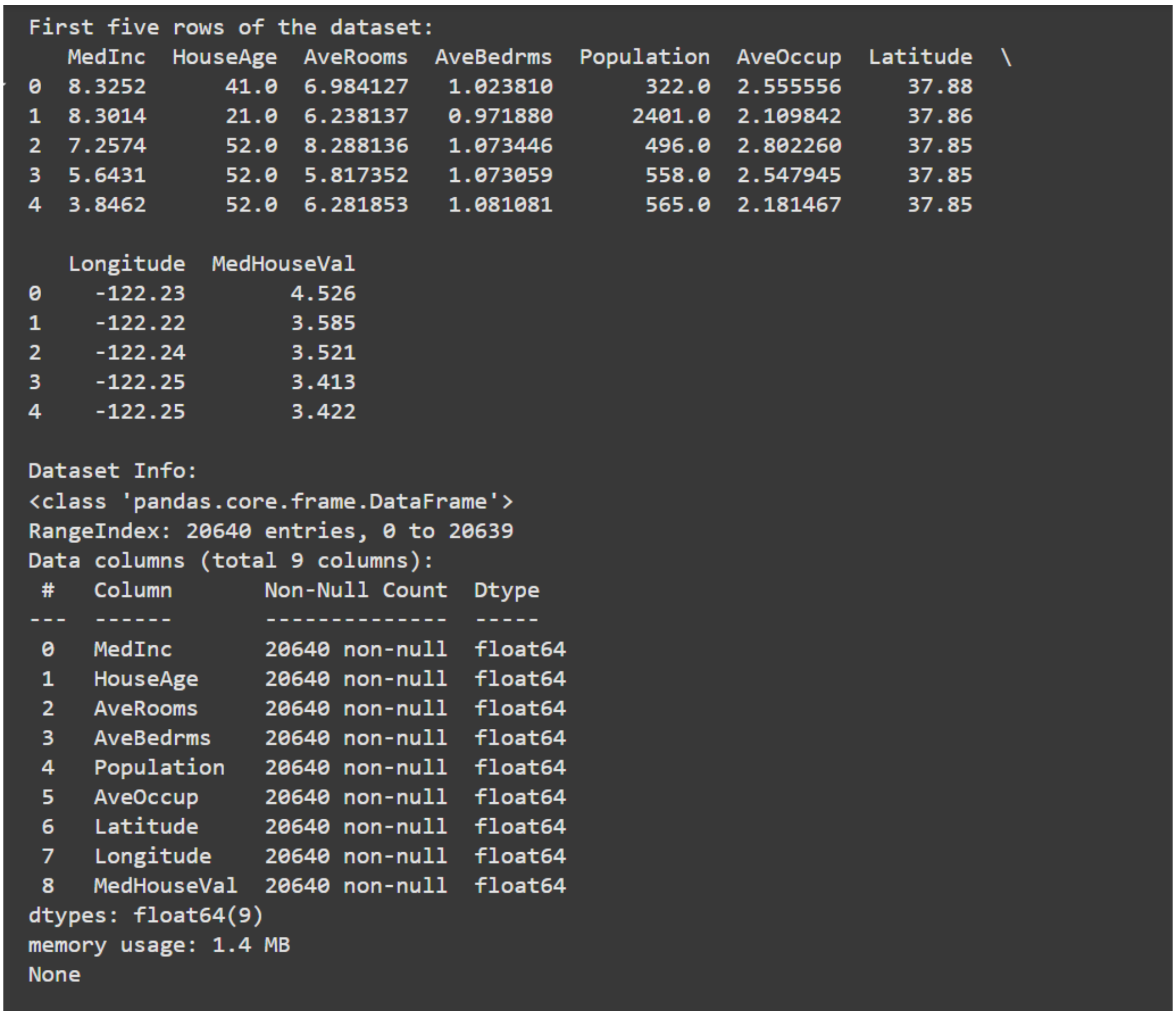
EXPERIMENT 1

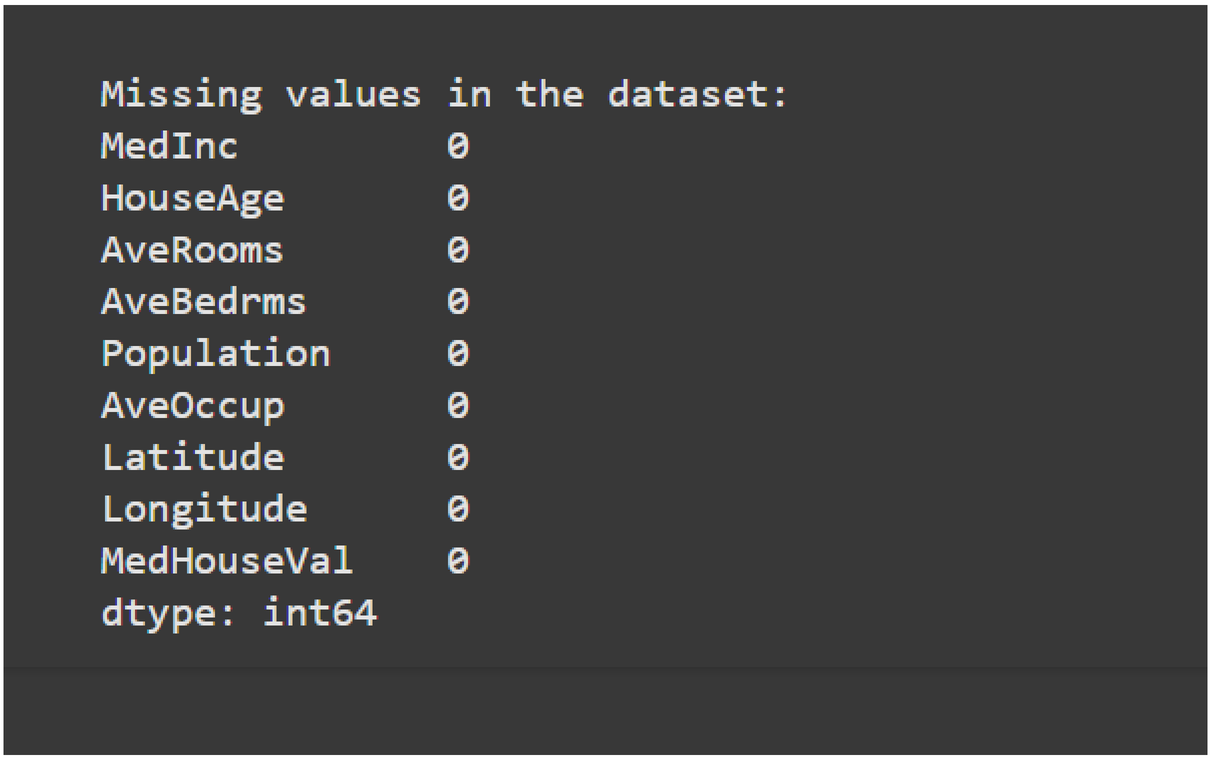
CODE:

```
 Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

 Load California Housing dataset
california = fetch_california_housing(as_frame=True)
data = california.frame

 Display first few rows and dataset information
print("First five rows of the dataset:")
print(data.head())

print("\nDataset Info:")
print(data.info())

 Check for missing values
print("\nMissing values in the dataset:")
print(data.isnull().sum())
```

OUTPUT:



```
First five rows of the dataset:
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  MedHouseVal
0    -122.23        4.526
1    -122.22        3.585
2    -122.24        3.521
3    -122.25        3.413
4    -122.25        3.422

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   MedInc       20640 non-null  float64
 1   HouseAge     20640 non-null  float64
 2   AveRooms     20640 non-null  float64
 3   AveBedrms    20640 non-null  float64
 4   Population   20640 non-null  float64
 5   AveOccup     20640 non-null  float64
 6   Latitude     20640 non-null  float64
 7   Longitude    20640 non-null  float64
 8   MedHouseVal  20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
None
```



```
Missing values in the dataset:
MedInc         0
HouseAge       0
AveRooms       0
AveBedrms      0
Population     0
AveOccup       0
Latitude       0
Longitude      0
MedHouseVal    0
dtype: int64
```

EXPLAINATION:

1. Loading the California Housing Dataset

   - The dataset is fetched using the `fetch_california_housing` function from scikit-learn's dataset module. This dataset is widely used for regression tasks and includes features like median income, average number of rooms, and median house value (target variable).

   - The `as_frame=True` argument ensures the dataset is loaded as a pandas DataFrame, making it easy to work with using familiar pandas functions.

2. Exploring the Dataset

   - The `data.head()` function displays the first five rows of the dataset. This provides an initial view of the dataset's structure, column names, and the type of data available.

   - The dataset includes columns like `MedInc` (median income), `HouseAge` (age of the house), and `MedHouseVal` (median house value). These features will be used as inputs to model the target variable, `MedHouseVal`.

   - The `data.info()` function gives insights into the dataset's columns, data types, and non-null counts. It helps confirm that all columns are of numeric type and that there are no missing values.

3. Checking for Missing Values

   - The `data.isnull().sum()` function is used to check if there are any missing values in the dataset. In this case, all columns have zero missing values, so no preprocessing steps like imputation or removal are needed.

Summary

   - The dataset appears clean and ready for regression modeling. The next steps typically involve preprocessing (like scaling or encoding categorical variables), splitting the data into training and testing sets, and building a regression model to predict `MedHouseVal`.

   - The information from this initial exploration helps identify potential features (`MedInc`, `HouseAge`, etc.) and the target variable (`MedHouseVal`) that could be used to build a predictive model.

CODE:

```
 Basic statistics

print("\nDataset Summary Statistics:")

print(data.describe())
```

Heatmap to visualize correlations

plt.figure(figsize=(10, 8))

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap")

plt.show()

Scatter plot: Median Income vs Median House Value

plt.scatter(data['MedInc'], data['MedHouseVal'], alpha=0.6)

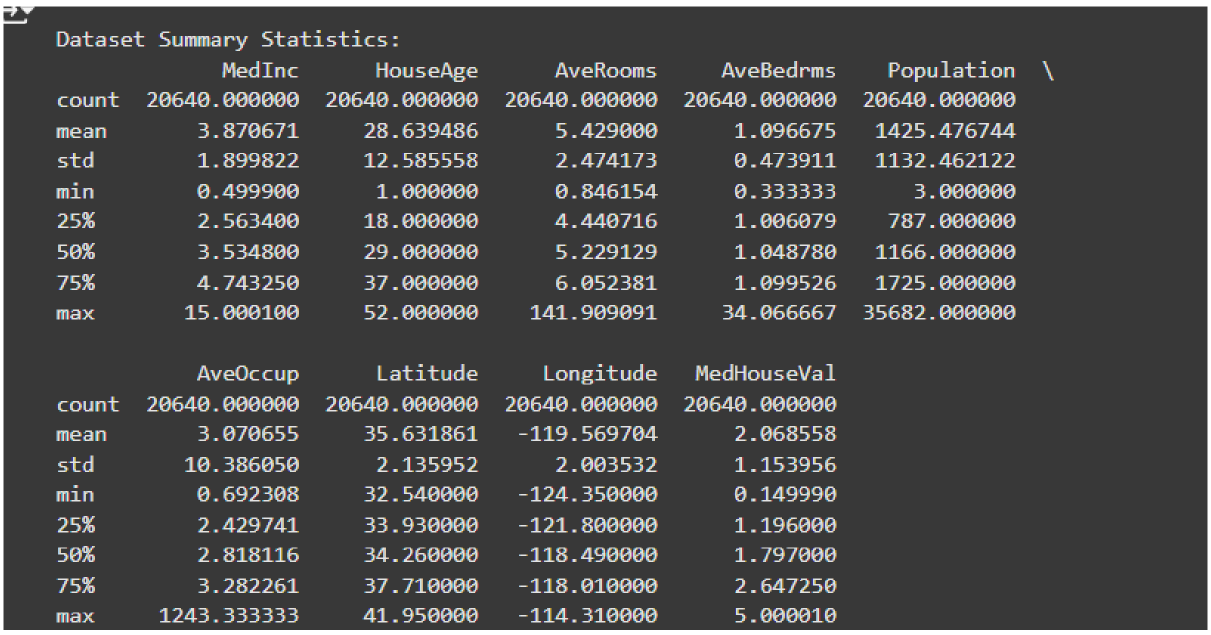
plt.title("Scatter Plot: MedInc vs MedHouseVal")

plt.xlabel("Median Income (MedInc)")

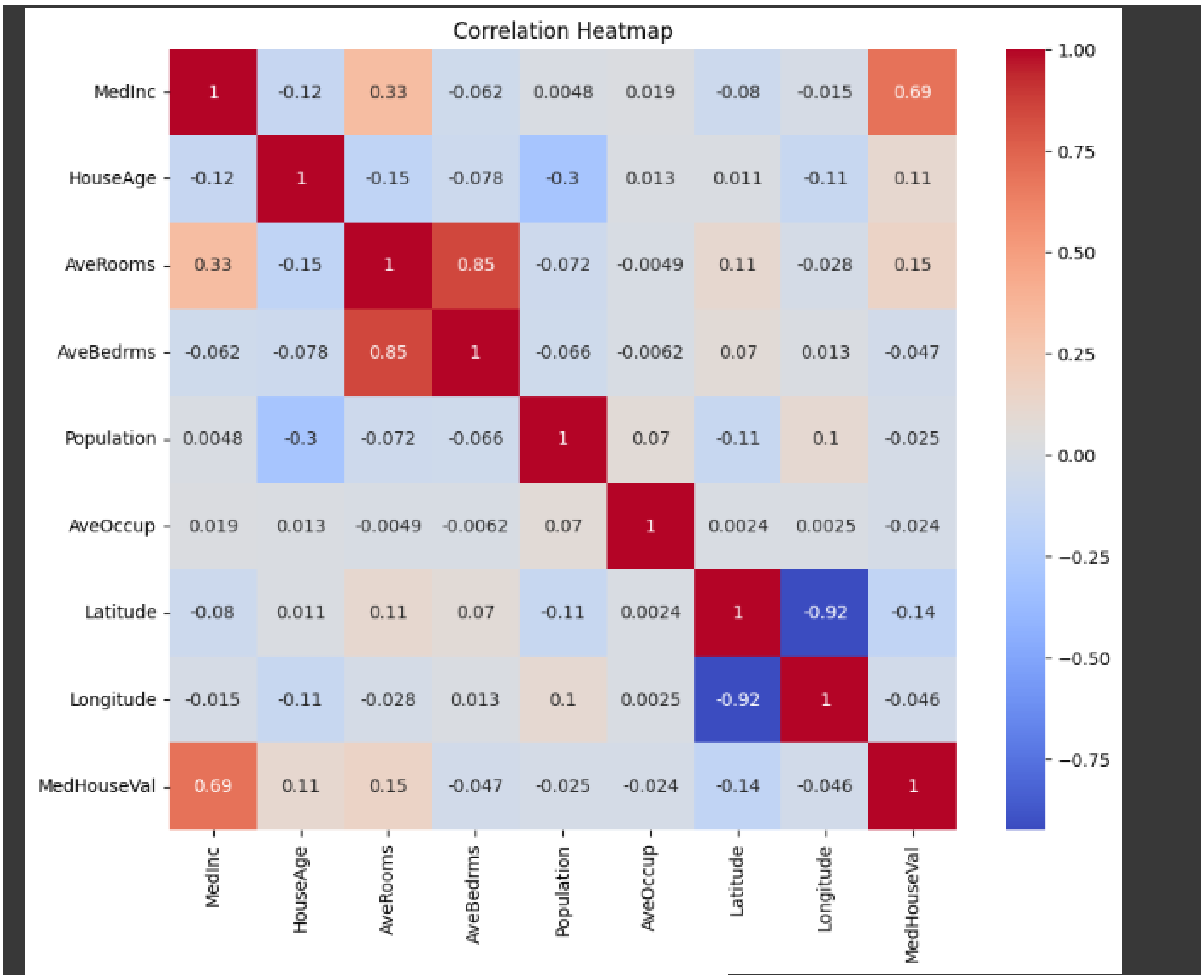
plt.ylabel("Median House Value (MedHouseVal)")

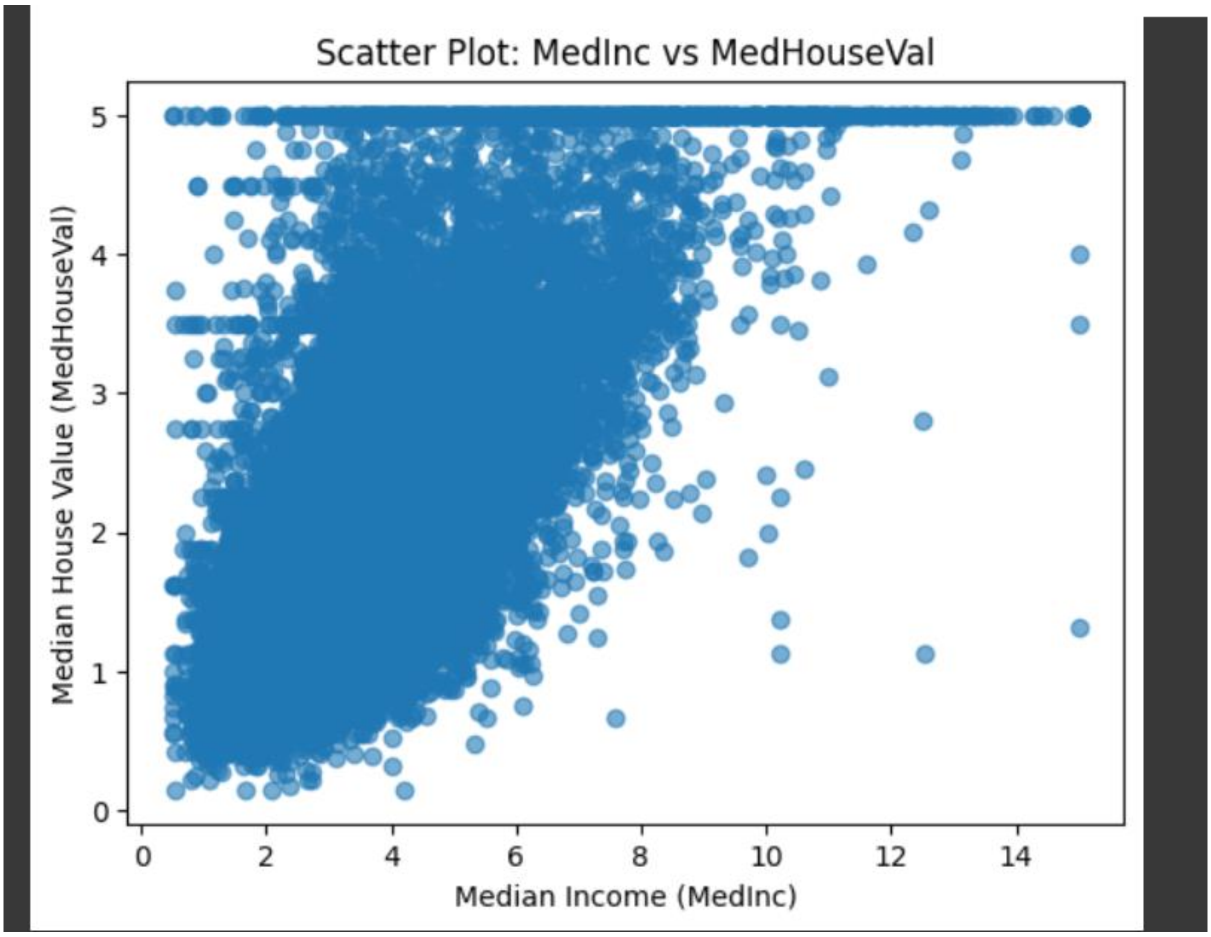
plt.show()

OUTPUT:



```
Dataset Summary Statistics:
             MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.870671     28.639486      5.429000      1.096675   1425.476744
std        1.899822     12.585558      2.474173      0.473911   1132.462122
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563400     18.000000      4.440716      1.006079    787.000000
50%        3.534800     29.000000      5.229129      1.048780   1166.000000
75%        4.743250     37.000000      6.052381      1.099526   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

           AveOccup      Latitude     Longitude   MedHouseVal
count  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704      2.068558
std       10.386050      2.135952      2.003532      1.153956
min        0.692308     32.540000   -124.350000      0.149990
25%        2.429741     33.930000   -121.800000      1.196000
50%        2.818116     34.260000   -118.490000      1.797000
75%        3.282261     37.710000   -118.010000      2.647250
max     1243.333333     41.950000   -114.310000      5.000010
```


Correlation Heatmap
MedInc 1 -0.12 0.33 -0.062 0.0048 0.019 -0.08 -0.015 0.69
HouseAge -0.12 1 -0.15 -0.078 -0.3 0.013 0.011 -0.11 0.11
AveRooms 0.33 -0.15 1 0.85 -0.072 -0.0049 0.11 -0.028 0.15
AveBedrms -0.062 -0.078 0.85 1 -0.066 -0.0062 0.07 0.013 -0.047
Population 0.0048 -0.3 -0.072 -0.066 1 0.07 -0.11 0.1 -0.025
AveOccup 0.019 0.013 -0.0049 -0.0062 0.07 1 0.0024 0.0025 -0.024
Latitude -0.08 0.011 0.11 0.07 -0.11 0.0024 1 -0.92 -0.14
Longitude -0.015 -0.11 -0.028 0.013 0.1 0.0025 -0.92 1 -0.046
MedHouseVal 0.69 0.11 0.15 -0.047 -0.025 -0.024 -0.14 -0.046 1
MedInc
HouseAge
AveRooms
AveBedrms
Population
AveOccup
Latitude
Longitude
MedHouseVal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



EXPLAINATION:

1. Basic Statistics Summary

- The `data.describe()` function generates basic statistical metrics for each column in the dataset, including:

  - Count: The number of non-null values in each column.

  - Mean: The average value for each feature.

  - Standard Deviation (std): Measures the spread of the values in each column.

  - Min and Max: The minimum and maximum values in the dataset for each feature.

  - Percentiles (25th, 50th, 75th): These provide insights into the distribution of data, showing where the middle 50% of values lie.

This summary is helpful to understand the general distribution and range of the data, identify potential outliers, and decide if any feature scaling or normalization is necessary.

2. Correlation Heatmap

- A heatmap is created to visualize the correlation between different features in the dataset. Correlation indicates the strength and direction of the relationship between pairs of features, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation). A correlation of 0 means no linear relationship.

  - Positive correlations (e.g., between `MedInc` and `MedHouseVal`) indicate that as one feature increases, the other tends to increase as well.

  - Negative correlations suggest that as one feature increases, the other tends to decrease.

  - No correlation implies that changes in one feature do not systematically relate to changes in another.

The heatmap uses color coding (e.g., cool colors for negative correlations and warm colors for positive correlations) to make these relationships easier to identify at a glance.

3. Scatter Plot: Median Income vs Median House Value

- A scatter plot is used to visually explore the relationship between Median Income (`MedInc`) and Median House Value (`MedHouseVal`).

  - Each point on the plot represents a data entry, with the x-axis showing the median income for a region and the y-axis showing the median house value.

  - The plot helps identify any trends or patterns between these two variables, such as:

    - A positive correlation suggests that regions with higher median incomes tend to have higher median house values.

    - Any potential outliers can also be identified on the plot as points that deviate significantly from the general trend.

By using the scatter plot, you get a clearer visual understanding of how these two features relate to each other, and whether a linear regression model might be appropriate to model their relationship.

Summary

- Basic statistics provide an overview of the data's central tendencies, variability, and distribution.

- The correlation heatmap reveals the strength of relationships between various features, which can inform feature selection for modeling.

- The scatter plot offers a visual representation of the relationship between `MedInc` and `MedHouseVal`, helping to confirm any trends or patterns that might inform the choice of modeling techniques.

CODE :

Feature selection

X = data[['MedInc']]   Using Median Income as the independent variable

y = data['MedHouseVal']   Median House Value as the target variable

Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Train a Linear Regression model

model = LinearRegression()

model.fit(X_train, y_train)

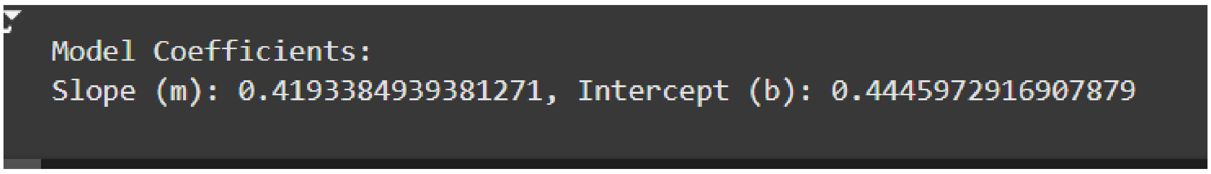
Make predictions

y_pred = model.predict(X_test)

print("\nModel Coefficients:")

print(f"Slope (m): {model.coef_[0]}, Intercept (b): {model.intercept_}")

OUTPUT:



```
Model Coefficients:
Slope (m): 0.4193384939381271, Intercept (b): 0.4445972916907879
```

EXPLAINATION:

1. Feature Selection

   - The feature selection step involves choosing which variables (features) will be used as inputs (independent variables) for the regression model. In this case, only Median Income (`MedInc`) is selected as the feature (independent variable), and Median House Value (`MedHouseVal`) is selected as the target (dependent variable). This simplifies the model to a univariate regression, focusing on the relationship between just these two variables.

2. Splitting the Data

   - The dataset is divided into two parts:

- Training Set: 80% of the data is used to train the model, which means that the model will learn the relationship between `MedInc` and `MedHouseVal` based on this subset.

- Testing Set: 20% of the data is reserved for testing the model's performance. This ensures that the model is evaluated on unseen data, which provides a more accurate assessment of its predictive ability.

- The `train_test_split` function performs this division, and the `random_state=42` ensures that the split is reproducible (i.e., the same split will occur each time the code is run).

3. Training the Linear Regression Model

- A Linear Regression model is created using the `LinearRegression()` class from scikit-learn. This model will attempt to find the best-fit line (or hyperplane) that minimizes the difference between predicted and actual values of `MedHouseVal` for the training data.

- The model learns the parameters (slope and intercept) that define the linear relationship between `MedInc` and `MedHouseVal`.

4. Making Predictions

- After training, the model is used to make predictions on the testing set (`X_test`). The predicted values (`y_pred`) are the model's estimates for `MedHouseVal` based on the input values of `MedInc` in the test set.

5. Model Coefficients

- The coefficients of the linear regression model are printed out:

- Slope (m): This represents the change in the target variable (`MedHouseVal`) for each unit increase in the independent variable (`MedInc`). A positive slope indicates that as median income increases, median house value also tends to increase.

- Intercept (b): This is the value of `MedHouseVal` when `MedInc` is zero. In practical terms, it may not always be meaningful (e.g., `MedInc` being zero might not make sense in the context of housing data), but it's part of the equation of the best-fit line.

Summary

- This step involves selecting the independent feature (`MedInc`) and the dependent target (`MedHouseVal`), splitting the data into training and testing sets, and then training a linear regression model to predict house values based on income levels.

- The model coefficients (slope and intercept) provide a mathematical understanding of how median income is expected to influence median house value in the dataset.

CODE :

Calculate metrics

mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

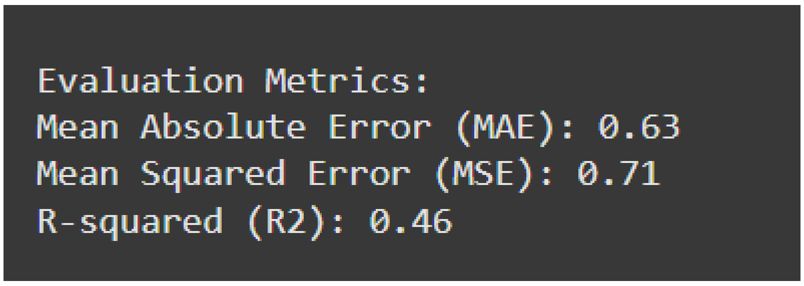
print("\nEvaluation Metrics:")

print(f"Mean Absolute Error (MAE): {mae:.2f}")

print(f"Mean Squared Error (MSE): {mse:.2f}")

print(f"R-squared (R2): {r2:.2f}")

OUTPUT:



```
Evaluation Metrics:
Mean Absolute Error (MAE): 0.63
Mean Squared Error (MSE): 0.71
R-squared (R2): 0.46
```

EXPLAINATION:

1. Mean Absolute Error (MAE)

- MAE measures the average absolute difference between the actual values (`y_test`) and the predicted values (`y_pred`). It provides a straightforward interpretation of the model's error:

  - Formula: MAE = $(1/n)\ \Sigma|y_i - \hat{y}_i|$

    - Where `$y_i$` is the actual value, and `$\hat{y}_i$` is the predicted value for each data point.

  - A lower MAE indicates better model performance. It's easy to interpret since it represents the average amount by which the predictions are off, in the same units as the target variable (in this case, house values).

2. Mean Squared Error (MSE)

- MSE calculates the average squared difference between the actual and predicted values. Squaring the differences gives more weight to larger errors, making MSE more sensitive to outliers.

  - Formula: MSE = $(1/n)\ \Sigma(y_i - \hat{y}_i)^2$

    - Where `$y_i$` is the actual value, and `$\hat{y}_i$` is the predicted value for each data point.

- MSE is widely used but can be more difficult to interpret directly because the error is in squared units. It penalizes large errors more significantly than MAE, so if there are outliers, the MSE will be higher.

3. R-squared (R^2)

- R-squared is a measure of how well the model explains the variance in the target variable. It tells us the proportion of the variance in the target variable that is predictable from the independent variables.

  - Formula: $R^2 = 1 - (\Sigma(y_i - \hat{y}_i)^2 / \Sigma(y_i - \bar{y})^2)$

    - Where `$y_i$` is the actual value, `$\hat{y}_i$` is the predicted value, and `$\bar{y}$` is the mean of the actual values.

  - R^2 values range from 0 to 1:

    - 0 means the model does not explain any of the variance in the target variable.

    - 1 means the model perfectly explains the variance in the target variable.

  - An R^2 closer to 1 indicates that the model does a good job of explaining the relationship between the independent and dependent variables.

Summary

- MAE gives a clear indication of average prediction error in the original units of the target variable.

- MSE penalizes larger errors more than MAE, making it more sensitive to outliers.

- R^2 provides a measure of the model's goodness of fit, indicating how well the model explains the variability in the target variable.

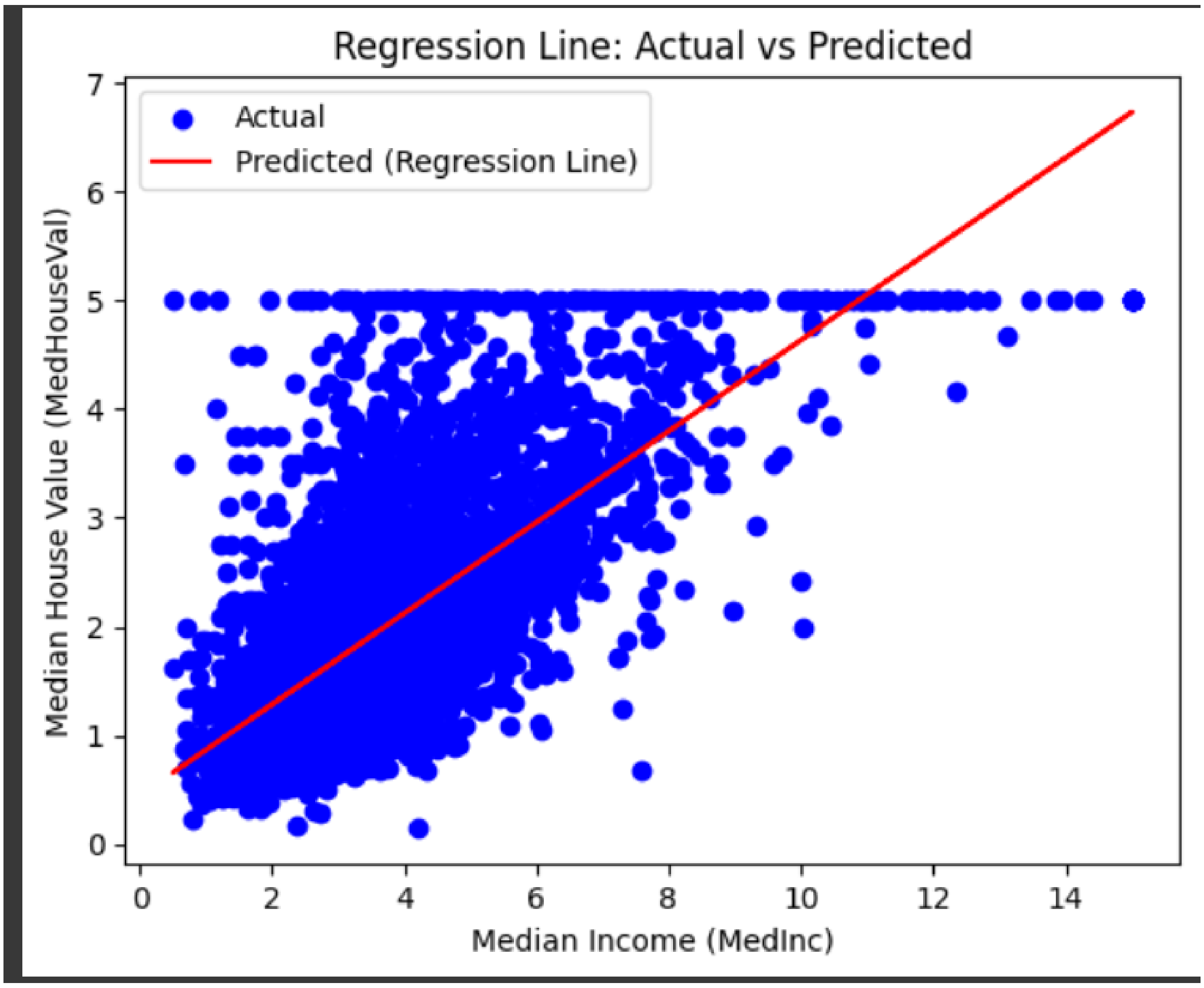
- By examining these metrics, you can evaluate the accuracy and reliability of your linear regression model. A good model will have low MAE and MSE values and an R^2 value close to 1.

CODE :

Regression line visualization

```
plt.scatter(X_test, y_test, color='blue', label='Actual')

plt.plot(X_test, y_pred, color='red', label='Predicted (Regression Line)')

plt.title("Regression Line: Actual vs Predicted")

plt.xlabel("Median Income (MedInc)")

plt.ylabel("Median House Value (MedHouseVal)")

plt.legend()
plt.show()
```

OUTPUT:



EXPLAINATION:

Visualization of Actual vs Predicted Values

- The plot shows the relationship between the actual values of the target variable (`MedHouseVal`) and the predicted values (`y_pred`) produced by the linear regression model. It provides a visual representation of how well the model fits the data.

1. Scatter Plot of Actual Values

- The scatter plot displays the actual values of the target variable (`MedHouseVal`) from the test set (`X_test` and `y_test`). Each point on the scatter plot represents a data point, with the x-axis showing the median income (`MedInc`) and the y-axis showing the actual house value (`MedHouseVal`).

- This part of the plot helps us see the distribution of actual house values in relation to median income.

2. Plot of Predicted Regression Line

- The red line represents the predicted values of the target variable (`MedHouseVal`) based on the model's predictions (`y_pred`) for the test set. This line is the best-fit regression line found by the linear regression model.

- The line illustrates the relationship between median income and house value as estimated by the model. It is essentially the linear equation (slope and intercept) derived from the training data.

3. Comparison Between Actual and Predicted

- The scatter points represent the actual data points, while the red regression line represents the model's predictions. If the model is a perfect fit, the red line would pass exactly through all the blue points (actual values).

- The closer the red line is to the blue points, the better the model's predictions are. If the points are spread out significantly from the line, it indicates the model has more error in predicting those values.

Summary

- The plot provides a visual check of how well the linear regression model is capturing the relationship between median income and median house value. The closer the predicted regression line is to the actual data points, the better the model's performance.

- This type of visualization is crucial for assessing whether the linear model is a good fit for the data, helping to identify any significant discrepancies between predicted and actual values.