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]:
```
import warnings
warnings.filterwarnings("ignore")
```

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

In [4]:
```
data = pd.DataFrame(boston.data)
```

In [5]:
```
data.head()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	10	11	12
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 [6]:
```
#Adding the feature names to the dataframe
data.columns = boston.feature_names
data.head()
```

Out[6]:

	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 [7]:
```
#Adding target variable to dataframe
data['PRICE'] = boston.target
```

In [8]:
```
#Check the shape of dataframe
data.shape
```

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

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

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

In [10]:
```
data.dtypes
```

Out[10]:
```
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
PRICE      float64
dtype: object
```

In [11]:
```
# Identifying the unique number of values in the dataset
data.nunique()
```

Out[11]:
```
CRIM       504
ZN          26
INDUS       76
CHAS         2
NOX         81
RM         446
AGE        356
DIS        412
RAD          9
TAX         66
PTRATIO     46
B          357
LSTAT      455
PRICE      229
dtype: int64
```

In [12]:
```
# Check for missing values
data.isnull().sum()
```

Out[12]:
```
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
PRICE      0
dtype: int64
```

In [13]:
```
# See rows with missing values
data[data.isnull().any(axis=1)]
```

Out[13]:

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

In [14]:
```
# Viewing the data statistics
data.describe()
```

Out[14]:

	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 [15]:

```
# Finding out the correlation between the features
corr = data.corr()
corr.shape
```

Out[15]:

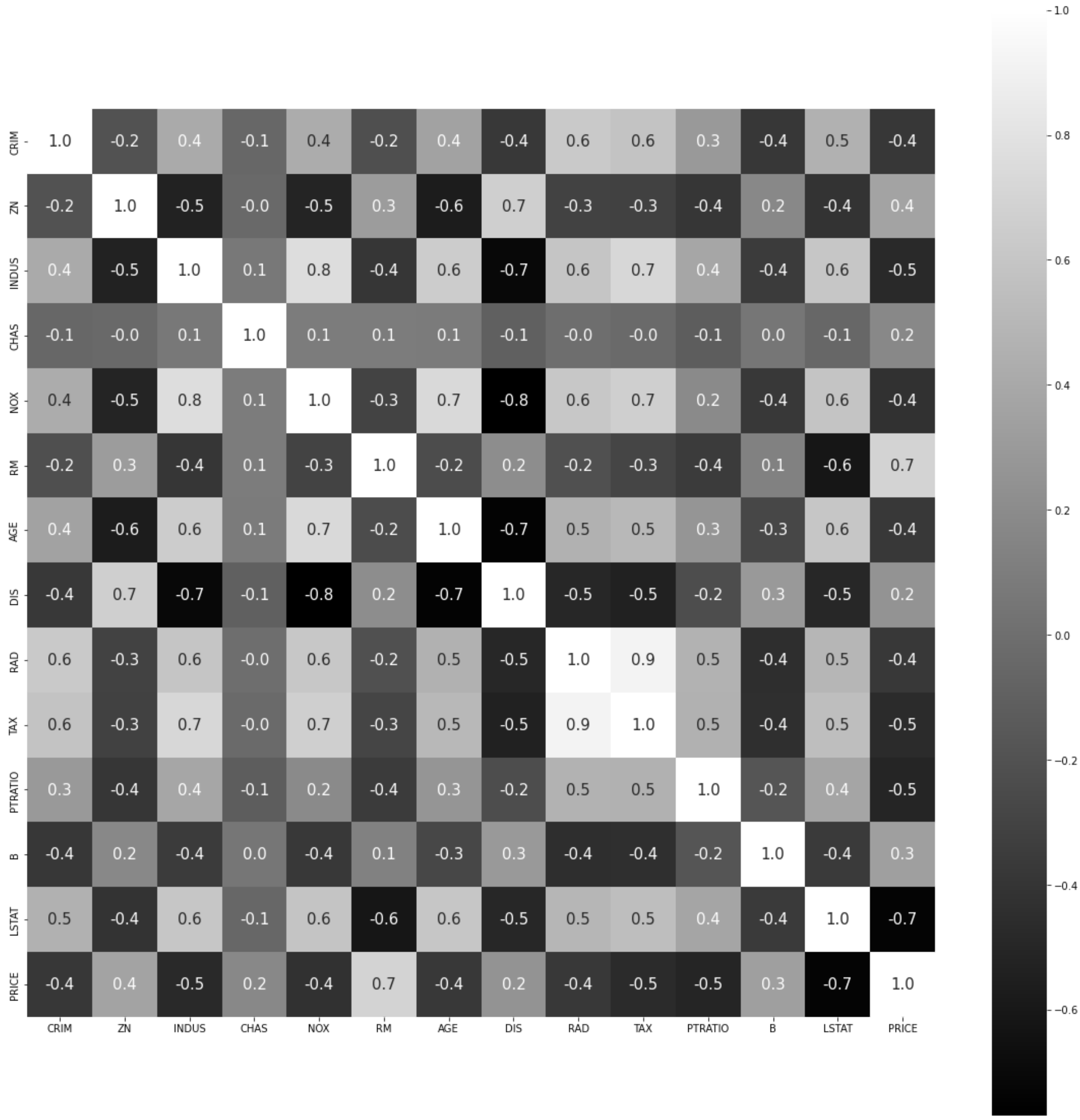
```
(14, 14)
```

In [16]:

```
# Plotting the heatmap of correlation between features
plt.figure(figsize=(20,20))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size
```

Out[16]:

```
<AxesSubplot:>
```



In [17]:
```
# Spliting target variable and independent variables
X = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

In [18]:
```
# Splitting to training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_s
```

In [19]:
```
# Import library for Linear Regression
from sklearn.linear_model import LinearRegression
```

In [20]:
```
# Create a Linear regressor
lm = LinearRegression()

# Train the model using the training sets
lm.fit(X_train, y_train)
```

Out[20]:
```
LinearRegression()
```

In [21]:
```
# Value of y intercept
lm.intercept_
```

Out[21]: 36.357041376595205

In [22]:
```
#Converting the coefficient values to a dataframe
coeffcients = pd.DataFrame([X_train.columns,lm.coef_]).T
coeffcients = coeffcients.rename(columns={0: 'Attribute', 1: 'Coefficients'})
coeffcients
```

Out[22]:

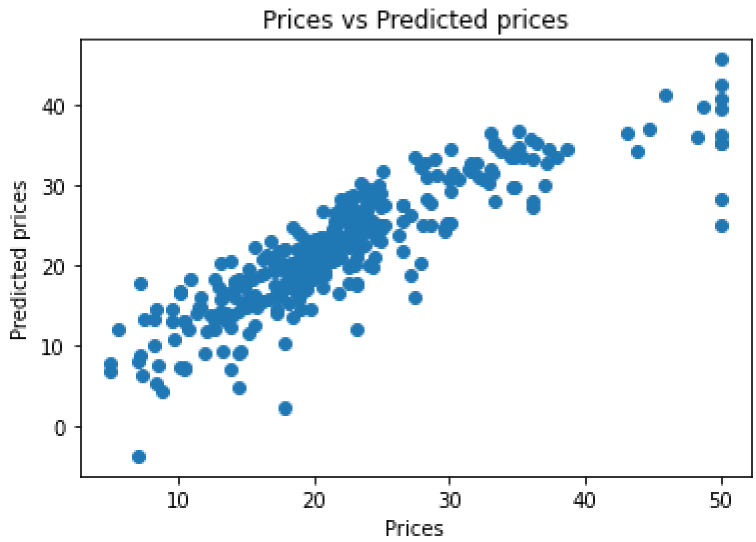
	Attribute	Coefficients
0	CRIM	-0.12257
1	ZN	0.055678
2	INDUS	-0.008834
3	CHAS	4.693448
4	NOX	-14.435783
5	RM	3.28008
6	AGE	-0.003448
7	DIS	-1.552144
8	RAD	0.32625
9	TAX	-0.014067
10	PTRATIO	-0.803275
11	B	0.009354
12	LSTAT	-0.523478

In [23]:
```
# Model prediction on train data
y_pred = lm.predict(X_train)
```

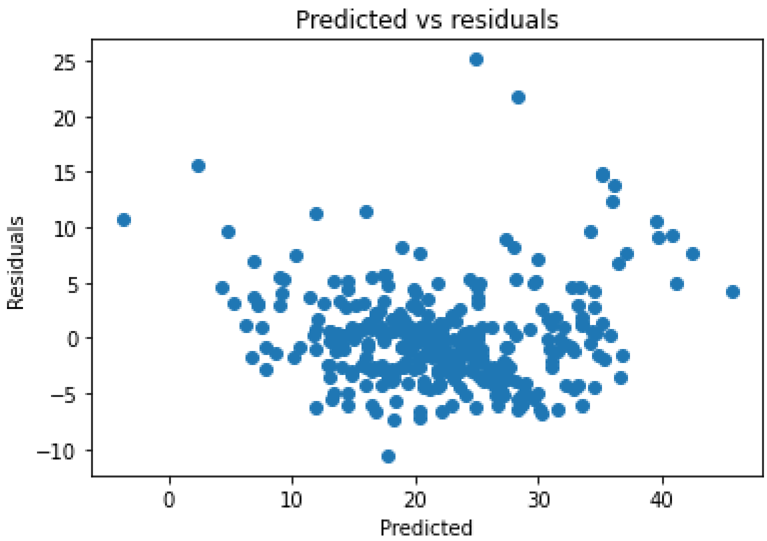
In [24]:
```
# Model Evaluation
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.7465991966746854
Adjusted R^2: 0.736910342429894
MAE: 3.08986109497113
MSE: 19.07368870346903
RMSE: 4.367343437774162
```

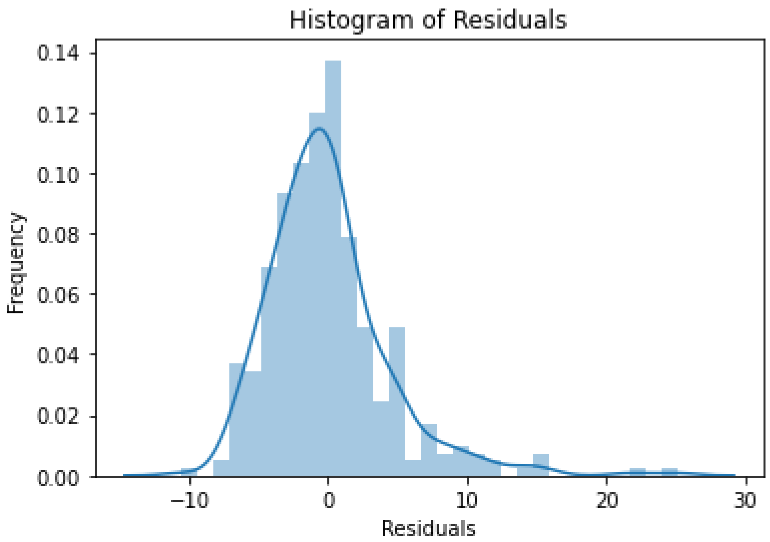
In [25]:
```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("Prices vs Predicted prices")
plt.show()
```



In [26]:
```
# Checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.title("Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```



In [27]:
```
# Checking Normality of errors
sns.distplot(y_train-y_pred)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```



In [28]:
```
# Predicting Test data with the model
y_test_pred = lm.predict(X_test)
```

In [29]:
```
# Model Evaluation
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
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
```

In [30]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [31]:
```
import keras
from keras.layers import Dense, Activation,Dropout
from keras.models import Sequential

model = Sequential()

model.add(Dense(128,activation  = 'relu',input_dim =13))
model.add(Dense(64,activation  = 'relu'))
model.add(Dense(32,activation  = 'relu'))
model.add(Dense(16,activation  = 'relu'))
model.add(Dense(1))
model.compile(optimizer = 'adam',loss = 'mean_squared_error')
```

In [32]:
```
model.fit(X_train, y_train, epochs = 100)
```

```
Epoch 1/100
12/12 [==============================] - 2s 5ms/step - loss: 536.4907
Epoch 2/100
12/12 [==============================] - 0s 4ms/step - loss: 445.2772
Epoch 3/100
12/12 [==============================] - 0s 4ms/step - loss: 260.4927
Epoch 4/100
12/12 [==============================] - 0s 4ms/step - loss: 94.9290
Epoch 5/100
12/12 [==============================] - 0s 10ms/step - loss: 48.9381
Epoch 6/100
12/12 [==============================] - 0s 8ms/step - loss: 29.5932
Epoch 7/100
12/12 [==============================] - 0s 8ms/step - loss: 24.5667
Epoch 8/100
12/12 [==============================] - 0s 6ms/step - loss: 21.6446
Epoch 9/100
12/12 [==============================] - 0s 5ms/step - loss: 19.7617
Epoch 10/100
12/12 [==============================] - 0s 6ms/step - loss: 18.1928
Epoch 11/100
12/12 [==============================] - 0s 7ms/step - loss: 17.3536
Epoch 12/100
12/12 [==============================] - 0s 7ms/step - loss: 16.1515
Epoch 13/100
12/12 [==============================] - 0s 5ms/step - loss: 15.3579
Epoch 14/100
12/12 [==============================] - 0s 4ms/step - loss: 14.7798
Epoch 15/100
12/12 [==============================] - 0s 6ms/step - loss: 14.0140
Epoch 16/100
12/12 [==============================] - 0s 4ms/step - loss: 13.4308
Epoch 17/100
12/12 [==============================] - 0s 5ms/step - loss: 13.3195
Epoch 18/100
12/12 [==============================] - 0s 5ms/step - loss: 13.4252
Epoch 19/100
12/12 [==============================] - 0s 5ms/step - loss: 12.5759
Epoch 20/100
12/12 [==============================] - 0s 5ms/step - loss: 12.3779
Epoch 21/100
12/12 [==============================] - 0s 10ms/step - loss: 12.0664
Epoch 22/100
12/12 [==============================] - 0s 8ms/step - loss: 11.9065
Epoch 23/100
12/12 [==============================] - 0s 5ms/step - loss: 12.2365
Epoch 24/100
12/12 [==============================] - 0s 5ms/step - loss: 11.4611
Epoch 25/100
12/12 [==============================] - 0s 7ms/step - loss: 11.0854
Epoch 26/100
12/12 [==============================] - 0s 4ms/step - loss: 10.9809
Epoch 27/100
12/12 [==============================] - 0s 3ms/step - loss: 10.7466
Epoch 28/100
12/12 [==============================] - 0s 3ms/step - loss: 10.5044
Epoch 29/100
12/12 [==============================] - 0s 5ms/step - loss: 10.5765
Epoch 30/100
12/12 [==============================] - 0s 4ms/step - loss: 10.3270
Epoch 31/100
12/12 [==============================] - 0s 2ms/step - loss: 9.9691
Epoch 32/100
12/12 [==============================] - 0s 2ms/step - loss: 9.9284
```

```
Epoch 33/100
12/12 [==============================] - 0s 2ms/step - loss: 9.7064
Epoch 34/100
12/12 [==============================] - 0s 2ms/step - loss: 9.4673
Epoch 35/100
12/12 [==============================] - 0s 2ms/step - loss: 9.7000
Epoch 36/100
12/12 [==============================] - 0s 3ms/step - loss: 9.7727
Epoch 37/100
12/12 [==============================] - 0s 3ms/step - loss: 9.1311
Epoch 38/100
12/12 [==============================] - 0s 3ms/step - loss: 9.0136
Epoch 39/100
12/12 [==============================] - 0s 4ms/step - loss: 9.1342
Epoch 40/100
12/12 [==============================] - 0s 3ms/step - loss: 8.9718
Epoch 41/100
12/12 [==============================] - 0s 3ms/step - loss: 8.7594
Epoch 42/100
12/12 [==============================] - 0s 3ms/step - loss: 11.4910
Epoch 43/100
12/12 [==============================] - 0s 3ms/step - loss: 9.8499
Epoch 44/100
12/12 [==============================] - 0s 3ms/step - loss: 9.6734
Epoch 45/100
12/12 [==============================] - 0s 4ms/step - loss: 9.8016
Epoch 46/100
12/12 [==============================] - 0s 4ms/step - loss: 8.8640
Epoch 47/100
12/12 [==============================] - 0s 4ms/step - loss: 8.4947
Epoch 48/100
12/12 [==============================] - 0s 3ms/step - loss: 8.0373
Epoch 49/100
12/12 [==============================] - 0s 6ms/step - loss: 7.9219
Epoch 50/100
12/12 [==============================] - 0s 4ms/step - loss: 7.8490
Epoch 51/100
12/12 [==============================] - 0s 3ms/step - loss: 7.6934
Epoch 52/100
12/12 [==============================] - 0s 4ms/step - loss: 7.5885
Epoch 53/100
12/12 [==============================] - 0s 4ms/step - loss: 7.4943
Epoch 54/100
12/12 [==============================] - 0s 4ms/step - loss: 7.4490
Epoch 55/100
12/12 [==============================] - 0s 5ms/step - loss: 7.4370
Epoch 56/100
12/12 [==============================] - 0s 4ms/step - loss: 7.4392
Epoch 57/100
12/12 [==============================] - 0s 4ms/step - loss: 10.0534
Epoch 58/100
12/12 [==============================] - 0s 5ms/step - loss: 9.1899
Epoch 59/100
12/12 [==============================] - 0s 4ms/step - loss: 7.8947
Epoch 60/100
12/12 [==============================] - 0s 4ms/step - loss: 7.2435
Epoch 61/100
12/12 [==============================] - 0s 4ms/step - loss: 6.8620
Epoch 62/100
12/12 [==============================] - 0s 5ms/step - loss: 7.0719
Epoch 63/100
12/12 [==============================] - 0s 4ms/step - loss: 6.9407
Epoch 64/100
12/12 [==============================] - 0s 3ms/step - loss: 6.8590
```

```
Epoch 65/100
12/12 [==============================] - 0s 4ms/step - loss: 6.6452
Epoch 66/100
12/12 [==============================] - 0s 4ms/step - loss: 6.3786
Epoch 67/100
12/12 [==============================] - 0s 4ms/step - loss: 6.1708
Epoch 68/100
12/12 [==============================] - 0s 4ms/step - loss: 6.2887
Epoch 69/100
12/12 [==============================] - 0s 4ms/step - loss: 6.0254
Epoch 70/100
12/12 [==============================] - 0s 4ms/step - loss: 6.1180
Epoch 71/100
12/12 [==============================] - 0s 5ms/step - loss: 6.5012
Epoch 72/100
12/12 [==============================] - 0s 6ms/step - loss: 5.8355
Epoch 73/100
12/12 [==============================] - 0s 4ms/step - loss: 5.9898
Epoch 74/100
12/12 [==============================] - 0s 4ms/step - loss: 6.0399
Epoch 75/100
12/12 [==============================] - 0s 4ms/step - loss: 6.2273
Epoch 76/100
12/12 [==============================] - 0s 4ms/step - loss: 5.9231
Epoch 77/100
12/12 [==============================] - 0s 4ms/step - loss: 6.2560
Epoch 78/100
12/12 [==============================] - 0s 3ms/step - loss: 5.7622
Epoch 79/100
12/12 [==============================] - 0s 3ms/step - loss: 5.7326
Epoch 80/100
12/12 [==============================] - 0s 4ms/step - loss: 5.5256
Epoch 81/100
12/12 [==============================] - 0s 5ms/step - loss: 5.8990
Epoch 82/100
12/12 [==============================] - 0s 4ms/step - loss: 6.4134
Epoch 83/100
12/12 [==============================] - 0s 4ms/step - loss: 7.2626
Epoch 84/100
12/12 [==============================] - 0s 4ms/step - loss: 9.7168
Epoch 85/100
12/12 [==============================] - 0s 3ms/step - loss: 7.5276
Epoch 86/100
12/12 [==============================] - 0s 4ms/step - loss: 5.7990
Epoch 87/100
12/12 [==============================] - 0s 4ms/step - loss: 5.4174
Epoch 88/100
12/12 [==============================] - 0s 4ms/step - loss: 5.1878
Epoch 89/100
12/12 [==============================] - 0s 4ms/step - loss: 5.6626
Epoch 90/100
12/12 [==============================] - 0s 4ms/step - loss: 5.3683
Epoch 91/100
12/12 [==============================] - 0s 3ms/step - loss: 5.1334
Epoch 92/100
12/12 [==============================] - 0s 5ms/step - loss: 4.8568
Epoch 93/100
12/12 [==============================] - 0s 3ms/step - loss: 4.9240
Epoch 94/100
12/12 [==============================] - 0s 5ms/step - loss: 5.8374
Epoch 95/100
12/12 [==============================] - 0s 7ms/step - loss: 4.9842
Epoch 96/100
12/12 [==============================] - 0s 7ms/step - loss: 4.9352
```

```
Epoch 97/100
12/12 [==============================] - 0s 4ms/step - loss: 4.9905
Epoch 98/100
12/12 [==============================] - 0s 5ms/step - loss: 4.8173
Epoch 99/100
12/12 [==============================] - 0s 5ms/step - loss: 4.5281
Epoch 100/100
12/12 [==============================] - 0s 4ms/step - loss: 5.1569
```

Out[32]:
```
<keras.callbacks.History at 0x2649d91b1f0>
```

In [33]:
```python
y_pred = model.predict(X_test)
```

```
5/5 [==============================] - 0s 3ms/step
```

In [34]:
```python
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print(r2)
```

```
0.8402226765414398
```

In [35]:
```python
from sklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
print(rmse)
```

```
4.084600306583784
```

In [ ]: