Bayesian Linear Regression for Real Estate Price Prediction

Zijie Wang

1 Introduction

Housing prices are a topic of widespread interest as they reflect economic trends, influence investment decisions, and impact quality of life. Accurate prediction of housing prices, particularly the unit price per square meter, enables market analysis and property appraisal. With the increasing availability of real estate data, statistical models offer powerful tools to analyze and forecast these prices. In this project, we leverage a Bayesian linear regression model to predict residential property prices in Taipei, Taiwan, harnessing its ability to provide uncertainty-aware predictions that account for unobserved influences common in real estate valuation.

Bayesian regression has emerged as a robust approach to predictive modeling, evolving from classical linear regression by incorporating prior knowledge and quantifying uncertainty through posterior distributions. Unlike traditional methods that yield point estimates, Bayesian models produce a full probability distribution over possible outcomes. This distribution captures not only the most likely prediction but also the uncertainty associated with it, reflecting the range of plausible values and their relative likelihoods, making them particularly suited for complex domains such as real estate, where factors such as neighborhood dynamics or building-specific attributes may not be fully captured in the data.

We used the Real Estate Valuation dataset from the UCI Machine Learning Repository, which contains 414 transaction records for residential properties in the Taipei metropolitan area. The data set includes six predictors: house age, distance to the nearest Mass Rapid Transit (MRT) station, number of nearby convenience stores, transaction date, and geographical coordinates (latitude and longitude). The target variable is the unit price of the house, measured in New Taiwan Dollars (NTD) per square meter. This data set is well suited for Bayesian linear regression, as the response variable is continuous, the features are numerical, and the residuals are approximately Gaussian.

The primary objectives of this project are as follows. First, we predict the unit price of residential properties in Taipei, Taiwan, using a Bayesian linear regression model. Second, quantify the uncertainty in these predictions by providing credible intervals, which reflect the range of plausible price values. Third, evaluate the performance of the model through visualizations and predictive checks post-reference, ensuring that it captures both the underlying relationships and the variability of the data.By addressing the question of how well a Bayesian framework can model real estate prices while accounting for unobserved factors, we aim to provide a tool for informed decision-making in property valuation and market analysis.

2 Data Description

This project uses the real estate valuation data set from the UCI Machine Learning Repository, comprising 414 transaction records of residential properties in Taipei, Taiwan's metropolitan area. The data set includes six explanatory variables and one target variable, all numeric and complete with no missing values, ensuring robust data for analysis.

The explanatory variables are designed to capture key property characteristics: First, the transaction date is recorded in a fractional year format, providing temporal context. Second, the age of the home, measured in years since construction, reflects the condition and depreciation of the property.

Third, the distance to the nearest Mass Rapid Transit (MRT) station, measured in meters, serves as a proxy for transportation accessibility. Fourth, the number of nearby convenience stores indicates neighborhood convenience and amenities. Finally, the latitude and longitude of each property pinpoint its geographic location, capturing spatial influences on pricing.

The target variable is the house price per unit area, expressed in New Taiwan Dollars (NTD) per square meter, which enables standardized price comparisons across properties. This well-structured dataset supports the project's goal of predicting unit prices using Bayesian linear regression while accounting for the multifaceted factors influencing real estate valuation.

3 Model and Methods

We consider a Bayesian linear regression model to describe the relationship between the variables and the unit house price. Note that even with the same observable features, housing prices can vary due to unobservable factors such as neighborhood dynamics, building orientation, or floor level. These unexplained variations are modeled as Gaussian noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, where the variance σ^2 reflects the overall degree of such unobserved variability. We let y_i denote the unit price of the i-th property, and let $x_i \in \mathbb{R}^p$ be the corresponding feature vector including an intercept. The model assumes:

$$y_i = x_i^\top \boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

where $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of regression coefficients, and σ^2 is the variance of the observation noise.

Given this model, the likelihood of the observed data $(y_1, \dots, y_n)$ under parameters $\boldsymbol{\beta}$ and σ^2 is:

$$p(y \mid X, \boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}(y_i \mid x_i^\top \boldsymbol{\beta}, \sigma^2)$$

To complete the Bayesian specification, we use a conjugate prior:

$$\boldsymbol{\beta} \mid \sigma^2 \sim \mathcal{N}(\boldsymbol{\mu}_0, \sigma^2 \Lambda_0^{-1}), \quad \sigma^2 \sim \text{Inv-Gamma}(a_0, b_0),$$

where $\boldsymbol{\mu}_0 \in \mathbb{R}^p$ is the prior mean of $\boldsymbol{\beta}$, Λ_0 is the precision matrix (inverse covariance), and a_0, b_0 are shape and scale hyperparameters of the inverse-gamma prior on the noise variance. ... and a_0, b_0 are shape and scale hyperparameters of the inverse-gamma prior on the noise variance.This prior structure is conjugate to the Gaussian likelihood, which is suitable for our dataset, where both the target variable and predictors are numeric and well modeled by a linear Gaussian model.

Given n observations $\{(x_i, y_i)\}_{i=1}^n$, the posterior distribution is also conjugate. Specifically, the posterior distribution of $\boldsymbol{\beta} \mid \sigma^2, \mathcal{D}$ is Gaussian:

$$\boldsymbol{\beta} \mid \sigma^2, \mathcal{D} \sim \mathcal{N}(\boldsymbol{\mu}_n, \sigma^2 \Lambda_n^{-1}),$$

where

$$\Lambda_n = \Lambda_0 + X^\top X, \quad \boldsymbol{\mu}_n = \Lambda_n^{-1}(\Lambda_0 \boldsymbol{\mu}_0 + X^\top y),$$

and the marginal posterior for σ^2 is:

$$\sigma^2 \mid \mathcal{D} \sim \text{Inv-Gamma}(a_n, b_n),$$

with

$$a_n = a_0 + \frac{n}{2}, \quad b_n = b_0 + \frac{1}{2}(y^\top y + \boldsymbol{\mu}_0^\top \Lambda_0 \boldsymbol{\mu}_0 - \boldsymbol{\mu}_n^\top \Lambda_n \boldsymbol{\mu}_n).$$

For prediction, the posterior predictive distribution of a new observation y^* with input x^* is:

$$y^* \mid x^*, \mathcal{D} \sim t_{2a_n}\left(x^{*\top} \boldsymbol{\mu}_n, \frac{b_n}{a_n}\left(1 + x^{*\top} \Lambda_n^{-1} x^*\right)\right).$$

This formulation provides not only point estimates for new predictions but also credible intervals that reflect parameter uncertainty.

4 Implementation

The model was implemented using `Python`. The dataset was first read from an Excel file and filtered to include only the most relevant explanatory variables: house age, distance to the nearest MRT station, and number of nearby convenience stores. A column of ones was also added to the feature matrix to account for the intercept term. The resulting design matrix X had dimension $n \times p$, where $n = 414$ and $p = 4$. We also implemented an extended model including all six features to examine whether additional information improves predictive performance.

We specified a conjugate prior for the parameters:

$$\boldsymbol{\beta} \mid \sigma^2 \sim \mathcal{N}(\mathbf{0}, \sigma^2(0.01I)^{-1}), \quad \sigma^2 \sim \text{Inv-Gamma}(1.0, 1.0),$$

and calculated the posterior parameters $\boldsymbol{\mu}_n$, Λ_n, a_n, and b_n analytically using the formulas described in Section 3. With the closed-form posterior, we may directly calculate posterior mean and the predictive expectation.

However, to visualize the model's predictive uncertainty and perform posterior predictive checks, we used sampling. Specifically, we drew samples of σ^2 from its posterior Inv-Gamma(a_n, b_n), and for each sampled value, generated a $\boldsymbol{\beta}$ from $\mathcal{N}(\mu_n, \sigma^2\Lambda_n^{-1})$.

These sampled parameters were used to simulate replicated datasets and generate predictive distributions. The results were then compared with the observed data for model assessment.

5 Results and Discussion

Model performance is evaluated through three complementary visualizations.

First, we compare the posterior predictive mean for each data point with its actual observed value.

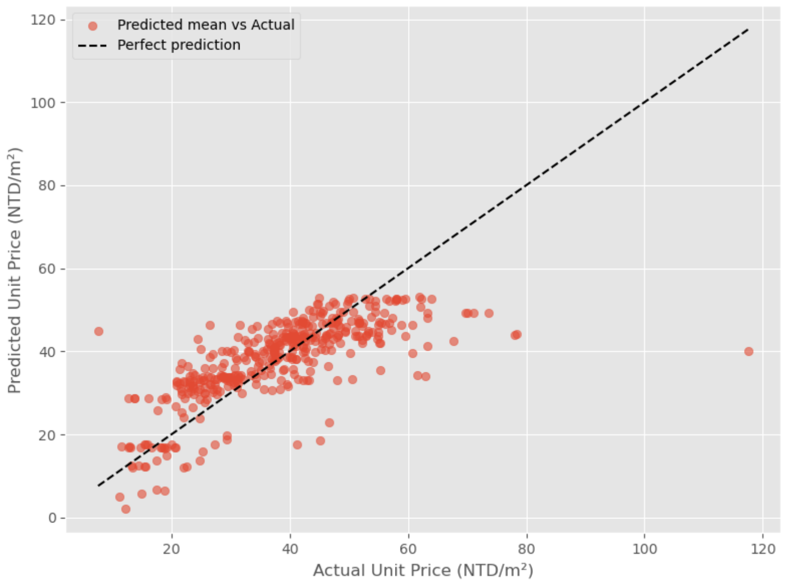

Figure 1: Predicted unit price vs actual unit price.

The predicted values are scattered around the $y = x$ diagonal, which indicates a good fit. Despite a few outliers, the model captures the underlying relationship between features and unit price.

Next, we plot the predicted mean and the 95% credible interval for each property.

We visualizes the 95% credible interval for each prediction as a vertical light blue bar, with actual unit prices shown as red dots. Most red dots fall within the corresponding interval, indicating that the model successfully captures predictive uncertainty.

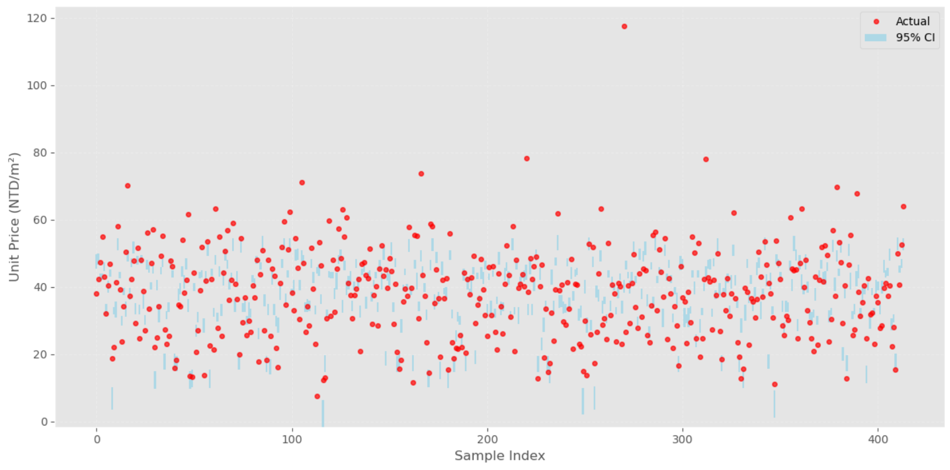

Figure 2: Bayesian predictions with 95% credible intervals.

To further examine model adequacy, we conduct posterior predictive checks (PPC) by simulating replicated datasets from the posterior predictive distribution and comparing their histograms with the observed data.

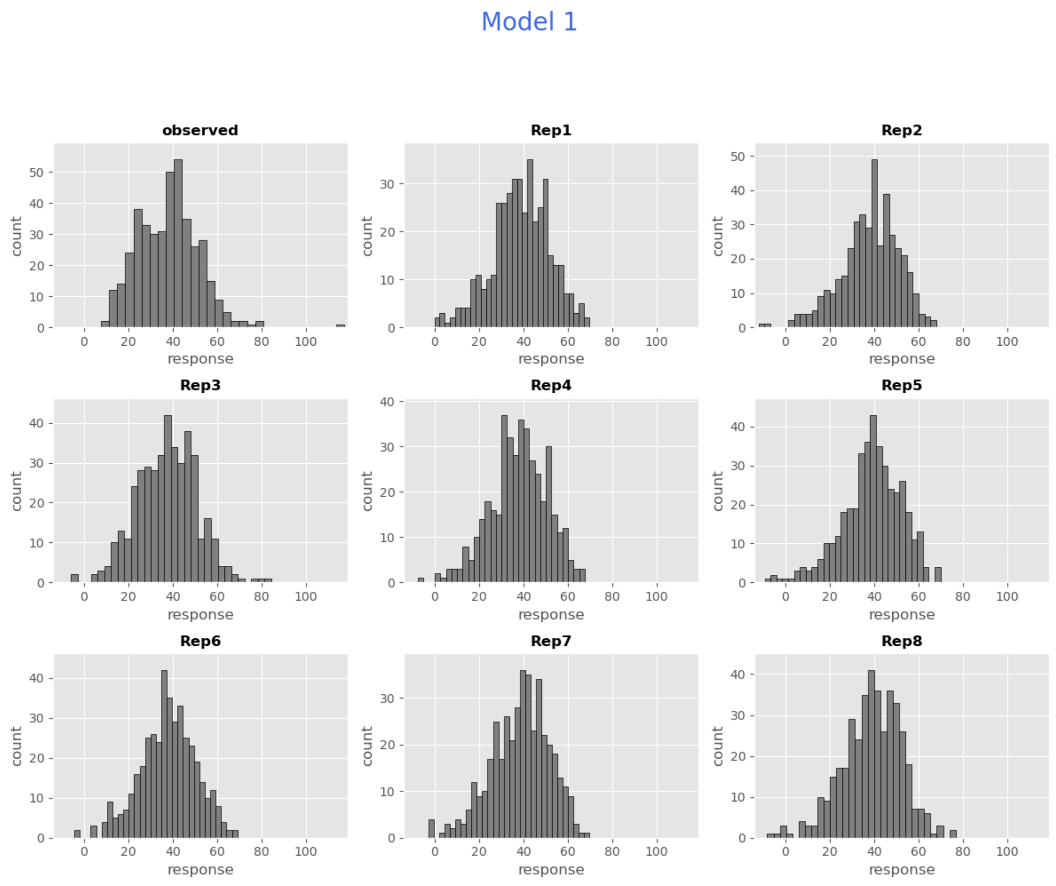

Figure 3: PPC for Model 1

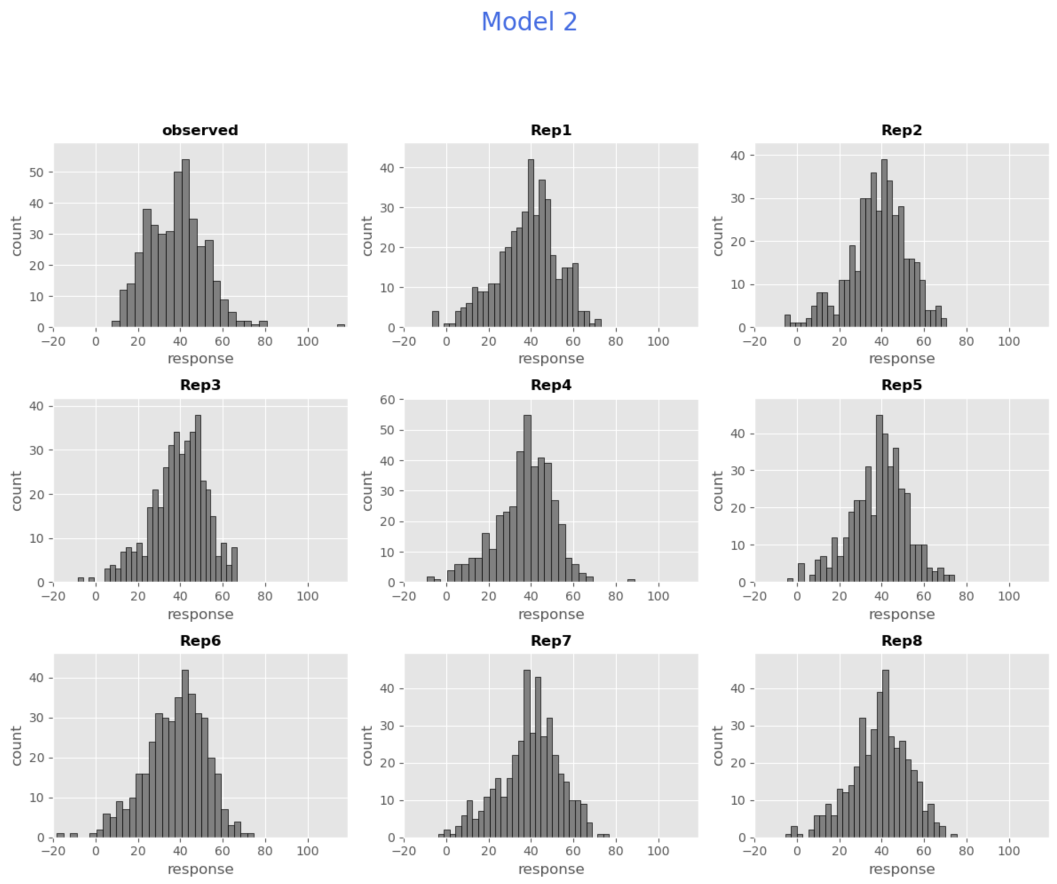

Figure 4: PPC for Model 2

Model 2 shows better alignment between the replicated and observed distributions in terms of symmetry and peak location. This suggests that incorporating additional features such as transaction date and geographic coordinates improves the model's ability to reproduce the true data distribution.

Overall, the Bayesian linear regression model provides reasonable predictive accuracy and credible interval coverage. The reduced model offers good interpretability, while the full model demonstrates improved predictive calibration through posterior predictive checks. The trade-off between simplicity and performance remains a key consideration when choosing between models.

6 Appendix

```python
import pandas as pd
import numpy as np
df = pd.read_excel("data/real estate.xlsx")

X_raw = df[['X2 house age',
            'X3 distance to the nearest MRT station',
            'X4 number of convenience stores']].values

X = np.hstack([np.ones((X_raw.shape[0], 1)), X_raw])

y = df['Y house price of unit area'].values

n, p = X.shape

mu0 = np.zeros(p)
Lambda0 = 0.01 * np.identity(p)
a0 = 1.0
b0 = 1.0

Lambda_n = Lambda0 + X.T @ X
mu_n = np.linalg.inv(Lambda_n) @ (Lambda0 @ mu0 + X.T @ y)
a_n = a0 + n / 2
b_n = b0 + 0.5 * (
    y.T @ y + mu0.T @ Lambda0 @ mu0 - mu_n.T @ Lambda_n @ mu_n
)
from scipy.stats import invgamma, multivariate_normal
import matplotlib.pyplot as plt

n_samples = 500

sigma2_samples = invgamma.rvs(a=a_n, scale=b_n, size=n_samples)

beta_samples = np.array([
    multivariate_normal.rvs(mean=mu_n, cov=np.linalg.inv(Lambda_n) *
    s2)
    for s2 in sigma2_samples
])

y_pred_samples = X @ beta_samples.T  # shape: (n, n_samples)

y_pred_mean = y_pred_samples.mean(axis=1)
y_pred_lower = np.percentile(y_pred_samples, 2.5, axis=1)
y_pred_upper = np.percentile(y_pred_samples, 97.5, axis=1)

# Figure 1
plt.figure(figsize=(8, 6))
plt.scatter(y, y_pred_mean, alpha=0.6, label='Predicted mean vs Actual
    ')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', label='Perfect
     prediction')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Listing 1: Code for Figure 1

```python
n = len(y)
x_pos = np.arange(n)

ci_height = y_pred_upper - y_pred_lower
ci_bottom = y_pred_lower
```

```python

plt.figure(figsize=(12, 6))

plt.bar(
    x=x_pos,
    height=ci_height,
    bottom=ci_bottom,
    color='lightblue',
    edgecolor='none',
    width=1.0,
    label='95% CI'
)

plt.plot(x_pos, y, 'o', color='red', alpha=0.7, markersize=4, label='
    Actual')

# Figure 2
plt.xlabel('Sample Index')
plt.legend(loc='upper right')
plt.grid(True, linestyle='--', alpha=0.3)
plt.tight_layout()
plt.show()
```

Listing 2: Code for Figure 2

```python
# FIgure 3
plt.style.use('ggplot')

n_replications = 8
n_samples = X.shape[0]
replicated_ys = [y]

for _ in range(n_replications):
    sigma2_sample = invgamma.rvs(a=a_n, scale=b_n)
    beta_sample = multivariate_normal.rvs(mean=mu_n, cov=sigma2_sample
     * np.linalg.inv(Lambda_n))
    y_rep = X @ beta_sample + np.random.normal(0, np.sqrt(
    sigma2_sample), size=n_samples)
    replicated_ys.append(y_rep)

fig, axes = plt.subplots(3, 3, figsize=(12, 10))
axes = axes.flatten()
y_all = np.concatenate(replicated_ys)
x_min = np.floor(np.min(y_all)) - 1
x_max = np.ceil(np.max(y_all)) + 1

for i in range(9):
    axes[i].hist(replicated_ys[i], bins=30, color='gray', edgecolor='
    black')

    axes[i].set_xlim(x_min, x_max)
    counts, _ = np.histogram(replicated_ys[i], bins=30)
    ymax = np.max(counts)
    axes[i].set_ylim(0, ymax * 1.1)
    axes[i].set_xlabel("response")
    axes[i].set_ylabel("count")
    axes[i].set_title('observed' if i == 0 else f'Rep{i}', fontsize
    =12, fontweight='bold')

fig.suptitle('Model 1', fontsize=20, color='royalblue', x=0.5, y=1.02)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Figure 4
X_raw = df[['X1 transaction date',
              'X2 house age',
              'X3 distance to the nearest MRT station',
              'X4 number of convenience stores',
              'X5 latitude',
              'X6 longitude']].values

X = np.hstack([np.ones((X_raw.shape[0], 1)), X_raw])
y = df['Y house price of unit area'].values
n, p = X.shape
mu0 = np.zeros(p)
Lambda0 = 0.01 * np.identity(p)
a0 = 1.0
b0 = 1.0

Lambda_n = Lambda0 + X.T @ X
mu_n = np.linalg.inv(Lambda_n) @ (Lambda0 @ mu0 + X.T @ y)
a_n = a0 + n / 2
b_n = b0 + 0.5 * (y.T @ y + mu0.T @ Lambda0 @ mu0 - mu_n.T @ Lambda_n
    @ mu_n)

n_replications = 8
replicated_ys = [y]

for _ in range(n_replications):
    sigma2_sample = invgamma.rvs(a=a_n, scale=b_n)
    beta_sample = multivariate_normal.rvs(mean=mu_n, cov=sigma2_sample
     * np.linalg.inv(Lambda_n))
    y_rep = X @ beta_sample + np.random.normal(0, np.sqrt(
    sigma2_sample), size=n)
    replicated_ys.append(y_rep)

y_all = np.concatenate(replicated_ys)
x_min = np.floor(np.min(y_all)) - 1
x_max = np.ceil(np.max(y_all)) + 1

fig, axes = plt.subplots(3, 3, figsize=(12, 10))
axes = axes.flatten()

for i in range(9):
    axes[i].hist(replicated_ys[i], bins=30, color='gray', edgecolor='
    black')
    axes[i].set_xlim(x_min, x_max)
    counts, _ = np.histogram(replicated_ys[i], bins=30)
    ymax = np.max(counts)
    axes[i].set_ylim(0, ymax * 1.1)
    axes[i].set_xlabel("response")
    axes[i].set_ylabel("count")
    axes[i].set_title('observed' if i == 0 else f'Rep{i}', fontsize
    =12, fontweight='bold')
fig.suptitle('Model 2', fontsize=20, color='royalblue', x=0.5, y=1.02)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```

Listing 3: Code for Figure 3&Figure 4