In [2]:

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error  # For calculating RMSE
import numpy as np
import matplotlib.pyplot as plt
```

Loading the Dataset

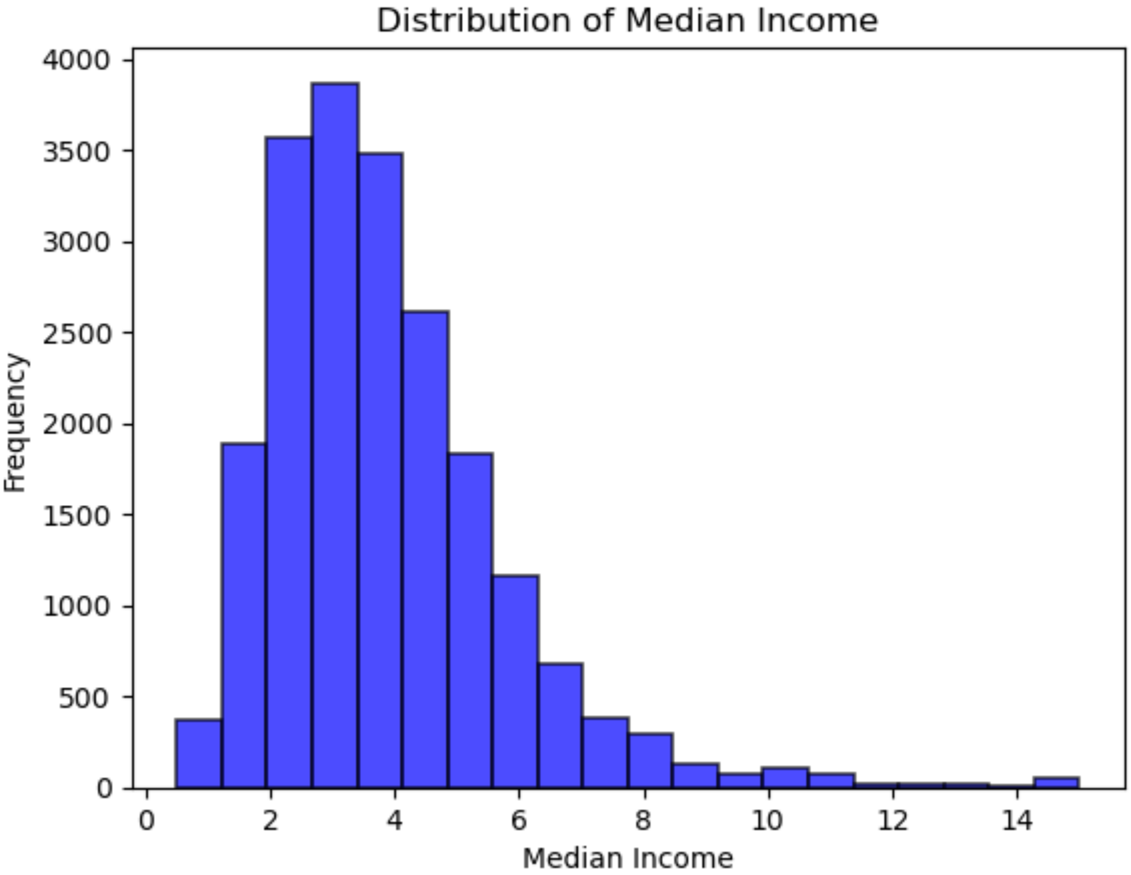
In [5]:

```
housing_data = pd.read_csv("C:/Users/SMIT YENKAR/OneDrive/Desktop/T.Y .Sub/M
```

Visualize the Dataset

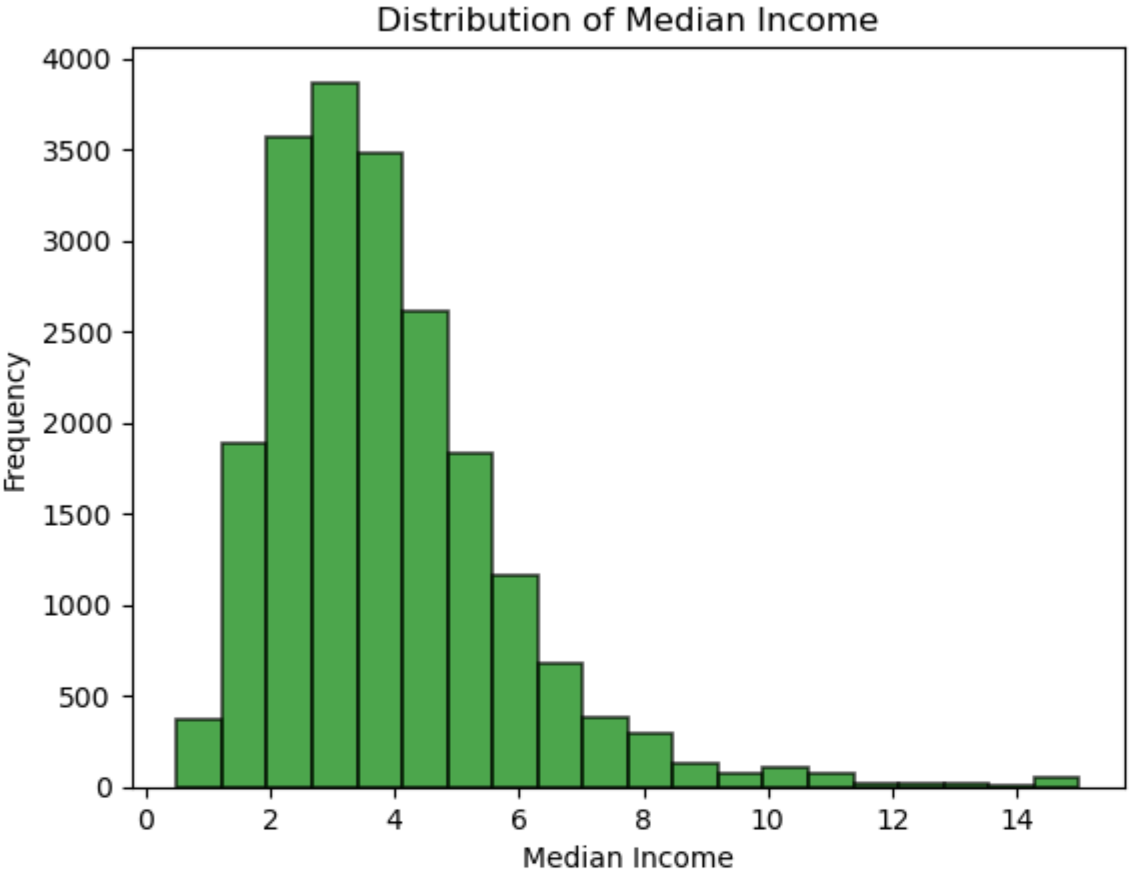
In [7]:

```
# Visualize the distribution of the 'median_income' column
plt.hist(housing_data['median_income'], bins=20, color='blue', alpha=0.7, ed
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.title('Distribution of Median Income')
plt.show()
```



In [10]:

```
# The relationship between 'median_income' and 'median_house_value'
plt.hist(housing_data['median_income'], bins=20, color='green', alpha=0.7, e
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.title('Distribution of Median Income')
plt.show()
```



In [12]:

```
# Drop rows with missing values
housing_data.dropna(inplace=True)

# Reset the index after dropping rows
housing_data.reset_index(drop=True, inplace=True)
```

K-Fold Cross-Validation

In [14]:

```
# Define the number of folds
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Initialize a list to store RMSE values for each fold
rmse_values = []

# Step 3: K-Fold Cross-Validation
for train_index, test_index in kf.split(housing_data):
    # Split the data into train and test sets for the current fold
    train_data = housing_data.iloc[train_index]
    test_data = housing_data.iloc[test_index]
    # Extract features and target variables
    X_train = train_data.drop(columns=["ocean_proximity", "median_house_valu
    y_train = train_data["median_house_value"]
    X_test = test_data.drop(columns=["ocean_proximity", "median_house_value"
    y_test = test_data["median_house_value"]
```

Regression-based Classification

In [22]:

```
    # Training a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Making  predictions
    y_pred = model.predict(X_test)

    # Calculate the RMSE for the current fold
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    rmse_values.append(rmse)

# Calculate the average RMSE across all folds
average_rmse = np.mean(rmse_values)

print("Average RMSE:", average_rmse)
```

```
Average RMSE: 70578.47363260883
```

Evaluation with Root Mean Square Error

In [23]:

```
X = housing_data.drop(columns=["ocean_proximity", "median_house_value"])
y = housing_data["median_house_value"]

# Training a linear regression model on the entire dataset
model = LinearRegression()
model.fit(X, y)

# Make predictions on the entire dataset
y_pred = model.predict(X)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y, y_pred))
print("RMSE on the entire dataset:", rmse)
```

```
RMSE on the entire dataset: 69556.14839566677
```