 21.98069747,  8.3160445 ,
       21.49942727, 16.565317  , 33.22680671, 24.39944797, 24.99055027,
       38.37532568, 28.82033178, 14.76287071, 34.72063358, 35.40686066,
       32.93299015, 20.95996646, 16.74324182, 34.54769634, 38.9463732 ,
       21.47971843, 15.54203787, 27.46719043, 18.62729405, 27.3819011 ,
       21.13549164, 26.24403323, 22.22010257, 23.34295812, 28.29923408,
       20.53788412, 23.94639233, 29.8421088 , 10.55075792, 27.02280253,
       32.41641579, 13.94180705, 13.62057726, 33.00870341, 14.2836798 ,
       17.70899424, 16.30194776, 17.14377392, 28.71467061, 34.07548297,
       20.31288796, 24.55801246, 17.32995936, 28.3602341 , 20.79422791,
       33.6981572 , 13.51882716])
```

In [76]:
```
residual=Y_test-Y_pred
residual
```

Out[76]:
```
307   -4.191465
343   -4.044013
47    -1.237628
67     0.330586
362    1.863604
         ...
467    1.770041
95     0.039766
122   -0.294228
260    0.101843
23     0.981173
Name: MEDV, Length: 152, dtype: float64
```

In [77]:
```
sns.displot(residual,kind='kde')
```

Out[77]:
```
<seaborn.axisgrid.FacetGrid at 0x1d839db1990>
```



In [79]:
```
from sklearn.metrics import mean_squared_error,r2_score,mean_absolute_error

mse=mean_squared_error(Y_test,Y_pred)
mae=mean_absolute_error(Y_test,Y_pred)
r2=r2_score(Y_test,Y_pred)
print(f'Mean Squared Error: {mse}')
print(f'Mean Absolute Error: {mae}')
print(f'R2 Score: {r2}')
```

```
Mean Squared Error: 19.831323672063046
Mean Absolute Error: 3.344665503598754
R2 Score: 0.7836295385076302
```

In [86]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))


sns.scatterplot(x=Y_test, y=Y_pred, ax=axes[0], color='blue', alpha=0.7)
axes[0].plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], color='
axes[0].set_title("Actual vs Predicted")
axes[0].set_xlabel("Actual Values")
axes[0].set_ylabel("Predicted Values")

sns.histplot(residual, bins=20, kde=True, ax=axes[1], color='purple')
axes[1].axvline(0, color='red', linestyle='--')
axes[1].set_title("Residual (Error) Distribution")
axes[1].set_xlabel("Residuals")
axes[1].set_ylabel("Frequency")


metrics = ["MSE", "MAE", "R² Score"]
```

```
values = [mse, mae, r2]
sns.barplot(x=metrics, y=values, ax=axes[2], palette="viridis",hue=metrics)
axes[2].set_title("Regression Metrics")
axes[2].set_ylabel("Value")

plt.tight_layout()
plt.show()
```