Bayesian statistics - Model Comparison

Akhil Antony

June 2025

Abstract

In this article, I compare a simple linear regression model, XGBoost model and a Bayesian linear regression model. The study analyzes the models in their vanilla settings (baseline) and also a fine-tuned version of each. For linear regression, we vary regularization parameters, for XGBoost we do hyper-parameter tuning and for the Bayesian model we use different priors for the feature parameters. I use mean squared error and R^2 score for model comparison. In the baseline analysis, we see that Bayesian model and the XGBoost model perform at similar levels, with XGBoost with lowest mean squared error (0.15) and Bayesian model has the highest R^2 score (0.841). However, upon first order improvement, XGBoost outperforms other two models with a MSE 0.12 and R^2 score 0.89.

1 Introduction

Predictive modeling is a cornerstone of modern data analysis, with applications spanning economics, healthcare, real estate, and beyond. Among the vast landscape of modeling techniques, linear regression remains a widely used baseline due to its simplicity and interpretability. However, its performance often suffers when confronted with complex, nonlinear relationships.

To address these limitations, more advanced techniques like XGBoost — a scalable, ensemble-based gradient boosting algorithm — have gained popularity for their ability to model nonlinear patterns and handle high-dimensional data. While powerful, such models can behave like black boxes, offering limited insights into uncertainty and model structure.

Bayesian modeling presents a compelling alternative by incorporating prior beliefs and providing full posterior distributions over model parameters. This probabilistic framework, implemented here via PyMC, allows not only for flexible model construction but also for robust uncertainty quantification, which is particularly valuable in high-stakes decision-making.

In this study, we compare the performance and interpretability of three modeling approaches—linear regression, XGBoost, and a Bayesian regression model implemented in PyMC—on a structured dataset. We aim to evaluate their predictive accuracy, robustness to overfitting, and ability to express uncertainty, providing practical guidance for model selection in real-world applications.

2 Theory

2.1 Linear Regression

Linear regression is a foundational statistical method used to model the relationship between a dependent variable y and one or more independent variables X. The core assumption is that this relationship is linear, meaning it can be expressed as:

$$\bar{y} = A \cdot X + B \tag{1}$$

where A contains the set of parameters and is of the same size as X. B is the intercept of the hyperplane. The residuals, which represent the difference between observed and predicted values, are given by:

$$r = y - \bar{y}. \tag{2}$$

The objective in ordinary least squares (OLS) linear regression is to minimize the sum of squared residuals, leading to the cost function:

$$\mathcal{L}_{\text{OLS}} = \sum_i \|y - \bar{y}\|_2^2. \tag{3}$$

While OLS performs well in many scenarios, it can suffer from overfitting, especially when the number of predictors is large or when multicollinearity is present. To mitigate this, regularization techniques are introduced by augmenting the loss function with a penalty term that discourages overly complex models.

Two commonly used regularization methods are **Ridge** and **Lasso**. Ridge regression adds an L_2 penalty on the coefficients, resulting in the cost function:

$$\mathcal{L}_{\text{Ridge}} = \sum_i \|y - \bar{y}\|_2^2 + \lambda \|A\|_2^2, \tag{4}$$

which encourages smaller coefficient magnitudes without enforcing sparsity. In contrast, Lasso regression uses an L_1 penalty:

$$\mathcal{L}_{\text{Lasso}} = \sum_i \|y - \bar{y}\|_2^2 + \lambda \|A\|_1, \tag{5}$$

which promotes sparsity by allowing some coefficients to shrink exactly to zero. These regularization strategies offer a principled way to improve model generalization by balancing bias and variance.

2.2 XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful ensemble learning method based on gradient boosted decision trees. It builds a strong predictive model by iteratively combining the outputs of many weak learners—typically shallow trees—each trained to correct the residual errors of its predecessors. XGBoost is designed for speed and performance, incorporating advanced features like regularization (to prevent overfitting), tree pruning, parallel computation, and efficient handling of missing data. Unlike linear models, XGBoost can naturally capture complex, non-linear interactions between features, making it highly effective for structured tabular data. However, its predictive power comes at the cost of interpretability, often making it less transparent compared to simpler models. XGBoost builds an ensemble of decision trees sequentially. At each step, it adds a new tree to correct the errors (residuals) made by the sum of the previous trees. The prediction at step t is:

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + f_t \tag{6}$$

where f_t is the correction t^{th} learner adds to the prediction. XGBoost minimizes a regularized objective function and it is given by

$$\mathcal{L}^{(t)} = \sum_{i=0}^{n} l(y, \hat{y}^{(t)}) + \Omega(f_t(x_i)) \tag{7}$$

Here l is a differential loss funtion (MSE for regression, log-loss for classification) and Ω is the regularizing the term given by

$$\Omega = \gamma T + \frac{\lambda}{2} \sum_j w_j^2 \tag{8}$$

where T is no of leaf nodes and w_j is the weight of j^{th} leaf node. λ and γ are hyper-parameters. To optimize loss efficiency, the loss function can be Taylor expanded around the region $\hat{y}^{(t)} - \hat{y}^{(t-1)}$

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{9}$$

$$g_i = \frac{\partial l}{\partial \hat{y}^{(t)}}\bigg|_{\hat{y}^{(t)} = \hat{y}^{(t-1)}}, \quad h_i = \frac{\partial^2 l}{\partial (\hat{y}^{(t)})^2}\bigg|_{\hat{y}^{(t)} = \hat{y}^{(t-1)}}$$

Fine-tuning an XGBoost model is crucial because its performance is highly sensitive to hyperparameter settings, which govern model complexity, learning dynamics, and regularization. Proper tuning enhances predictive accuracy, prevents overfitting, and ensures the model generalizes well to unseen data. The process

involves systematically searching for the combination of hyperparameters that yields the best predictive performance on validation data. In this study, a randomized search approach was employed over a predefined hyperparameter grid. Key parameters included the number of boosting rounds (`n_estimators`), tree depth (`max_depth`), learning rate, and sampling ratios (`subsample`, `colsample_bytree`), all of which control model complexity and generalization. Regularization terms such as `reg_alpha` (L1 penalty), `reg_lambda` (L2 penalty), and `gamma` (minimum loss reduction for a split) were also tuned to prevent overfitting. The search was conducted using `RandomizedSearchCV` with five-fold cross-validation and negative mean squared error as the scoring metric. This process allowed for efficient exploration of the hyperparameter space and helped identify a configuration that balances model accuracy and robustness.

2.3 Bayesian Modeling

Bayesian modelling is a probabilistic approach to statistical inference that incorporates prior knowledge and updates beliefs based on observed data using Bayes' Theorem. In this framework, model parameters are treated as random variables, and uncertainty is expressed through probability distributions. The posterior distribution—obtained by combining the prior and the likelihood—captures updated beliefs about the parameters after seeing the data.

$$Posterior = Likelihood \times Prior \tag{10}$$

This modelling philosophy differs fundamentally from traditional machine learning (ML) methods, which often focus on optimizing a single set of parameters to minimize prediction error. Bayesian models instead generate a full distribution over possible parameter values, offering a more nuanced understanding of uncertainty. Central to Bayesian analysis is the likelihood function, which quantifies how probable the observed data are, given specific parameter values. The log-likelihood is often used for numerical stability and interpretability. If the data are assumed independent and identically distributed under a model $f(x \mid \theta)$, the log-likelihood becomes:

$$\log L(\theta) = \sum_{i=1}^{n} \log f(x_i \mid \theta). \tag{11}$$

This expression forms the likelihood component in Bayes' theorem, to be combined with the prior on θ to obtain the posterior. Log-posterior can be written as

$$\log \mathcal{P}(\theta|D) = \log \mathcal{L}(\theta) + \log \mathcal{P}(\theta) \tag{12}$$

For gaussian prior on θ, the expression becomes equivalent Ridge regularization in ML. For $\mathcal{P}(\theta) \propto e^{-\theta^2/2\sigma^2}$, log posterior becomes,

$$\log \mathcal{P}(\theta|D) = \log \mathcal{L}(\theta) - \sum_i \theta_i^2/2\sigma^2 \tag{13}$$

and for Laplace prior ($\mathcal{P}(\theta) \propto e^{-|\theta|/2\lambda}$), expression corresponds to Lasso regularization.

$$\log \mathcal{P}(\theta|D) = \log \mathcal{L}(\theta) - \sum_i |\theta_i|/2\lambda \tag{14}$$

Once the posterior distribution is obtained, it becomes the foundation for making predictions and deriving credible intervals. However, in most practical scenarios, analytic solutions are intractable. To address this, computational methods such as Markov Chain Monte Carlo (MCMC) and variational inference are employed to draw samples from or approximate the posterior. Tools like `PyMC` and `Stan` streamline this process by automating both the log-likelihood evaluation and posterior estimation.

This computational framework enables one of Bayesian modelling's core strengths: the seamless integration of prior information. Such a feature is particularly advantageous in data-scarce environments, where conventional machine learning models may overfit or fail to generalize. By providing full posterior distributions rather than mere point estimates, Bayesian methods offer credible intervals that naturally quantify uncertainty.

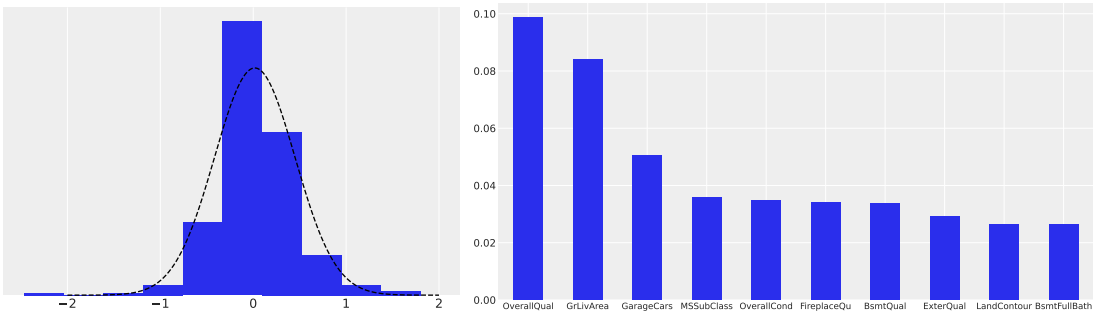

Figure 1: On the left we have the residual density distribution from the test dataset under the linear regression model. The distribution is approximately Gaussian in nature, indicating that the model's error terms are symmetrically distributed and homoscedastic. On the right we have bar chart of model features with highest weight values. Here the given weights are normalized using sum of all weights

Moreover, the generative and transparent nature of Bayesian models enhances interpretability, allowing for clearer insight into the assumptions driving predictions. This clarity also supports rigorous model evaluation and comparison through principled criteria like the Widely Applicable Information Criterion (WAIC) and Bayes factors, moving beyond the reliance on cross-validation scores alone. In this way, Bayesian modelling prioritizes understanding and uncertainty, standing in contrast to black-box ML approaches and offering a robust framework for decision-making—especially in high-stakes or low-data settings.

3 Data

In this study, we explore the housing price dataset from a Kaggle competition (data), featuring 1,460 entries and 80 features, including the target variable 'SalePrice'. Home prices in the dataset range from $\$34,900$ to $\$755,000$, but to limit the impact of outliers, we focus only on properties priced under $\$400,000$. The dataset includes 37 numerical and 43 categorical features. Categorical variables are encoded based on the rank of their mean target values. Features 'Fence' and 'PoolQC' are removed due to their high proportion of missing values. We also eliminate features with strong correlations (above 0.6) to reduce redundancy. After preprocessing, we're left with a clean set of 65 features for analysis.

4 Results

This section presents the outcome of my analysis using three models: Linear regression, XGBoost and Bayesian linear regression. I split the full dataset into training and testing subset in a 4:1 ratio. Each model has been trained on the training set and evaluated on the test dataset. Model performance was compared using the Mean Squared Error (MSE) and the R^2 score and is summarised in Table 1. I have studied all three models in their standard setting for comparison.

4.1 baseline

I fit the training set using Linear regression and evaluated the model on test data set, obtaining MSE of 0.188 and R^2 score of 0.808. The residuals of the predicted values closely follow a normal distribution, consistent with $\mathcal{N}(0, 0.4)$, as shown in Figure 1. Additionally, I have plotted top 10 features model identified along with their normalised weights in Figure 1.

Subsequently, I trained an XGBoost model on the same dataset. The model yielded an MSE of 0.155 and an R^2 score of 0.833. The residuals again exhibit a Gaussian distribution, approximately $\mathcal{N}(0, 0.4)$, as shown in Figure 2. The top 10 features ranked by gain are shown in Figure 2.

Finally, I employed a Bayesian linear regression model, which achieved an MSE of 0.187 and an R^2 score of 0.841. The residuals follow a normal distribution with a standard deviation of approximately 0.4, as shown

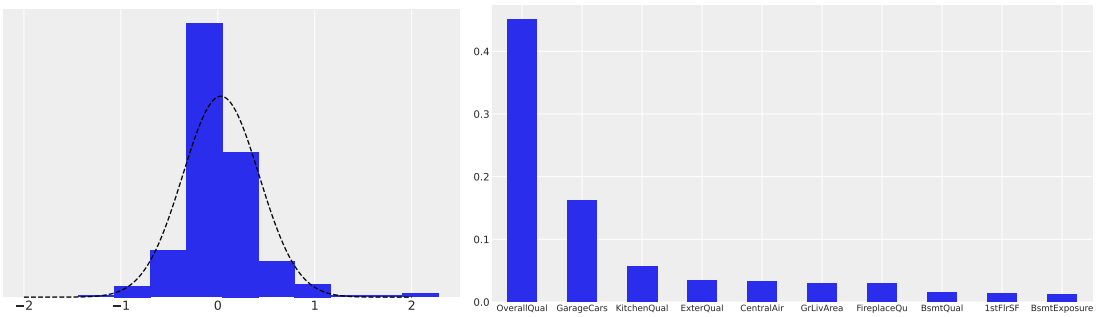

Figure 2: **Left**: Residual density distribution of the test dataset with the XGBoost predictions. The distribution is gaussian nature. **Right**: The plot shows the top 10 feature that gave the highest gains for the model splitting.

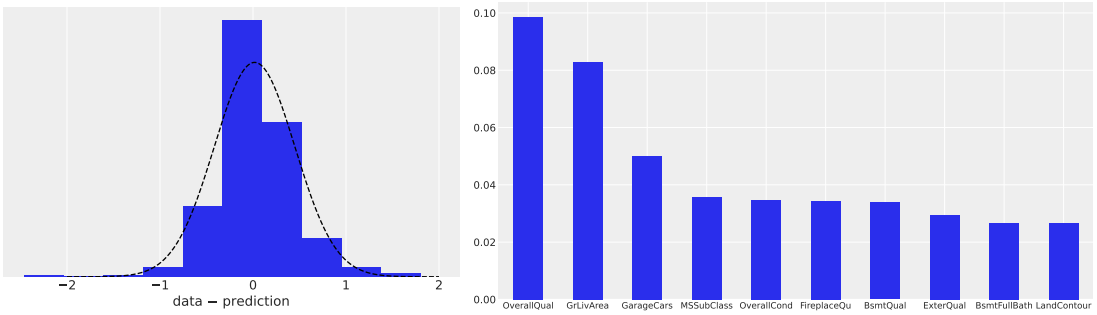

Figure 3: **Left**: Residual density distribution of the test dataset with the Bayesian model predictions. The distribution is gaussian nature. **Right**: The plot shows the top 10 feature that gave the highest feature mean weights from the posterior distribution.

in Figure 3. The top 10 features, based on the means of their posterior distributions (normalized), are shown in Figure 3. Figure 4 depicts the full posterior distributions of these parameters, along with the confidence levels indicating how significantly each parameter deviates from zero (i.e., the number of standard deviations the posterior mean lies from zero).

4.2 Finetuning

We now fine-tune each of these models and compare the results. We fine-tune the linear regression model by implementing cross-validation allowing both Lasso(L1) and Ridge(L2) regularization. We fine-tune the α, the strength of regularization, varying it in log-space. Upon implementing the regularization and finetuning α, we can see that the MSE has improved to 0.1622 and R2 score to 0.8621. The above scores were given by Lasso regularization. Ridge gave relatively less improvement compared to Lasso. The updated list of features for best-fit implementation is given in Figure 5.

Next, we tune the XGBoost model. We do a RandomizedSearchCV hyperparameter tuning on various XGBoost parameters and obtain significant improvement in fit. The fit improved with an MSE of 0.124 and R^2 score of 0.894, which is a substantial improvement. The updated list of top features contributing to the sales price is given in Figure 6.

For bayesian modelling, we can implement tuning to some extend by playing around with the prior functional form. We can use a Laplace prior for features which is equivalent to implementing Lasso regularization and a StudentT prior to check for effect of heavy tails. We can see that StudentT prior only adds a marginal improvement MSE while decreasing R^2 score. However, a Laplace prior improved the MSE considerably while the retaining the R^2 score. Refer to Table 1 for the results. The updated feature list and parameter significance are given in Figure 7 and Figure 8.

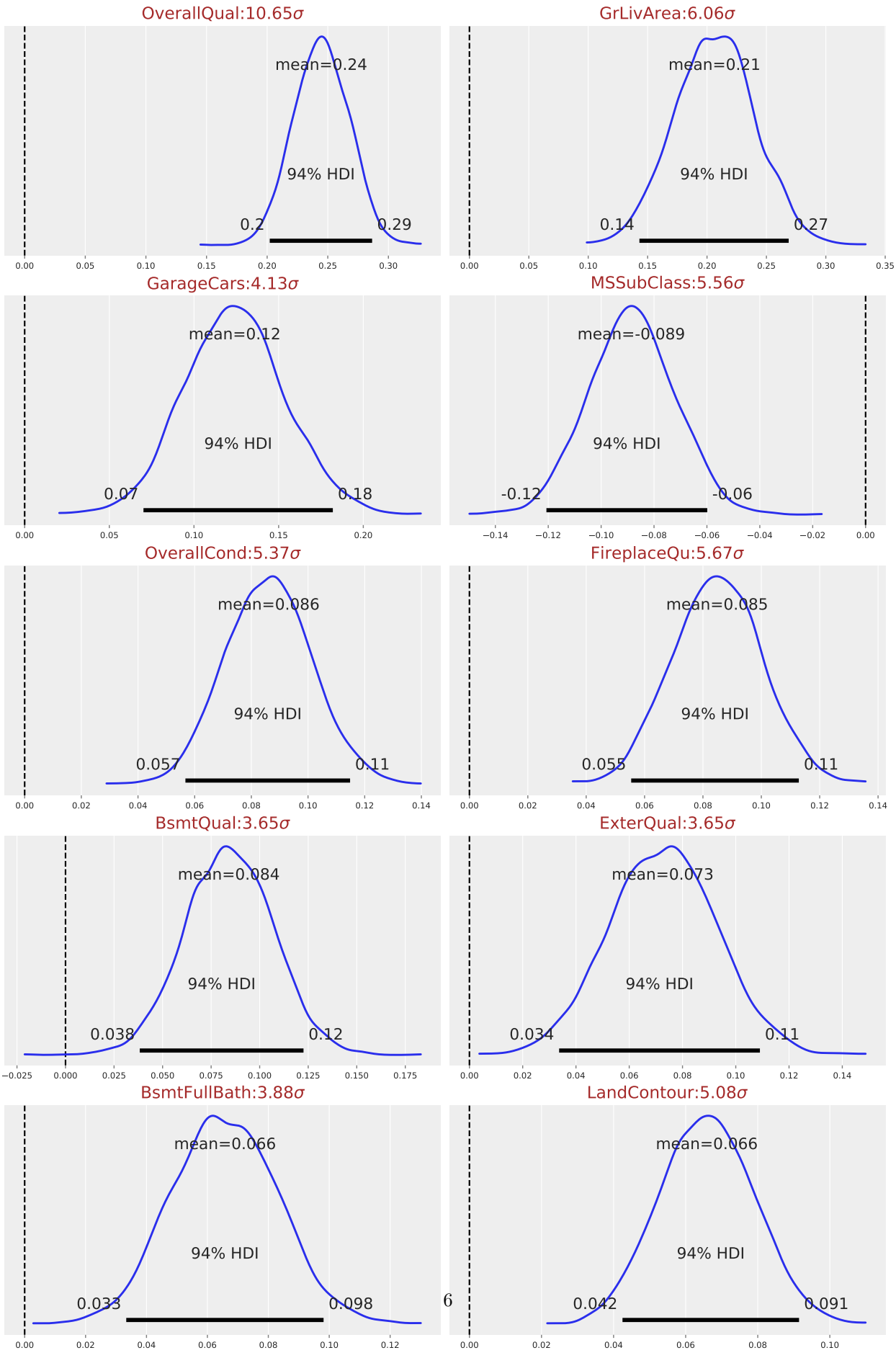

Figure 4: Features parameter posterior with its significance and confidence interval for the bayesian model with Gaussian prior on feature weights.Here significance is the distance of the mean from the zero line in terms of its standard deviation(σ)

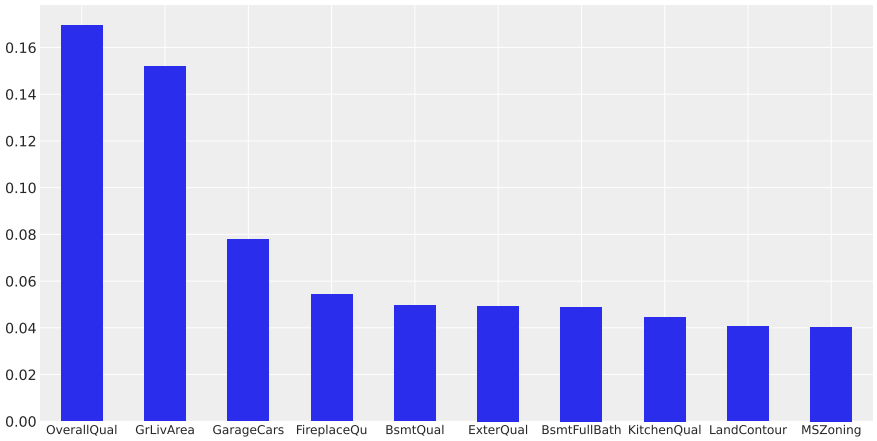

Figure 5: Features with highest weights for linear regression with Lasso regularization

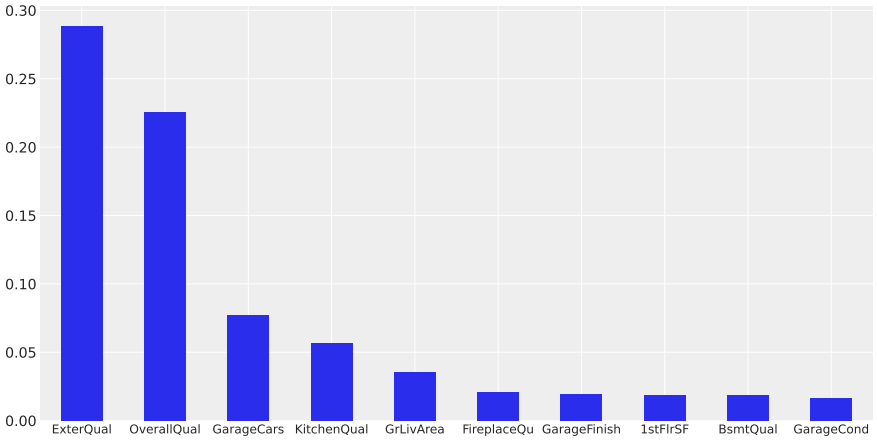

Figure 6: Features with highest gains for fine tuned XGBoost model

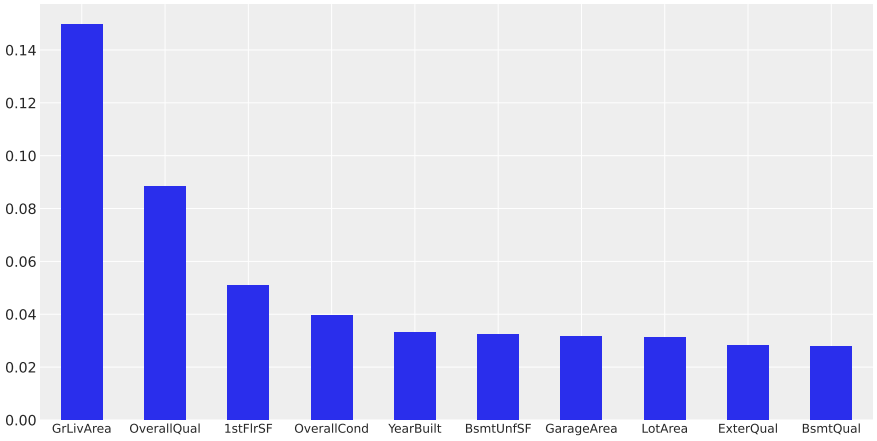

Figure 7: Features with highest weights for the bayesian model with Laplacian prior on feature weights

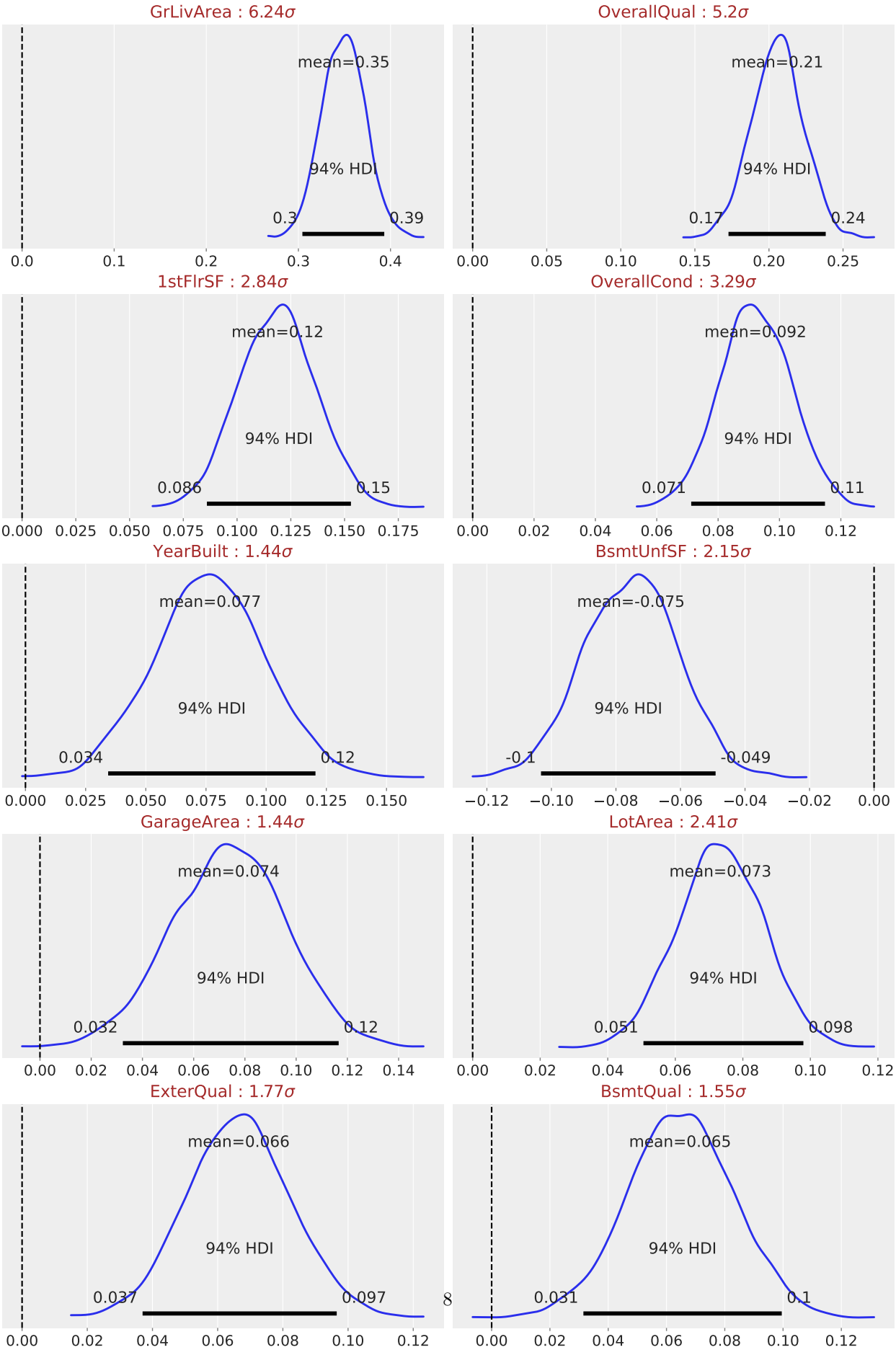

Figure 8: Features parameter posterior with its significance and confidence interval for the bayesian model with Laplacian prior on feature weights

	Linear Regression			XGBoost		Bayesian Modelling		
	baseline	Ridge	Lasso	baseline	finetuned	baseline	StudenT	Laplace
MSE	0.188	0.171	0.162	0.155	0.125	0.187	0.170	0.158
R2 score	0.808	0.855	0.862	0.833	0.894	0.841	0.835	0.841

Table 1: MSE and R2 score for various models

5 Discussion

In this article we study three models, linear regression, XGBoost and Bayesian model implemented using PyMC packages. We first analyse the models in their standard setting and calculate the mean squared error and R^2 score for comparison. We see that with standard settings (baseline), XGBoost has the lowest MSE and Bayesian model has the highest the R^2 score. We can also see that all three models identifies total living area, overall building quality and garage area as the main deciding factors to the price of the house. While linear regression and bayesian linear regression predicts similar weights to overall quality and total living space XGBoost selects former as main contributor to sales price with much higher weights compared to later. Though Bayesian model identifies living area as the feature with highest mean weight, it shows a much high statistical significance for overall-quality (9.5σ). All top ten features have a significance above 3.5σ.

When we do hyper parameter tuning to the model, we see that XGBoost out perform the liner regression methodologies as it is able to capture the complexities of the data. XGBoost has a much higher R^2 score (0.89) and a lower MSE (0.125) compared to other methods. For linear regression with Lasso regularization, we see a 0.25 improvement in MSE compared to the baseline fit and a 0.54 for R^2 score. Bayesian model (Laplace prior) improves MSE by 0.3 and retains the same R^2 score. After finetuning, XGBoost finds exterior quality as the top ranked feature followed by overall quality and garage space. The Bayesian model retains its top two features however with reduced significance. This is because the Laplace prior now prefers lower weights compared to the normal priors. As a result, the mean value of the features come down thereby reducing the total significance of the parameter.

Both linear regression models producing similar results as both try to fit a hyperplane to the data. The XGBoost which implements decision trees can account for non-linear relation between the features and target value. The Bayesian model could account for these nonlinearities if the underlying model is a nonlinear model. This could be accomplished by implementing multi-level or hierarchical models.