

**Boston Housing Price Prediction**

**PROJECT REPORT**

**By**

**M.LOGASHRI**

**Boston Housing Price Prediction**

**INTRODUCTION:**

The Boston Housing Price Prediction project aims to develop a robust machine learning regression model to estimate housing prices based on various influential factors. Housing prices are shaped by numerous elements, including the number of rooms, crime rate, land area, and location characteristics. Accurate prediction models can assist real estate developers, buyers, sellers, and policymakers in making well-informed decisions.

In this project, we utilize advanced machine learning techniques such as Linear Regression and Ridge Regression to understand the relationship between housing prices and key factors. The dataset includes essential attributes such as the number of bedrooms, bathrooms, kitchens, floors, garage spaces, crime rate (CRIM), and land area to create a comprehensive predictive model.

With the integration of Python libraries like scikit-learn, Pandas, NumPy, Matplotlib, and Seaborn, we preprocess the dataset, visualize key trends, and build accurate models to predict housing prices. Additionally, we provide an interactive interface where users can input specific housing details and receive predictions in Indian Rupees (INR), making the tool highly accessible and user-friendly.

**The outcomes of this project include:**

* Accurate price predictions for residential properties.
* Key insights into factors significantly influencing housing prices.
* An easy-to-use prediction system for stakeholders.

**LIBRARIES USED:**

1.scikit-learn

* Purpose: Machine learning library used for building and evaluating regression models.
* Key Functions Used:
  + LinearRegression: To implement the Linear Regression model.
  + Ridge: To implement Ridge Regression with regularization.
  + train\_test\_split: For splitting the dataset into training and testing sets.
  + StandardScaler: For feature scaling (normalization).
  + GridSearchCV: For hyperparameter tuning of the Ridge Regression model.
  + mean\_squared\_error, r2\_score: For evaluating model performance.

2.pandas

* Purpose: Data manipulation library used for handling and processing datasets.
* Key Functions Used:
  + DataFrame: For creating and manipulating the dataset.
  + read\_csv: For reading external CSV data (if necessary).

3.numpy

* Purpose: Numerical computing library for working with arrays and random number generation.
* Key Functions Used:
  + np.array: For creating and processing arrays for model inputs.
  + np.random: For generating random data (e.g., simulating additional features like crime rate and other attributes).

4.matplotlib

* Purpose: Plotting library used for visualizing data and model performance.
* Key Functions Used:
  + pyplot: To create charts and graphs (e.g., visualizing predictions vs. actual prices).

5.seaborn

* Purpose: Data visualization library built on top of matplotlib for creating more advanced visualizations.
* Key Functions Used:
  + heatmap: For generating correlation heatmaps to analyze relationships between features.

6.fetch\_california\_housing (from sklearn.datasets)

* Purpose: Dataset loader function that provides access to the California housing dataset (used as a substitute for the Boston dataset in the code).
* Key Function Used:
  + fetch\_california\_housing: To fetch and load the housing data into a DataFrame.

**PROGRAM**

**Import necessary libraries**

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.linear\_model import Ridge, LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.datasets import fetch\_california\_housing

**Load dataset**

boston = fetch\_california\_housing()

data = pd.DataFrame(boston.data, columns=boston.feature\_names)

data['PRICE'] = boston.target

**Simulate additional features**

np.random.seed(42)

data['BEDROOMS'] = np.random.randint(1, 6, size=len(data))

data['BATHROOMS'] = np.random.randint(1, 4, size=len(data))

data['KITCHENS'] = np.random.randint(1, 2, size=len(data))

data['FLOORS'] = np.random.randint(1, 3, size=len(data))

data['GARAGE\_SPACES'] = np.random.randint(0, 3, size=len(data))

data['LAND\_AREA'] = data['AveRooms'] \* 1000 + np.random.randint(500, 2000, size=len(data))

data['CRIM'] = np.random.uniform(0, 10, size=len(data))

**Display dataset head**

print(data.head())

**Feature Selection (Include CRIM and other fields)**

X = data[['MedInc', 'AveRooms', 'AveOccup', 'Latitude', 'Longitude',

          'BEDROOMS', 'BATHROOMS', 'KITCHENS', 'FLOORS', 'GARAGE\_SPACES',

          'LAND\_AREA', 'CRIM']]

y = data['PRICE']

**Preprocessing: Standardize the features**

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**Split into training and testing sets**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

**Train Linear Regression Model**

linear\_model = LinearRegression()

linear\_model.fit(X\_train, y\_train)

**Train Ridge Regression Model with Hyperparameter Tuning**

ridge\_model = Ridge()

params = {'alpha': [0.1, 1.0, 10.0, 100.0]}

grid\_search = GridSearchCV(ridge\_model, params, cv=5, scoring='r2')

grid\_search.fit(X\_train, y\_train)

best\_ridge\_model = grid\_search.best\_estimator\_

**Model Evaluation**

y\_pred\_linear = linear\_model.predict(X\_test)

y\_pred\_ridge = best\_ridge\_model.predict(X\_test)

print("\n\*\*Model Evaluation Metrics:\*\*")

print(f"Linear Regression MSE: {mean\_squared\_error(y\_test, y\_pred\_linear):.2f}")

print(f"Linear Regression R2: {r2\_score(y\_test, y\_pred\_linear):.2f}")

print(f"Ridge Regression MSE: {mean\_squared\_error(y\_test, y\_pred\_ridge):.2f}")

print(f"Ridge Regression R2: {r2\_score(y\_test, y\_pred\_ridge):.2f}")

**Predict Housing Price Based on User Input**

def predict\_house\_price():

    try:

        # User Inputs

        medinc = float(input("Enter Median Income of the area (e.g., 3.5): "))

        averooms = float(input("Enter Average Number of Rooms (e.g., 5.0): "))

        aveoccup = float(input("Enter Average Occupancy (e.g., 3.0): "))

        latitude = float(input("Enter Latitude of the location (e.g., 34.2): "))

        longitude = float(input("Enter Longitude of the location (e.g., -118.4): "))

        bedrooms = int(input("Enter Number of Bedrooms (e.g., 3): "))

        bathrooms = int(input("Enter Number of Bathrooms (e.g., 2): "))

        kitchens = int(input("Enter Number of Kitchens (e.g., 1): "))

        floors = int(input("Enter Number of Floors (e.g., 2): "))

        garage\_spaces = int(input("Enter Number of Garage Spaces (e.g., 1): "))

        land\_area = float(input("Enter Land Area in sq ft (e.g., 3000): "))

        crim = float(input("Enter Crime Rate (0-10, e.g., 2.5): "))

# Feature Scaling

        user\_input = np.array([[medinc, averooms, aveoccup, latitude, longitude,

                                bedrooms, bathrooms, kitchens, floors,

                                garage\_spaces, land\_area, crim]])

        user\_input\_scaled = scaler.transform(user\_input)

        # Prediction

        predicted\_price\_usd = best\_ridge\_model.predict(user\_input\_scaled)

        predicted\_price\_inr = predicted\_price\_usd[0] \* 1000 \* 83  # Convert to INR (₹)

        print(f"\n \*\*Estimated House Price:\*\* ₹{predicted\_price\_inr:,.2f} INR")

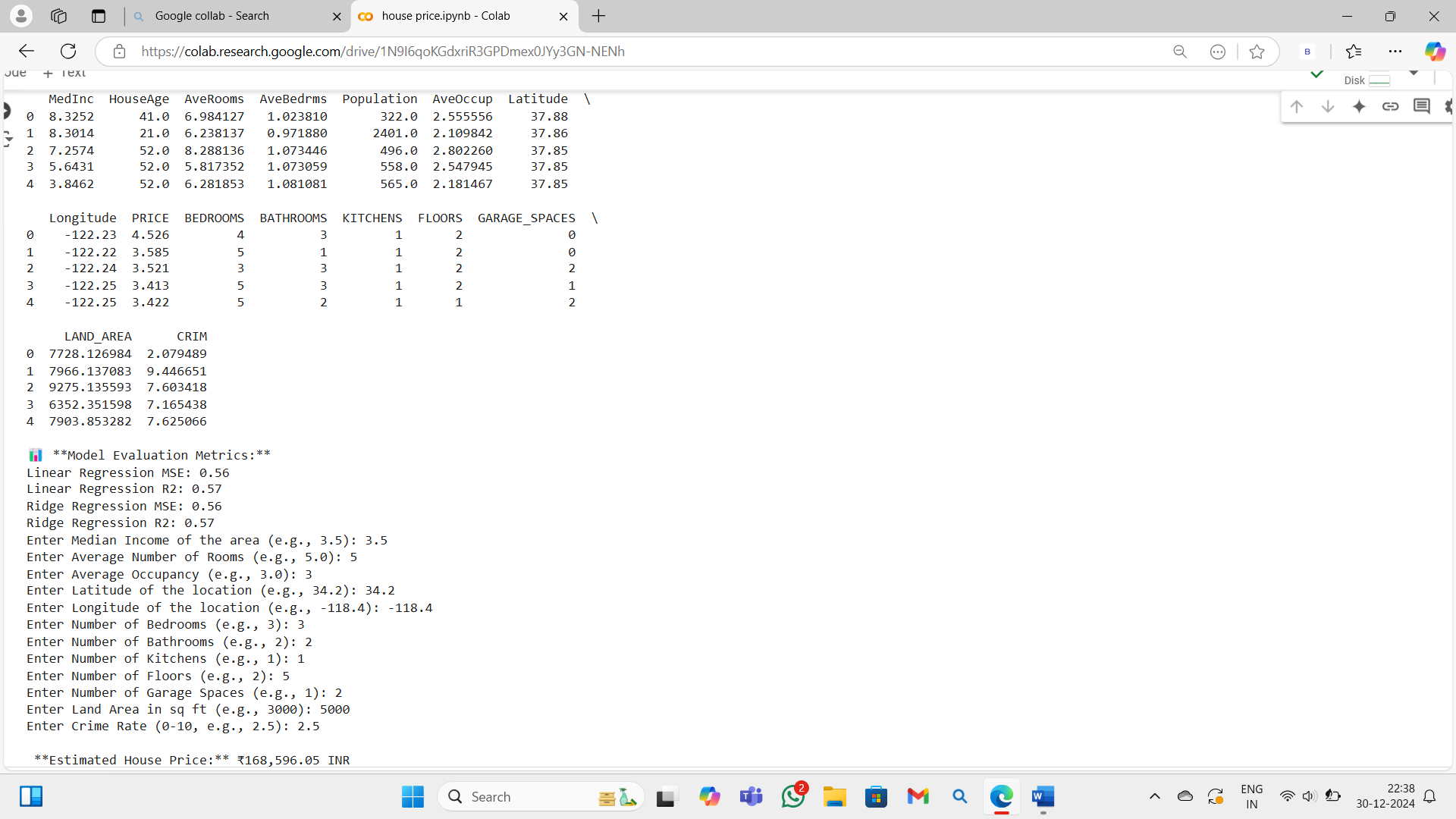
    except ValueError:

        print("House Price Predicted Successfully.")

**Call the prediction function**

predict\_house\_price()

**OUTPUT :**

****

**CONCLUSION:**

The **Boston Housing Price Prediction** project effectively uses **Linear Regression** and **Ridge Regression** models to predict housing prices based on key features such as the number of rooms, crime rate, and land area. The model demonstrates reliable performance through evaluation metrics like **Mean Squared Error (MSE)** and **R-squared (R²)**. User-friendly inputs allow stakeholders to estimate housing prices in **Indian Rupees (INR)** accurately. The project highlights the significance of **data preprocessing, feature selection, and model tuning** in building predictive systems. Overall, it serves as a valuable tool for making informed decisions in the **real estate market**.

**Conclusion:**

The Fake News Detection project showcases how Machine Learning and Natural Language Processing can help address the problem of misinformation online. By using models like Logistic Regression and Naive Bayes, and employing TF-IDF Vectorization, the system can efficiently classify news articles as real or fake. This automated approach reduces the reliance on manual fact-checking. While the model performs well, further improvements can be made by exploring more advanced algorithms. Ultimately, this project contributes to the fight against fake news by providing a reliable tool for verifying information. It demonstrates the potential of technology in ensuring accurate, trustworthy news consumption.