In [ ]:
```
#Introduction
'''
In this project, I delve into the Boston Housing dataset, utilizing Supervised Learning to
predict median home values.

My analysis includes exploring data features and applying Linear Regression, enhanced with polynomial features,
to improve prediction accuracy.


'''
```

In [11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
```

In [12]:
```
# Load the dataset
file_path = 'boston.csv'
boston_data = pd.read_csv(file_path, header=1)
boston_data.head()
```

Out[12]:

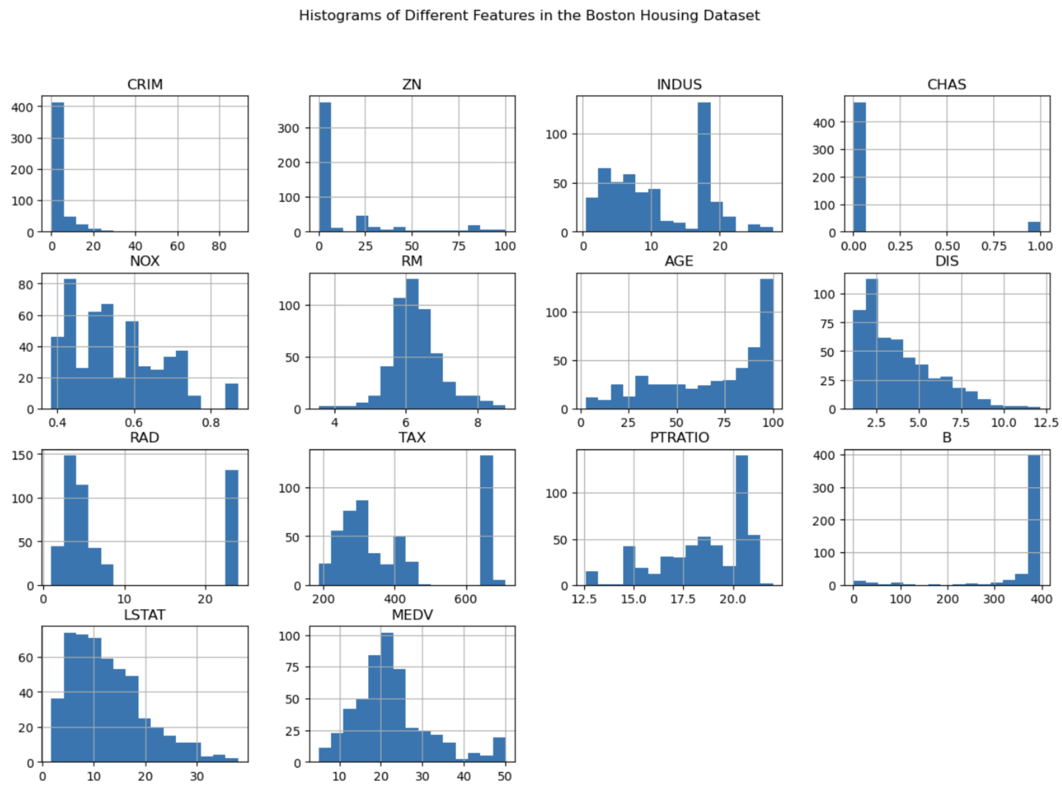
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	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 [13]:
```
#statistical summary of the dataset
boston_data.describe()
```

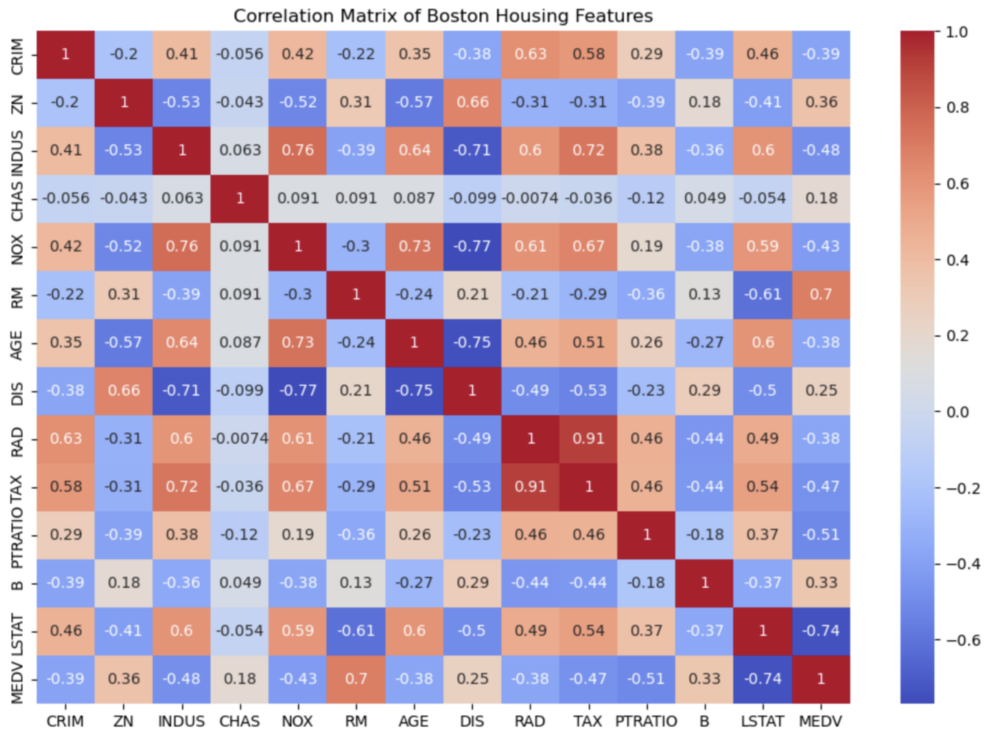
Out[13]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.

In [14]:
```python
# EDA
boston_data.hist(bins=15, figsize=(15, 10), layout=(4, 4))
plt.suptitle("Histograms of Different Features in the Boston Housing Dataset")
plt.show()
```



In [15]:
```
# correlation matrix
correlation_matrix = boston_data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix of Boston Housing Features")
plt.show()
```

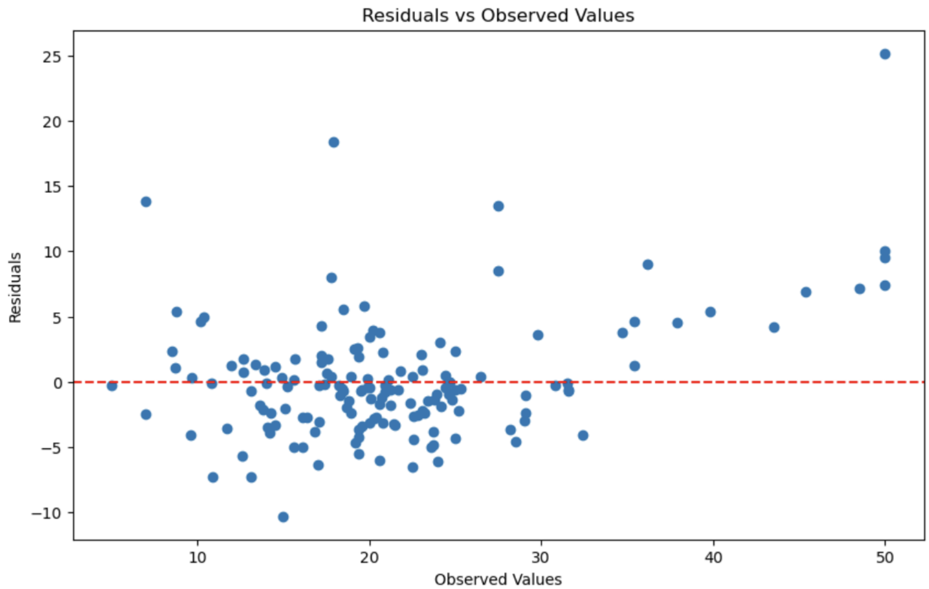


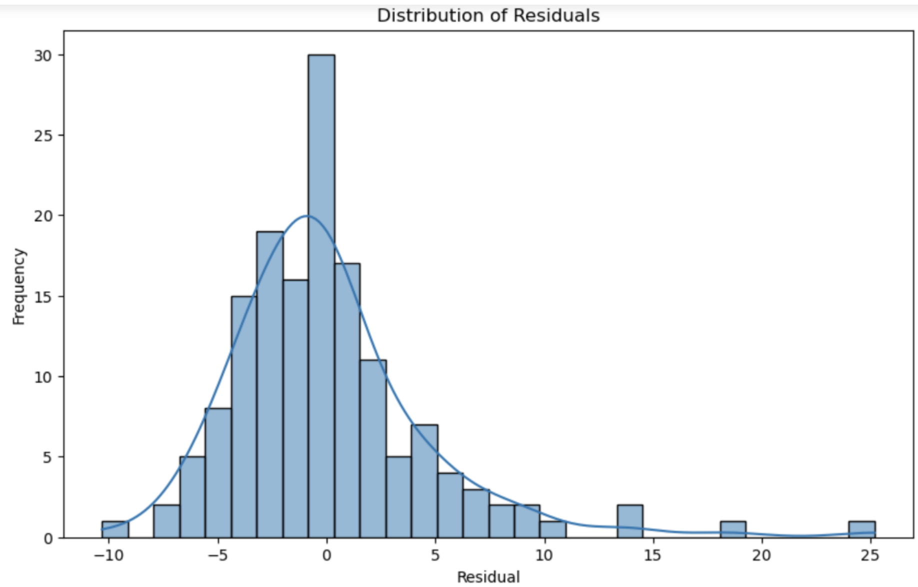
In [16]:
```
X = boston_data.drop('MEDV', axis=1)
y = boston_data['MEDV']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [17]:
```
# Training a Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred = lr_model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

Out[17]:
```
(21.517444231177038, 0.7112260057484956)
```

In [18]:
```
# plotting residuals
residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
plt.scatter(y_test, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Observed Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Observed Values')
plt.show()

# Histogram
plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribution of Residuals')
plt.xlabel('Residual')
plt.ylabel('Frequency')
plt.show()
```


Residuals vs Observed Values
Residuals
Observed Values
25
20
15
10
5
0
−5
−10
10
20
30
40
50



In [19]:
```
# Creating polynomial features
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)

X_train_poly, X_test_poly, y_train_poly, y_test_poly = train_test_split(X_poly, y, test_size=0.3, random_state=42)

lr_poly_model = LinearRegression()
lr_poly_model.fit(X_train_poly, y_train_poly)

y_pred_poly = lr_poly_model.predict(X_test_poly)
mse_poly = mean_squared_error(y_test_poly, y_pred_poly)
r2_poly = r2_score(y_test_poly, y_pred_poly)

mse_poly, r2_poly
```

Out[19]: (25.257540302651694, 0.6610321969559437)

In [1]:
```
#Conclusion
''''
My project concludes with a model that predicts Boston housing prices.

I improved my predictions by adding polynomial features to our analysis.

The model's accuracy and potential improvements are visible in the residual plots.

This study provides a framework for similar predictive modeling tasks.
''''
```

In [ ]:
```
#Future
#Real Estate Valuation, Urban Planning, Investment Analysis, Housing Market Analysis and more!
```