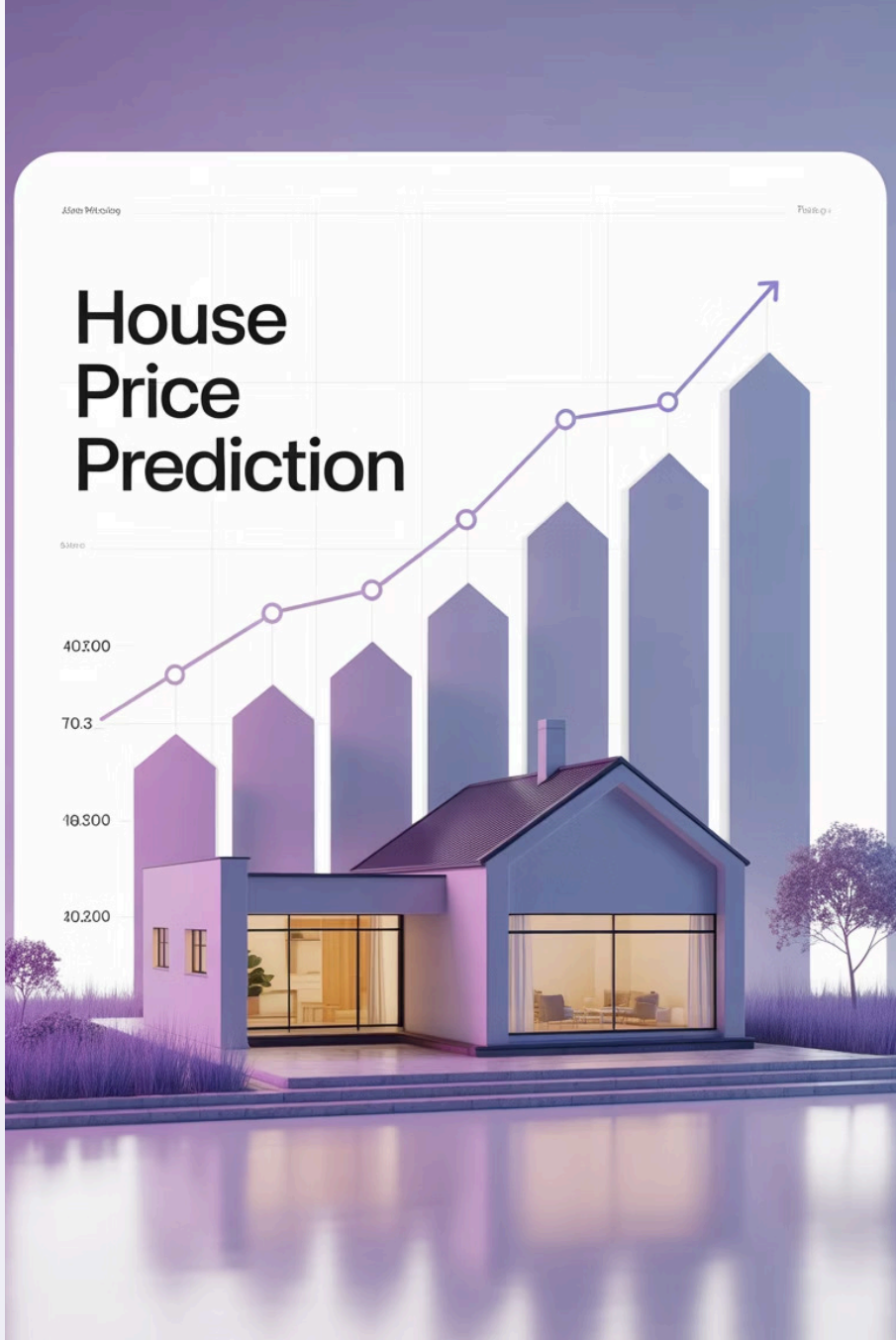
# Mini Project: Predict House Prices

by Yonca Kurt

## 📘 Mini Project Code Steps

- [ ] 📂 Load the Dataset
- [ ] 🧹 Check and Clean Missing Values
- [ ] 📊 Select Features (Independent Variables)
- [ ] 🧠 Build and Train Your Linear Regression Model
- [ ] 🔮 Make Predictions
- [ ] 📈 Visualize Predictions vs Actual Values
- [ ] 📌 Predict the Price of a New House



## ✅ Step 1: Load the Dataset

Start by loading the housing dataset using Pandas. This dataset contains real features like area size, number of bedrooms, and price.

```
import pandas as pd

df = pd.read_csv('/content/Real estate.csv') # adjust the path if needed
df.head()
```

**Download House Prices Dataset (CSV)**

## ✅ Step 1: Load the Dataset (from Google Drive)

Start by loading the housing dataset using Pandas. This dataset contains real features like area size, number of bedrooms, and price.

```
# Connect Google Drive to access files
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

# Set the path to the CSV file
file_path = '/content/drive/My Drive/ai-camp/Real estate.csv'

# Readt the CSV file
df = pd.read_csv(file_path)

# Display the first 5 rows of the dataset
df.head()
```

**Download House Prices Dataset (CSV)**

## 🧹 Step 2: Explore & Clean the Data

Real-world data is often messy. We'll check for missing values or outliers and clean the dataset to ensure accurate predictions.

Check for missing values, column types, and rename the columns if needed.

```
# Check the shape of the dataset (rows, columns)
print("Shape of the dataset:", df.shape)

df.info() # Gives Structural Information About the Dataset
df.describe() # Gives Statistical Summary
df.isnull().sum()
```

You can also rename columns for easier reference:

```
df = df.rename(columns={
    'X1 transaction date': 'transaction_date',
    'X2 house age': 'house_age',
    'X3 distance to the nearest MRT station': 'mrt_distance',
    'X4 number of convenience stores': 'convenience_stores',
    'X5 latitude': 'latitude',
    'X6 longitude': 'longitude',
    'Y house price of unit area': 'price'
})
```

## 📊 Step 3: Select Features (Independent Variables)

Decide which features (X) you want to use to predict the target variable (y = price).

```
X = df[['house_age', 'mrt_distance', 'convenience_stores']]
y = df['price']
```

## ✅ Step 4: Split the Data

Split into training and testing sets.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## 🧠 Step 5: Build and Train Your Model

Use Linear Regression to train your model.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)

print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

## 🔮 Step 6: Make Predictions

Use the trained model to predict the target values for test data

```
# Make Predictions

y_pred = model.predict(X_test)
```

## 🔮 Step 7: Evaluate the Model

Now it's time to test your model! Compare predictions with actual values using common regression metrics

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R2 Score:", r2)
```

## ✅ Step 7: Evaluate the Model

**Let's understand what our evaluation metrics actually tell us:**

### 📊 Mean Squared Error (MSE)

Measures the average of squared differences between predicted and actual values.
→ In our case: **58.88** — lower is better.

### 📉 Mean Absolute Error (MAE)

Shows the average prediction error in actual units (price per square meter).
→ Our model is off by **5.63** units on average.

### 📈 R² Score

Explains how much of the variability in the data is captured by the model.
→ Score: **0.649** → Our model explains **~65%** of the variation in house prices.

### ⓘ Conclusion

This is a strong first model!
✅ It predicts reasonably well.
✅ Results make sense.
✅ There's room for improvement—but we've built a working ML model!

## 📈 Step 8: Visualize Results

Create plots to compare predicted vs actual prices. Visualization helps you understand how well your model is performing.

```
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted House Prices')
plt.grid(True)
plt.show()
```

## 📌 Step 9: *Try New Data!* Predict a New House Price

Enter the features of a new house (e.g., 1800 sq ft, 3 bedrooms) and let your model predict the price.
This is your chance to test the model with data that wasn't in the original dataset.
See if the result feels realistic, and reflect on how well the model generalizes!

```
new_house = pd.DataFrame({
    'house_age': [15],
    'mrt_distance': [200],
    'convenience_stores': [4]
})
predicted_price = model.predict(new_house)
print(f"Predicted Price for the new house: ${predicted_price[0]:.2f}")
```

```
new_house = pd.DataFrame({
    'house_age': [10],
    'mrt_distance': [300],
    'convenience_stores': [5]
})
predicted_price = model.predict(new_house)
print(f"Predicted Price for the new house: ${predicted_price[0]:.2f}")
```

# 🎯 End of Day 3 – What You Learned Today

✅ What is **Machine Learning** and how it works
✅ The logic behind **Linear Regression**
✅ How to **prepare and clean a dataset**
✅ The difference between **.info()** and **.describe()**
✅ How to **select features** and **split** data
✅ How to **train a model** and **make predictions**
✅ How to **evaluate your model** (MSE, MAE, $R^2$)
✅ How to **visualize results** clearly
✅ How to use your model to **predict new data**

🎉 You've built your **first Machine Learning project** from start to finish.
You cleaned data, trained a model, and predicted house prices like a real data scientist!