MAHATI AKELLA 22070521027

```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
# Step 2: Load Dataset
df = pd.read_csv("HousingData.csv")

# Step 3: Inspect Dataset
print(df.head())
print(df.info())
print(df.isnull().sum())
```

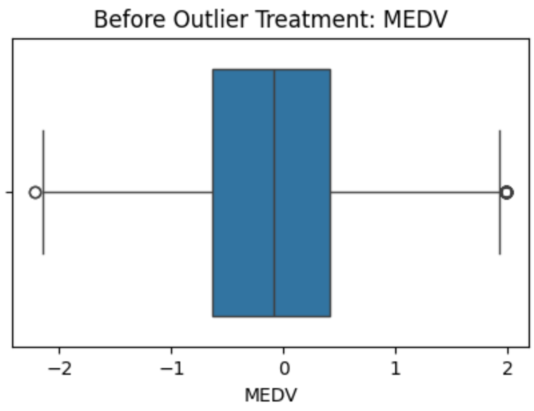
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     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    NaN  36.2
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    int64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
# Step 4: Handle Missing Values
# Separate numerical and categorical columns
num_cols = df.select_dtypes(include=[np.number]).columns
cat_cols = df.select_dtypes(include=['object']).columns
```

```
# Fill missing values for numerical columns with median
for col in num_cols:
    df[col].fillna(df[col].median(), inplace=True)

# Fill missing values for categorical columns with mode
for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

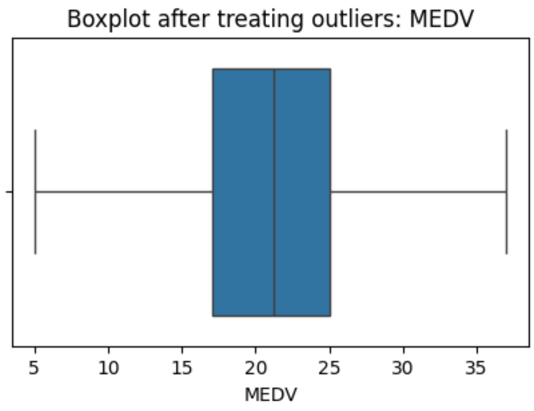
```
/tmp/ipython-input-3101236779.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series throug
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df[col].fillna(df[col].median(), inplace=True)
```

```
# Boxplot BEFORE
plt.figure(figsize=(5,3))
sns.boxplot(x=df[col])
plt.title(f'Before Outlier Treatment: {col}')
plt.show()
```



```
# Step 5: Outlier Detection & Treatment (IQR method)
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower_bound, lower_bound,np.where(df[col] > upper_bound, upper_bound, df[col]))
```

```
#Visualize boxplot
plt.figure(figsize=(5,3))
sns.boxplot(x=df[col])
plt.title(f'Boxplot after treating outliers: {col}')
plt.show()
```



```
# Step 6: Encode Categorical Variables (Label Encoding)
label_encoder = LabelEncoder()
for col in cat_cols:
    df[col] = label_encoder.fit_transform(df[col])
```

```
# Step 7: Feature Scaling (Standardization)
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])
```

```
# Step 8: Save Cleaned Dataset
df.to_csv("HousingData_Cleaned.csv", index=False)

print("Data cleaning and preprocessing completed. Saved as HousingData_Cleaned.csv")
```

```
Data cleaning and preprocessing completed. Saved as HousingData_Cleaned.csv
```

SUMMARY TABLE FOR BEFORE AND AFTER

```
# Outlier count summary BEFORE treatment
outlier_summary_before = {}
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()
    outlier_summary_before[col] = outliers

# Treat outliers using IQR method
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower_bound, lower_bound,
                       np.where(df[col] > upper_bound, upper_bound, df[col]))

# Outlier count summary AFTER treatment
outlier_summary_after = {}
for col in num_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()
    outlier_summary_after[col] = outliers

# Combine into one DataFrame
outlier_comparison = pd.DataFrame({
    'Before_Treatment': outlier_summary_before,
    'After_Treatment': outlier_summary_after
}).T

print("\nOutlier Count Comparison:\n")
print(outlier_comparison)
```

```

Outlier Count Comparison:

                  CRIM  ZN  INDUS  CHAS  NOX  RM  AGE  DIS  RAD  TAX  PTRATIO  \
Before_Treatment    81   0      0     0    0  22    0    5    0    0       15
After_Treatment      0   0      0     0    0   0    0    0    0    0        0

                   B  LSTAT  MEDV
Before_Treatment  77      0    40
After_Treatment    0      0     0
```