- Name: Deep Salunkhe
- Roll No.:21102A0014
- SEM-7 ML Lab1 Github Link

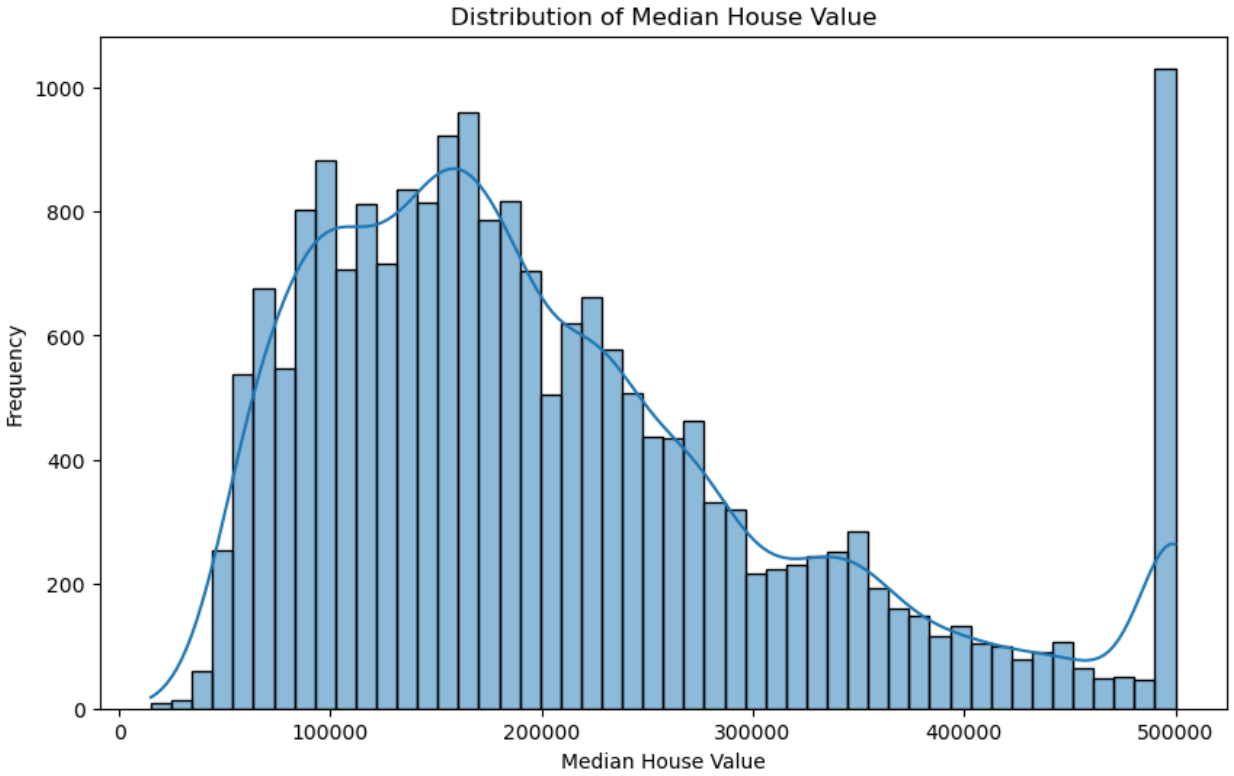
```
# Import necessary libraries
import pandas as pd
import numpy as np
import joblib
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('housing.csv')

# Handle missing values by filling with median
df['total_bedrooms'].fillna(df['total_bedrooms'].median(),
inplace=True)

# Identify categorical and numerical columns
categorical_columns = ['ocean_proximity']
numerical_columns = df.columns.drop(categorical_columns +
['median_house_value']).tolist()

# Create a column transformer for preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_columns),  # Scale
numerical columns
        ('cat', OneHotEncoder(drop='first', sparse=False),
categorical_columns)  # One-hot encode categorical columns
    ])

# Separate features and target
X = df.drop('median_house_value', axis=1)
y = df['median_house_value']

# 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)

# Fit the preprocessor on the training data and transform both
training and test data
```

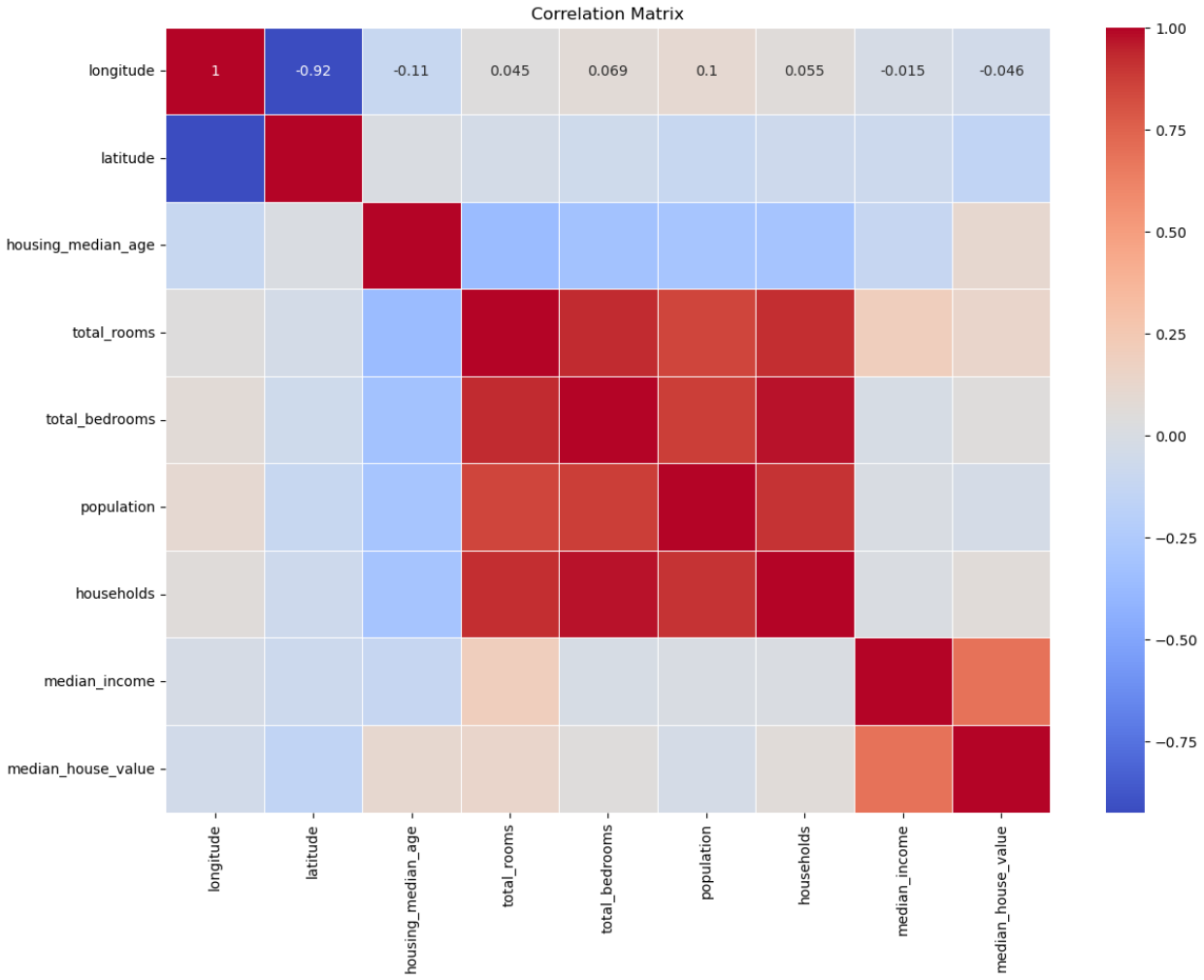
```
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)

# Get feature names after preprocessing
feature_names = (numerical_columns +
                 preprocessor.named_transformers_['cat']
                 .get_feature_names_out(categorical_columns).tolist())

# Convert preprocessed data to DataFrames
X_train_preprocessed = pd.DataFrame(X_train_preprocessed,
columns=feature_names)
X_test_preprocessed = pd.DataFrame(X_test_preprocessed,
columns=feature_names)

# Train the Linear Regression model
model = LinearRegression()
model.fit(X_train_preprocessed, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_preprocessed)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Save the model and preprocessor
joblib.dump((model, preprocessor), 'housing_price_model.pkl')
print("Model and preprocessor saved successfully!")

# Function to preprocess and predict user input
def preprocess_and_predict(user_input):
    # Load the model and preprocessor
    loaded_model, loaded_preprocessor =
joblib.load('housing_price_model.pkl')

    # Convert user input to DataFrame
    user_df = pd.DataFrame([user_input])

    # Preprocess user input
    user_preprocessed = loaded_preprocessor.transform(user_df)

    # Make prediction
    prediction = loaded_model.predict(user_preprocessed)

    return prediction[0]
```

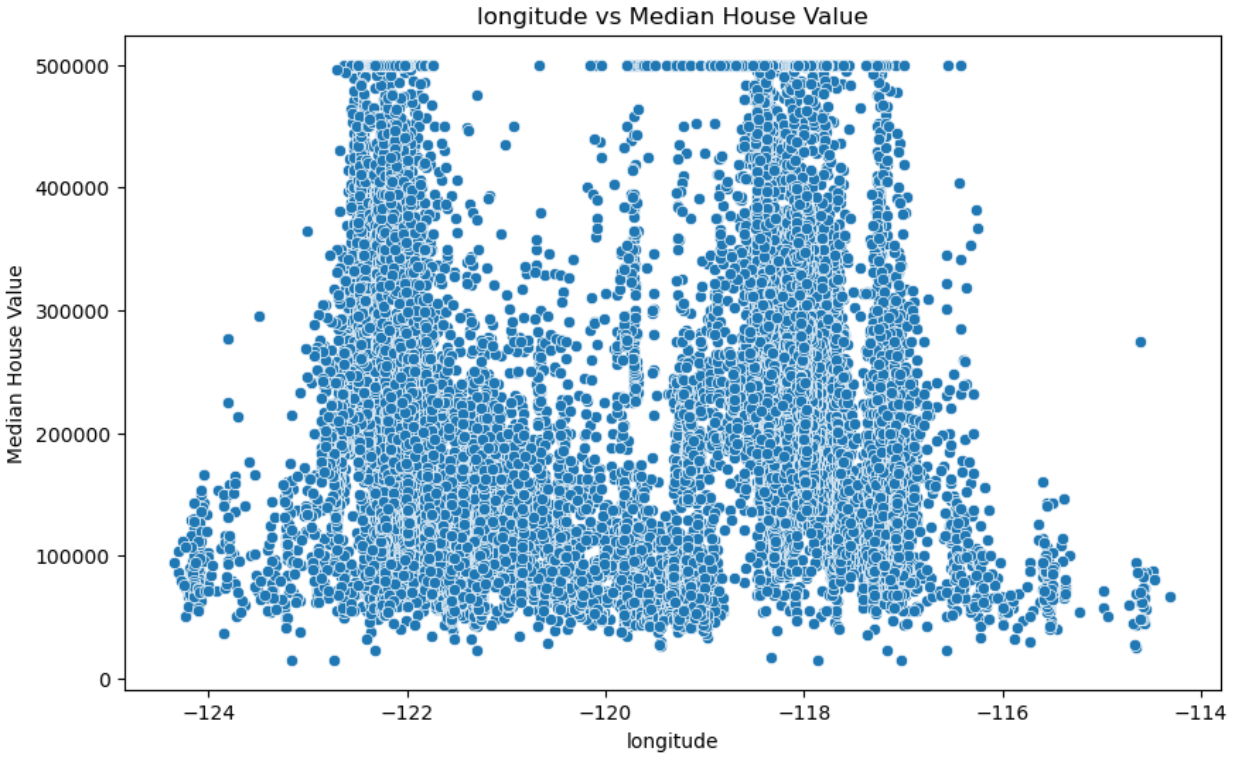
```
# Take user input
user_input = {}
for column in numerical_columns:
    user_input[column] = float(input(f"Enter {column}: "))
for column in categorical_columns:
    user_input[column] = input(f"Enter {column} (e.g., NEAR BAY,
INLAND, etc.): ")

# Predict based on user input
prediction = preprocess_and_predict(user_input)
print(f"Predicted Median House Value: ${prediction:.2f}")

# Visualize the distribution of the target variable
plt.figure(figsize=(10, 6))
sns.histplot(df['median_house_value'], bins=50, kde=True)
plt.title('Distribution of Median House Value')
plt.xlabel('Median House Value')
plt.ylabel('Frequency')
plt.show()

# Analyze the relationship between numerical features and the target
variable
plt.figure(figsize=(14, 10))
correlation_matrix = df[numerical_columns +
['median_house_value']].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()

# Scatter plots of key relationships
for column in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=df[column], y=df['median_house_value'])
    plt.title(f'{column} vs Median House Value')
    plt.xlabel(column)
    plt.ylabel('Median House Value')
    plt.show()

# Box plot for categorical variable
plt.figure(figsize=(12, 6))
sns.boxplot(x='ocean_proximity', y='median_house_value', data=df)
plt.title('Median House Value by Ocean Proximity')
plt.show()
```

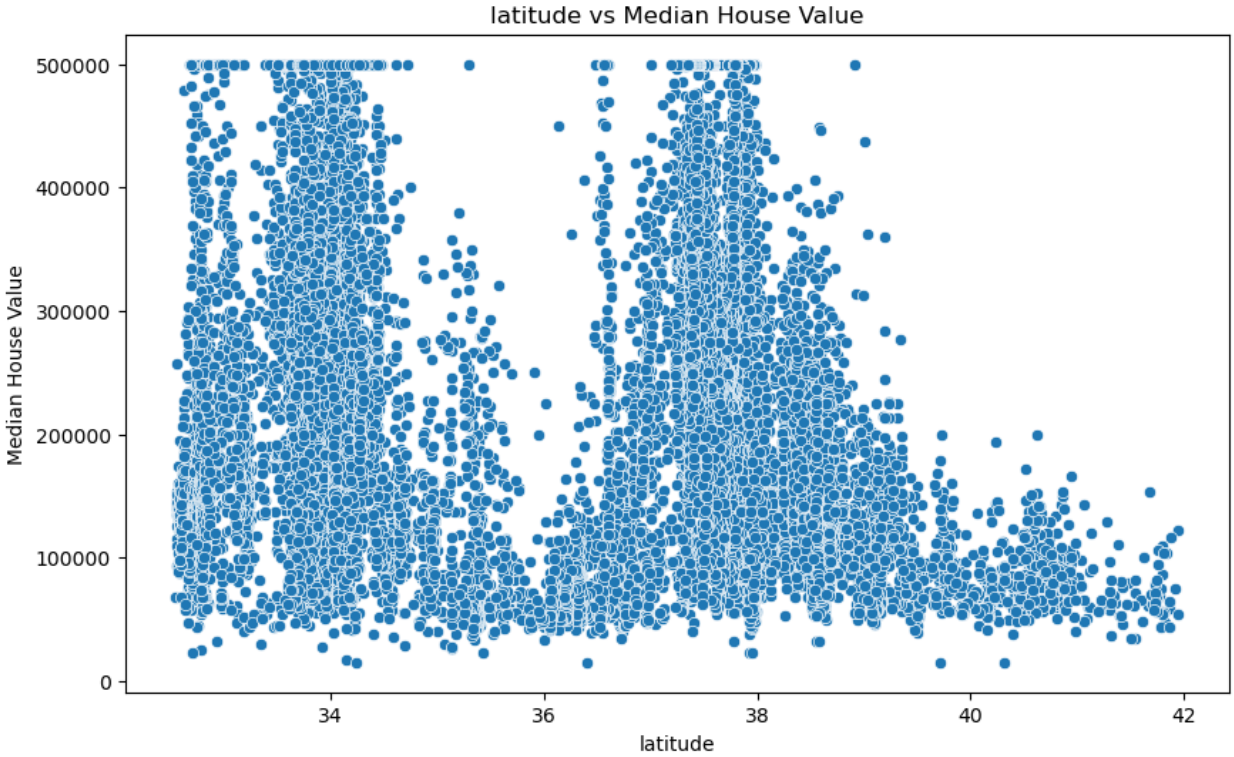
```
D:\Anaconda\Lib\site-packages\sklearn\preprocessing\_encoders.py:868:
FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2
and will be removed in 1.4. `sparse_output` is ignored unless you
```

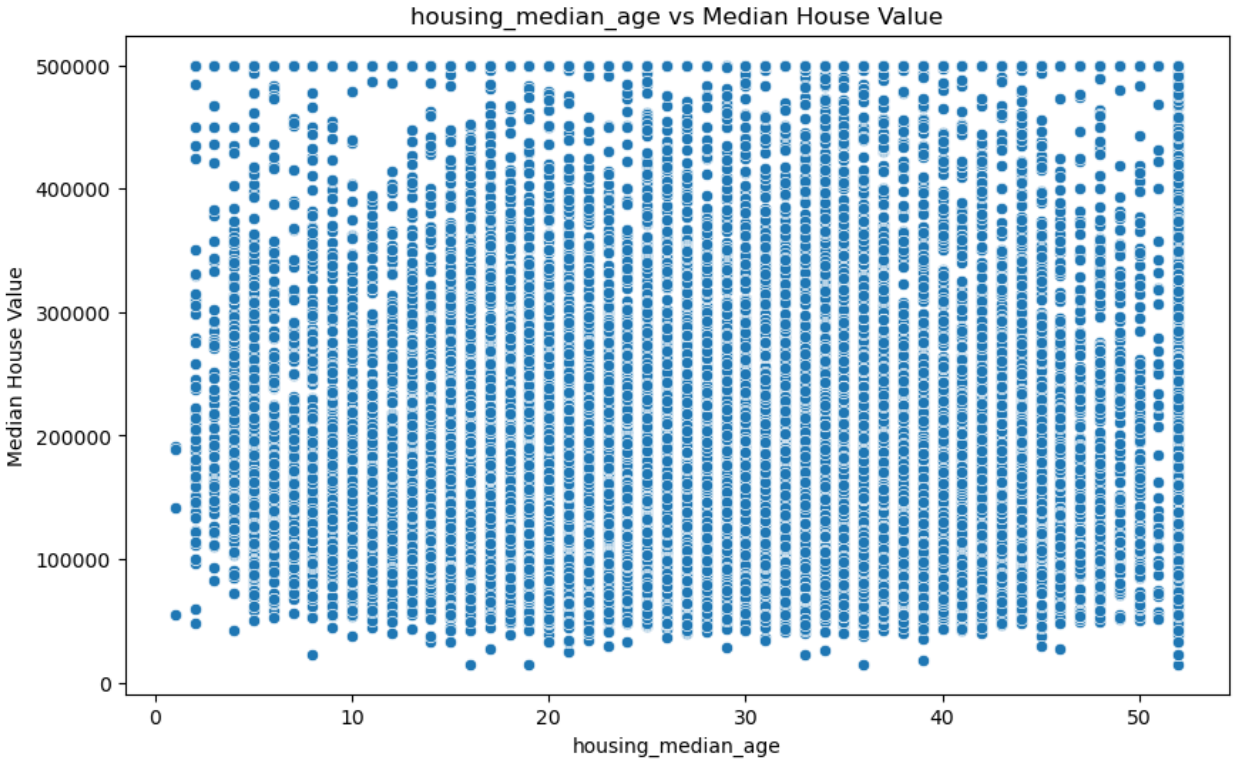
```
leave `sparse` to its default value.
  warnings.warn(

Mean Absolute Error: 50670.73824097188
Mean Squared Error: 4908476721.156617
R-squared: 0.6254240620553605
Model and preprocessor saved successfully!

Enter longitude:  -122
Enter latitude:  37
Enter housing_median_age:  41
Enter total_rooms:  880
Enter total_bedrooms:  129
Enter population:  322
Enter households:  126
Enter median_income:  8
Enter ocean_proximity (e.g., NEAR BAY, INLAND, etc.):  NEAR BAY

D:\Anaconda\Lib\site-packages\sklearn\base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with
feature names
  warnings.warn(
D:\Anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future
version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):

Predicted Median House Value: $413986.77
```

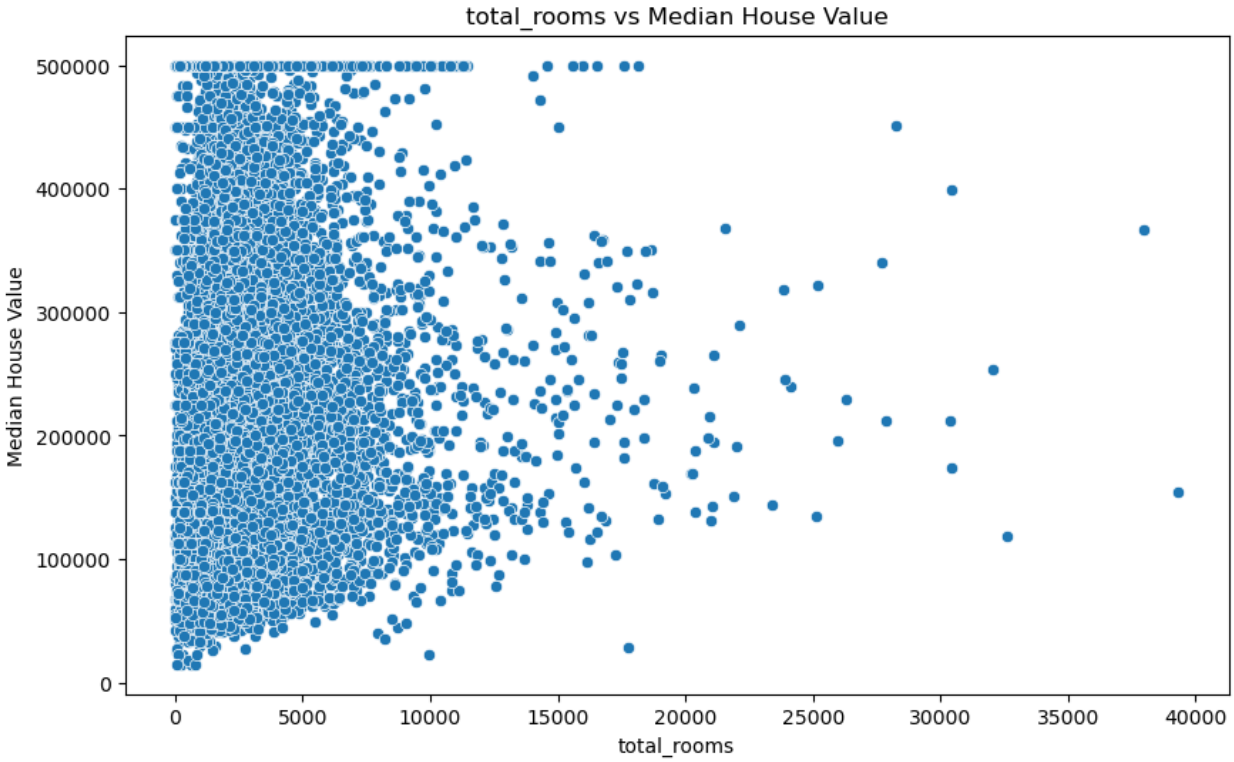

Distribution of Median House Value
Frequency
Median House Value
0
200
400
600
800
1000
0
100000
200000
300000
400000
500000

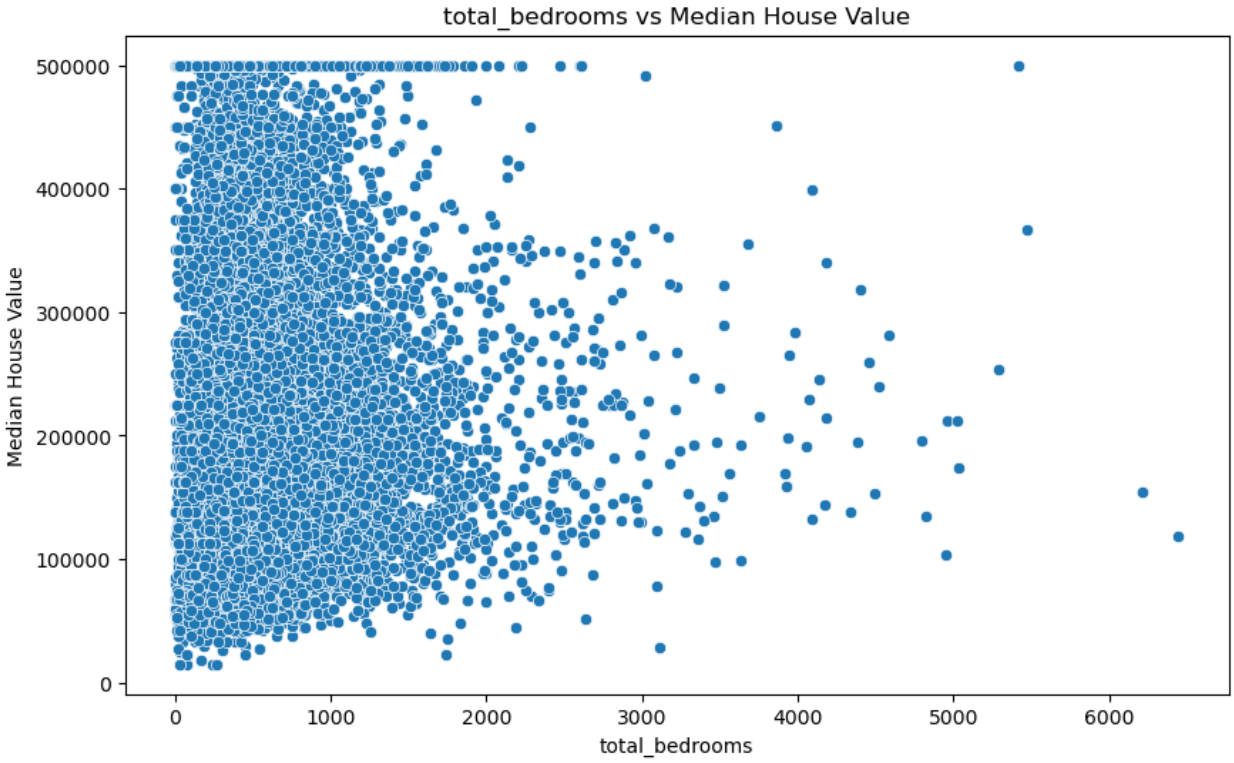

Correlation Matrix
longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
median_house_value
1
-0.92
-0.11
0.045
0.069
0.1
0.055
-0.015
-0.046
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

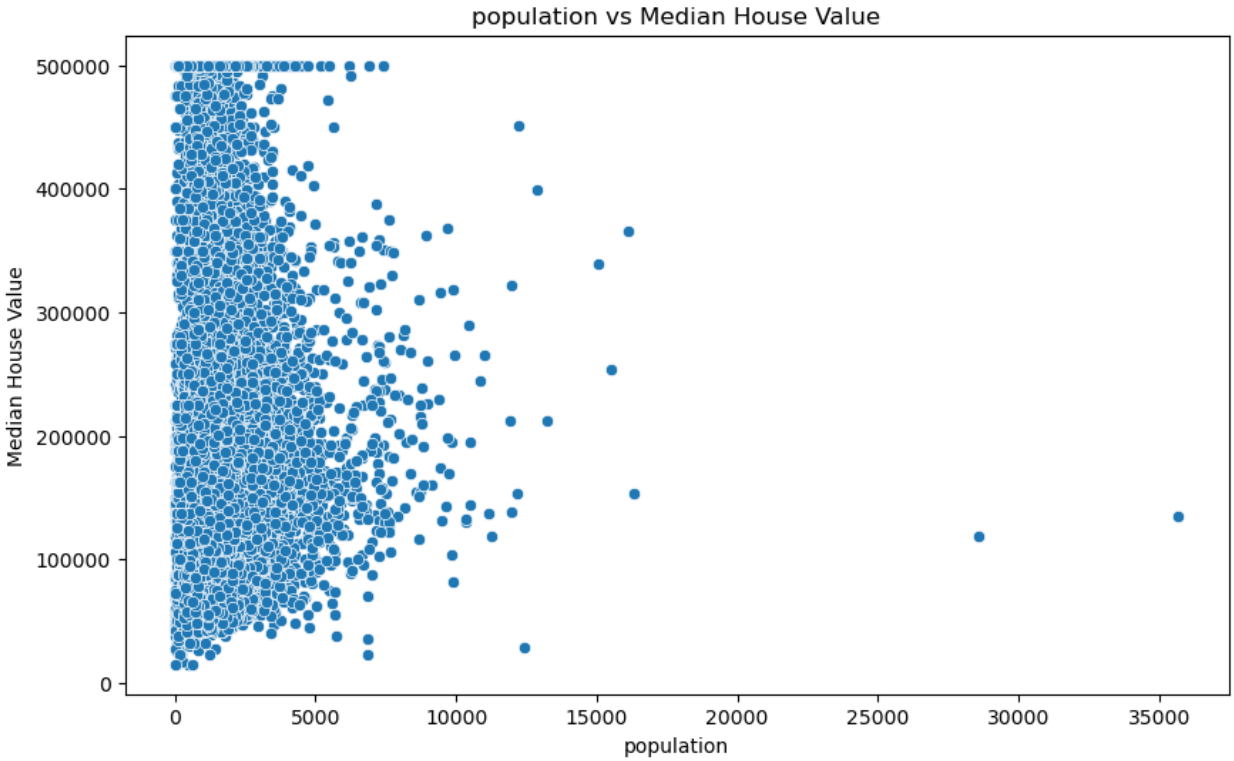

longitude vs Median House Value
Median House Value
500000
400000
300000
200000
100000
0
−124
−122
−120
−118
−116
−114
longitude

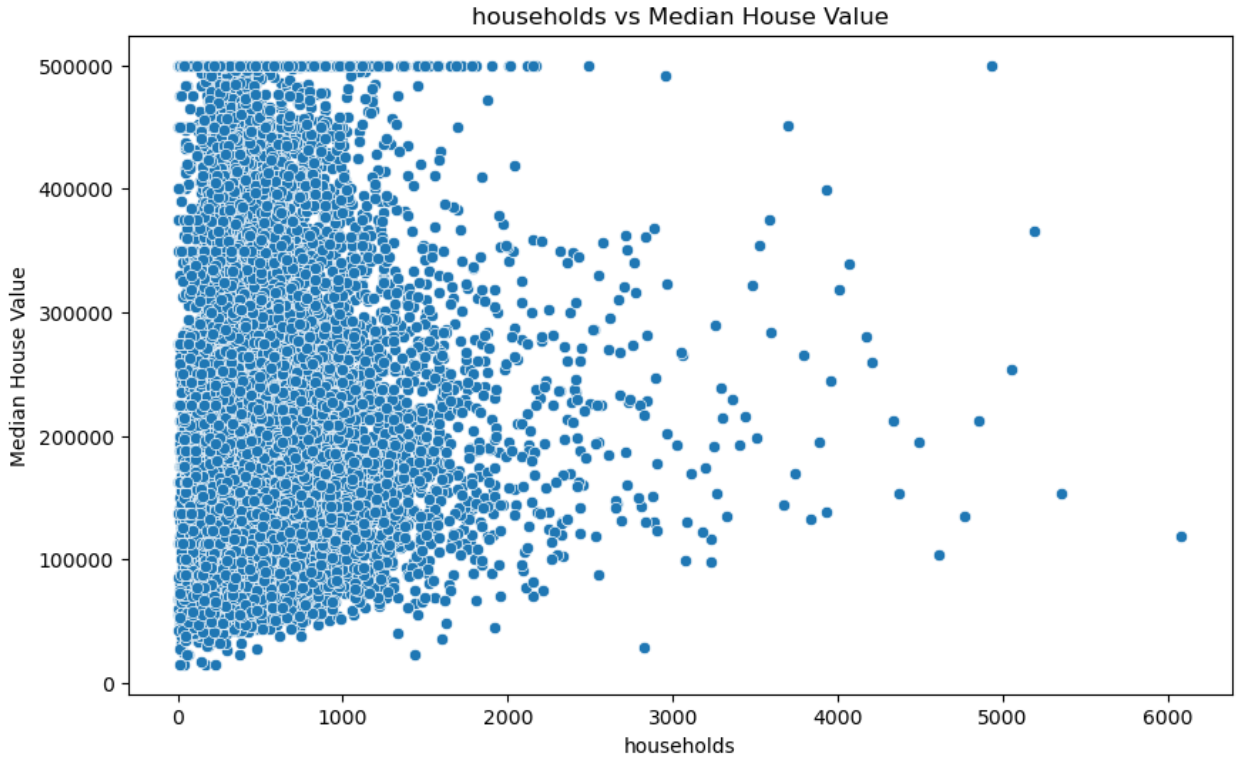

latitude vs Median House Value
Median House Value
500000
400000
300000
200000
100000
0
34
36
38
40
42
latitude

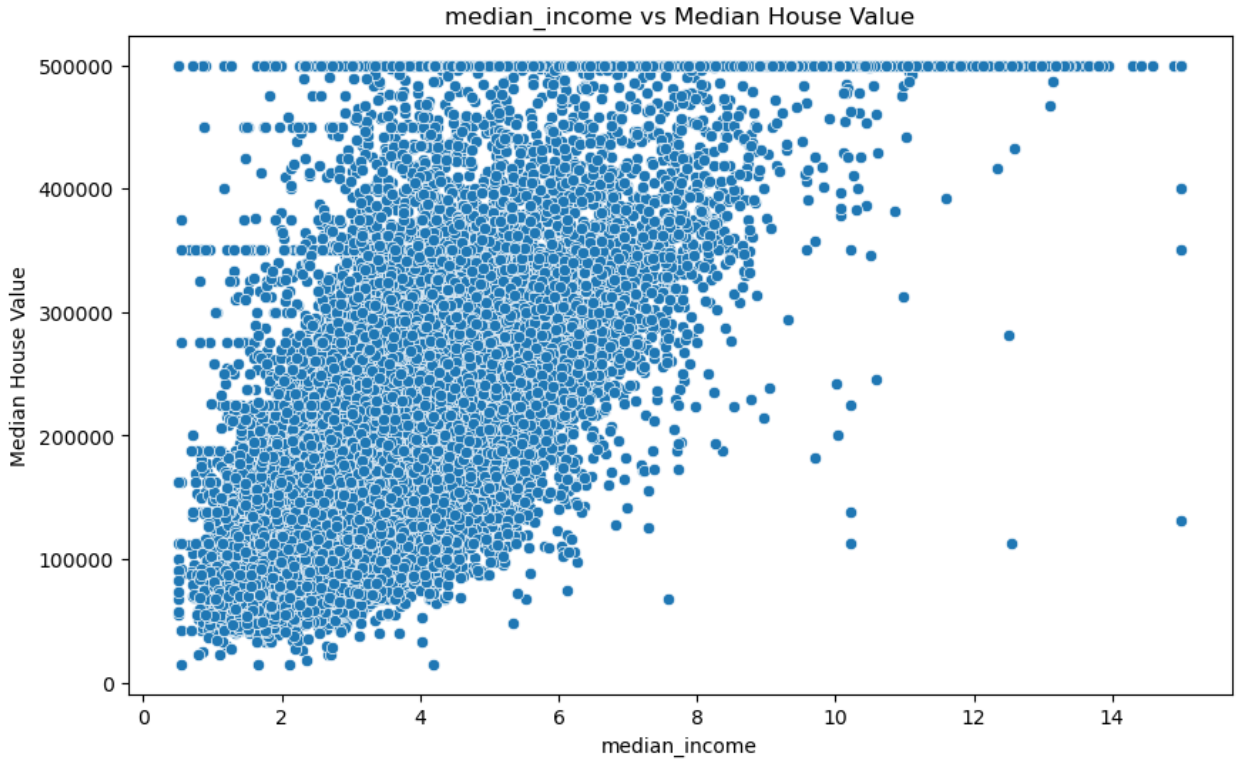

housing_median_age vs Median House Value
Median House Value
housing_median_age
0
100000
200000
300000
400000
500000
0
10
20
30
40
50

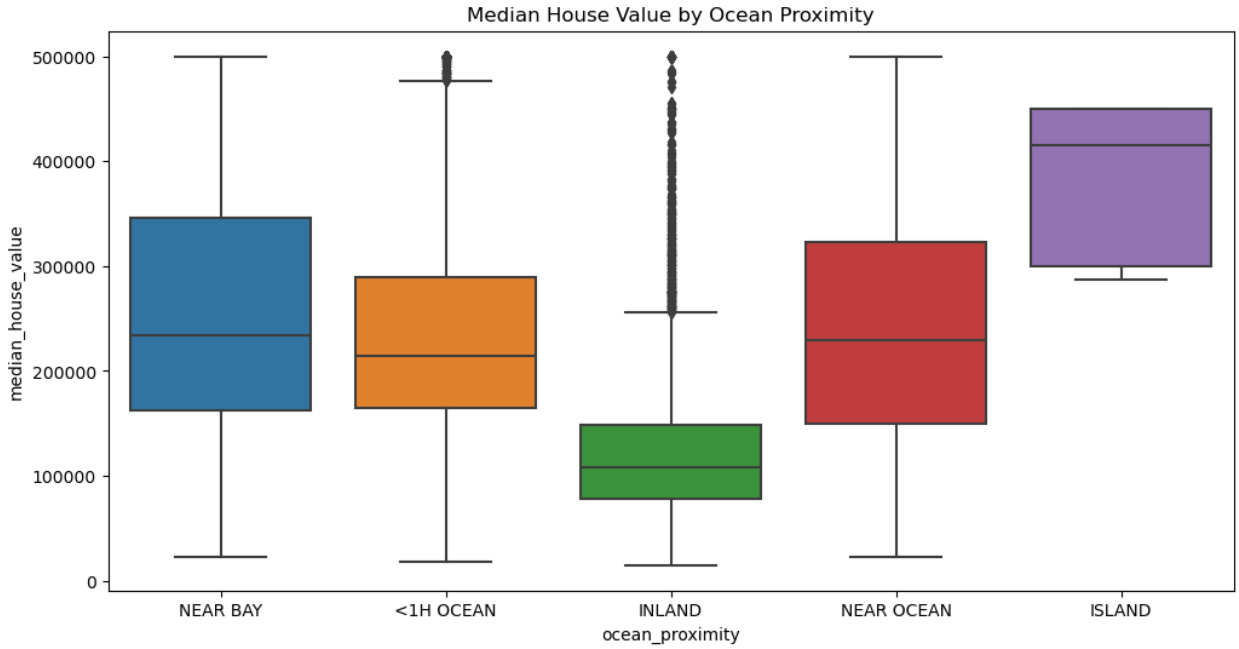

total_rooms vs Median House Value
Median House Value
total_rooms
0
100000
200000
300000
400000
500000
0
5000
10000
15000
20000
25000
30000
35000
40000


total_bedrooms vs Median House Value
Median House Value
total_bedrooms
0
100000
200000
300000
400000
500000
0
1000
2000
3000
4000
5000
6000


population vs Median House Value
Median House Value
population
0
100000
200000
300000
400000
500000
0
5000
10000
15000
20000
25000
30000
35000


households vs Median House Value
Median House Value
households
0
100000
200000
300000
400000
500000
0
1000
2000
3000
4000
5000
6000


median_income vs Median House Value
Median House Value
median_income
0
100000
200000
300000
400000
500000
0
2
4
6
8
10
12
14


Median House Value by Ocean Proximity
median_house_value
500000
400000
300000
200000
100000
0
NEAR BAY
<1H OCEAN
INLAND
NEAR OCEAN
ISLAND
ocean_proximity