Title: Simple House Price Prediction Model – Reflection (2025)

Section 1: Project Overview

This was a basic project to help me understand how machine learning models are trained. I created a simple house price prediction model using data with 100 entries. The dataset included features like area (in square feet), number of bedrooms, and location. The target variable was price (in lakhs).

I found a basic dataset online and manually copied 100 entries into my working file. The main goal of this project was to learn the steps of preprocessing and training a regression model.

Section 2: Preprocessing Steps

2.1 Encoding the Location

Since the "Location" column was in text format (categorical), I converted it into numeric form using encoding so that the model could process it.

2.2 Feature Scaling (StandardScaler)

To make the model training stable and accurate, I used **StandardScaler** from sklearn.preprocessing. Feature scaling normalizes the data so that each feature contributes equally during training.

Why Scaling Matters:

- Our model uses **SGDRegressor**, which is based on gradient descent.
- Gradient descent can behave unstably when there are large differences in feature values.
- For example, if "Area" is 1200 and "Bedrooms" is 3, the model might overemphasize "Area" just because it's a larger number.
- StandardScaler scales all features so they have a **mean of 0** and **standard deviation of 1**. This keeps the model training balanced and stable.

Section 3: Model and Parameters

- **Algorithm Used:** SGDRegressor (Stochastic Gradient Descent for Linear Regression)
- **Epochs (Iterations):** 5000
- **Learning Rate:** Adaptive (starting from 0.01)
- **Loss Function:** Squared Loss (Mean Squared Error)
- **Scaling:** StandardScaler used before training

- **Train-Test Split:** Not used in this version

Section 4: Learnings and Reflection

This was a beginner-level project, but it helped me understand some important parts of building ML models:

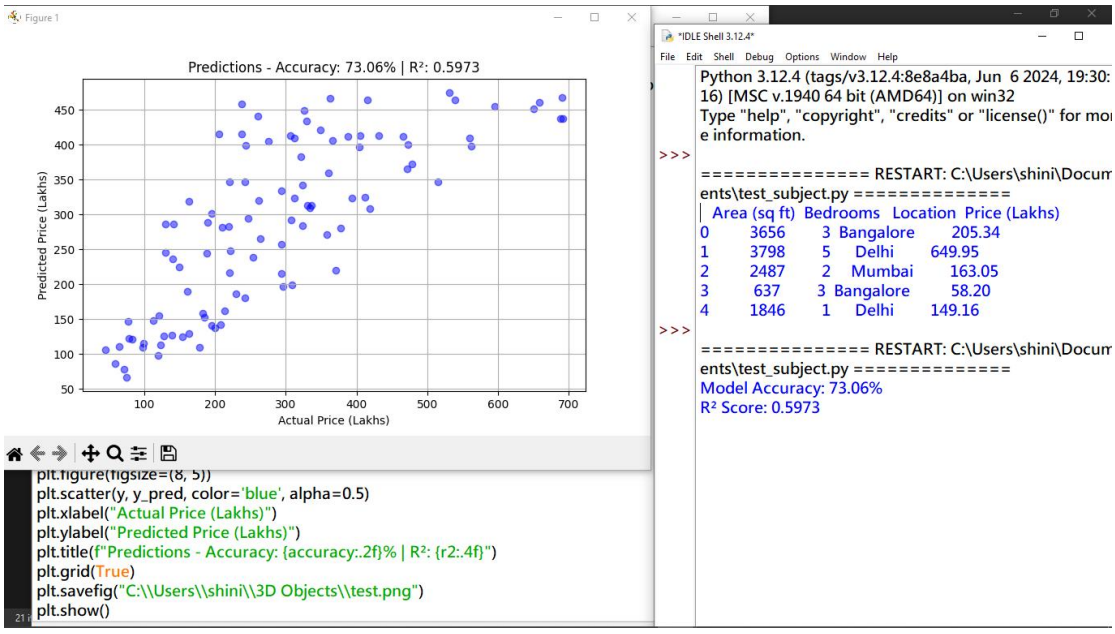
- How to preprocess data (encoding, scaling)
- Why scaling is necessary for stable learning
- Basics of using SGD for linear regression

Even though I didn't split the data for testing, it was useful to just see how models learn from patterns. This project gave me hands-on experience with preprocessing and model fitting.

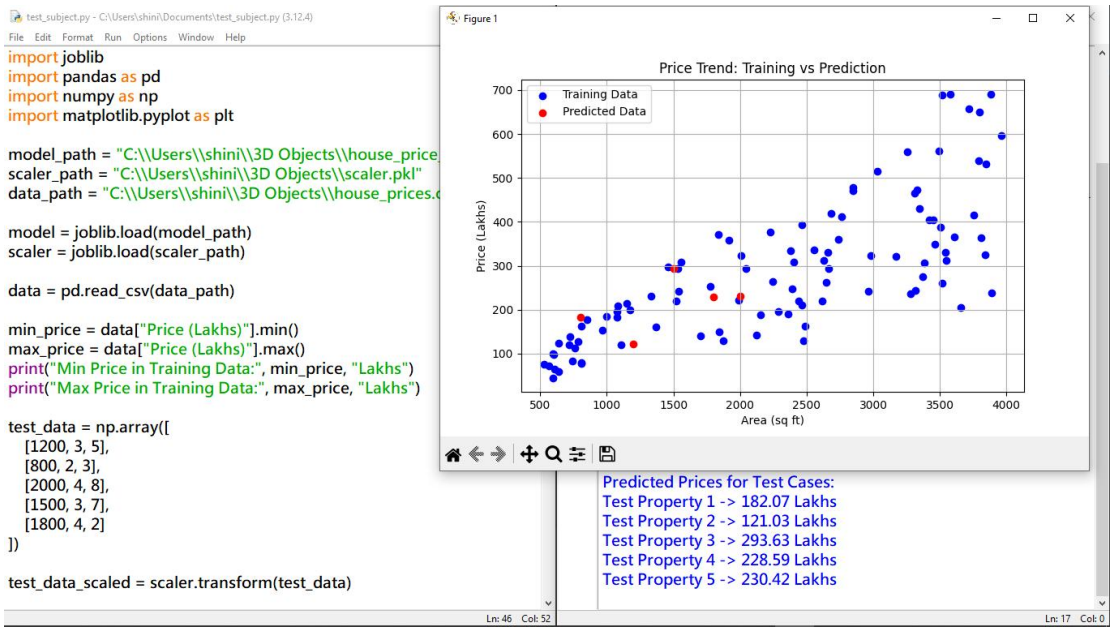
Attachments I May Add Later

- Sample dataset
- Training code snippets
- Output predictions vs. actual prices
- Graph of loss vs. iterations

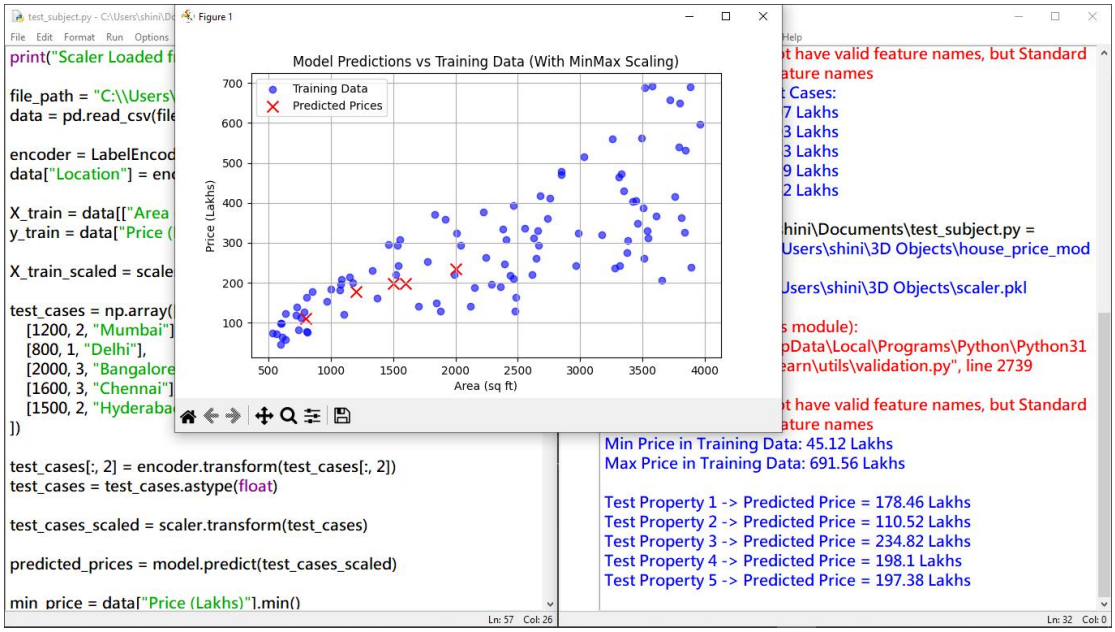
Python code: https://drive.google.com/file/d/1SC0DT_xJb5syHI-OAumO5HFUNqpw9iiC/view?usp=sharing
CSV File: https://drive.google.com/file/d/1DBPVDpXBxVluKDesm2JCeA_HW1HcHttP/view?usp=sharing
Output:



Test1:



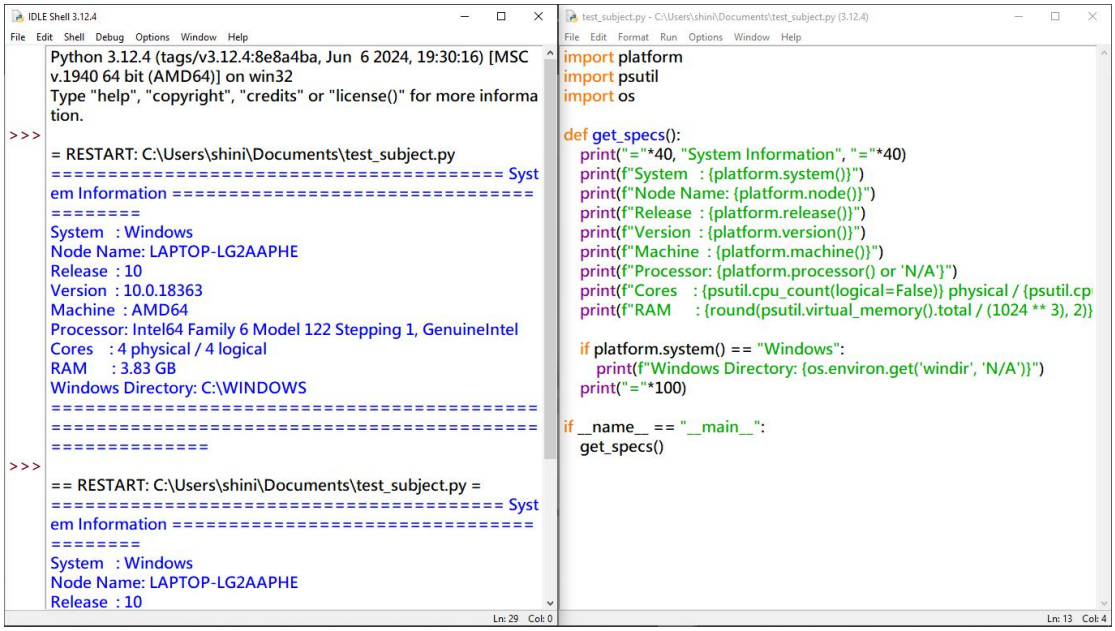
Test2:



Models: https://drive.google.com/drive/folders/1jDXh49vpQ35cpXr4eRH1fabzCF9-UQ-s?usp=sharing

Conclusion

Though simple, this project was an important step for me. It taught me foundational machine learning concepts and gave me the confidence to move toward more complex models. This experience will help in future projects where I aim to build smarter and more robust AI systems.


IDLE Shell 3.12.4
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\shini\Documents\test_subject.py
========== System Information ==========
System : Windows
Node Name: LAPTOP-LG2AAPHE
Release : 10
Version : 10.0.18363
Machine : AMD64
Processor: Intel64 Family 6 Model 122 Stepping 1, GenuineIntel
Cores : 4 physical / 4 logical
RAM : 3.83 GB
Windows Directory: C:\WINDOWS
>>>
== RESTART: C:\Users\shini\Documents\test_subject.py =
========== System Information ==========
System : Windows
Node Name: LAPTOP-LG2AAPHE
Release : 10
Ln: 29 Col: 0
test_subject.py - C:\Users\shini\Documents\test_subject.py (3.12.4)
File Edit Format Run Options Window Help
import platform
import psutil
import os
def get_specs():
print("="*40, "System Information", "="*40)
print(f"System : {platform.system()}")
print(f"Node Name: {platform.node()}")
print(f"Release : {platform.release()}")
print(f"Version : {platform.version()}")
print(f"Machine : {platform.machine()}")
print(f"Processor: {platform.processor() or 'N/A'}")
print(f"Cores : {psutil.cpu_count(logical=False)} physical / {psutil.cp
print(f"RAM : {round(psutil.virtual_memory().total / (1024 ** 3), 2)}
if platform.system() == "Windows":
print(f"Windows Directory: {os.environ.get('windir', 'N/A')}")
print("="*100)
if __name__ == "__main__":
get_specs()
Ln: 13 Col: 4