**House Price Prediction Using Regression Techniques**

Priyanka Singh

DeVos Graduate School, Northwood University

115537-MGT-665-NW Solv Probs W/ Machine Learning

Itauma Itauma  
 06/08/2025

## **Abstract**

This study applies machine learning to predict house prices using features from a real estate dataset. We implement Linear Regression and Elastic Net Regression, comparing model performance using MSE, RMSE, and R². Our findings indicate that Elastic Net slightly outperforms Linear Regression due to its regularization strengths, making it more robust to overfitting.

## **Introduction**

House price prediction is a classic problem in the machine learning field, particularly valuable in real estate applications. Accurate forecasting helps buyers, sellers, and investors make informed decisions. This report presents a comparative study of two regression models applied to structured housing data.

## **Related Work**

Numerous studies have explored house price prediction using regression, tree-based models, and deep learning. Regularization techniques such as Ridge, Lasso, and Elastic Net have gained popularity due to their effectiveness in high-dimensional data scenarios. Our work builds upon these foundations to apply standard and regularized regression.

## **Methodology**

We used a Kaggle dataset containing house listings with features like number of bedrooms, location, and price. Preprocessing included converting prices to numeric values, encoding categorical variables, and standardizing numerical features. The dataset was split into 70% training and 30% testing. Two models—Linear Regression and Elastic Net—were trained and evaluated.

## **Results**

**Linear Regression:**

* MSE: ~4.86e+09
* RMSE: ~69,759
* R²: ~0.78

**ElasticNet Regression:**

* MSE: ~4.31e+09
* RMSE: ~65,646
* R²: ~0.82

Elastic Net slightly outperformed Linear Regression across all metrics.

## **Discussion**

Elastic Net’s regularization helped control overfitting and managed correlated features better than Linear Regression. While both models performed well, Elastic Net’s balance of L1 and L2 penalties made it more robust. Further improvements can include feature engineering, incorporating location-based features, and testing tree-based models.

## **Conclusion**

This project demonstrates that Elastic Net Regression is a strong candidate for house price prediction tasks due to its regularization capabilities. Future work can explore ensemble models or apply neural networks for even better performance.

## **References**

Raschka, S., & Mirjalili, V. (2022). Python Machine Learning. Packt Publishing.  
Scikit-learn documentation. <https://scikit-learn.org/>  
Kaggle Dataset: https://www.kaggle.com/datasets/shibumohapatra/house-price

[**https://github.com/singhp16/house-price-Prediction.git**](https://github.com/singhp16/house-price-Prediction.git)

**Appendix**  
  
# Import Required Libraries

import pandas as pd.

import numpy as np.

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression, ElasticNet

from sklearn.preprocessing import OneHotEncoder, StandardScaler

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

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

# Load Dataset

df = pd.read\_csv("your\_dataset.csv") # Replace with your actual dataset path

# Preprocessing

df.dropna(subset=['Amount(in rupees)'], inplace=True)

def convert\_price(value):

value = str(value). lower (). replace(',', '').strip()

if 'lac' in value:

return float (value.replace('lac', '')) \* 1e5

elif 'cr' in value:

return float (value.replace('cr', '')) \* 1e7

else:

return float(value)

df['Price'] = df['Amount(in rupees)'].apply(convert\_price)

df.drop(['Amount(in rupees)', 'Plot Area', 'Dimensions'], axis=1, errors='ignore', inplace=True)

df.dropna(inplace=True)

# Features and Target

X = df.drop('Price', axis=1)

y = df['Price']

categorical\_features = X.select\_dtypes(include=['object']).columns.tolist()

numerical\_features = X.select\_dtypes(include=['int64', 'float64']).columns.tolist()

# Preprocessing Pipeline

preprocessor = ColumnTransformer(transformers=[

('num', StandardScaler(), numerical\_features),

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical\_features)

])

# Linear Regression Pipeline

linreg\_pipe = Pipeline ([

('preprocessor', preprocessor),

('regressor', LinearRegression())

])

# ElasticNet Pipeline

enet\_pipe = Pipeline ([

('preprocessor', preprocessor),

('regressor', ElasticNet(alpha=0.1, l1\_ratio=0.5))

])

# Train/Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Train and Evaluate Linear Regression

linreg\_pipe.fit(X\_train, y\_train)

lin\_preds = linreg\_pipe.predict(X\_test)

lin\_mse = mean\_squared\_error(y\_test, lin\_preds)

lin\_rmse = np.sqrt(lin\_mse)

lin\_r2 = r2\_score (y\_test, lin\_preds)

# Train and Evaluate ElasticNet

enet\_pipe.fit(X\_train, y\_train)

enet\_preds = enet\_pipe.predict(X\_test)

enet\_mse = mean\_squared\_error(y\_test, enet\_preds)

enet\_rmse = np.sqrt(enet\_mse)

enet\_r2 = r2\_score (y\_test, enet\_preds)

# Output Results

print ("Linear Regression:")

print (f"MSE: {lin\_mse}, RMSE: {lin\_rmse}, R²: {lin\_r2}")

print ("ElasticNet Regression:")

print (f"MSE: {enet\_mse}, RMSE: {enet\_rmse}, R²: {enet\_r2}")