Double-click (or enter) to edit

Assignment Simple machine Learning Project cycle

House Price Prediction Using California Housing Data

This dataset provides information about house prices in California. House Price Prediction

In this notebook, you'll follow the basic machine learning process to build a regression model to predict house prices using the "California Housing Dataset" from sklearn.

Follow the instructions and complete each TODO to complete the assessment on the essential steps in building and evaluating a regression model.

The following is a description of each column in the dataset:

Dataset Features (California Housing):

- MedInc: Median income in block group
- HouseAge: Median house age in block group
- AveRooms: Average number of rooms per household
- AveBedrms: Average number of bedrooms per household
- Population: Block group population
- AveOccup: Average number of household members
- Latitude: Block group latitude
- Longitude: Block group longitude
- MedHouseVal (Target): Median house value in block group

```
 1
 2
 3 # --- Imports ---
 4 # TODO: Import all the necessary libraries for data handling, visualization, and model building.
 5 # Example: import pandas as pd
 6
 7 # Add your imports here:
 8 import pandas as pd
 9 import numpy as np
10 import matplotlib.pyplot as plt
11 import seaborn as sns
12 from sklearn.datasets import fetch_california_housing
13 from sklearn.model_selection import train_test_split
14 from sklearn.linear_model import LinearRegression
15 from sklearn.metrics import mean_squared_error, r2_score
16
17 # Viz Setting
18
19 %matplotlib inline
20 sns.set_style('whitegrid')
21
22 # --- Data Collection and Loading ---
23 # TODO: Load the 'California Housing' dataset from sklearn and convert it into a pandas DataFrame.
24 # Hint: Use `fetch_california_housing()` from `sklearn.datasets`
25
26 # Task 1) Load dataset and convert to DataFrame:
27
28 # Add your code here:
29
30 df_carlifornia = fetch_california_housing(as_frame=True)
31 df_carlifornia = df_carlifornia.frame # conver to datafram
32
33 # --- Data Exploration ---
34
35 # --- Quick Check of Data ---
36 # TODO: Display the first few rows of the dataset to understand its structure.
37 # Hint: Use `.head()` to inspect the first few rows.
38
39 # Add your code here:
40
41
```

```
41
42 # TODO: Check the features and target variable. Identify which is continuous and categorical if applicable.
43 # Hint: Use `.info()` and `.describe()` to inspect data types and statistical properties.
44
45 # Add your code here:
46
47 # Print the Dataset:
48
49 print("Dataset Preview")
50 display(df_carlifornia.head())
51 print("-"  * 90)
52 print("Check the Data Feature (Continous / Categorical)")
53 display (df_carlifornia.info())
54 print("-"  * 90)
55 print("Check the Data Types")
56 display (df_carlifornia.dtypes)
57 print("-"  * 90)
58 print("Check the Statistical Properties")
59 display (df_carlifornia.describe())
60 print("-"  * 90)
61
62 # --- EDA and Data Preprocessing ---
63 # TODO: Check for missing/null values.
64 # Hint: Use `.isnull().sum()` to check for null values.
65
66 # Add your code here:
67
68 missing_values = df_carlifornia.isnull().sum()
69 print("Missing Values:")
70 display(missing_values)
71 print("-" * 120)
72
73 # Elimiate Duplicate Values
74
75 print("Eliminate Duplicate Values")
76 df_carlifornia = df_carlifornia.drop_duplicates()
77 print("-" * 120)
78
79 # Renaming my columns as best practice.
80
81 df_carlifornia.rename(columns={
82     'MedInc': 'Median_Income',
83     'HouseAge': 'House_Age',
84     'AveRooms': 'Average_Rooms',
85     'AveBedrms': 'Average_Bedrooms',
86     'Population': 'Population',
87     'AveOccup': 'Average_Occupancy',
88     'Latitude': 'Latitude',
89     'Longitude': 'Longitude',
90     'MedHouseVal': 'Median_House_Value'
91
92 }, inplace=True)
93
94 # Check column names currently in the dataset
95 print("Column names currently in df:")
96 print(df_carlifornia.columns.tolist())
97 print("-" * 120)
98
99 # --- Data Visualization ---
100
101 # TODO: Visualize the data. Create scatter plots to see the relationship between independent features and the target variab
102 # Example: Use `plt.scatter()` to visualize the relationship between 'MedInc' and 'MedHouseVal'.
103
104 # Add your code here:
105
106 # Scatter plot: Median Income vs Median House Value
107
108 plt.figure(figsize=(8,6))
109 plt.scatter(df_carlifornia['Median_Income'], df_carlifornia['Median_House_Value'], alpha=0.5, color='teal')
110 plt.title('Median_Income vs Median House Value')
111 plt.xlabel('Median_Income (in $10,000s)')
112 plt.ylabel('Median_Income Value ($100,000s)')
113 plt.show()
114
115
116 # TODO: Create a function to automate scatter plots for all features vs MedHouseVal.
117 # Hint: The function should loop over a list of features and plot scatter plots for each.
118
```

```
# Define your function here:

def plot_features_vs_target(df_carlifornia, target_col):  # get numerical cols only
  numerical_cols = df_carlifornia.select_dtypes(include=['float64', 'int64']).columns
  numerical_features = [col for col in numerical_cols if col != target_col] # remove the target cols from the feature list

  for features in numerical_features:
      plt.figure(figsize=(8,6))
      plt.scatter(df_carlifornia[features], df_carlifornia[target_col], alpha=0.5, color='teal')
      plt.title(f'{features} vs {target_col}')
      plt.xlabel(features)
      plt.ylabel(target_col)
      plt.show()
      print("-" * 120)

# --- Feature Engineering ---

# TODO: Use the function to visualize the relationships between multiple features and the target variable.
# Features: ['MedInc', 'AveRooms', 'AveOccup', 'HouseAge']
# Target: 'MedHouseVal'

# Add your code here:

selected_features = ['Median_Income', 'Average_Rooms', 'Average_Occupancy', 'House_Age']
target_col = 'Median_House_Value'

for features in selected_features:
    plt.figure(figsize=(8,6))
    plt.scatter(df_carlifornia[features], df_carlifornia[target_col], alpha=0.5, color='teal')
    plt.title(f'{features} vs {target_col}')
    plt.xlabel(features)
    plt.ylabel(target_col)
    plt.show()

# Additional visualization for skewed feature (Average_Rooms)
plt.figure(figsize=(8,6))
plt.scatter(df_carlifornia['Average_Rooms'], df_carlifornia['Median_House_Value'], alpha=0.5, color='darkcyan')
plt.xscale('log')
plt.title('Average_Rooms (Log Scale) vs Median_House_Value')
plt.xlabel('Average_Rooms (log scale)')
plt.ylabel('Median_House_Value')
plt.show()
print("-" * 120)

# --- ML Model Training ---
# TODO: Split the dataset into training and testing sets.
# Hint: Use `train_test_split()` from `sklearn.model_selection` with an 80/20 split.

# Define X (features) and y (target) and perform the train-test split:

# Define the independent features (X) and the target variable (y)
print("Define the independent features & dependent target:")
X = df_carlifornia[['Median_Income', 'House_Age', 'Average_Rooms', 'Average_Bedrooms',
                    'Population', 'Average_Occupancy', 'Latitude', 'Longitude']]
y = df_carlifornia['Median_House_Value']

print("Feature Matrix Shape:", X.shape)
print("Target Vector Shape:", y.shape)
print("-" * 120)

# Split data into training (80%) and testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
print("-" * 120)

# --- Model Building ---

# TODO: Choose an appropriate regression model.
# Hint: Use `LinearRegression()` from `sklearn.linear_model`.

# Define your regression model here:
model = LinearRegression()

# TODO: Train the model on the training data.
# Hint: Use `.fit()` to train the model.
```

```
196
197 # Add your code here:
198 model.fit(X_train, y_train)
199
200 # Initialize the Linear Regression model
201
202 coefficients = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
203 # Fit (train) the model
204
205 # Display model coefficients
206
207 print("Model Coefficients:")
208 display(coefficients)
209
210 # Intercept (constant term)
211 print("Intercept:", model.intercept_)
212 print("-" * 120)
213
214 # --- Model Evaluation ---
215 # TODO: Evaluate the performance of the model on the test set using relevant metrics (e.g., RMSE, R-squared).
216 # Hint: Use `mean_squared_error()` and `r2_score()` from `sklearn.metrics`.
217
218 # Predict on the test set and calculate the evaluation metrics:
219
220 # Predict on test data
221 y_pred = model.predict(X_test)
222
223 # Compute evaluation metrics
224 mse = mean_squared_error(y_test, y_pred)
225 rmse = np.sqrt(mse)
226 r2 = r2_score(y_test, y_pred)
227
228 print(f"Mean Squared Error (MSE): {mse:.4f}")
229 print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
230 print(f"R² Score: {r2:.4f}")
231 print("-" * 120)
232
233 # Personal Not for future reference. This residual provides additonal technical knowledge on the model traiing/testing.
234 # it tells me whether there was a difference between the actual vs the predicted values.
235
236 # Residuals = difference between actual and predicted values
237 residuals = y_test - y_pred
238
239 plt.figure(figsize=(8,6))
240 sns.histplot(residuals, bins=40, kde=True, color='mediumseagreen')
241 plt.title('Distribution of Residuals (Errors)')
242 plt.xlabel('Residual (Actual - Predicted)')
243 plt.ylabel('Frequency')
244 plt.show()
245 print("-" * 120)
246
247 # --- Model Prediction ---
248 # TODO: Predict Median House Value from a new set of feature inputs.
249 # Example new data: 'MedInc' = 3, 'HouseAge' = 30, 'AveRooms' = 6, 'AveOccup' = 3, 'Latitude' = 34, 'Longitude' = -118, 'Av
250 # Hint: Use `.predict()` on a new data array.
251
252 # Add your prediction code here:
253 # Example: predict for a new data point
254 sample_input = np.array([[3, 30, 6, 1, 1500, 3, 34, -118]])  # example feature values
255 predicted_value = model.predict(sample_input)
256 print(f"Predicted Median House Value: {predicted_value[0]:.2f}")
257
258
259
260
```

Dataset Preview

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Check the Data Feature (Continous / Categorical)

```
<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
```

Check the Data Types

	0
MedInc	float64
HouseAge	float64
AveRooms	float64
AveBedrms	float64
Population	float64
AveOccup	float64
Latitude	float64
Longitude	float64
MedHouseVal	float64

dtype: object

Check the Statistical Properties

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVa
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.00000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.06855
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.15395
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.14999
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.19600
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.79700
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.64725
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.00001

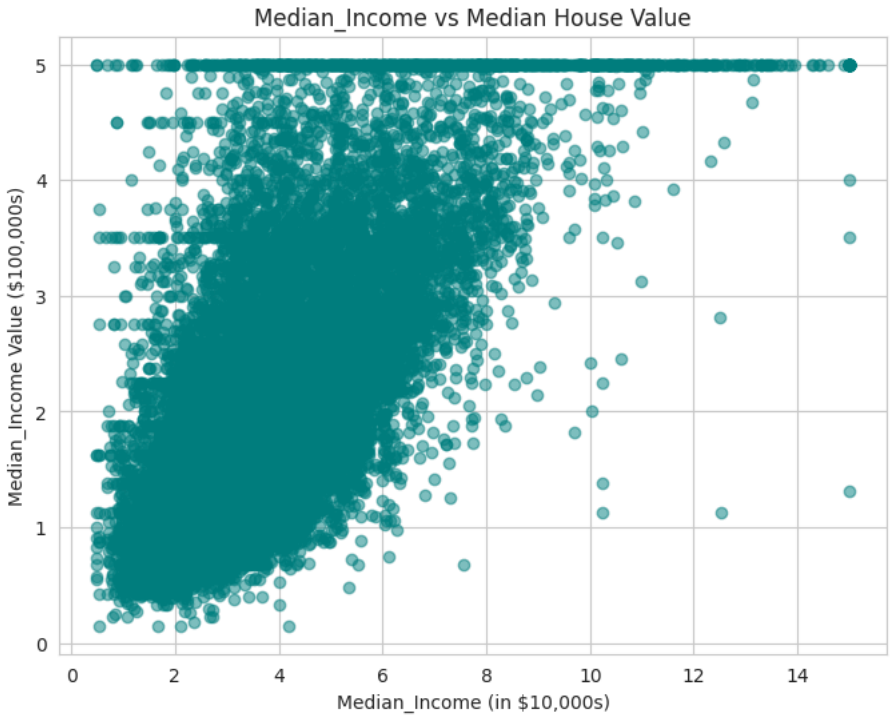
Missing Values:

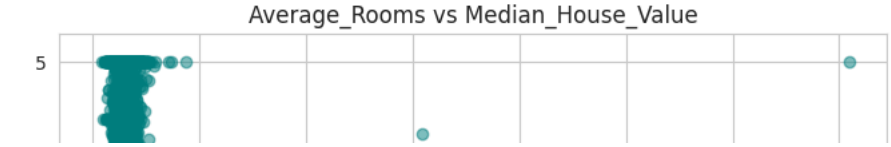
	0
MedInc	0
HouseAge	0
AveRooms	0
AveBedrms	0
Population	0

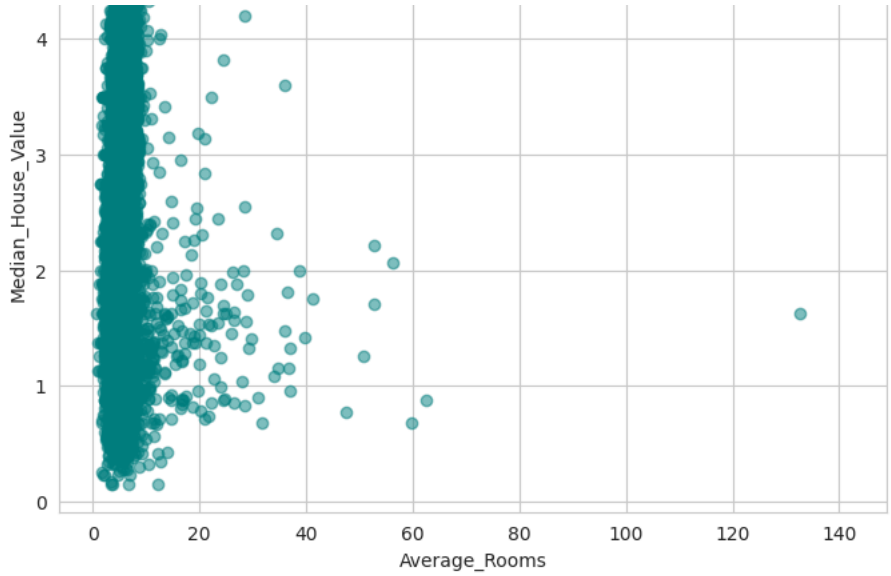
AveOccup	0
Latitude	0
Longitude	0
MedHouseVal	0

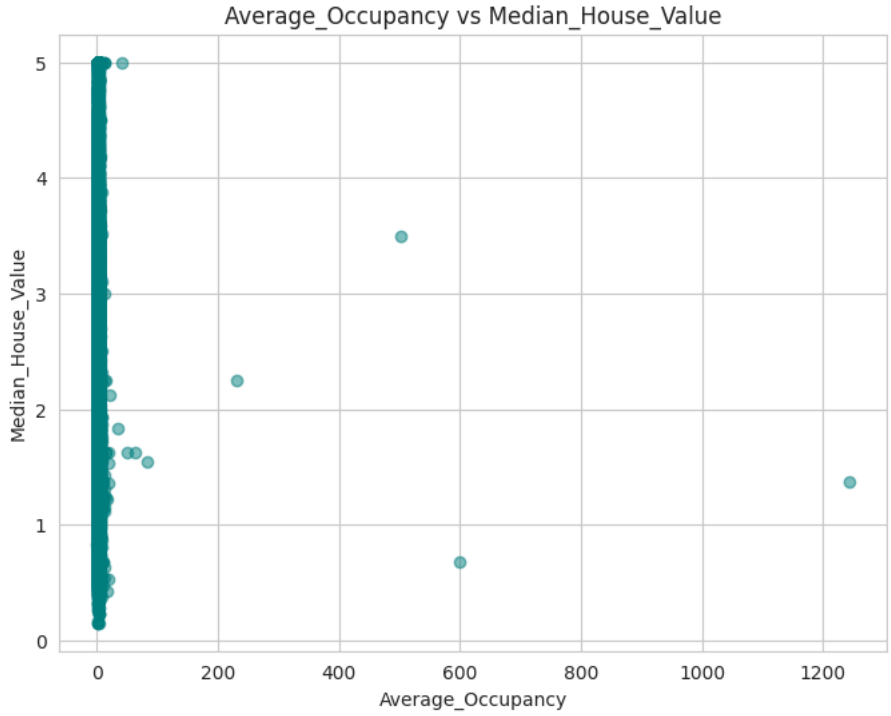
dtype: int64

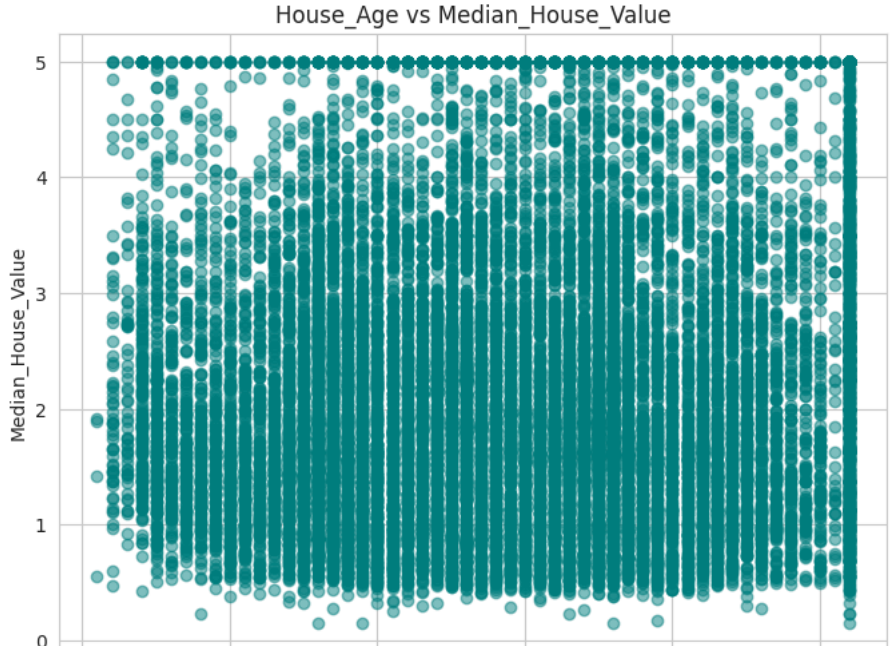
```
--------------------------------------------------------------------------------------------------------------------------------
Eliminate Duplicate Values
--------------------------------------------------------------------------------------------------------------------------------
Column names currently in df:
['Median_Income', 'House_Age', 'Average_Rooms', 'Average_Bedrooms', 'Population', 'Average_Occupancy', 'Latitude', 'Longitude']
--------------------------------------------------------------------------------------------------------------------------------
```

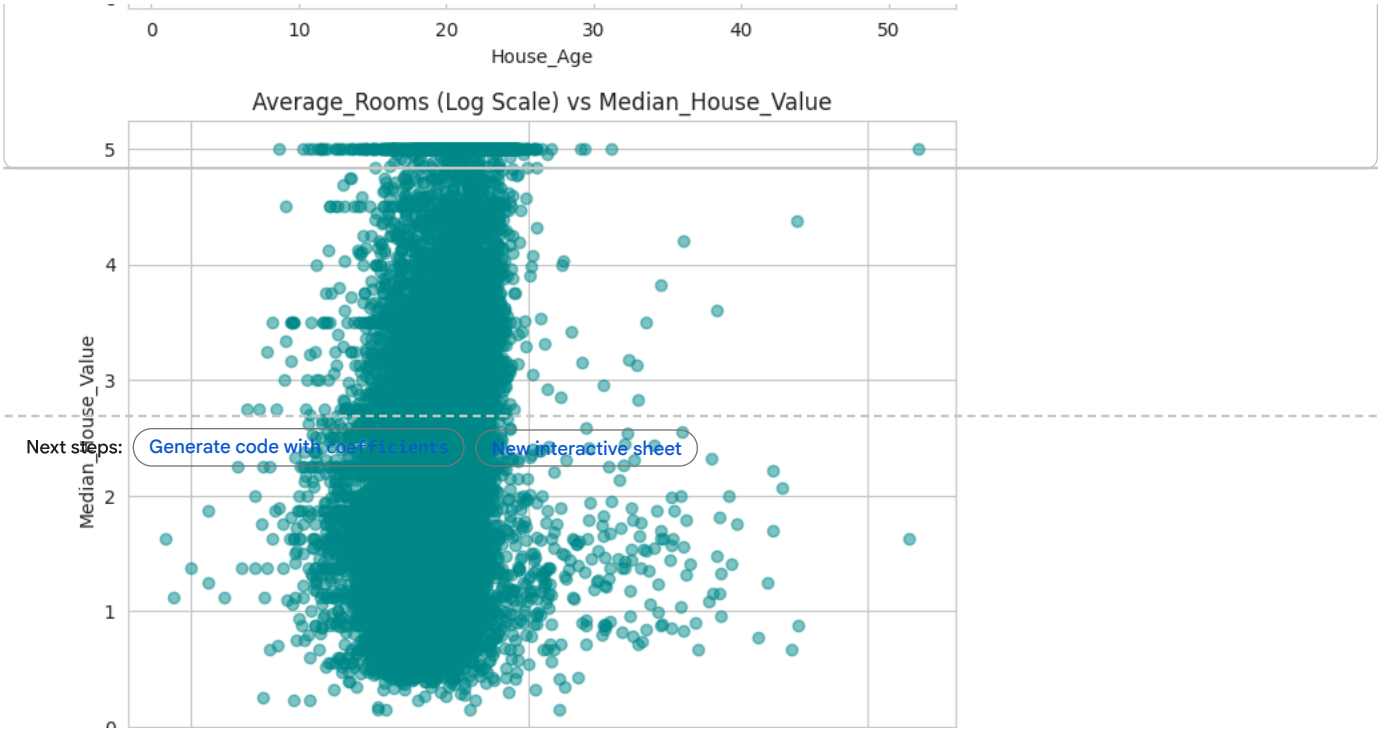





Median_House_Value
Average_Rooms
0
20
40
60
80
100
120
140
0
1
2
3
4


Average_Occupancy vs Median_House_Value
Median_House_Value
Average_Occupancy
0
200
400
600
800
1000
1200
0
1
2
3
4
5


House_Age vs Median_House_Value
Median_House_Value
0
1
2
3
4
5



Next steps: Generate code with coefficients New interactive sheet