Importing required modules

In [80]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Loading dataset and printing it

In [81]:

```
df = pd.read_csv("BostonHousing.csv")
df.head(100)
```

Out[81]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	0.12204	0.0	2.89	0	0.445	6.625	57.8	3.4952	2	276	18.0	357.98	6.65	28.4
96	0.11504	0.0	2.89	0	0.445	6.163	69.6	3.4952	2	276	18.0	391.83	11.34	21.4
97	0.12083	0.0	2.89	0	0.445	8.069	76.0	3.4952	2	276	18.0	396.90	4.21	38.7
98	0.08187	0.0	2.89	0	0.445	7.820	36.9	3.4952	2	276	18.0	393.53	3.57	43.8
99	0.06860	0.0	2.89	0	0.445	7.416	62.5	3.4952	2	276	18.0	396.90	6.19	33.2

100 rows × 14 columns

Preprocessing the dataset

In [82]:

```
df.describe()
```

Out[82]:

	crim	zn	indus	chas	nox	rm	age	dis

	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
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500

In [83]:

```
df.dtypes
```

Out[83]:

```
crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
lstat      float64
medv       float64
dtype: object
```

In [84]:

```
df.isna().sum()
```

Out[84]:

```
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 [85]:

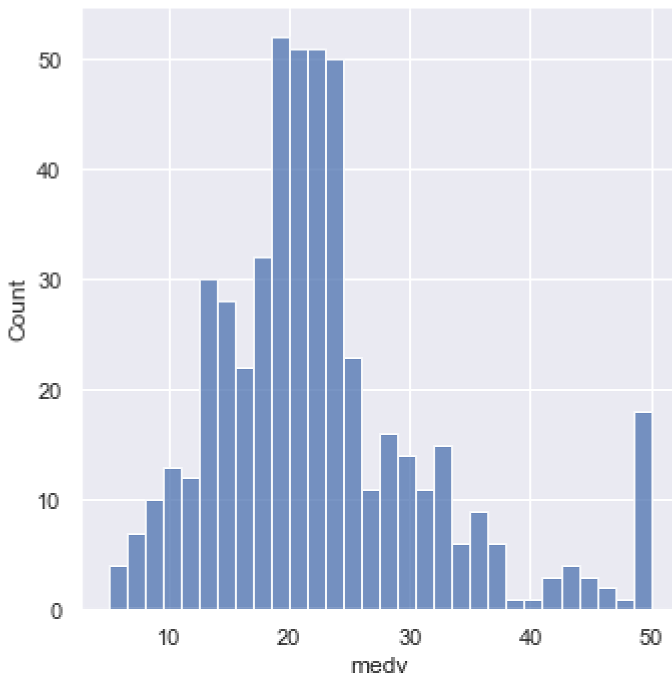
```
df.head()
```

Out[85]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [86]:

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.displot(df['medv'], bins=30)
plt.show()
```



correlation matrix that measures the linear relationships between the variables.

In [87]:

```
correlation_matrix = df.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```

Out[87]: <AxesSubplot:>

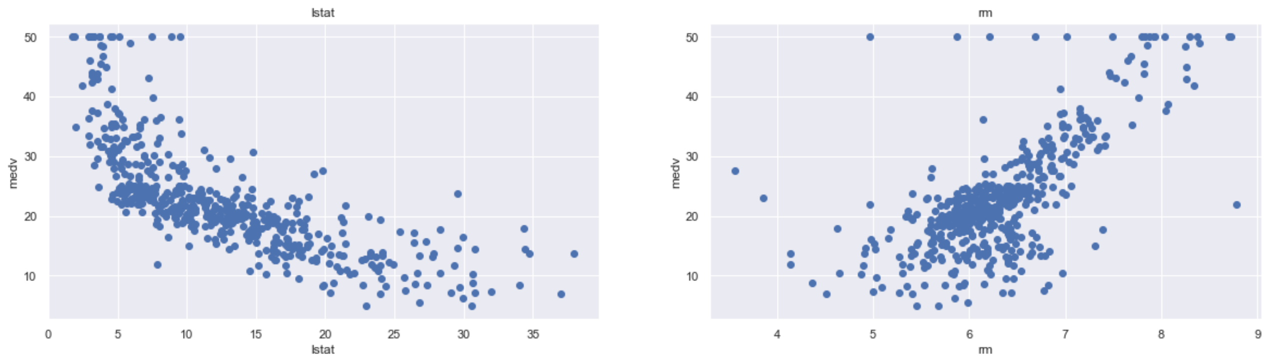
To fit a linear regression model, we select those features which have a high correlation with our target variable MEDV By looking at the correlation matrix we can see that RM has a strong positive correlation with MEDV (0.7) where as LSTAT has a high negative correlation with DV(-0.74)

In [88]:
```
plt.figure(figsize=(20,5))


features = ['lstat','rm']
target = df['medv']


for i, col in enumerate(features):
    plt.subplot(1,len(features), i+1)
    x = df[col]
    y = target
    plt.scatter(x, y, marker='o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('medv')


#Using a scatter plot let’s see how these features vary with MEDV
```



Observations

The prices tend to decrease with an increase in LSTAT

The prices increase as the value of RM increases linearly and there are few outliers.

Preparing the data for training the model

We concatenate the LSTAT and RM columns using np.c_ provided by the numpy library.

In [93]:
```
X = pd.DataFrame(np.c_[df['lstat'], df['rm']], columns = ['lstat','rm'])
Y = df['medv']
```

Spliting the data into training and testing sets

Train the model with 80% of the samples and test with the remaining 20% to assess the model's performance.

In [94]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

Training and testing the model

Using scikit-learn's LinearRegression to train model on both the training and test sets

In [109…
```
lin_model = LinearRegression()
```

```
lin_model.fit(X_train, Y_train)
```

Out[109… LinearRegression()

Model evaluation using and R2-score.

In [92]:

```
# model evaluation for training set
y_train_predict = lin_model.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("\n")

# model evaluation for testing set
y_test_predict = lin_model.predict(X_test)
rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
r2 = r2_score(Y_test, y_test_predict)

print("The model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
The model performance for training set
--------------------------------------
RMSE is 5.6371293350711955
R2 score is 0.6300745149331701


The model performance for testing set
--------------------------------------
RMSE is 5.13740078470291
R2 score is 0.6628996975186954
```