practice-assignment-4

May 4, 2025

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

[2]:
```python
df=pd.read_csv("BostonHousing.csv")
df
```

[2]:
```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       …     …     …     …     …     …    …       …   …   …
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       …       …     …     …
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
```

[3]:
```
print(df.head())
print(df.tail())
print(df.info())
print(df.describe())
```

```
      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     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
        crim   zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
501  0.06263  0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273     21.0
502  0.04527  0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273     21.0
503  0.06076  0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273     21.0
504  0.10959  0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273     21.0
505  0.04741  0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273     21.0

          b  lstat  medv
501  391.99   9.67  22.4
502  396.90   9.08  20.6
503  396.90   5.64  23.9
504  393.45   6.48  22.0
505  396.90   7.88  11.9
<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       501 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    int64
 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(11), int64(3)
memory usage: 55.5 KB
None
             crim          zn       indus        chas         nox          rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  501.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284341
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.705587
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.884000
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208000
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.625000
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              age         dis         rad         tax     ptratio           b  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000

            lstat        medv
count  506.000000  506.000000
mean    12.653063   22.532806
std      7.141062    9.197104
min      1.730000    5.000000
25%      6.950000   17.025000
50%     11.360000   21.200000
75%     16.955000   25.000000
max     37.970000   50.000000
```

[4]:
```
print(df.shape)
print(df.size)
```

```
(506, 14)
7084
```

[5]:
```
print(df.isna())
print(df.isna().sum())
```

```
      crim     zn  indus   chas    nox     rm    age    dis    rad    tax  \
0    False  False  False  False  False  False  False  False  False  False
1    False  False  False  False  False  False  False  False  False  False
```

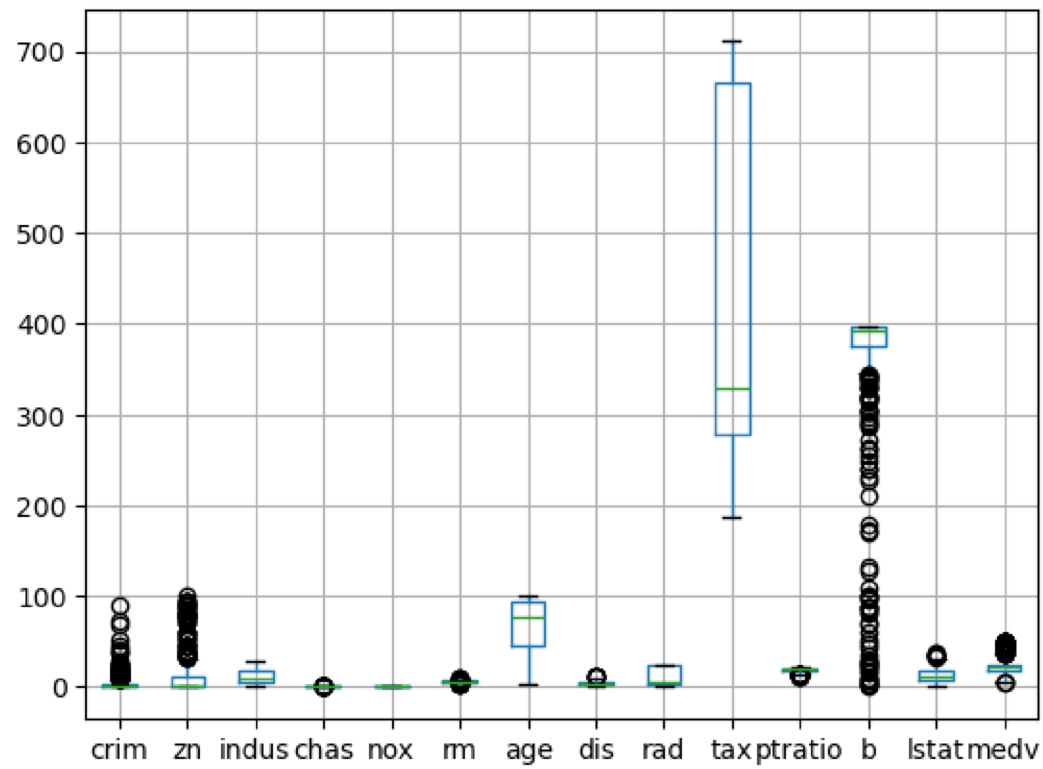
```
2    False  False  False  False  False  False  False  False  False  False
3    False  False  False  False  False  False  False  False  False  False
4    False  False  False  False  False  False  False  False  False  False
..     …      …      …      …      …      …      …      …      …      …
501  False  False  False  False  False  False  False  False  False  False
502  False  False  False  False  False  False  False  False  False  False
503  False  False  False  False  False  False  False  False  False  False
504  False  False  False  False  False  False  False  False  False  False
505  False  False  False  False  False  False  False  False  False  False

     ptratio      b  lstat   medv
0      False  False  False  False
1      False  False  False  False
2      False  False  False  False
3      False  False  False  False
4      False  False  False  False
..       …      …      …      …
501    False  False  False  False
502    False  False  False  False
503    False  False  False  False
504    False  False  False  False
505    False  False  False  False

[506 rows x 14 columns]
crim       0
zn         0
indus      0
chas       0
nox        0
rm         5
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

[6]:
```
df.boxplot()
```

[6]: `<Axes: >`



[7]:
```
Q1 = df['medv'].quantile(0.25)
Q3 = df['medv'].quantile(0.75)
IQR = Q3 - Q1
Lower_limit = Q1 - 1.5 * IQR
Upper_limit = Q3 + 1.5 * IQR
print(f'Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}, Lower_limit =␣
 ↪{Lower_limit},␣Upper_limit = {Upper_limit}')
```

```
Q1 = 17.025, Q3 = 25.0, IQR = 7.975000000000001, Lower_limit =
5.0624999999999964,␣Upper_limit = 36.962500000000006
```

[8]:
```
outliers_medv=[]
for i in df.medv:
    if i<Lower_limit or i>Upper_limit:
        outliers_medv.append(i)
print("outliers are",outliers_medv)
```

```
outliers are [38.7, 43.8, 41.3, 50.0, 50.0, 50.0, 50.0, 37.2, 39.8, 37.9, 50.0,
37.0, 50.0, 42.3, 48.5, 50.0, 44.8, 50.0, 37.6, 46.7, 41.7, 48.3, 42.8, 44.0,
50.0, 43.1, 48.8, 50.0, 43.5, 45.4, 46.0, 50.0, 37.3, 50.0, 50.0, 50.0, 50.0,
50.0, 5.0, 5.0]
```

[9]:
```
df[df.medv<Lower_limit].index
```

[9]: Index([398, 405], dtype='int64')

[10]:
```
outlier_indices=df[(df.medv<Lower_limit)|(df.medv>Upper_limit)].index
df1=df.drop(outlier_indices)
df1
```

[10]:
```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       …     …      …     …      …      …     …       …  …    …
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..         …       …     …     …
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[466 rows x 14 columns]
```

[11]:
```
df1.boxplot()
```

[11]: <Axes: >

[12]:
```
outliers_medv=[]
for i in df1.medv:
    if i<Lower_limit or i>Upper_limit:
        outliers_medv.append(i)
print("outliers are",outliers_medv)
```

```
outliers are []
```

[13]:
```
df1
```

[13]:
```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat  medv
0       15.3  396.90   4.98  24.0
1       17.8  396.90   9.14  21.6
2       17.8  392.83   4.03  34.7
3       18.7  394.63   2.94  33.4
4       18.7  396.90   5.33  36.2
..       ...     ...    ...   ...
501     21.0  391.99   9.67  22.4
502     21.0  396.90   9.08  20.6
503     21.0  396.90   5.64  23.9
504     21.0  393.45   6.48  22.0
505     21.0  396.90   7.88  11.9

[466 rows x 14 columns]
```

[14]:
```
X = df.drop(['medv'], axis = 1)
Y = df['medv']
```

[15]:
```
print(X)
print(Y)
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
```

```
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       …      …     …
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       …
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: medv, Length: 506, dtype: float64
```

[49]:

```python
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size =0.
 ↪2,random_state = 0)
```

[51]:

```python
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

[53]:

```python
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean') # You can choose other strategies like␣
 ↪'median' or 'most_frequent'
X = imputer.fit_transform(X)

# 2. Split the data after imputation
xtrain, xtest, ytrain, ytest = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=0)

# 3. Create and train the model
```

```python
lm = LinearRegression()
model = lm.fit(xtrain, ytrain)
```

[55]:
```python
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
ytrain_pred
```

[55]:
```
array([32.54370319, 21.932219  , 27.54552964, 23.62527133,  6.578493  ,
       14.97841931, 22.21772491, 29.1656419 , 33.23633623, 13.13958076,
       20.27455237, 20.68413823, 12.66852204, 23.3697738 ,  5.02810654,
       19.82447761,  9.43610482, 44.62829537, 30.78647869, 12.51813161,
       17.73377577, 21.38820917, 23.6372432 , 20.44934681, 34.99179512,
       13.88126045, 21.08693233, 35.14506971, 19.42370996, 13.14651005,
       14.07551034, 23.11554812, 14.35905012, 31.26696558, 25.30600475,
       15.4186878 , 24.22888048,  9.38631784, 14.92650845, 20.8121123 ,
       32.71894972, 27.98210777, 25.59531311, 15.57980306, 31.11707339,
       27.96643339, 13.98777371,  7.63296162, 28.439442  , 25.35431753,
        4.50921003, 28.38300415, 16.98778052, 29.76898387, 20.46579326,
       15.92063503, 17.90107052, 12.73790055,  8.74450855, 19.2209131 ,
       34.49718351, 32.92991179, 23.69727449, 19.56029786, 22.84821335,
       26.87821751, 21.82625519, 17.07317532, 32.05572536, 10.93501836,
       19.43965502, 32.49620518, 18.84673792, 15.94631864, 18.64040032,
       14.44326484, 24.59149605, 24.32961808, 16.64743526, 13.33257237,
       20.22362526, 25.14078145, 17.16142814, 24.71659709, 20.81755107,
       27.98436411, 35.61445424, 16.64616352, 11.82105901, 34.81722148,
       30.82293852, 20.74723986, 39.53769412, 28.93973655, 29.14758957,
       17.3850748 , 26.81988505, 39.99994502, 28.7436029 , 16.45233953,
       37.42520467, 35.50043267, 13.4301604 , 29.16282174, 21.61445353,
       24.33573863, 21.42879278, 23.70414569, 27.75720913, 29.66634603,
       14.17071791, 26.10123154, 23.301429  , 12.78870854, 13.69996698,
       25.25219339, 19.33992149, 30.53939588, 10.99425038, 23.59381765,
       16.97608003, 16.95715843, 22.63737653, 21.67809796, 11.7771548 ,
       25.20102569, 28.70665016, 20.15560512, 12.58969204, 25.491827  ,
       25.94150764, 25.1012182 , 23.54704847, 26.75021658, 16.60998619,
       21.8143152 , 36.13010572, 20.99151854, 35.84068525, 25.7240647 ,
       21.53807905, 15.86301742, 31.29104399, 21.25252923, 27.76555697,
       14.82988215, 32.22181362, 13.98471864,  1.73549252, 19.34419812,
       14.27907805, 37.50921956, 15.73128883, 14.43177617, 27.30962563,
       23.26035056, 18.51338996, 30.56987939, 27.27579387, 27.29636247,
       24.83120316, 24.16491603, 25.02281109, 11.15470273, 20.75645507,
       13.62885662, 17.19251126, 12.73588136, 28.36489633, 14.93423297,
       16.28759868, 28.71821577, 14.92423512, 21.2574851 , 12.84398221,
       13.88240604, 22.63704415, 21.20611254, 14.77408684, 20.9615425 ,
       16.95022031, 24.57066259, 12.55888031, 34.75093634, 12.01928324,
       43.11627076, 31.24437499, 35.2678383 , 21.46189294, 15.76203608,
       26.55522047, 29.48442291, 14.09084558, 26.53043308, 37.03279546,
       17.66599331, 10.58127564, 34.12553809, 35.60724777, 18.30871709,
```

```
22.55422403, 17.99645545, 24.37480858, 19.53272297, 27.33759414,
-4.37710883, 20.55848845, 35.20657023, 36.61182186, 25.09838298,
27.19384405, 20.76511499, 20.59850139, 15.88235358, 20.7109413 ,
20.56015402, 27.91074931, 19.64144048,  7.44448041, 16.38421739,
32.4058924 , 35.20068077, 17.52510488, 18.73818756, 23.38476377,
 6.9101272 , 21.474421  , 24.03945664, 16.48224996, 18.39477039,
21.89450064, 27.6087426 , 25.48978313, 36.99207872, 15.44312984,
28.60930065, 25.91061939, 22.289867  , 38.68527907, 20.87019495,
23.42254099, 22.8715894 , 12.46815126, 20.32563551, 33.59413957,
24.81623268, 18.00822181, 33.52610644, 21.60855452, 28.34626703,
32.25732348, 36.72521596, 22.24201483, 24.00412219, 22.43880668,
31.79684758, 22.36905758, 18.85033348, 21.83585813, 28.2440051 ,
22.53534761, 21.83903797, 17.02520576, 17.48214385, 16.95645501,
17.43479517, 16.49537109, 31.59311175, 23.80259153, 17.56107622,
19.79933557, 33.6701167 , 13.95877659, 24.9589198 , 17.37744496,
30.49453397, 29.97850997, 22.60575292, 20.826961  , 34.973321  ,
22.63480818, 32.88008718, 20.76686679, 31.40022292, 30.90899101,
37.56345649, 26.84896675, 21.94002219, 28.69666315, 16.17783819,
26.97516809, 21.11061277, 30.45452101,  9.96317971, 30.88843471,
 5.84554101, 15.63338238, 18.16309422, 35.40089044, 32.05626117,
11.05941886, 13.29457062, 21.62687263, 34.40247671, 18.64931021,
19.18796195, 14.99721921, 25.77438304, 41.10507292, 25.03604626,
42.00238887, 24.9324625 , 22.30586831, 12.24950075, 12.02438944,
14.16234864, 18.47767294,  3.08442408, 27.4989249 , 26.07818635,
40.99536581, 21.09643555, 21.1699912 , 34.05630845, 33.45094622,
 9.72577257, 24.74893414, 43.31925292, 16.92814859, 17.89933566,
25.52211908, 18.42060434,  6.13576017, 19.33435897, 34.89718181,
16.2265251 , 23.02431044, 13.589263  , 24.53484354, 18.77565936,
17.32092633, 18.75893027, 33.11765205, 19.46432946, 30.73066746,
32.75844496, 41.26823949, 19.17037266, 16.62926097, 37.52686446,
17.99662752,  9.42808957, 15.16127034, 24.93324217, 19.68070885,
16.62013838, 27.45888678, 12.96205602,  5.83839312, 19.02447968,
 9.86977983, 28.0976158 ,  4.52913758, 29.189982  , 32.14578868,
22.15775469, 16.77843434, 18.07456012, 20.72185108, 33.5981505 ,
27.78362104, 19.53364617, 20.72032337,  6.66638479, 28.92171723,
24.6198858 , 22.18005872, 13.66189533, 25.80558744, 19.360516  ,
 8.83678532, 26.69250461, 16.1742696 , 31.37220918, 32.61583684,
25.4444746 , 18.52294419, 30.58689102, 21.5617584 , 25.26864563,
25.90791006, 31.60220704, 24.53163973, 34.45150746, 17.12157442,
19.72932588, 18.55636829, 40.97364198, 25.1446076 , 19.51587917,
33.32027502, 23.79132039, 18.47117877, 23.25099877])
```

[57]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytrain, ytrain_pred)
print("The model performance for training set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
```

```python
print("\n")
# model evaluation for testing set
#y_test_predict = lin_model.predict(X_test)
mse = mean_squared_error(ytest, ytest_pred)

print("The model performance for testing set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print("\n\n\n")
rmse = (np.sqrt(mean_squared_error(ytrain, ytrain_pred)))
r2 = r2_score(ytrain, ytrain_pred)
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(ytest, ytest_pred)))
r2 = r2_score(ytest, ytest_pred)
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
--------------------------------------
MSE is 19.391606535694894


The model performance for testing set
--------------------------------------
MSE is 33.46497598891468




The model performance for training set
--------------------------------------
RMSE is 4.4035901870740535
R2 score is 0.7722485406464419


The model performance for testing set
--------------------------------------
RMSE is 5.784892046435671
R2 score is 0.5890259426012614
```

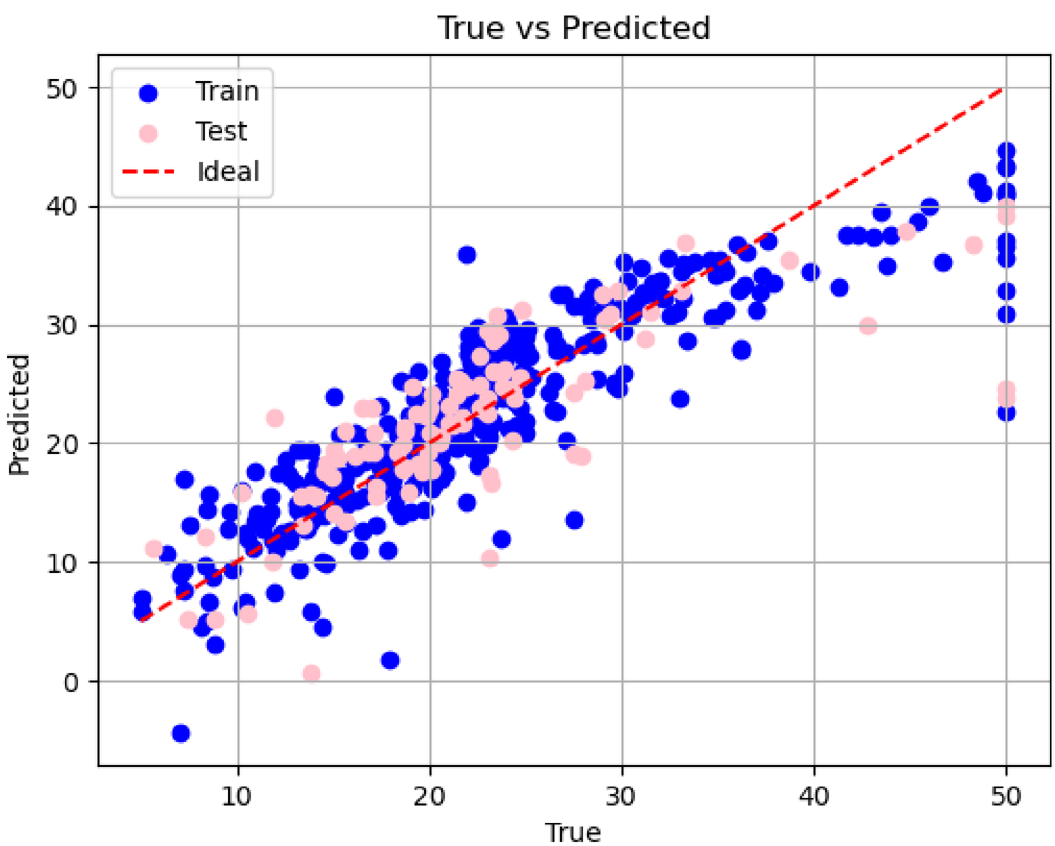
[59]:

```
import matplotlib.pyplot as plt

plt.scatter(ytrain, ytrain_pred, color='blue', label='Train')
plt.scatter(ytest, ytest_pred, color='pink', label='Test')

# Plot ideal line y = x
min_val = min(min(ytrain), min(ytest))
max_val = max(max(ytrain), max(ytest))
plt.plot([min_val, max_val], [min_val, max_val], 'r--', label='Ideal')

plt.xlabel('True')
plt.ylabel('Predicted')
plt.title('True vs Predicted')
plt.legend()
plt.grid(True)
plt.show()
```



[ ]: