Assingment 5

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

In [2]:
```
from sklearn.datasets import load_boston
boston = load_boston()
```

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

Out[3]:

	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 [4]:
```
data['MEDV'] = boston.target
```

In [5]:
```
data.shape
```

Out[5]:
```
(506, 14)
```

In [6]:
```
data.columns
```

Out[6]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

In [7]:
```
data.dtypes
```

Out[7]:
```
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
```

In [8]:
```
data.isnull().sum
```

Out[8]:
```
<bound method NDFrame._add_numeric_operations.<locals>.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]>
```

In [9]:
```
data[data.isnull().any(axis = 1)]
```

Out[9]:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV

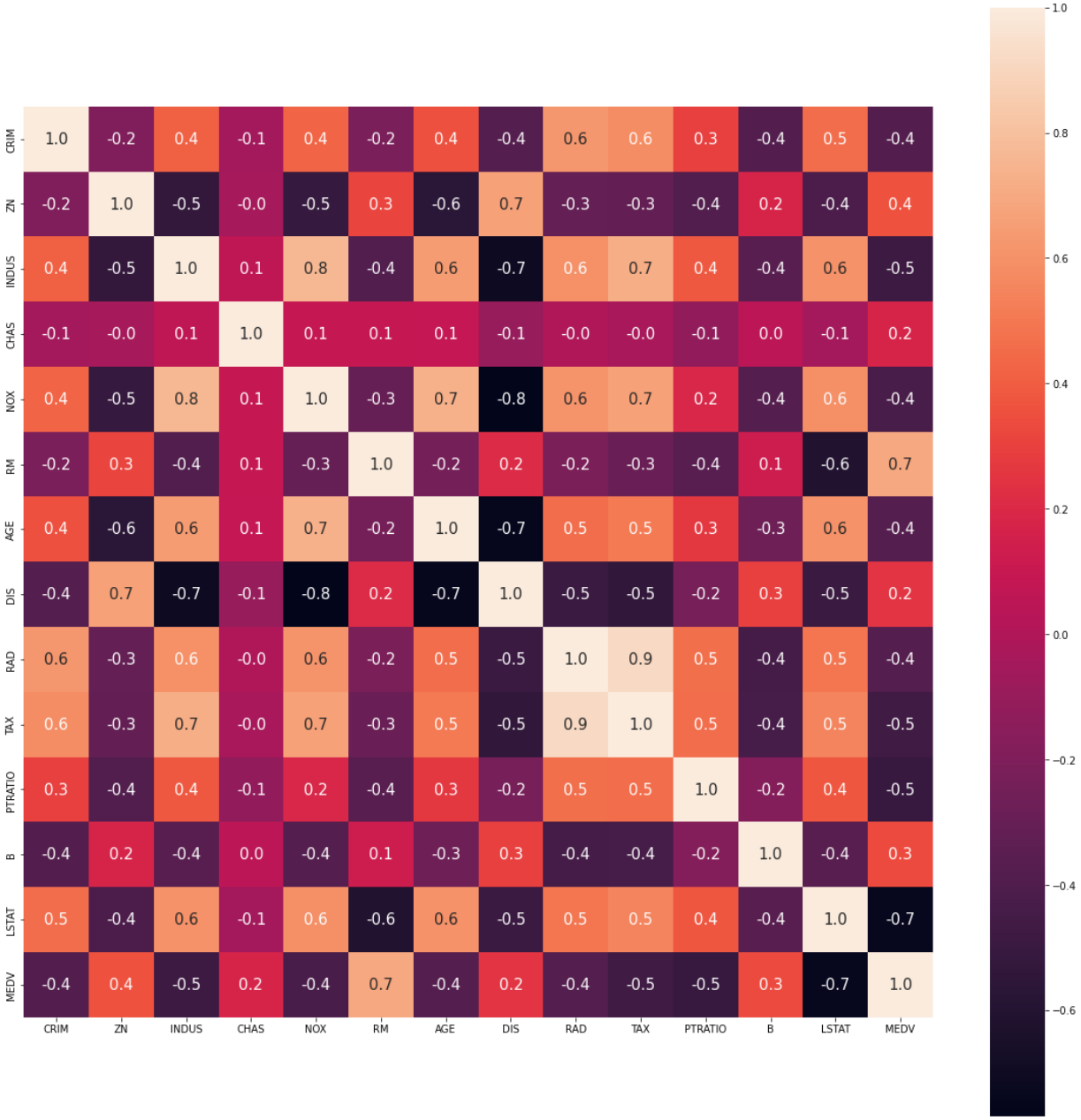
In [10]:
```
data.describe()
```

Out[10]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.79
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.10
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.12
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.10
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.20
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.18
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.12

In [11]:
```
corr = data.corr()
corr.shape
plt.figure(figsize=(20, 20))
sns.heatmap(corr, cbar = True, square = True, fmt = '.1f', annot = True, annot_kv
```

Out[11]: <AxesSubplot:>



In [12]:
```
x = data.drop(['MEDV'], axis = 1)
y = data['MEDV']
```

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random
```

In [15]:
```
from sklearn.linear_model import LinearRegression
```

In [16]:
```
lm = LinearRegression()
```

In [17]:
```
lm.fit(X_train, y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
lm.intercept_
```

Out[18]:
```
36.35704137659479
```

In [19]:
```
coefficients= pd.DataFrame([X_train.columns, lm.coef_]).T
coefficients = coefficients.rename(columns = {0: 'Attribute', 1: 'Coefficients'})
coefficients
```

Out[19]:

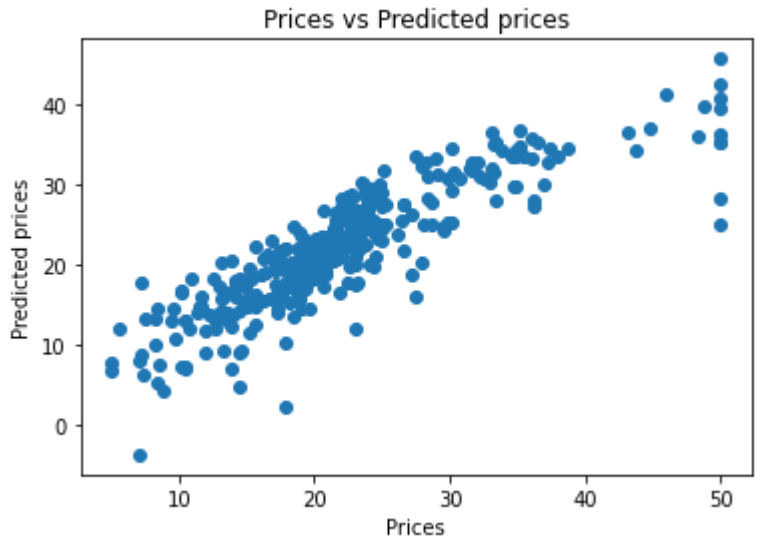
	Attribute	Coefficients
0	CRIM	-0.12257
1	ZN	0.055678
2	INDUS	-0.008834
3	CHAS	4.693448
4	NOX	-14.435783
5	RM	3.28008
6	AGE	-0.003448
7	DIS	-1.552144
8	RAD	0.32625
9	TAX	-0.014067
10	PTRATIO	-0.803275
11	B	0.009354
12	LSTAT	-0.523478

In [20]:
```
y_pred = lm.predict(X_train)
```

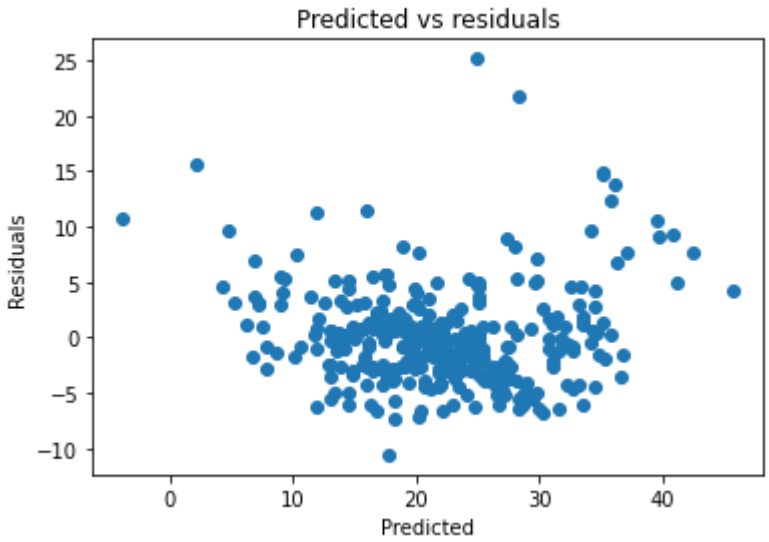
In [21]:
```
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/
(len(y_train)-X_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.7465991966746854
Adjusted R^2: 0.736910342429894
MAE: 3.089861094971131
MSE: 19.073688703469028
RMSE: 4.367343437774161
```

In [22]:
```
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```

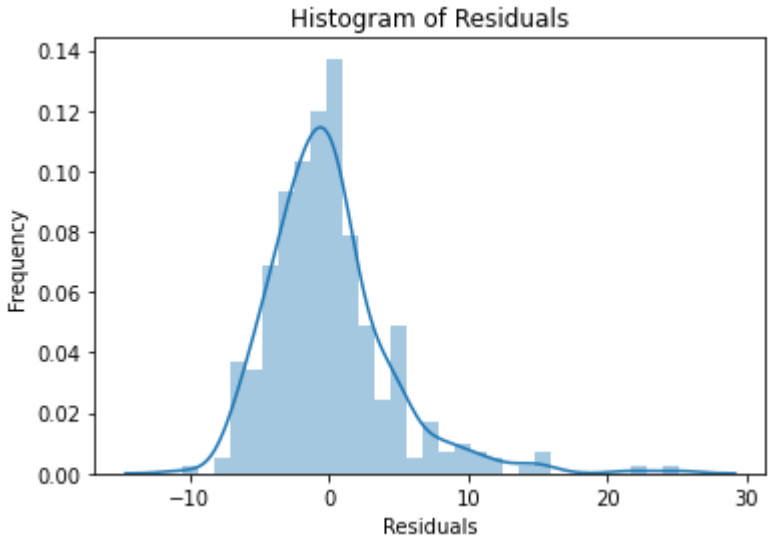


In [23]:
```
plt.scatter(y_pred,y_train-y_pred)
plt.title("Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```



In [24]:
```
sns.distplot(y_train-y_pred)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```

```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\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 `disp
lot` (a figure-level function with similar flexibility) or `histplot` (an axes-
level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [25]:
```
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)-
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: 0.7121818377409185
Adjusted R^2: 0.6850685326005702
MAE: 3.8590055923707407
MSE: 30.05399330712424
RMSE: 5.482152251362985
```