4

April 15, 2024

```
[1]: import pandas as pd
     import numpy as np
     from sklearn import metrics
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
```

```
[3]: from sklearn.datasets import load_boston
     boston = load_boston()
```

```
[5]: data = pd.DataFrame(boston.data)
     data.head()
     data.columns = boston.feature_names
     data.head()
```

```
[5]:       CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
     0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900  1.0  296.0
     1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671  2.0  242.0
     2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671  2.0  242.0
     3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622  3.0  222.0
     4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622  3.0  222.0

        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
```

```
[6]: data['MEDV'] = boston.target
```

```
[8]: data.shape
```

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

```
[9]: data.columns
```

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

```
[13]: data.dtypes
```

```
[13]: CRIM       float64
      ZN         float64
      INDUS      float64
      CHAS       float64
      NOX        float64
      RM         float64
      AGE        float64
      DIS        float64
      RAD        float64
      TAX        float64
      PTRATIO    float64
      B          float64
      LSTAT      float64
      MEDV       float64
      dtype: object
```

```
[14]: data.isnull().sum
```

```
[14]: <bound method DataFrame.sum of        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]>
```

[15]:

```
data[data.isnull().any(axis = 1)]
```

[15]:

```
Empty DataFrame
Columns: [CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT,
MEDV]
Index: []
```

[16]:

```
data.describe()
```

[16]:

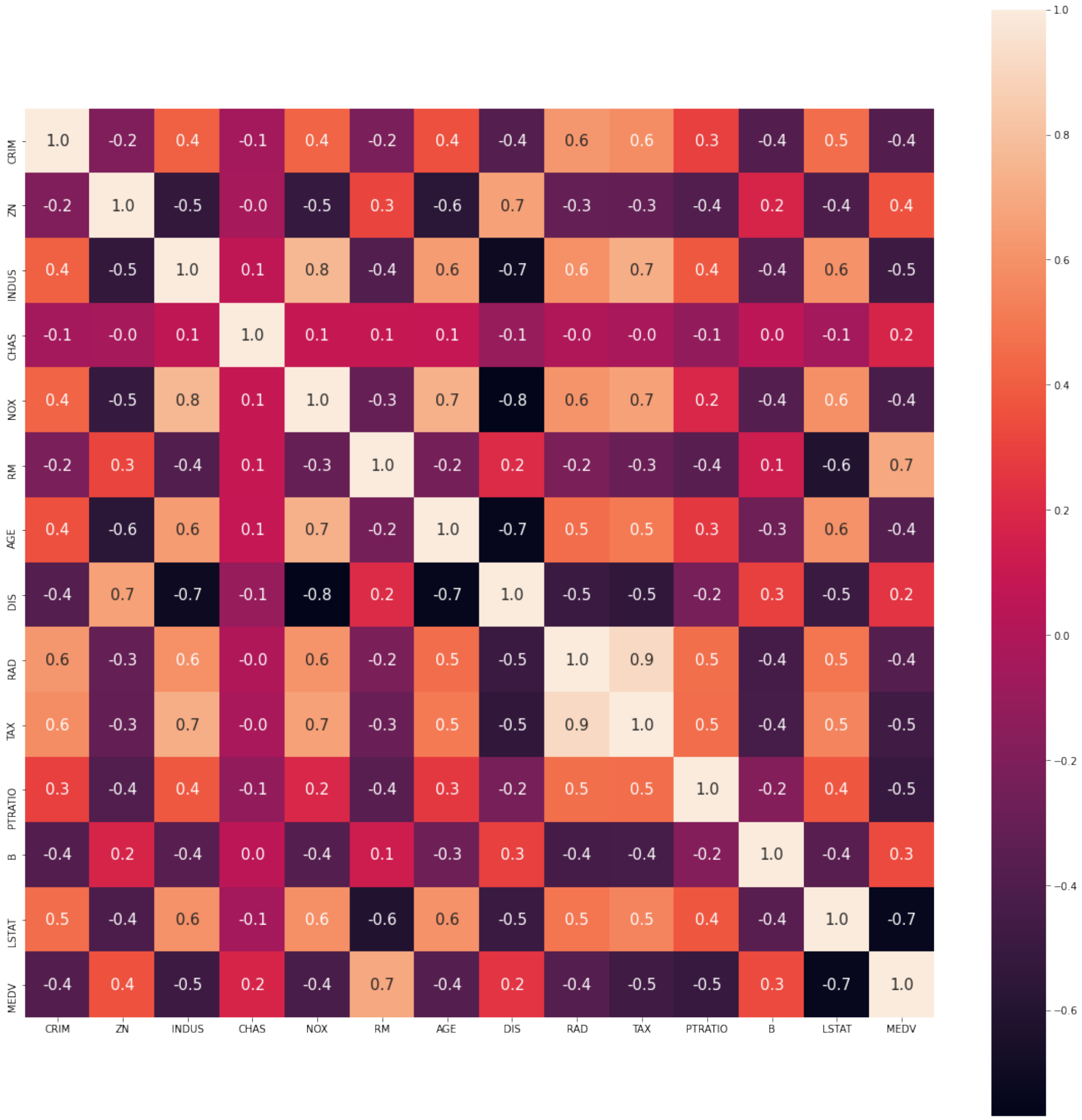
```
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
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
```

[19]:

```
corr = data.corr()
corr.shape
```

```
plt.figure(figsize=(20, 20))
sns.heatmap(corr, cbar = True, square = True, fmt = '.1f', annot = True,␣
 ↪annot_kws = {'size':15})
```

```
[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1d5bb5927c0>
```



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1.0	-0.2	0.4	-0.1	0.4	-0.2	0.4	-0.4	0.6	0.6	0.3	-0.4	0.5	-0.4
ZN	-0.2	1.0	-0.5	-0.0	-0.5	0.3	-0.6	0.7	-0.3	-0.3	-0.4	0.2	-0.4	0.4
INDUS	0.4	-0.5	1.0	0.1	0.8	-0.4	0.6	-0.7	0.6	0.7	0.4	-0.4	0.6	-0.5
CHAS	-0.1	-0.0	0.1	1.0	0.1	0.1	0.1	-0.1	-0.0	-0.0	-0.1	0.0	-0.1	0.2
NOX	0.4	-0.5	0.8	0.1	1.0	-0.3	0.7	-0.8	0.6	0.7	0.2	-0.4	0.6	-0.4
RM	-0.2	0.3	-0.4	0.1	-0.3	1.0	-0.2	0.2	-0.2	-0.3	-0.4	0.1	-0.6	0.7
AGE	0.4	-0.6	0.6	0.1	0.7	-0.2	1.0	-0.7	0.5	0.5	0.3	-0.3	0.6	-0.4
DIS	-0.4	0.7	-0.7	-0.1	-0.8	0.2	-0.7	1.0	-0.5	-0.5	-0.2	0.3	-0.5	0.2
RAD	0.6	-0.3	0.6	-0.0	0.6	-0.2	0.5	-0.5	1.0	0.9	0.5	-0.4	0.5	-0.4
TAX	0.6	-0.3	0.7	-0.0	0.7	-0.3	0.5	-0.5	0.9	1.0	0.5	-0.4	0.5	-0.5
PTRATIO	0.3	-0.4	0.4	-0.1	0.2	-0.4	0.3	-0.2	0.5	0.5	1.0	-0.2	0.4	-0.5
B	-0.4	0.2	-0.4	0.0	-0.4	0.1	-0.3	0.3	-0.4	-0.4	-0.2	1.0	-0.4	0.3
LSTAT	0.5	-0.4	0.6	-0.1	0.6	-0.6	0.6	-0.5	0.5	0.5	0.4	-0.4	1.0	-0.7
MEDV	-0.4	0.4	-0.5	0.2	-0.4	0.7	-0.4	0.2	-0.4	-0.5	-0.5	0.3	-0.7	1.0

```
[21]: x = data.drop(['MEDV'], axis = 1)
      y = data['MEDV']
```

```
[22]: from sklearn.model_selection import train_test_split
```

[23]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 4)
```

[24]:
```
from sklearn.linear_model import LinearRegression
```

[25]:
```
lm = LinearRegression()
```

[26]:
```
lm.fit(X_train, y_train)
```

[26]:
```
LinearRegression()
```

[28]:
```
lm.intercept_
```

[28]:
```
36.357041376595205
```

[35]:
```
coefficients= pd.DataFrame([X_train.columns, lm.coef_]).T
coefficients = coefficients.rename(columns = {0: 'Attribute', 1:␣
 ↪'Coefficients'})
coefficients
```

[35]:
```
   Attribute Coefficients
0       CRIM     -0.12257
1         ZN    0.0556777
2      INDUS  -0.00883428
3       CHAS      4.69345
4        NOX     -14.4358
5         RM      3.28008
6        AGE  -0.00344778
7        DIS     -1.55214
8        RAD      0.32625
9        TAX   -0.0140666
10   PTRATIO    -0.803275
11         B   0.00935369
12     LSTAT    -0.523478
```

[37]:
```
y_pred = lm.predict(X_train)
```

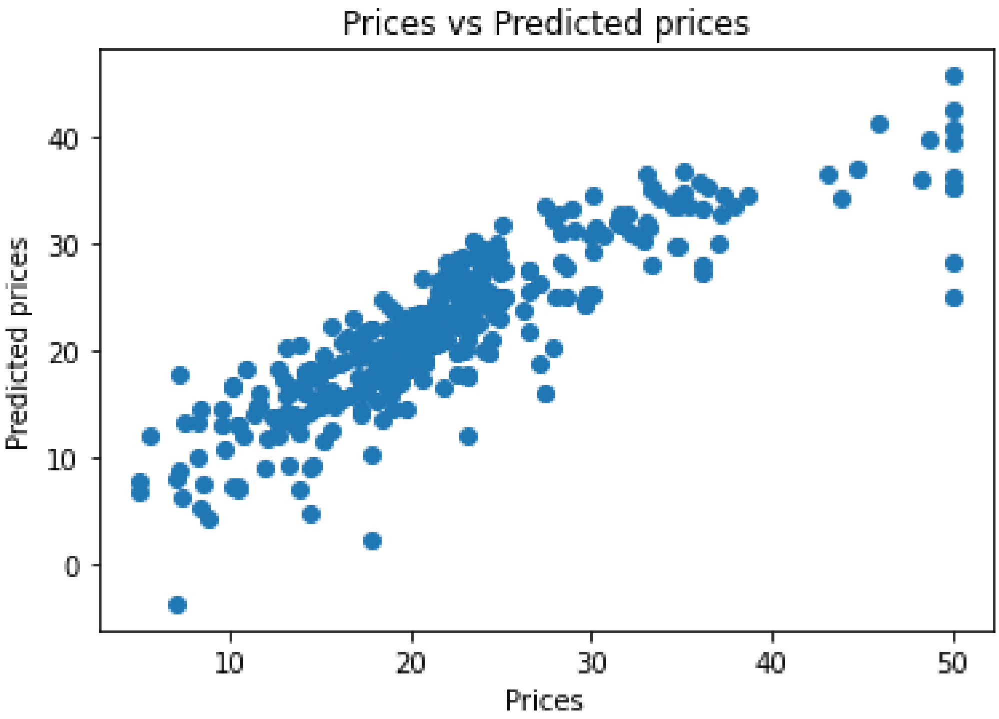
[39]:
```
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train,␣
 ↪y_pred))*(len(y_train)-1)/
(len(y_train)-X_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.7465991966746854
Adjusted R^2: 0.736910342429894
```

```
MAE: 3.08986109497113
MSE: 19.07368870346903
RMSE: 4.367343437774162
```

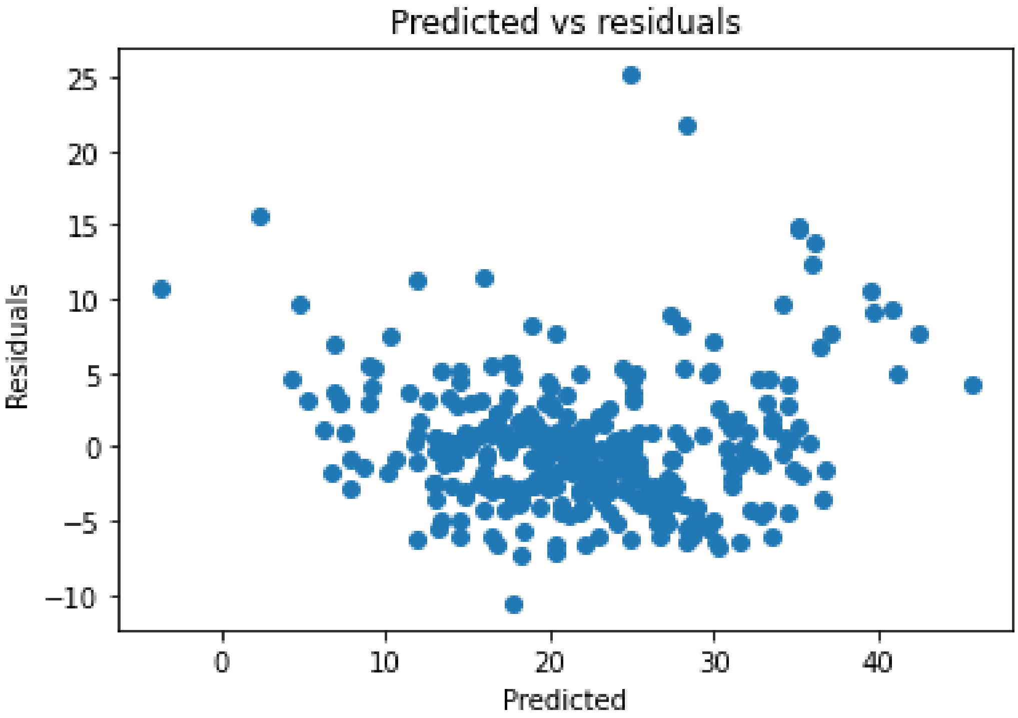
[40]:

```
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```



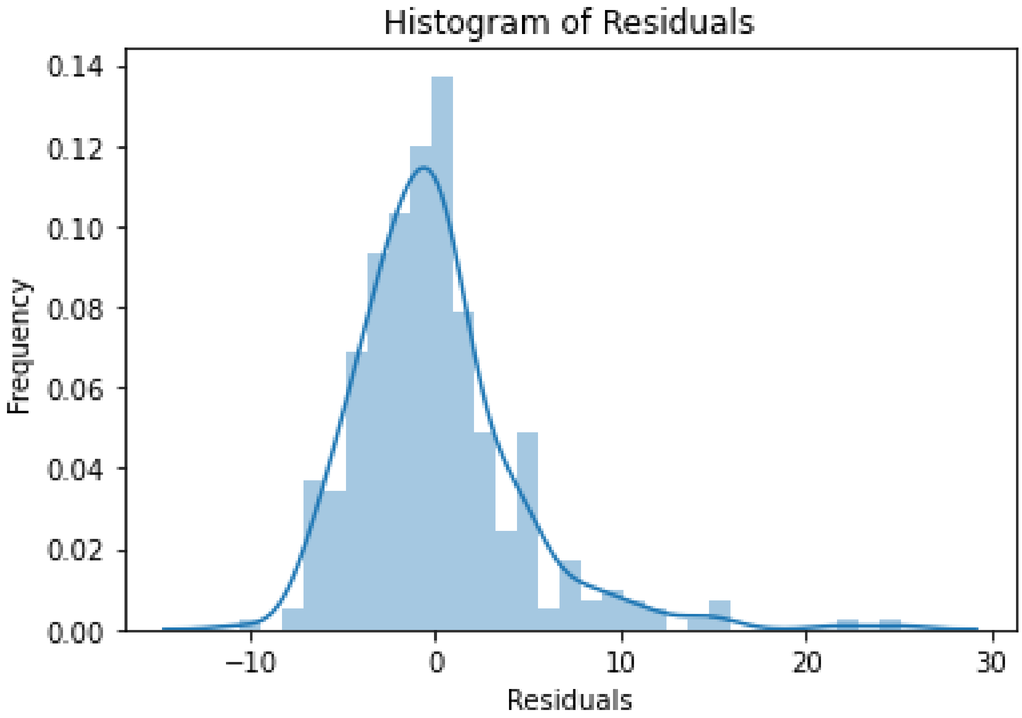
[41]:

```
plt.scatter(y_pred,y_train-y_pred)
plt.title("Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```



[42]:
```
sns.distplot(y_train-y_pred)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```

```
C:\Users\Tej\anaconda3\lib\site-packages\seaborn\distributions.py:2551:
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)
```



[43]:

```
y_test_pred = lm.predict(X_test)

acc_linreg = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_linreg)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test,␣
 ↪y_test_pred))*(len(y_test)-1)/(len(y_test)-X_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.7121818377409195
Adjusted R^2: 0.6850685326005713
MAE: 3.8590055923707407
MSE: 30.053993307124127
RMSE: 5.482152251362974
```