```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```python
df = pd.read_csv('Boston.csv')
```

```python
df.head()
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	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	2	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	3	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	4	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	5	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

Next steps: Generate code with df New interactive sheet

```python
df.columns
```

```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```


```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```python
df = pd.read_csv('Boston.csv')
```

```python
df.head()
```

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	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	2	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	3	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	4	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	5	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

Next steps: Generate code with df New interactive sheet

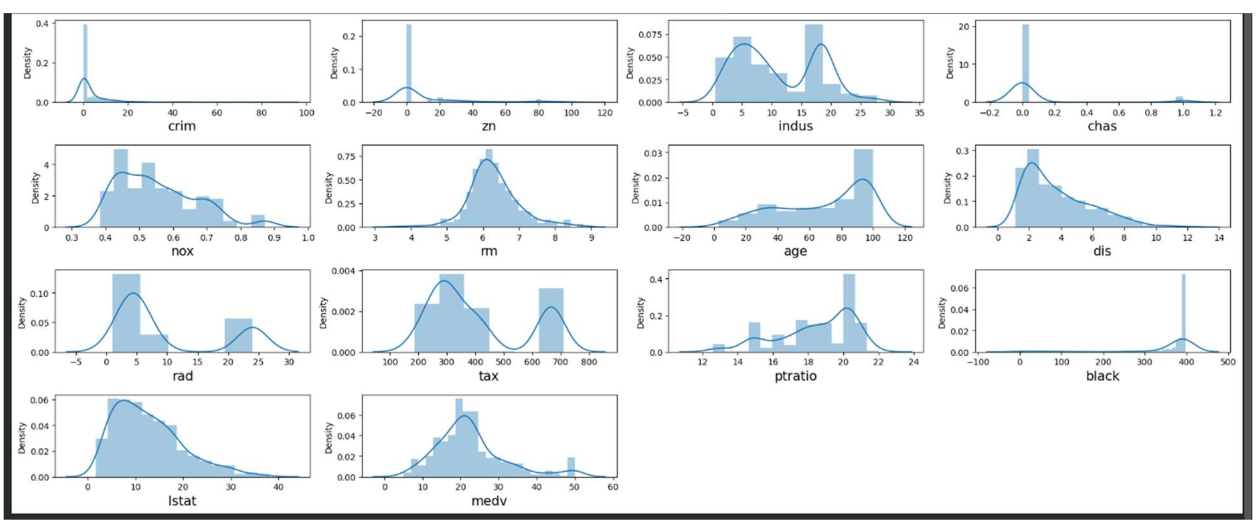
```python
df.columns
```

```
Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

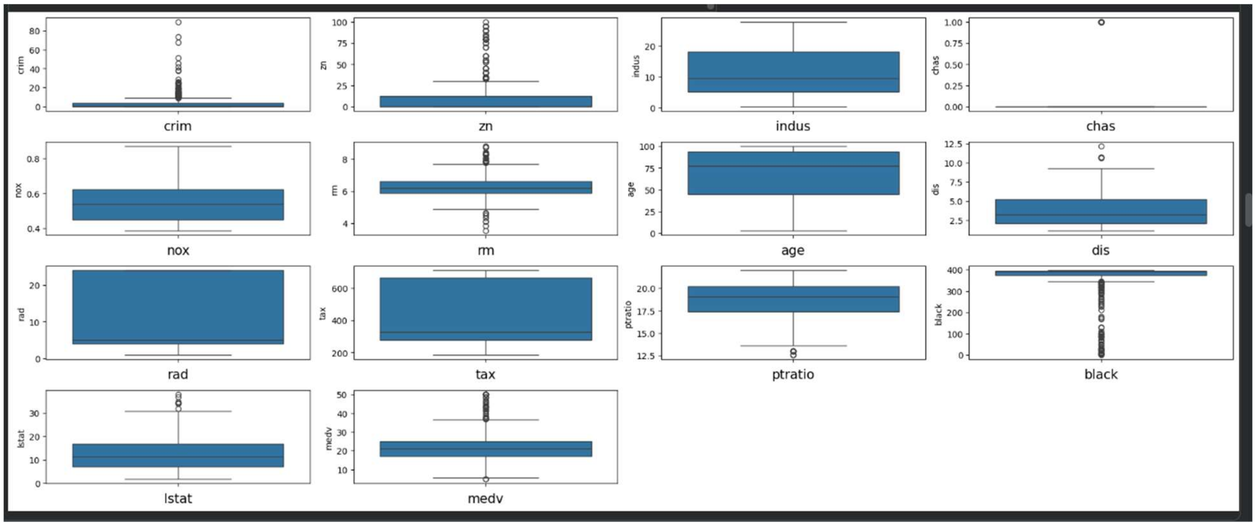
```python
df_clean.describe().T
```

	count	mean	std	min	25%	50%	75%	max
crim	506.0	3.613524	8.601545	0.00632	0.082045	0.25651	3.677083	88.9762
zn	506.0	11.363636	23.322453	0.00000	0.000000	0.00000	12.500000	100.0000
indus	506.0	11.136779	6.860353	0.46000	5.190000	9.69000	18.100000	27.7400
chas	506.0	0.069170	0.253994	0.00000	0.000000	0.00000	0.000000	1.0000
nox	506.0	0.554695	0.115878	0.38500	0.449000	0.53800	0.624000	0.8710
rm	506.0	6.284634	0.702617	3.56100	5.885500	6.20850	6.623500	8.7800
age	506.0	68.574901	28.148861	2.90000	45.025000	77.50000	94.075000	100.0000
dis	506.0	3.795043	2.105710	1.12960	2.100175	3.20745	5.188425	12.1265
rad	506.0	9.549407	8.707259	1.00000	4.000000	5.00000	24.000000	24.0000
tax	506.0	408.237154	168.537116	187.00000	279.000000	330.00000	666.000000	711.0000
ptratio	506.0	18.455534	2.164946	12.60000	17.400000	19.05000	20.200000	22.0000
black	506.0	356.674032	91.294864	0.32000	375.377500	391.44000	396.225000	396.9000
lstat	506.0	12.653063	7.141062	1.73000	6.950000	11.36000	16.955000	37.9700
medv	506.0	22.532806	9.197104	5.00000	17.025000	21.20000	25.000000	50.0000

```python
plt.figure(figsize=(20, 40))

plotnum = 1

for columns in df:
    if plotnum <= len(df_clean.columns):
        plt.subplot(20, 4, plotnum)
        sns.distplot(df_clean[columns])
        plt.xlabel(columns, fontsize=15)
    plotnum += 1
plt.tight_layout()
plt.show
```



```python
plt.figure(figsize=(20, 40))

plotnum = 1

for columns in df:
    if plotnum <= len(df_clean.columns):
        plt.subplot(20, 4, plotnum)
        sns.boxplot(df_clean[columns])
        plt.xlabel(columns, fontsize=15)
    plotnum += 1
plt.tight_layout()
plt.show
```




```python
colu = ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv']

corr = df_clean[colu].corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr, fmt='.2f', cbar=True, square=True, annot=True, annot_kws={'size':9})
```

```python
x = df_clean.drop(columns=['medv'], axis=1)
y = df_clean.medv
```

```python
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

```python
x_scaled = pd.DataFrame(x_scaled, columns=x.columns)
```

```python
print(f"x shape  :  {x_scaled.shape}")
print(f"y shape  :  {y.shape}")
```

```
x shape  :  (506, 13)
y shape  :  (506,)
```

Start coding or generate with AI.

```python
x_train,x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.3, random_state=42)
```

```python
model = LinearRegression()
model.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

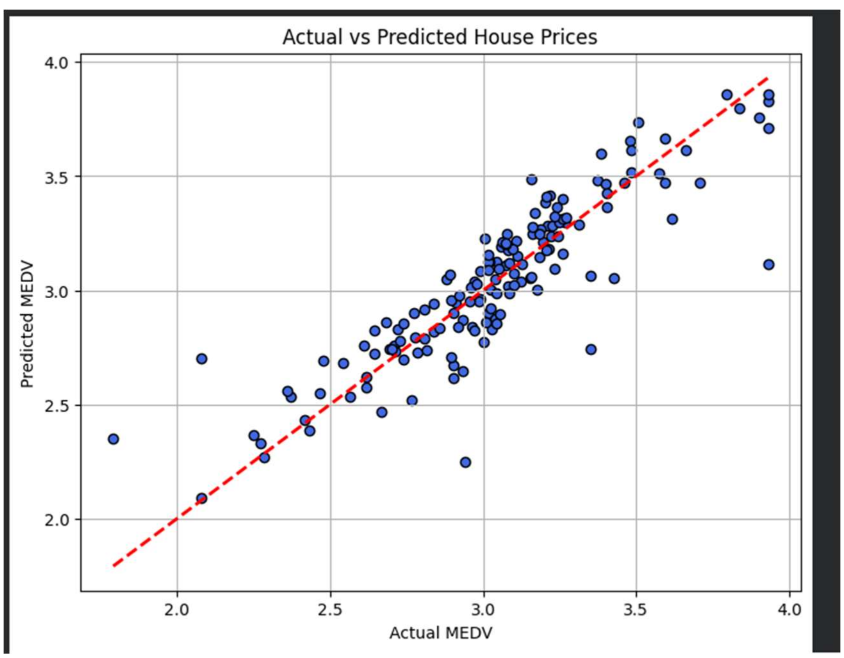
```python
y_predict = model.predict(x_test)
```

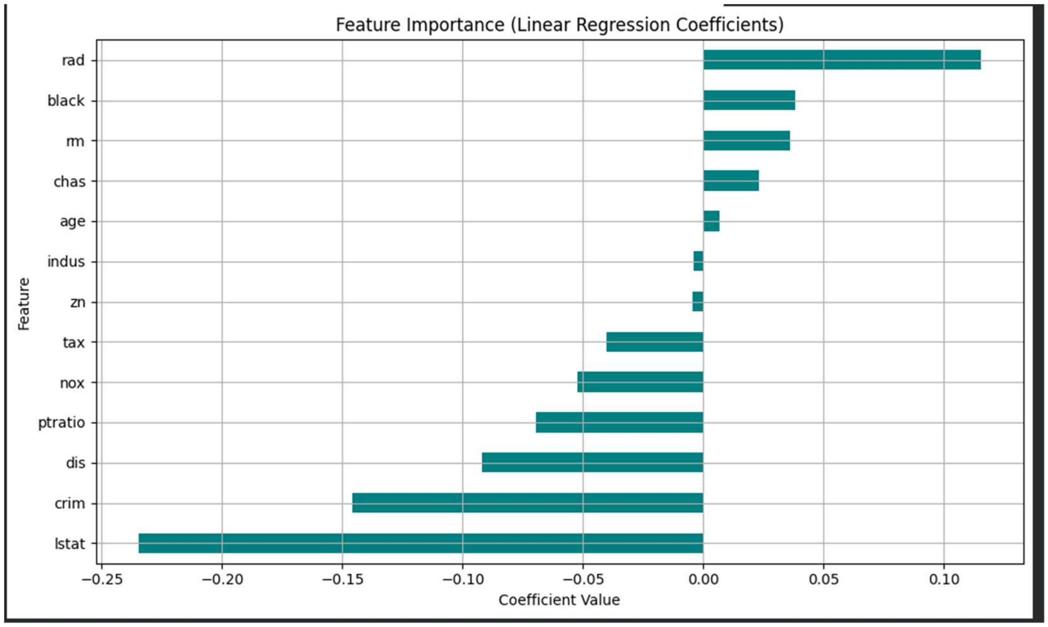
```python
MAE = mean_absolute_error(y_test, y_predict)
MSE = mean_squared_error(y_test, y_predict)
RMSE = np.sqrt(MSE)
R2 = r2_score(y_test, y_predict)
```

```
print("Model Evaluate Matrics")
print(f"Mean Absolute Error (MAE): {MAE:.2f}")
print(f"Mean Squared Error (MSE): {MSE:.2f}")
print(f"Root Mean Squared Error (RMSE): {RMSE:.2f}")
print(f"R-squared Score (R²): {R2:.2f}")
```

```
Model Evaluate Matrics
Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.03
Root Mean Squared Error (RMSE): 0.17
R-squared Score (R²): 0.78
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_predict, color='royalblue', edgecolor='k')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel("Actual MEDV")
plt.ylabel("Predicted MEDV")
plt.title("Actual vs Predicted House Prices")
plt.grid(True)
plt.show()

# ===== Analyze the model parameters (the effect of each feature on the MEDV) =====
coefficients = pd.Series(model.coef_, index=x.columns)
coefficients = coefficients.sort_values()

# Bar plot of the most important effects
plt.figure(figsize=(10, 6))
coefficients.plot(kind='barh', color='teal')
plt.title("Feature Importance (Linear Regression Coefficients)")
plt.xlabel("Coefficient Value")
plt.ylabel("Feature")
plt.grid(True)
plt.tight_layout()
plt.show()
```





1. Data Loading and Initial Inspection

Libraries Imported: Essential libraries like numpy, pandas, matplotlib, seaborn, sklearn. StandardScaler were imported for data manipulation, visualization, and machine learning.

Data Loading: The Boston.csv file was loaded into a pandas DataFrame named df.

Initial View: df.head() was used to display the first few rows, and df.columns showed all column names, revealing an 'Unnamed: 0' column.

Data Information: df.info() provided a summary of the DataFrame, including data types and non-null counts for each column.

2. Data Cleaning

Dropping Unnecessary Column: The 'Unnamed: 0' column, which appeared to be an artifact of the CSV export, was dropped using df.drop('Unnamed: 0', axis=1, inplace=True).

Duplicate Check: df.duplicated().sum() was used to check for duplicate rows, and no duplicates were found.

Missing Values: df.isnull().sum() confirmed that there were no missing values in the dataset initially.

3. Exploratory Data Analysis (EDA)

Descriptive Statistics: df_clean.describe().T provided summary statistics (mean, std, min, max, quartiles) for all numerical columns, giving insights into their distributions.

Distribution Plots: Histograms (using sns.distplot) were generated for all features to visualize their distributions. This helped identify skewed features and potential outliers.

Box Plots: Box plots (using sns.boxplot) were created for all features to visually identify outliers more clearly.

Outlier Detection & Handling: The Interquartile Range (IQR) method was applied to numerical columns to detect outliers. The code iterated through columns, calculated IQR, and identified outliers. While the code for removing outliers (df_clean = df_clean[(df_clean[col] >= lower_bound) | (df_clean[col] <= upper_bound)]) was present, it seems to have significantly reduced the DataFrame size potentially by filtering rows based on a problematic | (OR) condition instead of & (AND) to keep non-outliers. This might have led to an unexpected data loss or transformation.

Pair Plot: sns.pairplot(df_clean) was used to visualize pairwise relationships between all variables, which is useful for understanding correlations and dependencies.

Correlation Heatmap: A correlation heatmap was generated using sns.heatmap to show the correlation matrix between all features and the target variable (medv). This helps identify features that are strongly correlated with the target or with each other.

4. Feature Engineering

Log Transformation: To address skewness observed in the distribution plots, all columns in df_clean were transformed using np.log1p(). This transformation is often applied to positively skewed data to make it more Gaussian-like, which can improve the performance of linear models.

5. Model Preparation

Feature and Target Split: The dataset was split into features (x) and the target variable (y, which is medv).

Feature Scaling: StandardScaler was used to standardize the features (x_scaled). Standardization scales features to have a mean of 0 and a standard deviation of 1, which is crucial for many machine learning algorithms, including Linear Regression, to prevent features with larger scales from dominating the learning process.

Train-Test Split: The scaled features (x_scaled) and target (y) were divided into training and testing sets using train_test_split with a test_size of 30% and random_state=42 for reproducibility.

6. Model Training and Evaluation

Model Initialization & Training: A LinearRegression model was initialized and then trained (model.fit) using the training data (x_train, y_train).

Prediction: The trained model was used to make predictions (y_predict) on the test set (x_test).

Evaluation Metrics: The model's performance was evaluated using standard regression metrics:

Mean Absolute Error (MAE): 0.12
Mean Squared Error (MSE): 0.03
Root Mean Squared Error (RMSE): 0.17
R-squared Score (R^2): 0.78
Visualization of Predictions: A scatter plot of actual vs. predicted medv values was generated, along with a red dashed line representing perfect predictions, to visually assess model performance.

Feature Importance: A bar plot of the model coefficients was created to show the 'importance' or impact of each feature on the medv prediction. The coefficients, sorted, indicate which features have the strongest positive or negative influence on the target variable.