In [14]:
```
import pandas as pd
# Load the merged data
data = pd.read_csv('BostonHousing.csv')
```

In [15]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston

# Load the Boston Housing dataset
boston = load_boston()

# Create a Pandas DataFrame from the dataset
data = pd.DataFrame(boston.data, columns=boston.feature_names)

# Add the target variable (median value of owner-occupied homes) to the DataFrame
data['MEDV'] = boston.target

# Calculate the mean, min, max, and standard deviation for the selected numeric attrib
selected_attributes = ['CRIM', 'RM', 'AGE']
attribute_stats = data[selected_attributes].agg(['mean', 'min', 'max', 'std'])

# Print the statistics
print(attribute_stats)
```

```
           CRIM        RM         AGE
mean   3.613524  6.284634   68.574901
min    0.006320  3.561000    2.900000
max   88.976200  8.780000  100.000000
std    8.601545  0.702617   28.148861
```

```
C:\Users\Gladin\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWa
rning: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and wil
l be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

In [16]:

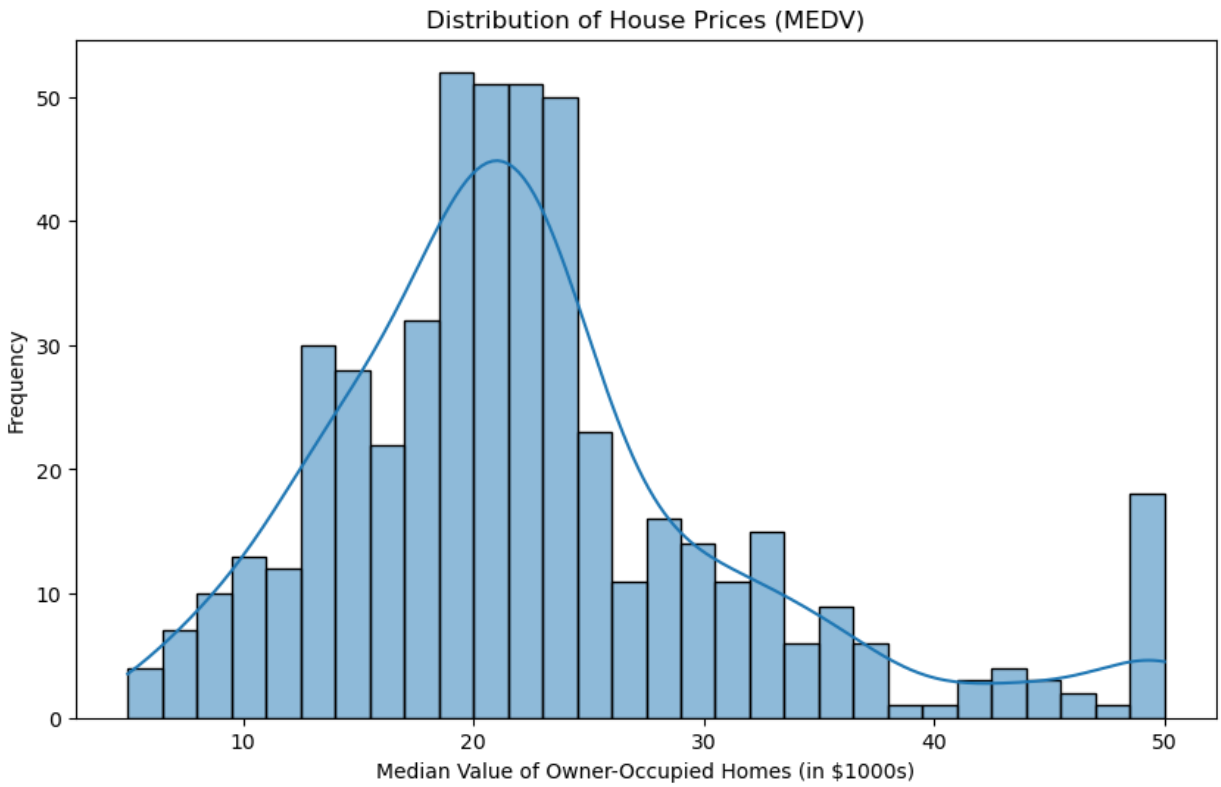
```python
import pandas as pd

def convertFile(input_file, output_file):
    try:
        # Read the input dataset
        data = pd.read_csv('BostonHousing.csv')

        # Open the output ARFF file for writing
        with open(output_file, 'w') as arff_file:
            # Write the ARFF header
            arff_file.write('@relation dataset_name\n\n')

            # Write attribute definitions
            for column in data.columns:
                # If the attribute is numeric, use real
                if pd.api.types.is_numeric_dtype(data[column]):
                    arff_file.write(f'@attribute {column} real\n')
                # If the attribute is non-numeric, use string
                else:
                    unique_values = data[column].unique()
                    unique_values_str = ','.join(map(str, unique_values))
```

```
                    arff_file.write(f'@attribute {column} {{{unique_values_str}}}\n')

            # Write data
            arff_file.write('\n@data\n')
            for index, row in data.iterrows():
                values = [str(row[column]) for column in data.columns]
                arff_file.write(','.join(values) + '\n')

        print("ARFF conversion complete.")
    except Exception as e:
        print(f"An error occurred: {str(e)}")

if __name__ == '__main__':
    input_file = 'in.data'  # Replace with your input data file
    output_file = 'out.arff'  # Specify the output ARFF file

    convertFile(input_file, output_file)
```

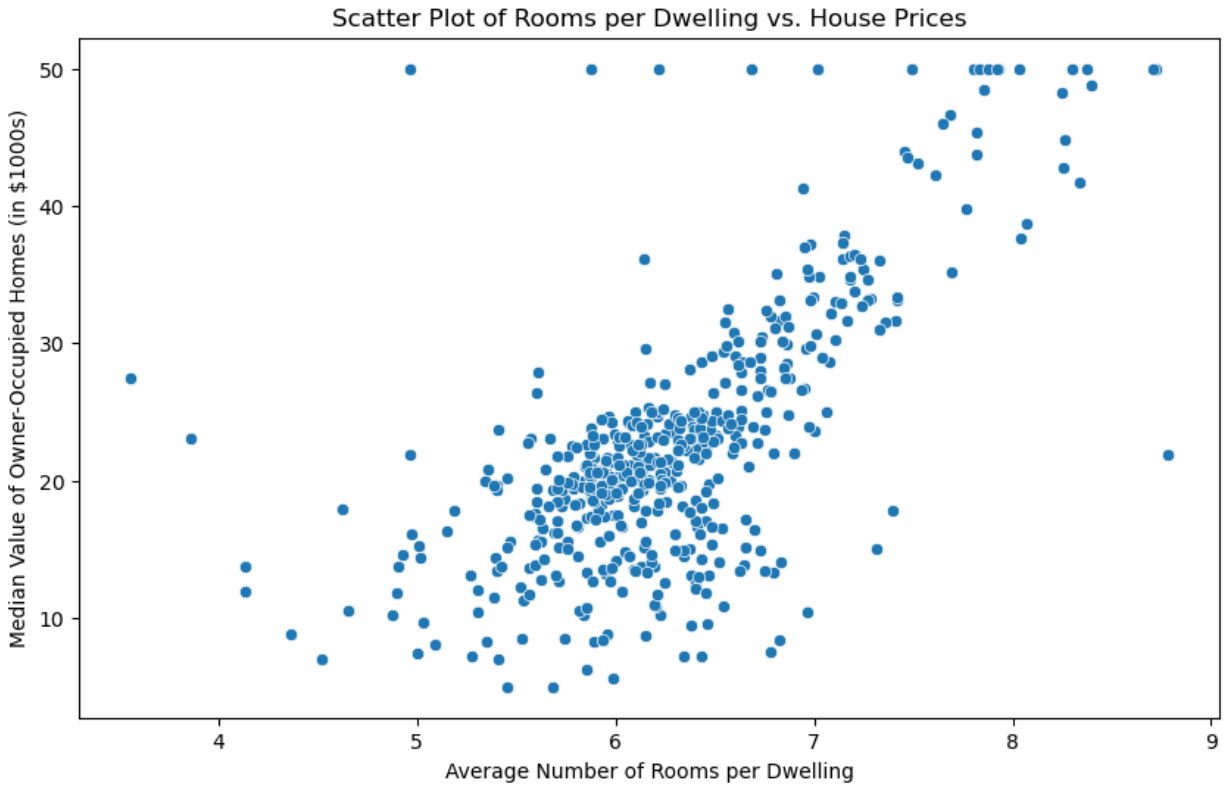
```
ARFF conversion complete.
```

In [17]:
```
output_file = 'output.arff'
```

In [11]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a histogram of house prices (MEDV)
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='MEDV', kde=True, bins=30)
plt.title("Distribution of House Prices (MEDV)")
plt.xlabel("Median Value of Owner-Occupied Homes (in $1000s)")
plt.ylabel("Frequency")
plt.show()
```



In [12]:
```
# Create a scatter plot of rooms per dwelling vs. house prices
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='RM', y='MEDV')
plt.title("Scatter Plot of Rooms per Dwelling vs. House Prices")
plt.xlabel("Average Number of Rooms per Dwelling")
plt.ylabel("Median Value of Owner-Occupied Homes (in $1000s)")
plt.show()
```



In [ ]: