In [1]:
```python
# Task 2: Linear Regression - Boston Housing Dataset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
```

In [2]:
```python
# Load dataset
boston = fetch_openml(name='boston', version=1, as_frame=True)
df = boston.frame
```

In [3]:
```python
# Display first few records
print("Sample Data:\n", df.head())
```

```
Sample Data:
       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  296.0     15.3
1  0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242.0     17.8
2  0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242.0     17.8
3  0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222.0     18.7
4  0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222.0     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
```

In [4]:
```python
# Dataset info
print("\nDataset Info:")
print(df.info())
```

```
Dataset Info:
<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    category
 4   NOX      506 non-null    float64 
 5   RM       506 non-null    float64 
 6   AGE      506 non-null    float64 
 7   DIS      506 non-null    float64 
 8   RAD      506 non-null    category
 9   TAX      506 non-null    float64 
 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: category(2), float64(12)
memory usage: 49.0 KB
None
```

In [5]:
```
# Target and features
X = df.drop(columns=['MEDV'])  # MEDV is the target (median value of owner-occ
y = df['MEDV']
```

In [6]:
```
# -------------------- Data Preprocessing --------------------
# Check for missing values
print("\nMissing values:\n", df.isnull().sum())
```

```
Missing values:
 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
MEDV       0
dtype: int64
```

In [7]:
```
# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [8]:
```python
# -------------------- Split Data --------------------
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2
```

In [9]:
```python
# -------------------- Train Linear Regression Model --------------------
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[9]:

▾ LinearRegression

Parameters

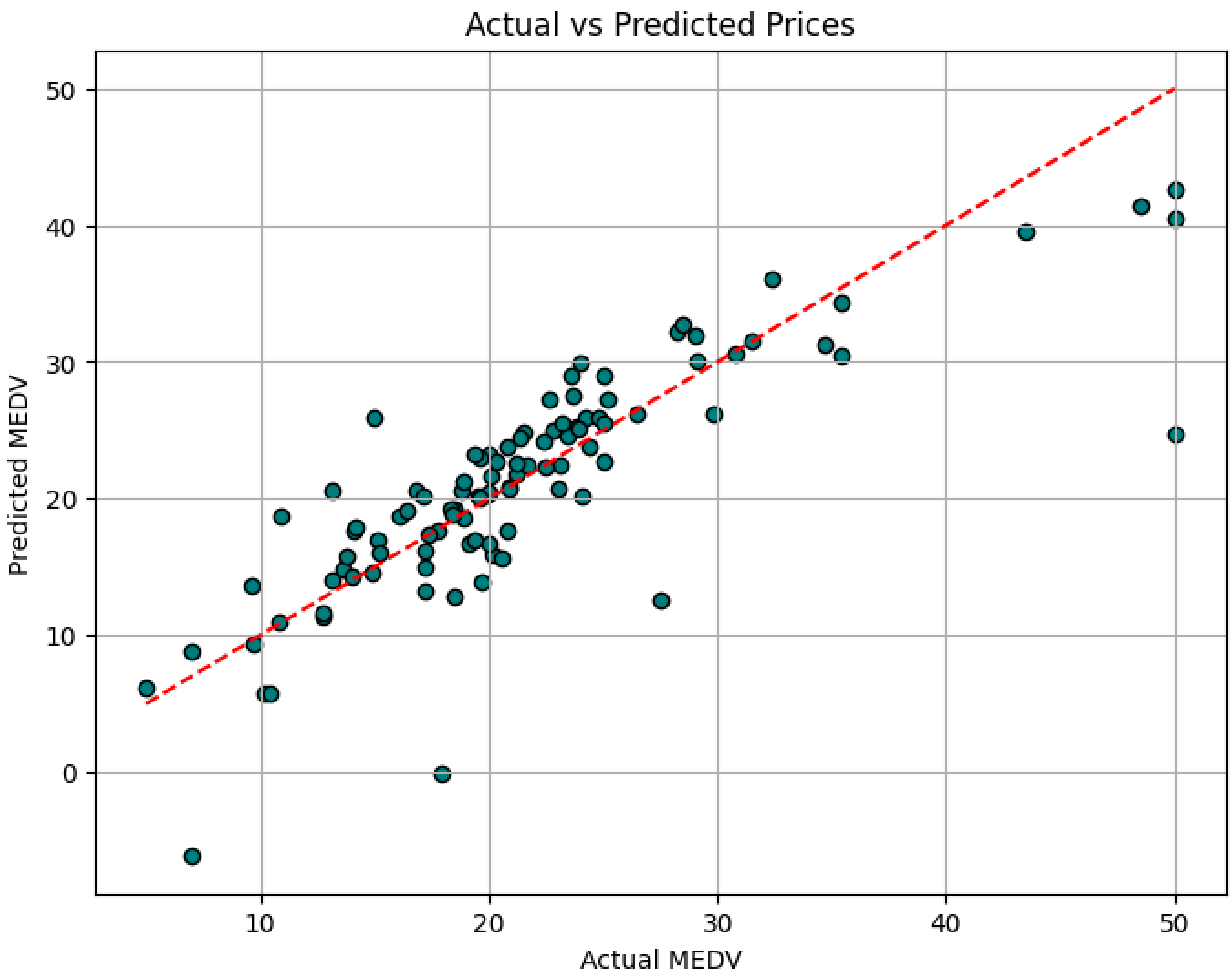
	fit_intercept	True
	copy_X	True
	tol	1e-06
	n_jobs	None
	positive	False

In [10]:
```python
# Predict on test set
y_pred = model.predict(X_test)
```

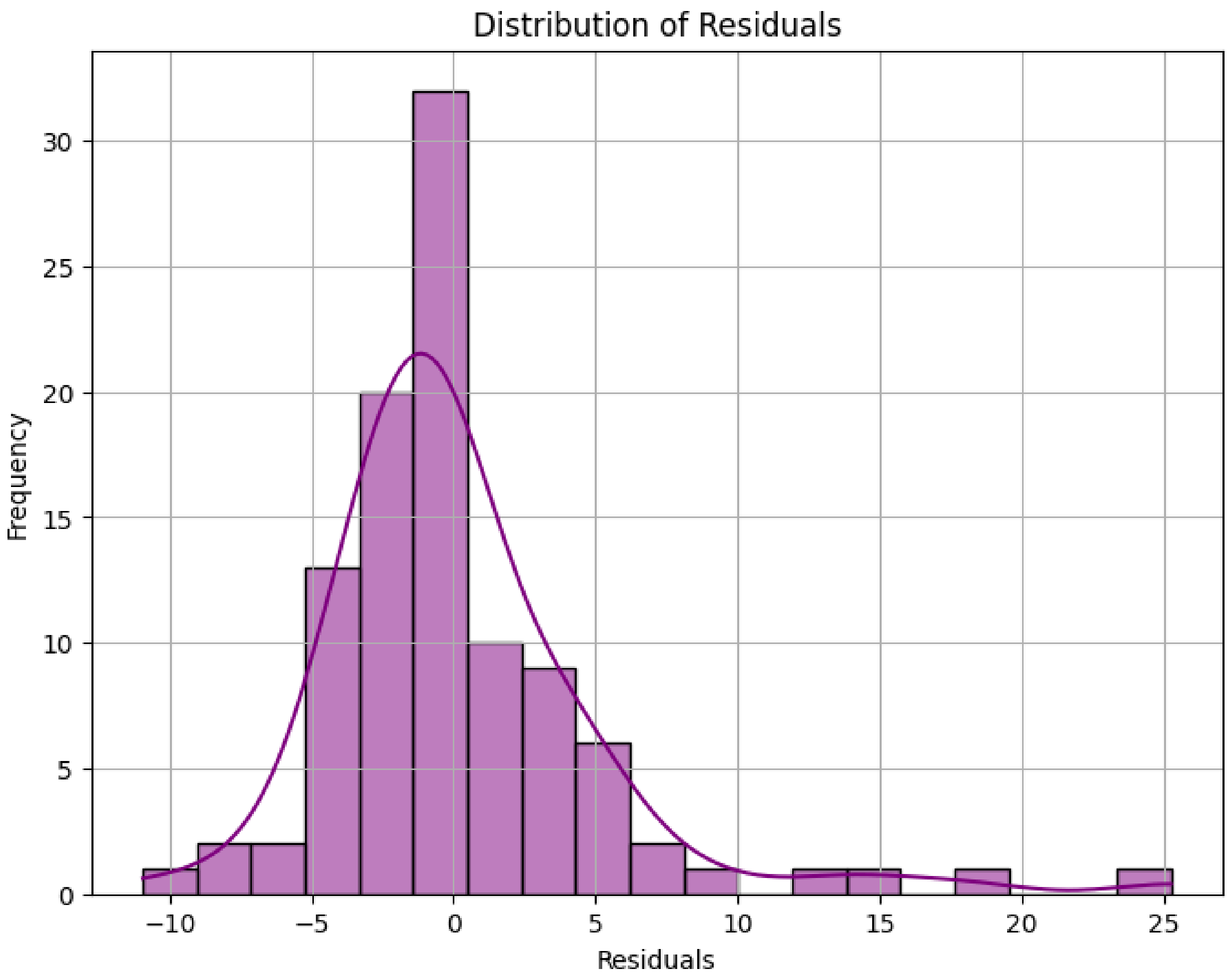
In [11]:
```python
# -------------------- Evaluation --------------------
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print("\nModel Performance:")
print("R² Score:", r2)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

```
Model Performance:
R² Score: 0.6687594935356318
Mean Squared Error: 24.291119474973527
Root Mean Squared Error: 4.928602182665338
```

In [12]:
```python
# -------------------- Plot: Actual vs Predicted --------------------
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color='teal', edgecolor='black')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
plt.xlabel('Actual MEDV')
plt.ylabel('Predicted MEDV')
plt.title('Actual vs Predicted Prices')
plt.grid(True)
plt.show()
```



In [13]:
```
# -------------------- Plot: Residuals --------------------
residuals = y_test - y_pred
plt.figure(figsize=(8,6))
sns.histplot(residuals, kde=True, color='purple')
plt.title('Distribution of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



Distribution of Residuals
Frequency
Residuals
0
5
10
15
20
25
30
−10
−5
0
5
10
15
20
25

In [ ]:

Task 2: Linear Regression Project – Predicting House Prices

Dataset Used: Boston Housing Dataset

Objective: Predict the median value of houses using regression and apply necessary preprocessing steps.

Dataset Overview

- The Boston Housing dataset consists of 506 entries and 14 attributes.
- The target variable is `MEDV` (Median value of owner-occupied homes in $1000s).
- Features include crime rate, number of rooms (`RM`), distance to employment centers (`DIS`), etc.

Preprocessing Steps

- Checked for missing values (None found).
- Categorical features like `CHAS` and `RAD` were kept as-is due to limited categories and presence of numerical encoding.
- Standardized all features using `StandardScaler` to bring them to a common scale for regression.

Model Used: Linear Regression

- **Why Linear Regression?**
  - Simple and interpretable model for continuous target prediction.
  - Assumes linear relationships between independent and dependent variables.
  - Ideal for understanding basic regression performance on a real dataset.

Evaluation Results

Metric	Value
R^2 Score	0.669

Metric	Value
MSE (Mean Squared Error)	24.29
RMSE (Root Mean Squared Error)	4.93

- **R^2 Score of 0.669** means that the model explains ~66.9% of the variance in the target variable.
- **Residuals** were normally distributed, indicating the model is fairly unbiased.
- **Predicted vs Actual Plot** showed good linear alignment but also some variance for higher-priced houses.

Conclusion

- The linear regression model performs reasonably well on this dataset.
- It can be improved by:
  - Feature engineering (interaction terms or polynomial features)
  - Using more robust models like Ridge, Lasso, or Random Forest
  - Applying cross-validation to reduce overfitting

TASK 2 - Model Fit and Residual Plots

Figure 1: Model Fit - Actual vs Predicted Prices

Figure 2: Residual Plot - Distribution of Residuals