DEVELOPERS HUB CORPORATION

PROJECT TITLE: HOUSING PRICE PREDICTION PROJECT

Name: **Ayesha Liaqat**
Intern ID: **DHC-1059**
Department: **AI / Machine Learning**
ayeshaliaqat908@gmail.com

Machine Learning –TASK 2 -

California Housing Price Prediction Project

Objective:

The goal of this project is to walk through the complete machine learning pipeline using the California Housing Prices dataset. It follows the steps outlined in Chapter 2 of *Hands-On Machine Learning with Scikit-Learn*. By the end of the task, we apply all stages of ML including data loading, cleaning, feature engineering, model training, evaluation, and comparison.

California Housing Price Prediction Project Summary

Step 1: Data Loading

Dataset used: *California housing dataset* (CSV or via fetch_california_housing()).

Loaded using pandas read_csv() function.

Displayed first few rows using head() and checked data types, shape, and null values.

Step 2: Exploratory Data Analysis

Plotted histograms for numerical columns like population, rooms, and income.

Checked correlation matrix using seaborn heatmap.

Scatter plots of longitude vs latitude colored by house value to observe geographical trends.

Step 3: Data Cleaning

Handled missing values in 'total_bedrooms' using median imputation.

Converted categorical variable 'ocean_proximity' using label encoding.

Standardized the features using StandardScaler to bring them to a common scale.

Step 4: Feature Selection

Selected all columns except the target column median_house_value.

Split the data into features (X) and target (y).

Performed train-test split using 80:20 ratio.

Step 5: Model Training

Trained and evaluated multiple models:

Linear Regression

Lasso Regression

Ridge Regression

Used r2_score, MAE, MAPE, and MSE to evaluate each model.

Compared models based on performance metrics.

Step 6: Visualization of Predictions

Plotted actual vs predicted prices for each model.

Used line plots to compare first 20 predicted and actual values.

Displayed side-by-side graphs for all three models.

Step 7: Results Summary (Model Comparison)

Created a performance table showing:

R^2 Score

Adjusted R^2

MAE

MAPE

MSE

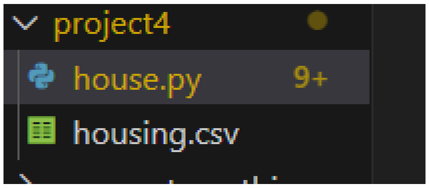
This helped identify which model performed best on unseen test data.

Conclusion:

This project allowed hands-on application of the complete machine learning pipeline. It strengthened understanding of core steps like data preprocessing, model fitting, and evaluation. The comparison between models showed how regularization (Lasso, Ridge) affects performance. This task directly applied the theoretical knowledge from Chapter 2 and provided practical experience with real-world data.

- .

Files and Folders



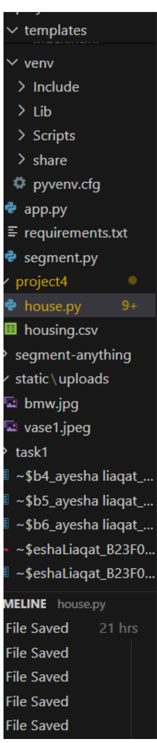
Libraries

```python
# 📦 Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 🔧 Preprocessing & model tools
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import r2_score, mean_absolute_error, mean_absolute_percentage_error, mean_squared_error

# 📊 Statsmodels for VIF (optional advanced diagnostics)
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
```

House.py




```python
# 🏠 Read dataset
df = pd.read_csv(r"f:\artificial intelligence\project4\housing.csv")
print(df.head())

# ---------------------------
# 🧹 Basic data exploration
# ---------------------------
print(f"Shape of dataset: {df.shape}")
print("\nMissing values per column:\n", df.isnull().sum())
print("\nDuplicate rows:", df.duplicated().sum())
print("\nData types:\n", df.dtypes)

# 🔥 Correlation heatmap
plt.figure(figsize=(16, 12))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# 🛠 Fill missing values
df['total_bedrooms'].fillna(df['total_bedrooms'].median(), inplace=True)

# 🔍 Re-check for missing values
print("\nMissing values after fill:\n", df.isnull().sum())

# -------------------------------------
# 📊 Distributions of numeric features
# -------------------------------------
fig, ax = plt.subplots(4, 2, figsize=(14, 12))
cols = ['longitude', 'latitude', 'housing_median_age', 'total_rooms',
        'total_bedrooms', 'population', 'households', 'median_house_value']
for i, col in enumerate(cols):
    sns.histplot(df[col], kde=True, ax=ax[i//2, i%2])
plt.tight_layout()
plt.show()
```

templates
venv
Include
Lib
Scripts
share
pyvenv.cfg
app.py
requirements.txt
segment.py
project4
house.py 9+
housing.csv
segment-anything
static\uploads
bmw.jpg
vase1.jpeg
task1
~$b4_ayesha liaqat_...
~$b5_ayesha liaqat_...
~$b6_ayesha liaqat_...
~$eshaLiaqat_B23F0...
~$eshaLiaqat_B23F0...
TIMELINE house.py
File Saved 21 hrs
File Saved
File Saved

```python
# 📊 Distributions of numeric features
# -------------------------------------
fig, ax = plt.subplots(4, 2, figsize=(14, 12))
cols = ['longitude', 'latitude', 'housing_median_age', 'total_rooms',
        'total_bedrooms', 'population', 'households', 'median_house_value']
for i, col in enumerate(cols):
    sns.histplot(df[col], kde=True, ax=ax[i//2, i%2])
plt.tight_layout()
plt.show()

# Categorical vs Target
sns.barplot(x='ocean_proximity', y='median_house_value', data=df, palette="Set1")
plt.title("House Value by Ocean Proximity")
plt.xticks(rotation=45)
plt.show()

# 🌍 House value across geographic coordinates
plt.figure(figsize=(12, 8))
sns.scatterplot(data=df, x='longitude', y='latitude', hue='median_house_value', palette='coolwarm', alpha=0.5)
plt.title("Geographic Distribution of Median House Value")
plt.show()

# 🧠 Label encode 'ocean_proximity'
le = LabelEncoder()
df['ocean_proximity'] = le.fit_transform(df['ocean_proximity'])

# 🔁 Re-check Correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="Pastel1")
plt.title("Correlation Matrix After Encoding")
plt.show()
```

house.py 9+ × california-housing-prices-prediction.ipynb ●

artificial intelligence > project4 > house.py > ...

```python
# 🧪 Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# -------------------------------------
# 📊 Fit Linear, Lasso & Ridge Models
# -------------------------------------
# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

# Lasso Regression
lasso = Lasso()
lasso.fit(X_train, y_train)
y_pred_lasso = lasso.predict(X_test)

# Ridge Regression
ridge = Ridge()
ridge.fit(X_train, y_train)
y_pred_ridge = ridge.predict(X_test)

# -------------------------------------
# 📏 Metrics Function
# -------------------------------------
def get_metrics(p, y, y_pred):
    n = len(y)
    r2 = r2_score(y, y_pred)
    adjusted_r2 = 1 - (((1 - r2) * (n - 1)) / (n - p - 1))
    mae = mean_absolute_error(y, y_pred)
    mape = mean_absolute_percentage_error(y, y_pred)
    mse = mean_squared_error(y, y_pred)
    return r2, adjusted_r2, mae, mape, mse
```

```python
# Linear Regression
df_lr = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index(drop=True)
axes[0].plot(df_lr[:20])
axes[0].set_title("Linear Regression")
axes[0].legend(["Actual", "Predicted"], loc="upper left")

# Lasso Regression
df_lasso = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_lasso}).reset_index(drop=True)
axes[1].plot(df_lasso[:20])
axes[1].set_title("Lasso Regression")
axes[1].legend(["Actual", "Predicted"], loc="upper left")

# Ridge Regression
df_ridge = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_ridge}).reset_index(drop=True)
axes[2].plot(df_ridge[:20])
axes[2].set_title("Ridge Regression")
axes[2].legend(["Actual", "Predicted"], loc="upper left")

plt.tight_layout()
plt.show()

# ----------------------------------------
# 📊 Compare Model Performance Metrics
# ----------------------------------------
p = X_train.shape[1]  # Number of features
performance_df = pd.DataFrame([
    get_metrics(p, y_test, y_pred),
    get_metrics(p, y_test, y_pred_lasso),
    get_metrics(p, y_test, y_pred_ridge)
],
columns=['R2', 'Adjusted R2', 'MAE', 'MAPE', 'MSE'],
index=['Linear Regression', 'Lasso Regression', 'Ridge Regression'])

print("\n📈 Model Performance Comparison:")
print(performance_df)
```

Outputs:

```
Arsalan@Aayeshaa MINGW64 /f/artificial intelligence (main)
$ python -u "f:\artificial intelligence\project4\house.py"
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  median_house_value ocean_proximity
0    -122.23     37.88                41.0        880.0           129.0       322.0       126.0         8.3252            452600.0        NEAR BAY
1    -122.22     37.86                21.0       7099.0          1106.0      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2    -122.24     37.85                52.0       1467.0           190.0       496.0       177.0         7.2574            352100.0        NEAR BAY
3    -122.25     37.85                52.0       1274.0           235.0       558.0       219.0         5.6431            341300.0        NEAR BAY
4    -122.25     37.85                52.0       1627.0           280.0       565.0       259.0         3.8462            342200.0        NEAR BAY
Shape of dataset: (20640, 10)

Missing values per column:
 longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64

Duplicate rows: 0

Data types:
 longitude             float64
latitude              float64
housing_median_age    float64
total_rooms           float64
total_bedrooms        float64
population            float64
households            float64
median_income         float64
median_house_value    float64
ocean_proximity        object
dtype: object
```

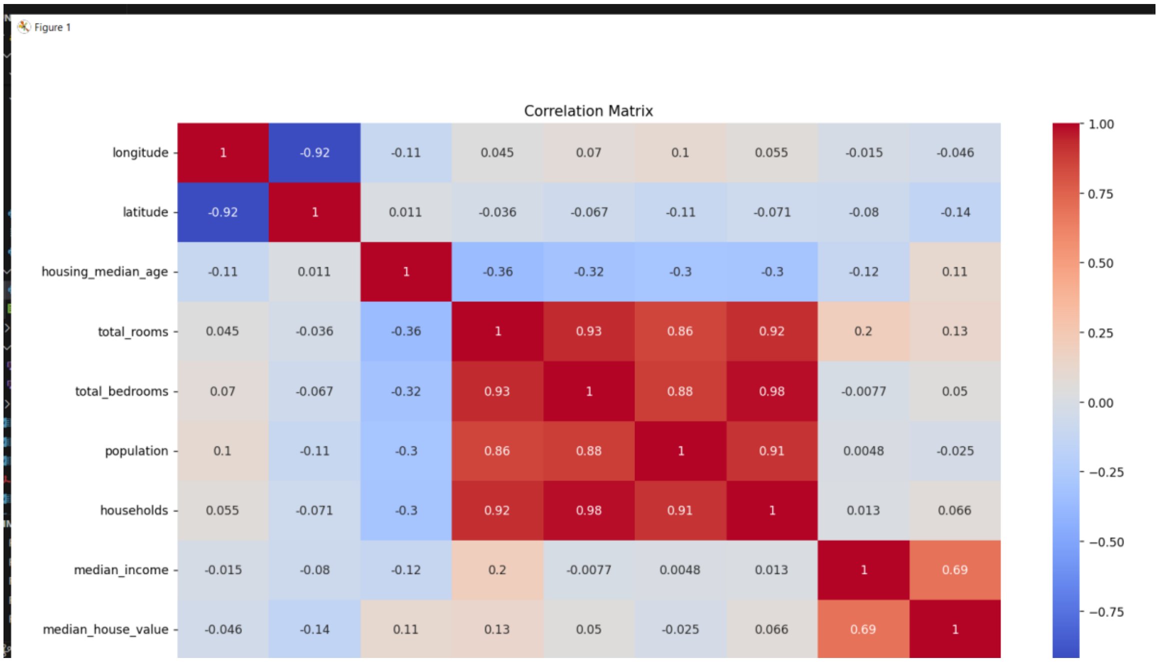
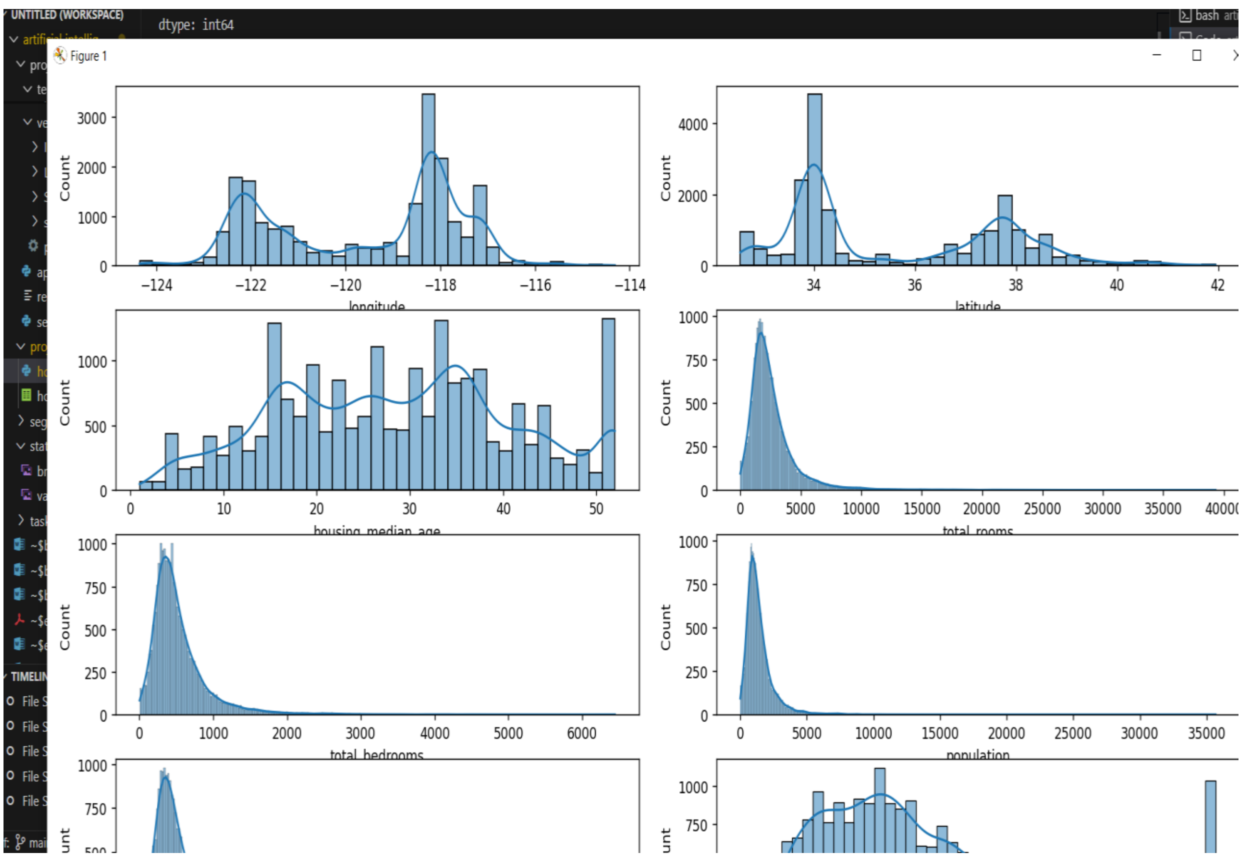
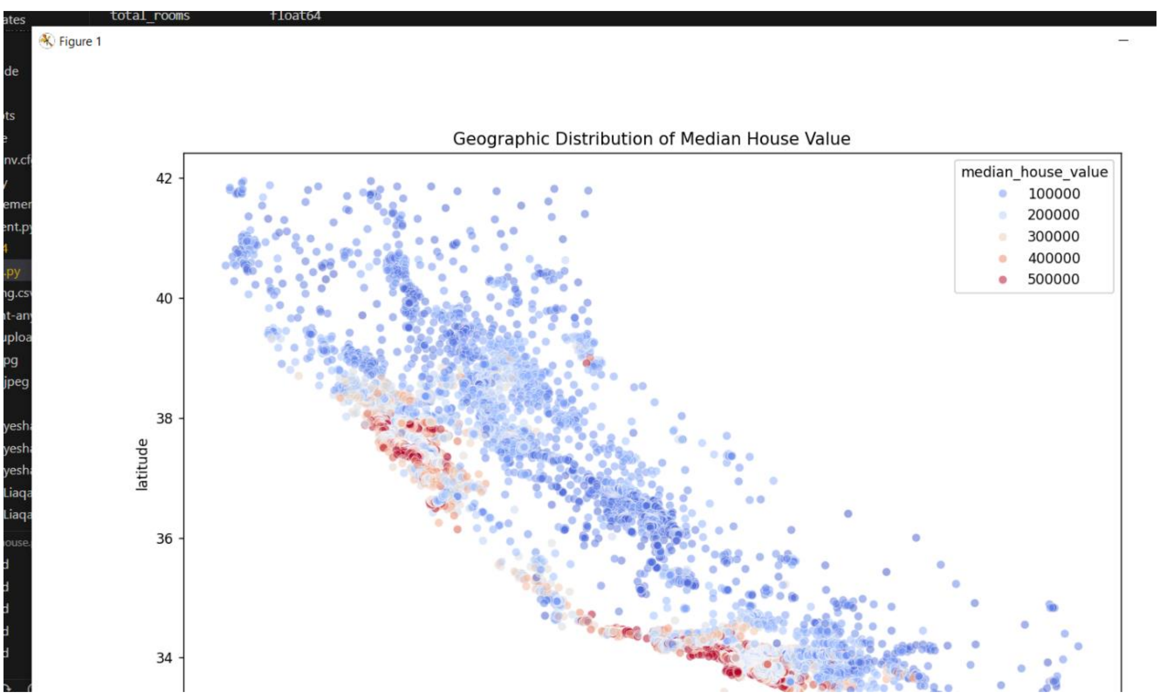
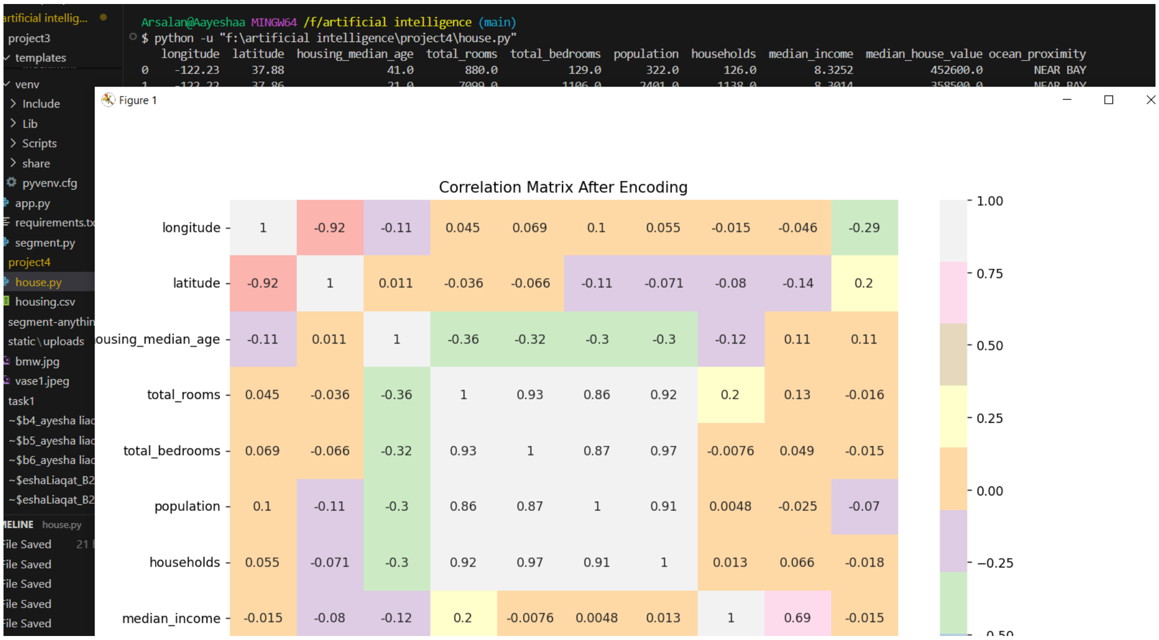


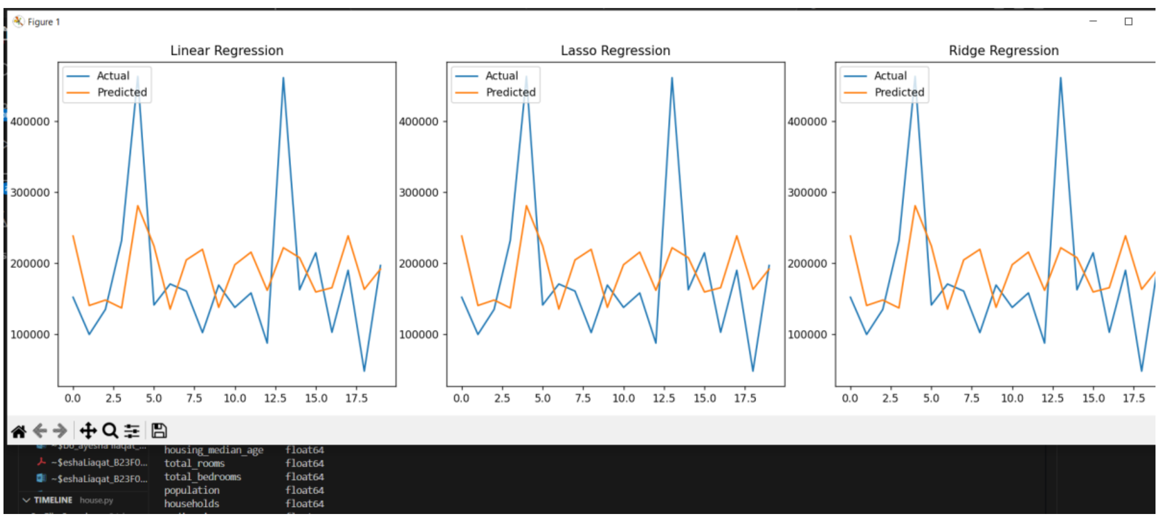
Figure 1
Correlation Matrix
longitude 1 -0.92 -0.11 0.045 0.07 0.1 0.055 -0.015 -0.046
latitude -0.92 1 0.011 -0.036 -0.067 -0.11 -0.071 -0.08 -0.14
housing_median_age -0.11 0.011 1 -0.36 -0.32 -0.3 -0.3 -0.12 0.11
total_rooms 0.045 -0.036 -0.36 1 0.93 0.86 0.92 0.2 0.13
total_bedrooms 0.07 -0.067 -0.32 0.93 1 0.88 0.98 -0.0077 0.05
population 0.1 -0.11 -0.3 0.86 0.88 1 0.91 0.0048 -0.025
households 0.055 -0.071 -0.3 0.92 0.98 0.91 1 0.013 0.066
median_income -0.015 -0.08 -0.12 0.2 -0.0077 0.0048 0.013 1 0.69
median_house_value -0.046 -0.14 0.11 0.13 0.05 -0.025 0.066 0.69 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75


UNTITLED (WORKSPACE)
dtype: int64
Figure 1
Count
longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population


House Value by Ocean Proximity
median_house_value
400000
300000
200000
100000
0
ISLAND
f:\artificial intelligence\project4\house.py:57: FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.


Geographic Distribution of Median House Value
median_house_value
100000
200000
300000
400000
500000
latitude
42
40
38
36
34





📈 Model Performance Comparison:

	R2	Adjusted R2	MAE	MAPE	MSE
Linear Regression	0.636100	0.635305	50121.081587	0.297397	4.839194e+09
Lasso Regression	0.636101	0.635306	50121.394309	0.297399	4.839174e+09
Ridge Regression	0.636107	0.635311	50121.337926	0.297395	4.839107e+09

Arsalan@Aayeshaa MINGW64 /f/artificial intelligence (main)
$