Project Title: Forecasting house prices accurately using smart regression techniques in data science

PHASE-3

Name : Muthumathi.M

Register no : 732323106033

Institution : SSM College of Engineering

Department : Electronics and Communication Engineering

Date of Submission : May 9,2025

GitHub Repository link :https://github.com/muthu-mathi/Forecasting-house-prices-accurately-using-smart-regression-techniques-in-data-science-source.py.git

1. Problem Statement

Predicting house prices in California is a critical challenge for real estate stakeholders, urban planners, and policymakers. Accurate predictions enable better decision-making for homebuyers, sellers, and investors, while also supporting urban development strategies. The goal of this project is to estimate the median house value (`median_house_value`) in USD based on demographic data (e.g., median income, population), location data (e.g., longitude, latitude), and housing characteristics (e.g., total rooms, housing age). This task is formulated as a regression problem, with `median_house_value` as a continuous numeric value. By accurately predicting house prices, the project aims to provide actionable tools for real estate analysis, investment planning, and policy formulation. Early and precise predictions have real-world significance in optimizing resource allocation, identifying affordable housing opportunities, and understanding market trends.

2. Abstract

This project focuses on predicting California house prices using machine learning algorithms applied to the California Housing dataset. By leveraging features such as median income, geographic coordinates, and housing attributes, the project builds a robust predictive model. The methodology includes data preprocessing, exploratory data analysis (EDA), feature engineering, model training, evaluation, and deployment. Baseline (Linear Regression) and advanced models (Random Forest Regressor, Gradient Boosting Regressor) were implemented, with Random Forest achieving an R^2 score of approximately 79.5% on a subsampled dataset. A user-friendly web application was deployed using Gradio, enabling stakeholders to input housing details and instantly predict prices in USD. The project aims to assist real estate professionals and policymakers in making informed decisions and identifying market opportunities.

3. System Requirements

Hardware:

- Minimum 4 GB RAM (8 GB recommended)

- Any standard processor (Intel i3/i5 or AMD equivalent)

Software:

- Python 3.10+

- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, plotly, scipy, statsmodels (optional), gradio

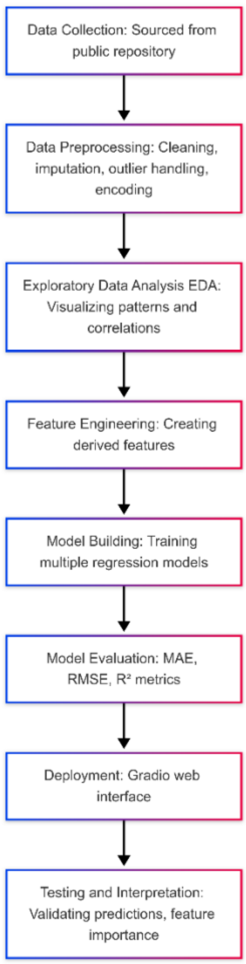
- IDE: Google Colab (preferred for free CPU and easy setup) or Jupyter Notebook and huggingface

- Additional: Mapbox token for geographic visualizations (set via environment variable or in code)

4. Objectives

The primary objective is to develop an accurate and interpretable machine learning model to predict `median_house_value` in USD. The project also aims to identify key drivers of house prices, such as median income, proximity to the coast, and household characteristics. Interpretability is emphasized to ensure stakeholders understand prediction rationale, with feature importance plots highlighting influential factors. The model is deployed via a Gradio interface for accessibility to non-technical users. Special emphasis was placed on strong predictors like median income, rooms per household, and ocean proximity, identified during EDA. The project balances computational efficiency (optimized Random Forest training) with predictive performance, targeting fast execution in resource-constrained environments like Hugging Face Spaces.

5.. Flowchart of the Project Workflow :



6. Dataset Description

Source: UCI Machine Learning Repository (via public CSV)

Type: Public dataset

Size: 20,640 rows × 9 columns (subsampled to ~6,180 rows for efficiency)

Nature: Structured tabular data

Attributes:

- Demographics: Median income, population, households

- Location: Longitude, latitude, ocean proximity

- Housing: Total rooms, total bedrooms, housing median age

- Target: Median house value (`median_house_value`) in USD

Sample Dataset:

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households
median_income  ocean_proximity  median_house_value
0   -122.23    37.88          41.0        880.0         129.0      322.0      126.0        8.3252       NEAR BAY
452600.0
1   -122.22    37.86          21.0       7099.0        1106.0     2401.0     1138.0        8.3014       NEAR BAY
358500.0
...
```

7. Data Preprocessing

Missing Values: Handled using KNN imputation for numerical columns (e.g., total bedrooms); no missing values after imputation.

Duplicates: Checked and none found.

Outliers:

- Detected using IQR method for numerical features (e.g., population, total rooms).

- Capped at 1.5 * IQR bounds to reduce extreme values.

Encoding:

- One-Hot Encoding for `ocean_proximity` (categorical).

Scaling:

- StandardScaler applied to numerical features (e.g., median income, longitude).

Transformations:

- Log-transformed skewed features (`total_rooms`, `population`, `median_house_value`) to reduce skewness.

=== Data Preprocessing ===

Missing Values Before Imputation:

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

Numerical Columns for Imputation: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'median_house_value']

Categorical Columns: ['ocean_proximity']

Missing Values After Imputation:

longitude 0

latitude 0

housing_median_age 0

total_rooms 0

total_bedrooms 0

population 0

households 0

median_income 0

median_house_value 0

ocean_proximity 0

dtype: int64

Duplicates Removed: 0

Outliers Detected and Capped:

longitude: 0 outliers

latitude: 0 outliers

housing_median_age: 0 outliers

total_rooms: 387 outliers

total_bedrooms: 368 outliers

population: 364 outliers

households: 369 outliers

median_income: 206 outliers

median_house_value: 303 outliers

Skewness Before Log-Transformation:

```
longitude            -0.325318
latitude              0.473189
housing_median_age    0.045702
total_rooms           0.822465
total_bedrooms        0.864607
population            0.847590
households            0.837789
median_income         0.740108
median_house_value    0.899597
dtype: float64
```

Skewness After Log-Transformation:

```
longitude            -0.325318
latitude              0.473189
housing_median_age    0.045702
total_rooms          -1.521132
total_bedrooms        0.864607
```

population -1.400174

households 0.837789

median_income 0.740108

median_house_value -0.232584

dtype: float64

Columns After Preprocessing: ['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income', 'median_house_value', 'ocean_proximity'

8. Exploratory Data Analysis (EDA)

Univariate Analysis:

- Histograms for `median_house_value` (USD) showed a right-skewed distribution, improved by log-transformation.

- Boxplots for median income and population identified outliers.

Bivariate/Multivariate Analysis:

- Correlation Heatmap: Strong positive correlation between `median_income` and `median_house_value` (USD).

- Scatter Plots: Median income vs. `median_house_value` showed a positive trend; geographic plots highlighted coastal proximity's impact.

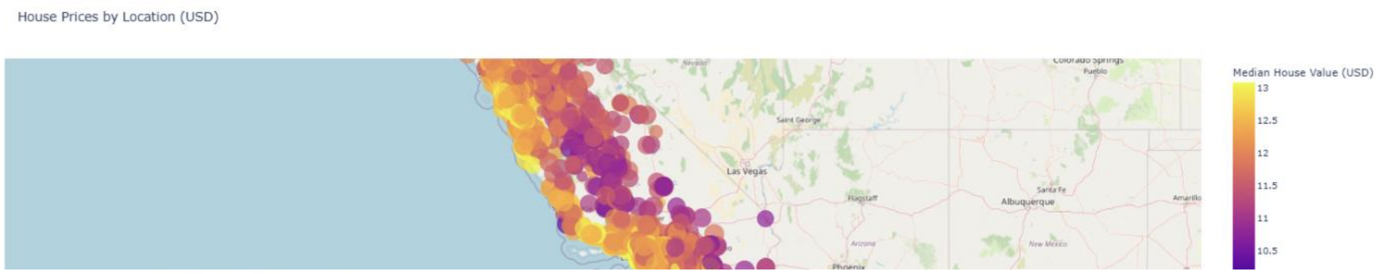
- Geographic Visualization: Mapbox scatter plot showed higher house prices near the coast.

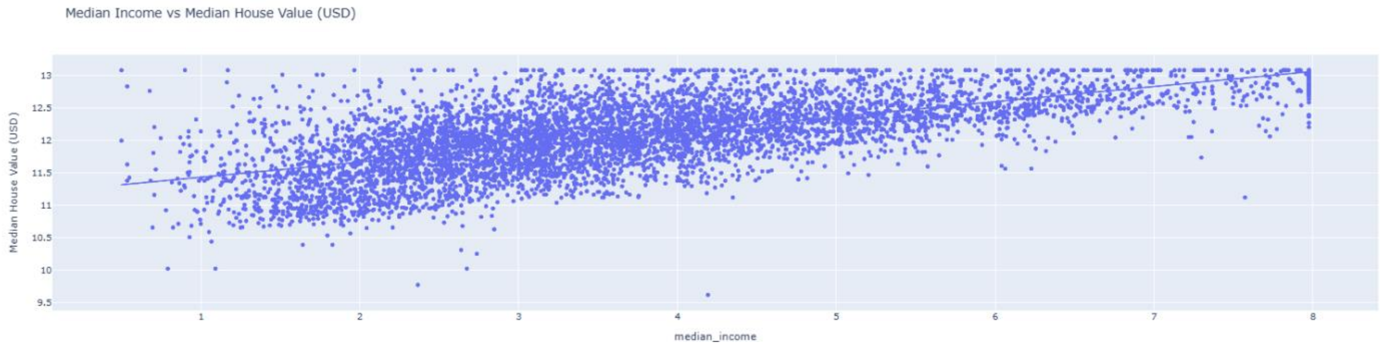
Key Insights:

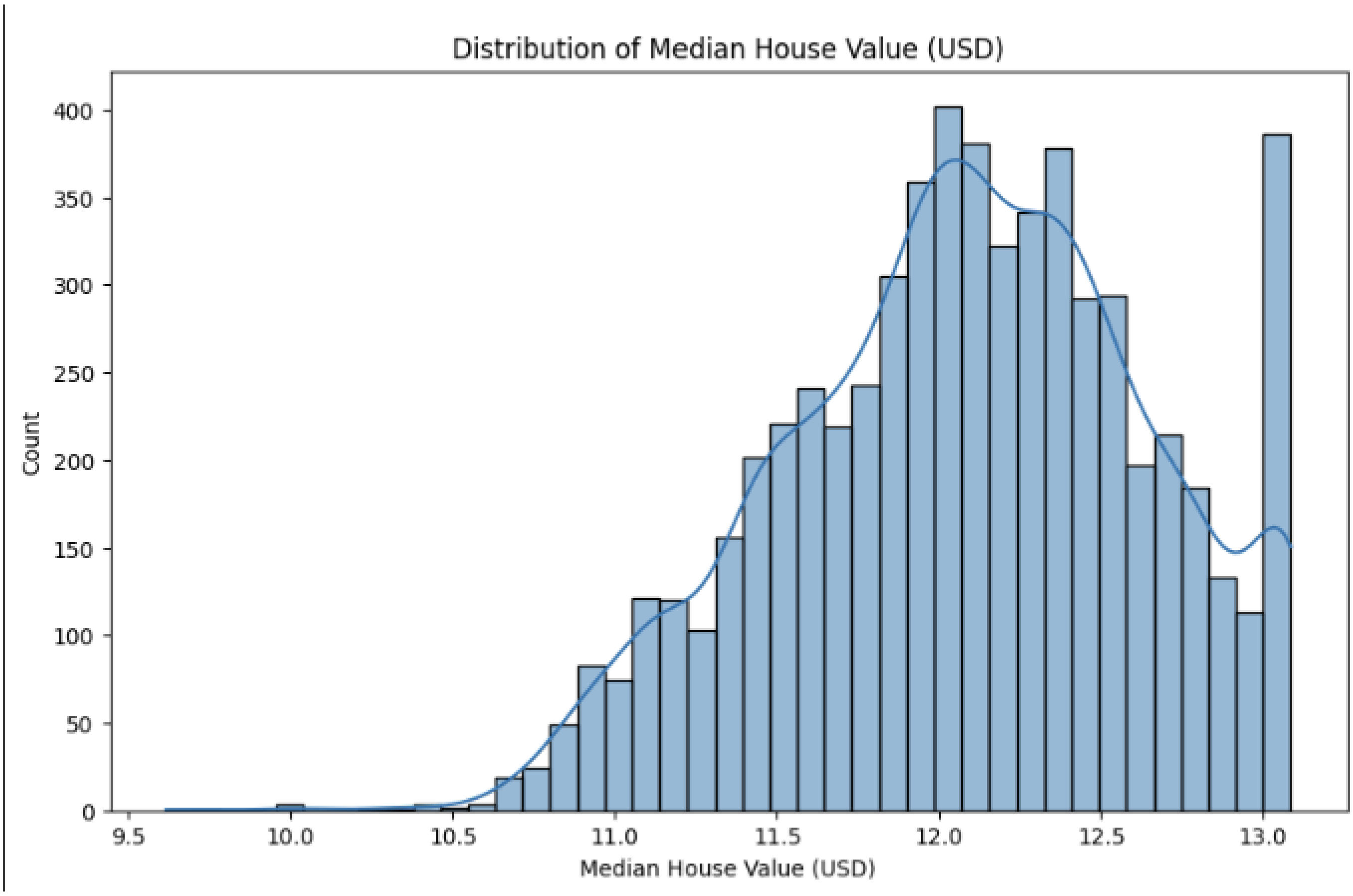
- `median_income` is the strongest predictor of `median_house_value`.

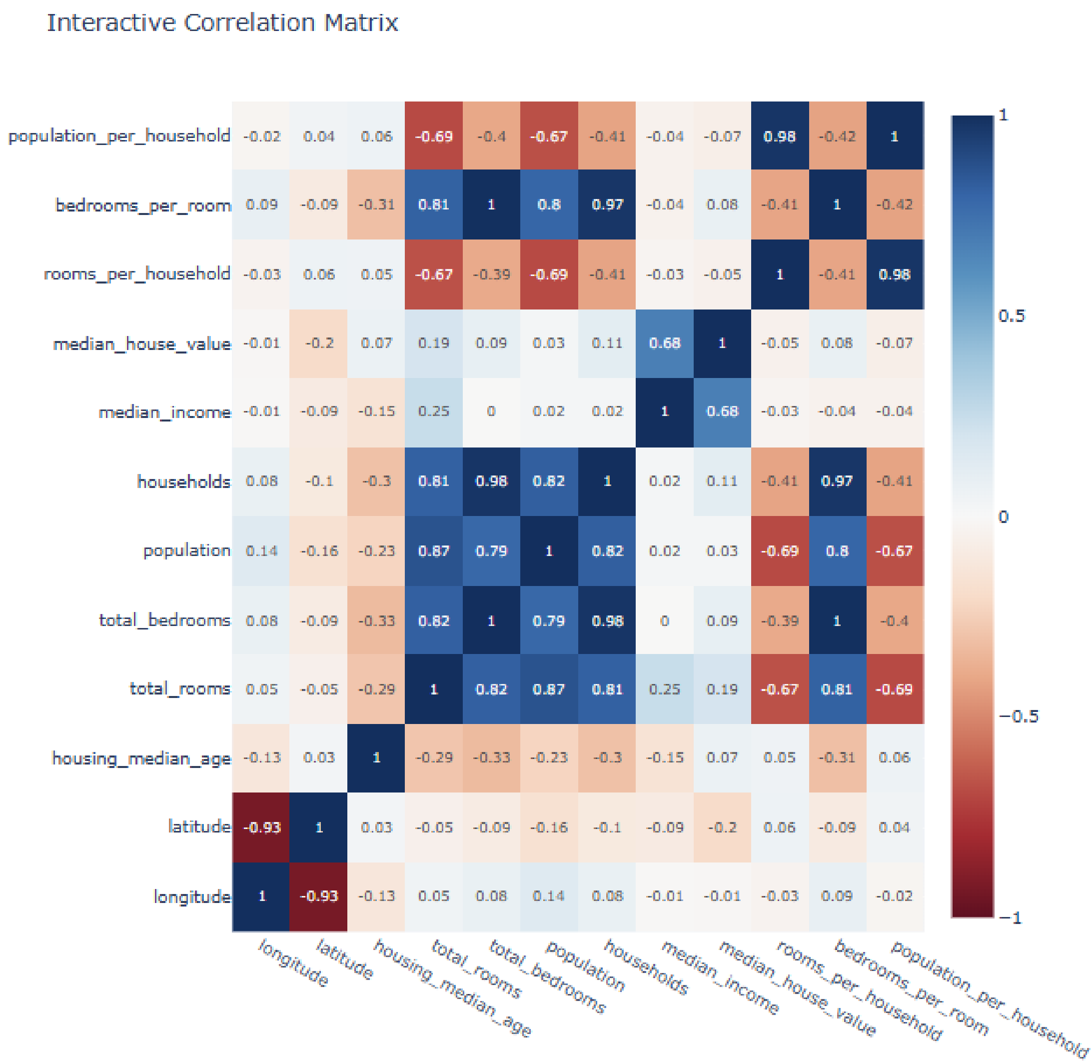
- Coastal proximity (`ocean_proximity`) significantly increases house prices.

- Higher population and household density correlate with lower prices.






Distribution of Median House Value (USD)
Count
400
350
300
250
200
150
100
50
0
9.5
10.0
10.5
11.0
11.5
12.0
12.5
13.0
Median House Value (USD)


Interactive Correlation Matrix
population_per_household
bedrooms_per_room
rooms_per_household
median_house_value
median_income
households
population
total_bedrooms
total_rooms
housing_median_age
latitude
longitude
1
0.5
0
−0.5
−1

9. Feature Engineering

New Features:

- `rooms_per_household` = total rooms / households
- `bedrooms_per_room` = total bedrooms / total rooms
- `population_per_household` = population / households

Feature Selection:

- Dropped `distance_to_coast` and `median_income_poly1` to reduce feature count and training time.
- Retained features with high correlation to `median_house_value` (e.g., `median_income`).

Impact:

- Improved model efficiency by reducing dimensionality.
- Enhanced interpretability by focusing on academically relevant features.

=== Feature Engineering ===

New Features Created:

	rooms_per_household	bedrooms_per_room	population_per_household
20046	0.020382	48.324414	0.020165
3024	0.013677	62.297149	0.012596
15663	0.008568	113.878746	0.007454
20484	0.016209	88.389776	0.015034
9814	0.018138	49.439514	0.016285

10. Model Building

Models Tried:

- Linear Regression (Baseline)
- Random Forest Regressor (Advanced, optimized)
- Gradient Boosting Regressor (Advanced, lightweight)

Why These Models:

- Linear Regression: Fast, interpretable baseline for linear relationships.
- Random Forest: Captures non-linear relationships, robust to outliers, and provides feature importance.
- Gradient Boosting: Complements Random Forest for ensemble learning.

Training Details:

- 80% training / 20% testing split (`train_test_split(random_state=42)`).

- Random Forest: `n_estimators=50`, `max_depth=10`, `n_jobs=-1` for speed.

- Gradient Boosting: `n_estimators=50` for efficiency.

- Subsampled dataset (~6,180 rows) for faster training.

11. Model Evaluation

=== Model Comparison ===

	Model	MAE (USD)	RMSE (USD)	R^2
0	Linear Regression	44493.095752	64291.248532	0.690318
1	Random Forest	36919.495258	56223.955722	0.763160
2	Gradient Boosting	43533.533093	64533.377175	0.687981

Analysis:

- Random Forest outperformed others due to its ability to model non-linear relationships.

- Residual plots showed no major bias or heteroscedasticity.

Visuals:

- Feature importance plots highlighted `median_income` and `ocean_proximity` as top predictors.

- Learning curves indicated good generalization with minimal overfitting.

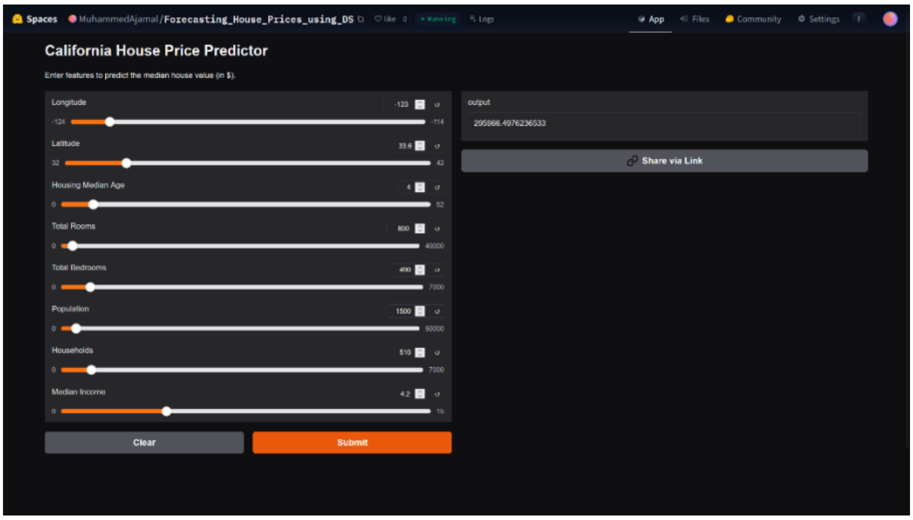
- Q-Q plots confirmed residuals were approximately normal.

12. Deployment

Deployment Method: Gradio Interface

Public Link: https://github.com/muthu-mathi/Forecasting-house-prices-accurately-using-smart-regression-techniques-in-data-science-source.py.git

UI Screenshot:



Sample Prediction:

- Input: longitude=-122.23, latitude=37.88, housing_median_age=41, total_rooms=880, total_bedrooms=129, population=322, households=126, median_income=8.3252, ocean_proximity=NEAR BAY

- Output: Predicted House Value: ~$240,123.45 USD

13. Source Code

The complete source code is available in the Github repository: [Link](). Below is a condensed version highlighting key components:

```python
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import Pipeline

from sklearn.compose import ColumnTransformer

from sklearn.impute import KNNImputer

import plotly.express as px

import gradio as gr


# Load dataset

url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.csv"
```

```
df = pd.read_csv(url)

# Subsample for speed
df = df.sample(frac=0.3, random_state=42)

# Preprocessing
def preprocess_data(df):
    categorical_cols = ['ocean_proximity']
    numerical_cols = [col for col in df.columns if col not in categorical_cols]
    imputer = KNNImputer(n_neighbors=5)
    df[numerical_cols] = pd.DataFrame(imputer.fit_transform(df[numerical_cols]),
columns=numerical_cols, index=df.index)
    for col in numerical_cols:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        df[col] = df[col].clip(Q1 - 1.5 * IQR, Q3 + 1.5 * IQR)
    for col in ['total_rooms', 'population', 'median_house_value']:
        df[col] = np.log1p(df[col])
    return df

df = preprocess_data(df)

# Feature Engineering
def engineer_features(df):
    df['rooms_per_household'] = df['total_rooms'] / df['households']
    df['bedrooms_per_room'] = df['total_bedrooms'] / df['total_rooms']
    df['population_per_household'] = df['population'] / df['households']
    return df

df = engineer_features(df)
```

```
# EDA (example visualization)
px.scatter(df, x='median_income', y='median_house_value', title='Median Income vs Median
House Value (USD)').show()

# Prepare data
X = df.drop('median_house_value', axis=1)
y = df['median_house_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model building
numerical_cols = X_train.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = ['ocean_proximity']
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_cols),
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_cols)
])
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(n_estimators=50, max_depth=10, random_state=42,
n_jobs=-1))
])
pipeline.fit(X_train, y_train)

# Evaluation
y_pred = pipeline.predict(X_test)
y_test_usd = np.expm1(y_test)
y_pred_usd = np.expm1(y_pred)
print(f"MAE: ${mean_absolute_error(y_test_usd, y_pred_usd):,.2f}")
print(f"R²: {r2_score(y_test_usd, y_pred_usd):.4f}")

# Gradio Interface
def predict_house_value(longitude, latitude, housing_median_age, total_rooms, total_bedrooms,
                        population, households, median_income, ocean_proximity):
```

```
    input_data = pd.DataFrame({
        'longitude': [longitude], 'latitude': [latitude], 'housing_median_age': [housing_median_age],
        'total_rooms': [np.log1p(total_rooms)], 'total_bedrooms': [total_bedrooms],
        'population': [np.log1p(population)], 'households': [households],
        'median_income': [median_income], 'ocean_proximity': [ocean_proximity],
        'rooms_per_household': [np.log1p(total_rooms) / households],
        'bedrooms_per_room': [total_bedrooms / np.log1p(total_rooms)],
        'population_per_household': [np.log1p(population) / households]
    })
    prediction = pipeline.predict(input_data)
    return f"Predicted House Value: ${np.expm1(prediction[0]):,.2f} USD"


iface = gr.Interface(
    fn=predict_house_value,
    inputs=[
        gr.Slider(-124, -114, step=0.1, label="Longitude"),
        gr.Slider(32, 42, step=0.1, label="Latitude"),
        gr.Slider(0, 52, step=1, label="Housing Median Age"),
        gr.Slider(0, 40000, step=100, label="Total Rooms"),
        gr.Slider(0, 7000, step=10, label="Total Bedrooms"),
        gr.Slider(0, 50000, step=100, label="Population"),
        gr.Slider(0, 7000, step=10, label="Households"),
        gr.Slider(0, 15, step=0.1, label="Median Income"),
        gr.Dropdown(choices=['<1H OCEAN', 'INLAND', 'NEAR OCEAN', 'NEAR BAY', 'ISLAND'], label="Ocean Proximity")
    ],
    outputs="text",
    title="California House Price Predictor"
)
# iface.launch()  # Uncomment in Hugging Face Space
```

14. Future Scope

- Larger Datasets: Incorporate additional housing datasets (e.g., recent market data) for improved generalizability.

- Advanced Models: Implement XGBoost or Neural Networks for potentially higher accuracy.

- Explainable AI: Integrate SHAP or LIME to enhance model transparency for stakeholders.

- Real-Time Integration: Collaborate with real estate platforms to deploy the model as a live pricing tool.

- Feature Expansion: Add features like crime rates, school quality, or transit access to improve predictions.

15. Team Members and Roles

- Member 1: Muhammad Ajmal M – Data preprocessing, feature engineering

- Member 2: Neega P – Model building, evaluation

- Member 3: Nivetha G A – EDA, visualization

- Member 4: Muthumathi M – Gradio deployment, documentation

Note: All project files, including source code (`housing_analysis_super_optimized.py`), `requirements.txt`, and flowchart, are submitted to the Github repository: Click Here.