MACHINE LEARNING

LAB WORK 4

Name: Arjun Unnikrishnan USN: 22BTRAD004

Boston Housing Dataset

Question 1. Load a dataset with outliers values (Boston Housing Dataset).

Code:

```
import pandas as pd

# Load the CSV file into a pandas DataFrame

boston_df_with_outliers = pd.read_csv('HousingData.csv')

# Display the DataFrame with outliers

print(boston_df_with_outliers.head())
```

Output:

Name: Arjun Unnikrishnan

USN: 22BTRAD004

Lab 4

Question 1. Load a dataset with outliers values (Boston Housing Dataset).

In [1]:
```
import pandas as pd
# Load the CSV file into a pandas DataFrame
boston_df_with_outliers = pd.read_csv('HousingData.csv')
# Display the DataFrame with outliers
print(boston_df_with_outliers.head())
```

```
      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
```

Question 2. Implement one hot encoding

Code:

```
# Add a categorical column for illustration purposes

boston_df['RAD_category'] = pd.cut(boston_df['RAD'], bins=[0, 5, 10, 25], labels=['Low', 'Medium', 'High'])

# Display the original DataFrame
```

print("Original DataFrame:")

print(boston_df.head())

Apply one-hot encoding to the categorical column

boston_encoded = pd.get_dummies(boston_df, columns=['RAD_category'], prefix='RAD')

Display the DataFrame after one-hot encoding

print("\nDataFrame after One-Hot Encoding:")

print(boston_encoded.head())

Output:

Question 2. Implement one hot encoding

In [5]:
```
# Add a categorical column for illustration purposes
boston_df['RAD_category'] = pd.cut(boston_df['RAD'], bins=[0, 5, 10, 25], labels=['Low', 'Medium', 'High'])
# Display the original DataFrame
print("Original DataFrame:")
print(boston_df.head())
# Apply one-hot encoding to the categorical column
boston_encoded = pd.get_dummies(boston_df, columns=['RAD_category'], prefix='RAD')
# Display the DataFrame after one-hot encoding
print("\nDataFrame after One-Hot Encoding:")
print(boston_encoded.head())
```

```
Original DataFrame:
      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 RAD_category
0  396.90   4.98  24.0          Low
1  396.90   9.14  21.6          Low
2  392.83   4.03  34.7          Low
3  394.63   2.94  33.4          Low
4  396.90    NaN  36.2          Low

DataFrame after One-Hot Encoding:
      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  RAD_Low  RAD_Medium  RAD_High
0  396.90   4.98  24.0        1           0         0
1  396.90   9.14  21.6        1           0         0
2  392.83   4.03  34.7        1           0         0
3  394.63   2.94  33.4        1           0         0
4  396.90    NaN  36.2        1           0         0
```

Question 3. Create visualizations for different aspects of a dataset using Matplotlib or Seaborn.

Code:

import matplotlib.pyplot as plt

import seaborn as sns

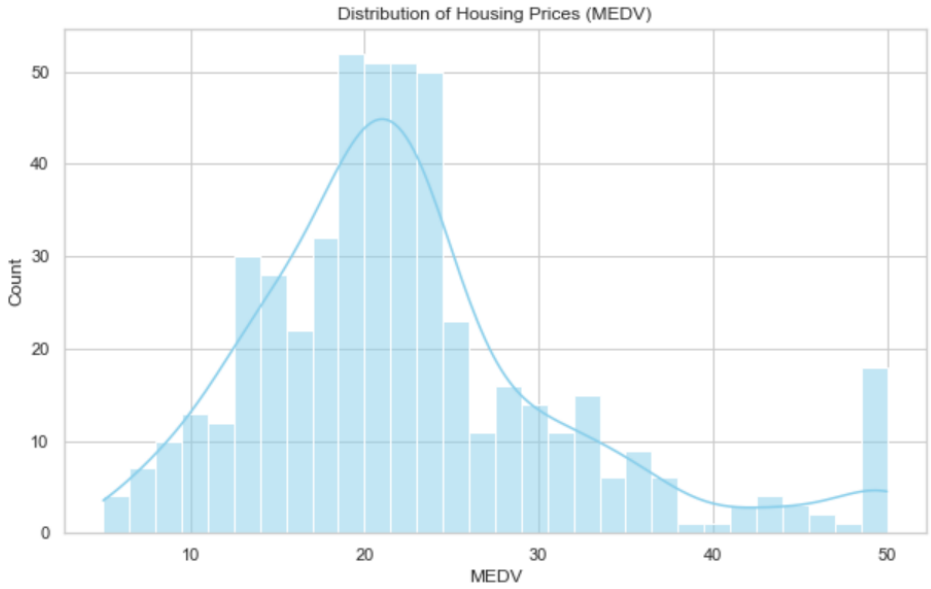
Set style for Seaborn plots

```
sns.set(style="whitegrid")
# Histogram of the target variable (MEDV)
plt.figure(figsize=(10, 6))
sns.histplot(boston_df['MEDV'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Housing Prices (MEDV)')
plt.xlabel('MEDV')
plt.show()

# Pairplot of selected features
selected_features = ['RM', 'LSTAT', 'DIS', 'TAX', 'MEDV']
sns.pairplot(boston_df[selected_features], height=2)
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()

# Box plot of selected features
plt.figure(figsize=(12, 6))
sns.boxplot(data=boston_df[selected_features], palette='Set2')
plt.title('Box Plot of Selected Features')
plt.show()

# Correlation matrix heatmap
correlation_matrix = boston_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

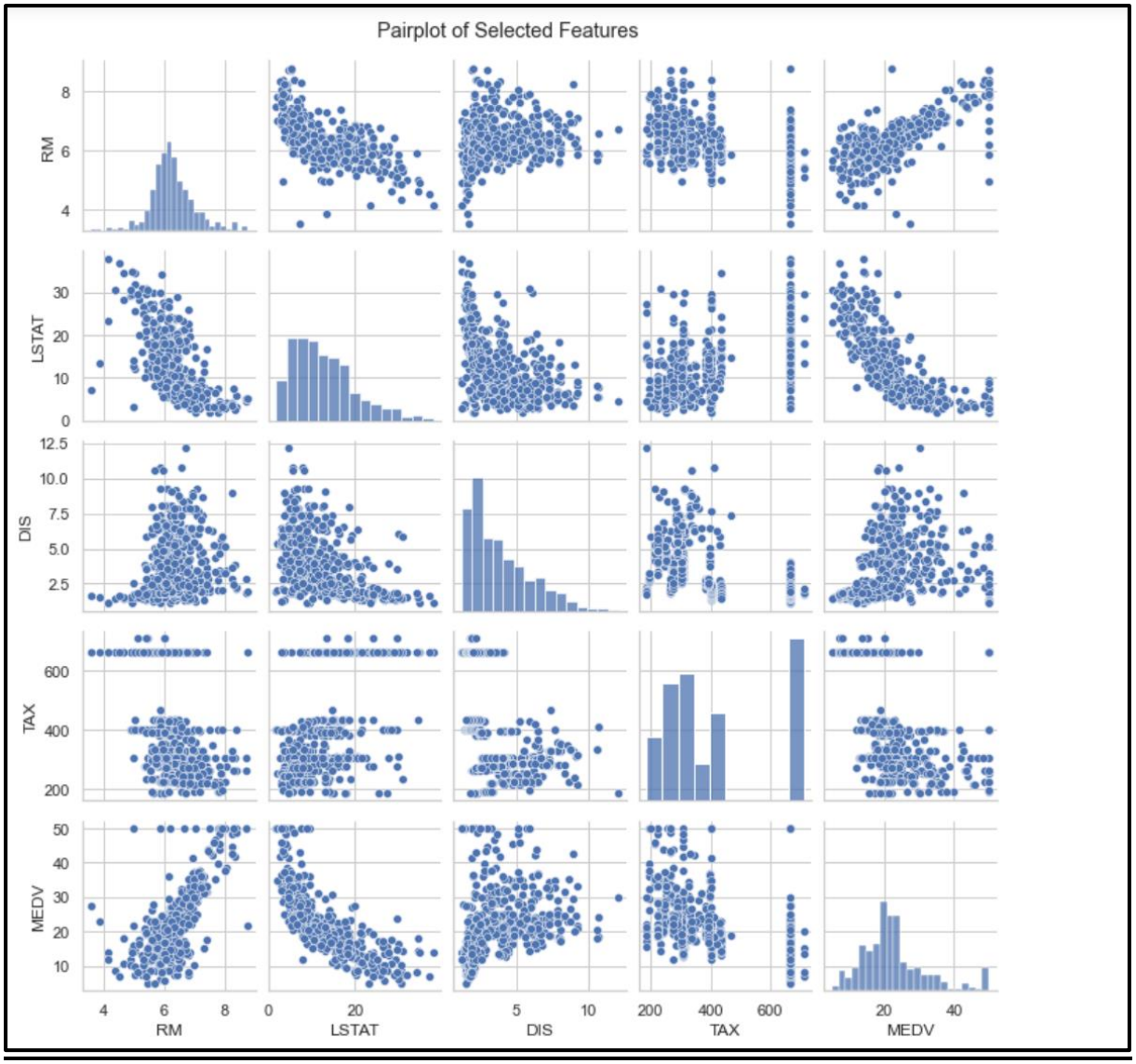
Output:

Question 3. Create visualizations for different aspects of a dataset using Matplotlib or Seaborn.

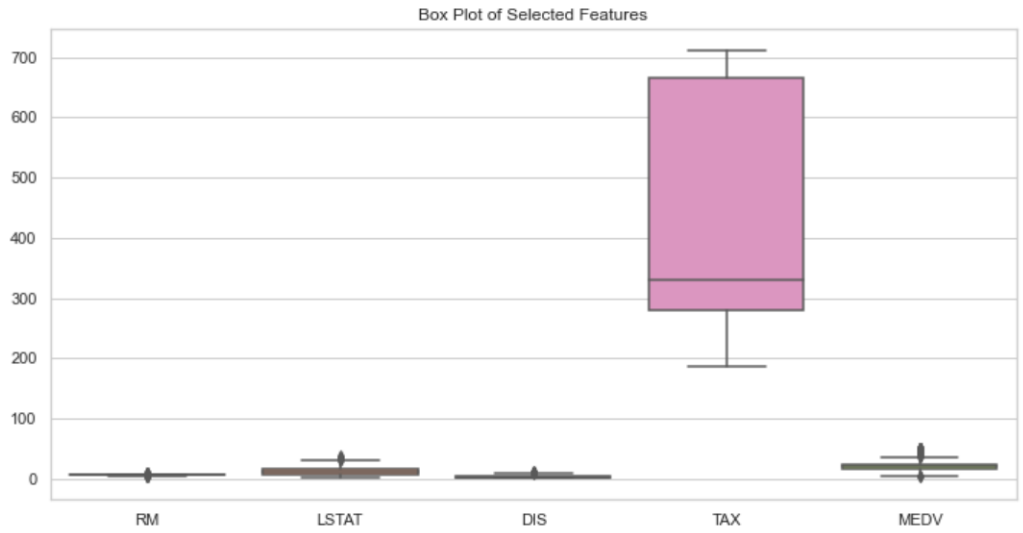
In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
# Set style for Seaborn plots
sns.set(style="whitegrid")
# Histogram of the target variable (MEDV)
plt.figure(figsize=(10, 6))
sns.histplot(boston_df['MEDV'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Housing Prices (MEDV)')
plt.xlabel('MEDV')
plt.show()
```



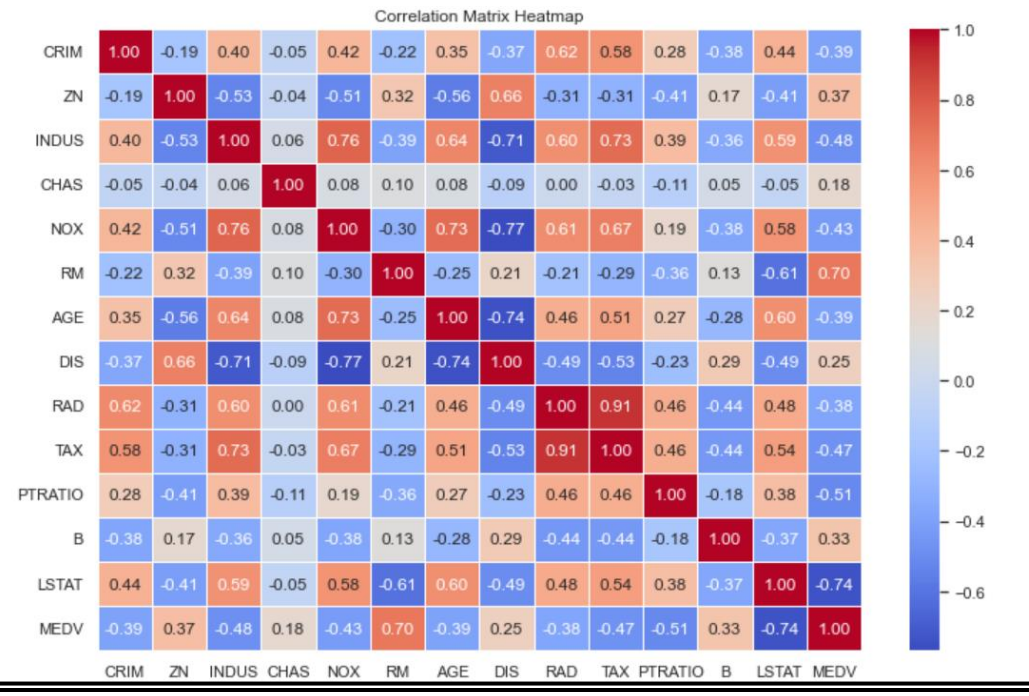
In [11]:
```
# Pairplot of selected features
selected_features = ['RM', 'LSTAT', 'DIS', 'TAX', 'MEDV']
sns.pairplot(boston_df[selected_features], height=2)
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
```


Pairplot of Selected Features
RM
LSTAT
DIS
TAX
MEDV

```
In [12]: # Box plot of selected features
plt.figure(figsize=(12, 6))
sns.boxplot(data=boston_df[selected_features], palette='Set2')
plt.title('Box Plot of Selected Features')
plt.show()
```




```
In [13]: # Correlation matrix heatmap
correlation_matrix = boston_df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```



Question 4. Interpret the visualizations to gain insights into the dataset.

Interpretation: The histogram is somewhat right-skewed, indicating that a significant portion of houses has lower prices, but there are also houses with higher prices.

Interpretation: The pairplot reveals potential relationships between features. For example, 'RM' shows a positive correlation with 'MEDV', suggesting that houses with more rooms tend to have higher prices. 'LSTAT' has a negative correlation with 'MEDV', indicating that areas with a higher percentage of lower-status residents tend to have lower housing prices.

Interpretation: The box plot highlights the distribution and variability of selected features. Outliers can be identified, and the spread of the data is visible. For instance, 'LSTAT' has a wider range, indicating higher variability.

Interpretation: The heatmap illustrates linear relationships between features. Darker colors represent stronger correlations. For example, the positive correlation between 'RM' and 'MEDV' is evident, while the negative correlation between 'LSTAT' and 'MEDV' is highlighted. The heatmap helps identify multicollinearity and understand feature relationships.

Question 5. Perform Univariate and multivariate analysis for the dataset.

Code:

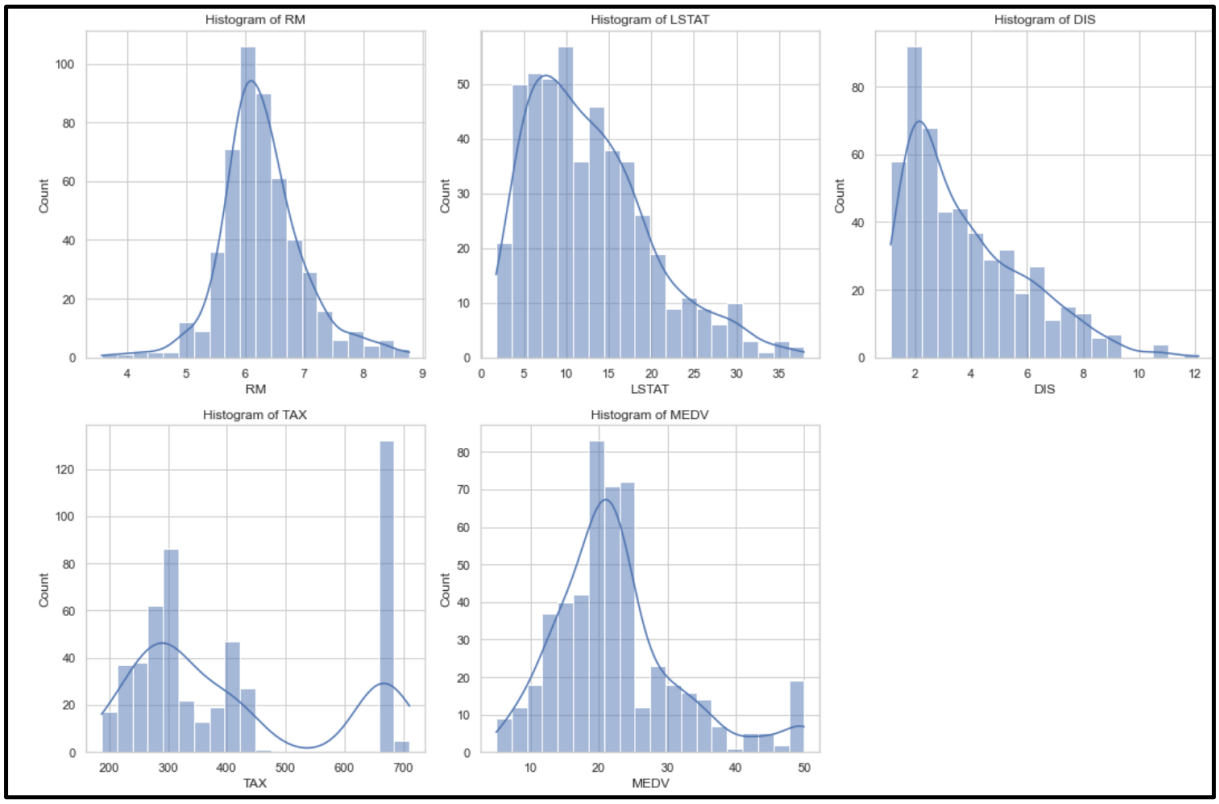
```
# Univariate Analysis - Histograms for Selected Features
selected_features = ['RM', 'LSTAT', 'DIS', 'TAX', 'MEDV']
plt.figure(figsize=(15, 10))
for i, feature in enumerate(selected_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(boston_df[feature], bins=20, kde=True)
    plt.title(f'Histogram of {feature}')
plt.tight_layout()
plt.show()
# Multivariate Analysis - Pairplot
plt.figure(figsize=(12, 8))
sns.pairplot(boston_df[selected_features], height=2)
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
# Multivariate Analysis - Correlation Matrix Heatmap
correlation_matrix = boston_df[selected_features].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
```

plt.title('Correlation Matrix Heatmap')

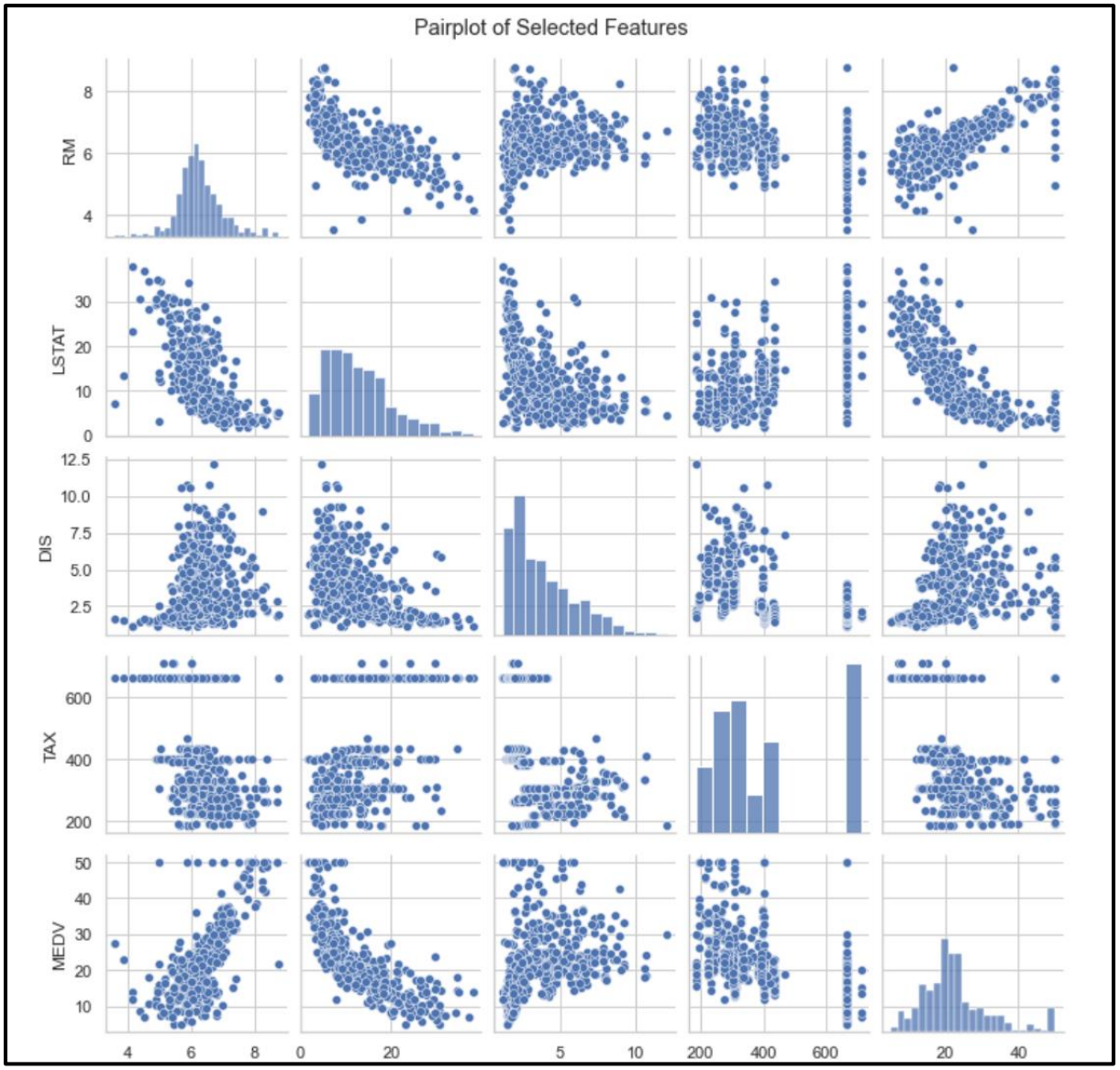
plt.show()

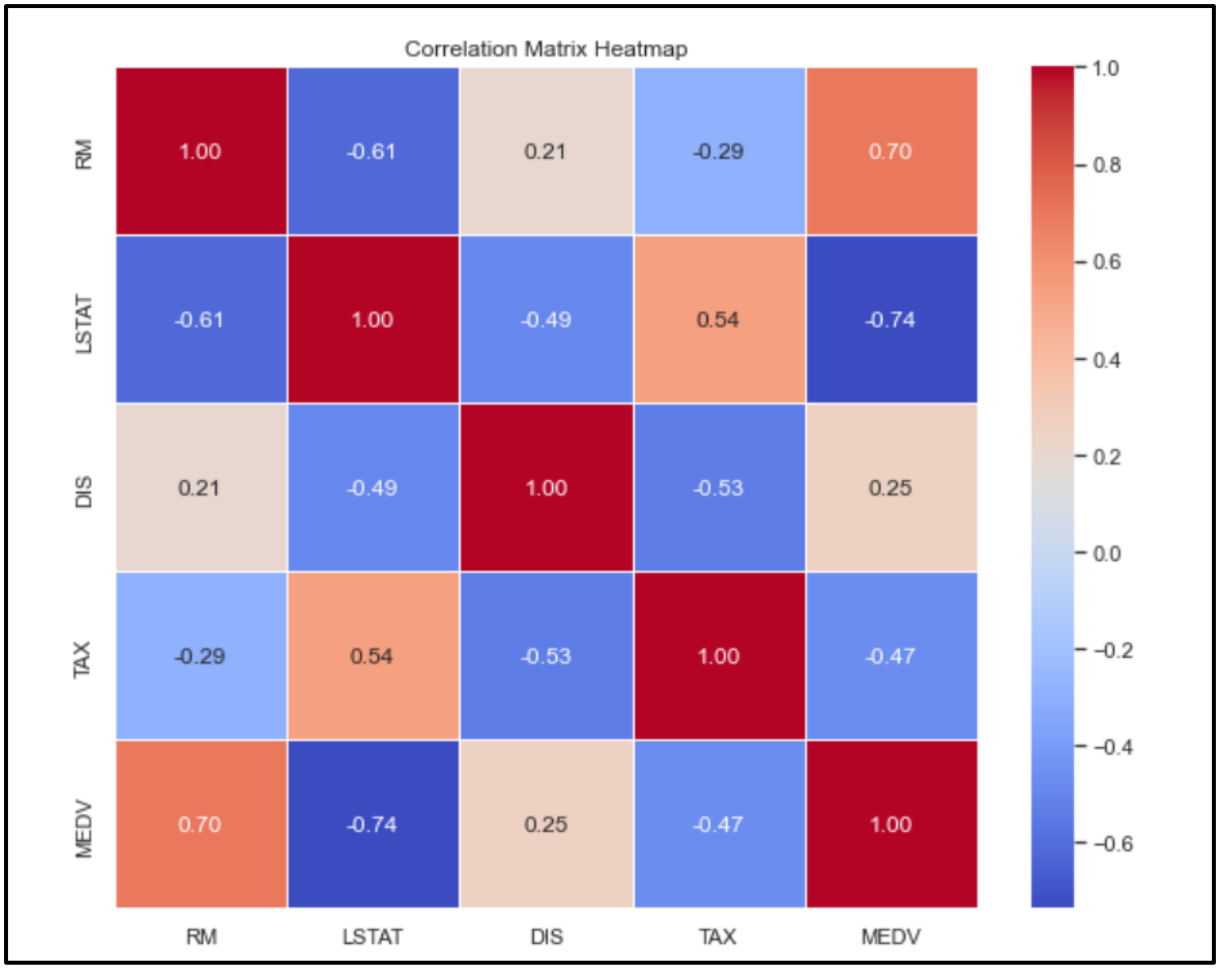
Output:

```
In [15]: # Univariate Analysis - Histograms for Selected Features
selected_features = ['RM', 'LSTAT', 'DIS', 'TAX', 'MEDV']
plt.figure(figsize=(15, 10))
for i, feature in enumerate(selected_features, 1):
    plt.subplot(2, 3, i)
    sns.histplot(boston_df[feature], bins=20, kde=True)
    plt.title(f'Histogram of {feature}')
plt.tight_layout()
plt.show()
```




```
In [17]: # Multivariate Analysis - Pairplot
plt.figure(figsize=(12, 8))
sns.pairplot(boston_df[selected_features], height=2)
plt.suptitle('Pairplot of Selected Features', y=1.02)
plt.show()
# Multivariate Analysis - Correlation Matrix Heatmap
correlation_matrix = boston_df[selected_features].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```


Pairplot of Selected Features
RM
LSTAT
DIS
TAX
MEDV



GitHub Link: https://github.com/arj1-1n/ML