Comparative Analysis of Statistical and Machine Learning Models for Housing Price Prediction

Jimmy Wu

2025-09-25

Introduction

Predictive modeling is a cornerstone of modern statistics and data science, where the goal is to identify factors that influence outcomes and build models that can accurately predict future values. In both academic and practical settings, the ability to forecast home prices is crucial. Specifically, it helps people in understanding the future perspective of real estate - a key aspect of the society that is not only a driver of the economy but also deeply tied to social and community development (Malpezzi, 2002). For policymakers, housing price models help evaluate affordability, guide regional planning, and provide insights regarding supply and demand. For individuals and businesses, housing price predictions can inform investment decisions and risk management strategies.

Statistically, the main challenge in the housing price problem is to balance the trade-offs between prediction accuracy, interpretability, and uncertainty. Statistical modeling and machine learning are the prevalent strategies to this challenge. Traditional frequentist regression offers coefficient estimates and hypothesis testing but may be insufficient when facing complex, nonlinear relationships. Machine learning methods such as random forests and gradient boosting can improve predictive performance, though often at the cost of interpretability due to the non-parametric nature of the tree-based model (Kuhn & Johnson, 2013). Meanwhile, Bayesian regression introduces a probabilistic foundation that allows uncertainty to be explicitly modeled through posterior distributions and credible intervals (Gelman et al., 2013). Together, these approaches represent complementary strategies for addressing sophisticated prediction problems.

This project investigates housing prices using the *Ames Housing dataset* (De Cock, 2011), a widely used benchmark in predictive modeling. The *Ames Housing dataset* is a dataset containing detailed information on nearly 3,000 residential properties in Ames, Iowa. It includes rich feature descriptions, such as lot size, neighborhood, building quality, and sale conditions, making it ideal for exploring predictive modeling and housing price analysis. We employed a full comparative framework that includes frequentist linear regression, regularized regressions (ridge and lasso), Bayesian linear regression, and machine learning techniques (random forest and gradient boosting). Each method is evaluated not only in terms of predictive accuracy (i.e. metrics such as RMSE, MAE, and R^2) but also in terms of interpretability and the treatment of uncertainty. By systematically comparing these approaches, this study provides insights into the strengths and weaknesses of each, offering both practical recommendations for prediction tasks and a deeper understanding of methodological trade-offs.

Methods

This study adopts a comparative modeling framework to evaluate multiple statistical and machine learning approaches for predicting housing prices using the *Ames Housing dataset* (De Cock, 2011), done in *R* using packages such as *glmnet*, *brms*, *ranger*, *xgboost*, etc. The dataset contains 2,930 housing records with 82 attributes describing structural, spatial, and qualitative features of the properties. Data preprocessing included the removal of missing values through pairwise deletion and the selection of key predictors, specifically *Overall Quality* (overall quality), *Gr.Liv.Area* (living area), *Garage.Cars* (garage capacity), *Garage.Area* (garage size), *Total.Bsmt.SF* (basement size), *X1st.Flr.SF* (first floor size), and *Year.Built* (year built), based on their correlation with *SalePrice*.

The data were randomly split into training (80%) and testing (20%) sets for model development and validation. A series of models were fitted to predict housing prices. The baseline model was a multiple linear regression estimated via ordinary least squares, followed by a log-transformed linear model to correct for heteroscedasticity and non-normal residuals. Regularized regression methods (ridge and lasso) were implemented using the *glmnet* package, with optimal penalty parameters λ determined via 10-fold cross-validation. Bayesian linear regression was then conducted using the *brms* package with normally distributed priors on coefficients and a Student-t prior on residual error to quantify uncertainty through posterior credible intervals.

To explore nonlinear and interaction effects, machine learning models were introduced: Random Forest using the *ranger* package and Gradient Boosting via *xgboost*. Both models utilized bootstrap aggregation and feature subsampling to enhance robustness and prevent overfitting. Model performance was evaluated using three key metrics on the test data: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2). Variable importance measures were extracted from the tree-based models to assess the relative influence of predictors.

Together, these methods allow a balanced comparison between interpretability and predictive accuracy, showcasing how traditional regression and modern ensemble learning methods perform under the same predictive framework.

Results

Data Exploration

We will first conduct data explorations.

```r
housing <- read.csv("AmesHousing.csv")
kable(dim(housing), caption = "AmesHousing Dataset Dimension", col.names = "Rows by Columns")
```

Table 1: AmesHousing Dataset Dimension

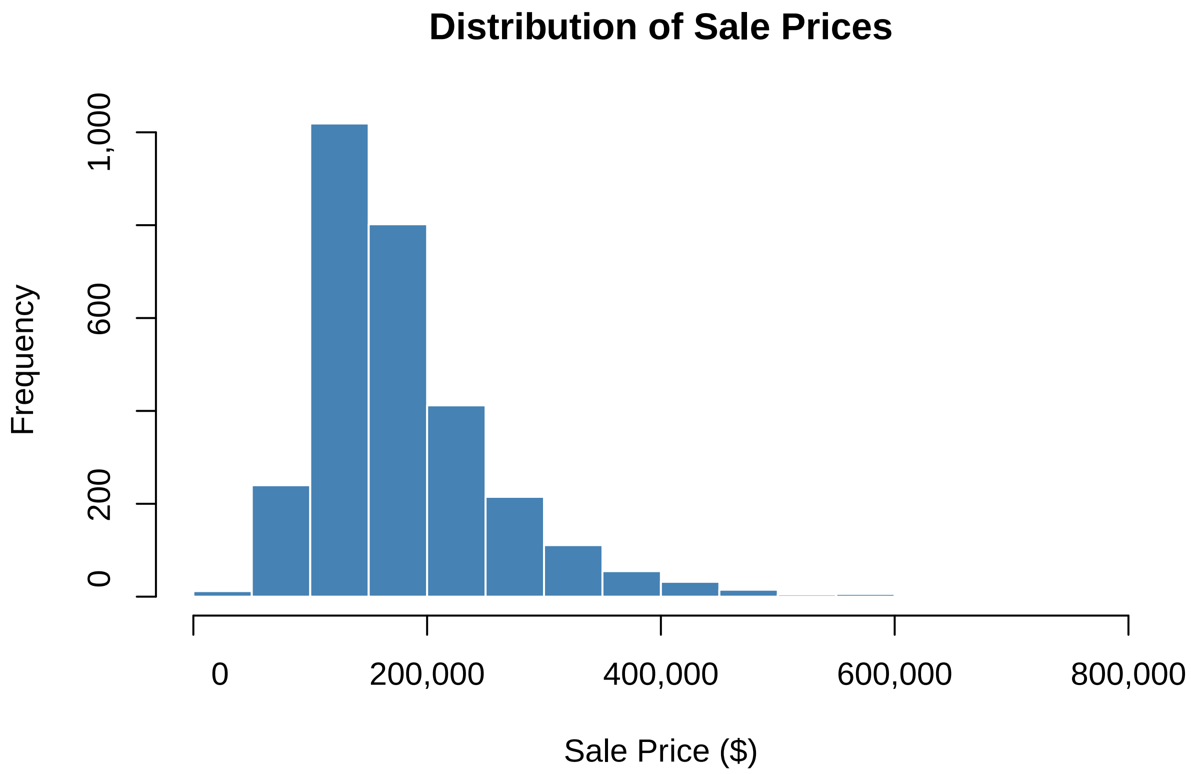
Rows by Columns
2930
82

This dataset contains data of the sales price in 2,930 sample houses. Housing conditions such as *Lot Frontage* (lot frontage), *Lot Area* (lot area), *Overall Quality*, etc were included to provide reference. We will be studying the relationship between sales price and these variables that describe the condition of the houses in different perspectives.

We are specifically interested in the price of the houses, and therefore we will be conducting a close exploration of the variable *SalePrice* (price of the house).

Explore the target variable 'SalePrice'

```r
options(scipen = 999)

hist(housing$SalePrice,
     main = "Distribution of Sale Prices",
     xlab = "Sale Price ($)",
     col  = "steelblue",
     border = "white",
     xaxt = "n",    # suppress x-axis
     yaxt = "n")    # suppress y-axis

x_ticks <- axTicks(1)
axis(1, at = x_ticks, labels = format(x_ticks, big.mark = ","))

y_ticks <- axTicks(2)
axis(2, at = y_ticks, labels = format(y_ticks, big.mark = ","))
```



The distribution of *SalePrice* is moderately right-skewed, while centering around the value of approximately $ 100k. The extreme values appears to go up to $ 600k dollar.

To better understand which housing characteristics are most strongly associated with *SalePrice*, we computed the correlation coefficients between *SalePrice* and all numeric predictors in the dataset.

Explore the correlation of some predictors

We will be using pairwise comparison observation method to compute the correlation between predictors.

```r
numeric_vars <- sapply(housing, is.numeric)
cor_matrix <- cor(housing[, numeric_vars], use = "pairwise.complete.obs")
cor_target <- sort(cor_matrix[,"SalePrice"], decreasing = TRUE)

kable(head(cor_target, 10), caption = "Correlation of SalePrice Predictors",
      col.names = c("Predictors", "Correlation Coefficient"))
```

Table 2: Correlation of SalePrice Predictors

Predictors	Correlation Coefficient
SalePrice	1.0000000
Overall.Qual	0.7992618
Gr.Liv.Area	0.7067799
Garage.Cars	0.6478766
Garage.Area	0.6404008
Total.Bsmt.SF	0.6322805
X1st.Flr.SF	0.6216761
Year.Built	0.5584261
Full.Bath	0.5456039
Year.Remod.Add	0.5329738

From the table above, we can observe the following:

- **Overall.Qual (≈ 0.80)**: The overall quality of the house shows the strongest positive correlation with sales price, suggesting that higher quality ratings are likely to result in higher property values.
- **Gr.Liv.Area (≈ 0.71)**: Above-ground living area is also highly correlated with sales price, reflecting the intuitive idea that larger houses tend to sell for more.
- **Garage.Cars (≈ 0.65) and Garage.Area (≈ 0.64)**: Both the number of garage spaces for vehicle and total garage area show strong correlations with sales price, indicating that garage capacity is a valued feature.
- **Total.Bsmt.SF (≈ 0.63) and 1st.Flr.SF ≈ (0.62)**: Basement and first-floor space are also highly associated with sale price, also reflecting the intuitive sense that larger house tend to value more.
- **Year.Built (≈ 0.56) and Year.Remod.Add (≈ 0.53)**: More recently built or remodeled houses tend to worth higher values, showing buyers' interest for modern features and up-to-date construction.
- **Full.Bath (0.55)** (number of full bathrooms): The number of full bathrooms is moderately correlated with price, reflecting buyers' moderate preference for more complete bathroom facilities.

With the basic understanding of how different predictors are related to our target variable *SalePrice*, we now attempt to fit a linear model.

Fit a Linear Model

We will first be clearning the dataset and leaving only the interested variables, then proceed to fit a linear model using the *lm_model* function.

```r
# Here, we pull out the variables of interest
vars <- c("SalePrice",
          "Overall.Qual", "Gr.Liv.Area", "Garage.Cars",
          "Garage.Area", "Total.Bsmt.SF", "X1st.Flr.SF", "Year.Built")
```

```r
# We then clean the dataset and leave only the interested variables
dat <- na.omit(housing[, vars])

set.seed(20250929)
n <- nrow(dat)

# We will randomly chose 80% of the SalePrice data for training purposes
train_index <- sample(seq_len(n), size = floor(0.8 * n), replace = FALSE)
train_data  <- dat[train_index, ]

# The remaining 20% of the SalePrice data could be used for performance checking
test_data   <- dat[-train_index, ]

lm_model <- lm(SalePrice ~ Overall.Qual + Gr.Liv.Area + Garage.Cars +
                 Garage.Area + Total.Bsmt.SF + X1st.Flr.SF + Year.Built,
               data = train_data)

lm_model_summary <- summary(lm_model)

coefs <- as.data.frame(lm_model_summary$coefficients)
colnames(coefs)[4] <- "Pr"

rsq <- lm_model_summary$r.squared
adj_rsq <- lm_model_summary$adj.r.squared
sigma <- lm_model_summary$sigma
f_stat <- lm_model_summary$fstatistic[1]
df1 <- lm_model_summary$fstatistic[2]
df2 <- lm_model_summary$fstatistic[3]

f_statistic <- lm_model_summary$fstatistic
p <- pf(f_statistic[1], f_statistic[2], f_statistic[3], lower.tail = FALSE)

overview <- data.frame(
  `R-squared` = rsq,
  `Adjusted R-squared` = adj_rsq,
  `Residual Std. Error` = sigma,
  `F-statistic` = f_stat,
  `df1 (num)` = df1,
  `df2 (den)` = df2,
  `p-value` = signif(p,5)
)

kable(coefs, caption = "lm Model Coefficients", digits = 5, escape = FALSE)
```

Table 3: lm Model Coefficients

	Estimate	Std. Error	t value	Pr
(Intercept)	-684365.35367	64859.00999	-10.55158	0.00000
Overall.Qual	21665.86186	870.46642	24.88995	0.00000
Gr.Liv.Area	46.70605	2.12655	21.96329	0.00000
Garage.Cars	6433.33135	2416.72839	2.66200	0.00782
Garage.Area	27.86652	8.29705	3.35861	0.00080

	Estimate	Std. Error	t value	Pr
Total.Bsmt.SF	20.18181	3.13365	6.44035	0.00000
X1st.Flr.SF	13.90348	3.57918	3.88454	0.00011
Year.Built	304.96391	34.05928	8.95392	0.00000

```r
kable(overview, caption = "lm Model Fit Statistics", digits = 5, escape = FALSE)
```

Table 4: lm Model Fit Statistics

	R.squared	Adjusted.R.squared	Residual.Std..Error	F.statistic	df1..num.	df2..den.	p.value
value	0.78137	0.78071	37143.82	1191.632	7	2334	0

The model explains approximately 78% of the variation in SalePrice ($R^2 = 0.781$). The overall F-test is highly significant (p-value ≈ 0). All predictors are statistically significant. The strongest are *Overall.Qual*, *Gr.Liv.Area*, and *Year.Built*. Each of them has p-value ≈ 0.

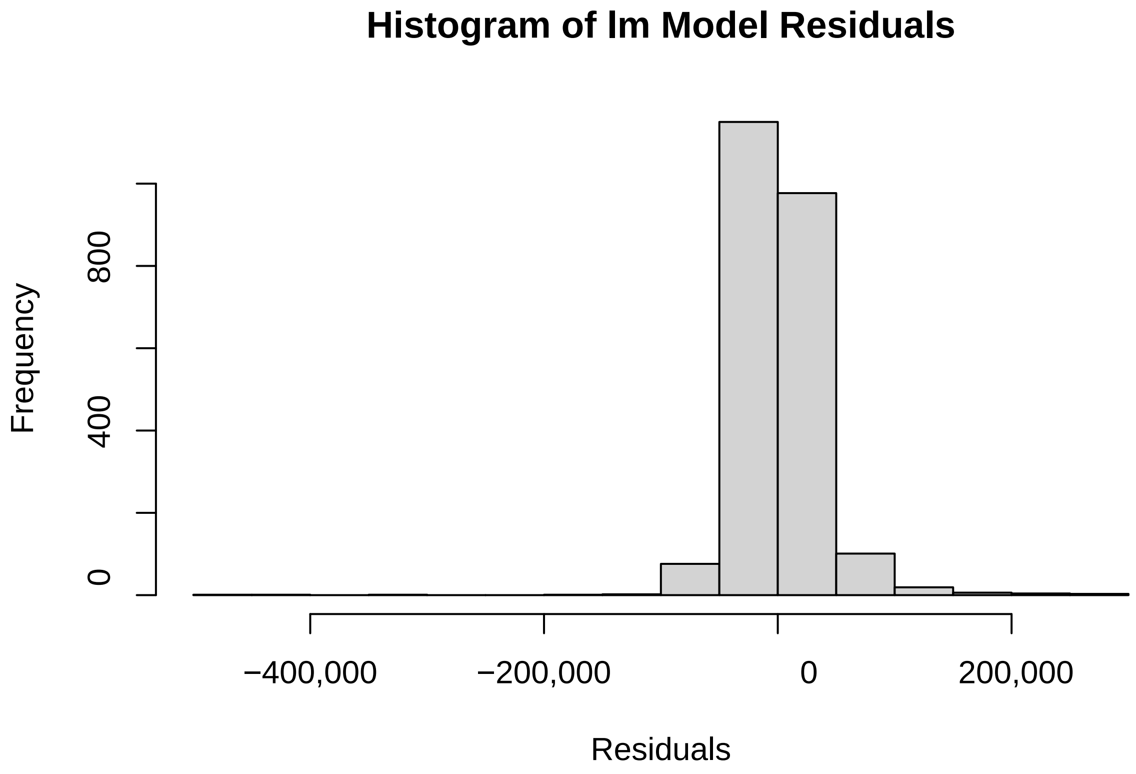
All coefficients are positive, meaning higher quality, larger area, or newer houses all increase *SalePrice*. For example, each unit increase in *Overall.Qual* adds about \$21,666 to *SalePrice*. The residual standard error is about \$37,143, showing predictions can still miss actual prices by tens of thousands.

To ensure the validity of our model, we now conduct the normality check for the model residuals.

Linear Model Assumptions Check

```r
# Histogram of the Residuals
options(scipen = 999)

hist(lm_model$residuals,
     main = "Histogram of lm Model Residuals",
     xlab = "Residuals",
     col  = "lightgray",
     border = "black",
     xaxt = "n",    # turn off default x-axis
     yaxt = "n")    # turn off default y-axis

x_ticks <- axTicks(1)
axis(1, at = x_ticks, labels = format(x_ticks, big.mark = ","))

y_ticks <- axTicks(2)
axis(2, at = y_ticks, labels = format(y_ticks, big.mark = ","))
```

Normality of the Data



The residuals distribution of our model appears to be relatively normal with moderate skewness to the right. Specifically, the distribution is centered around 0, with the lower tail located around 10,000 and upper tail located around 30,000.

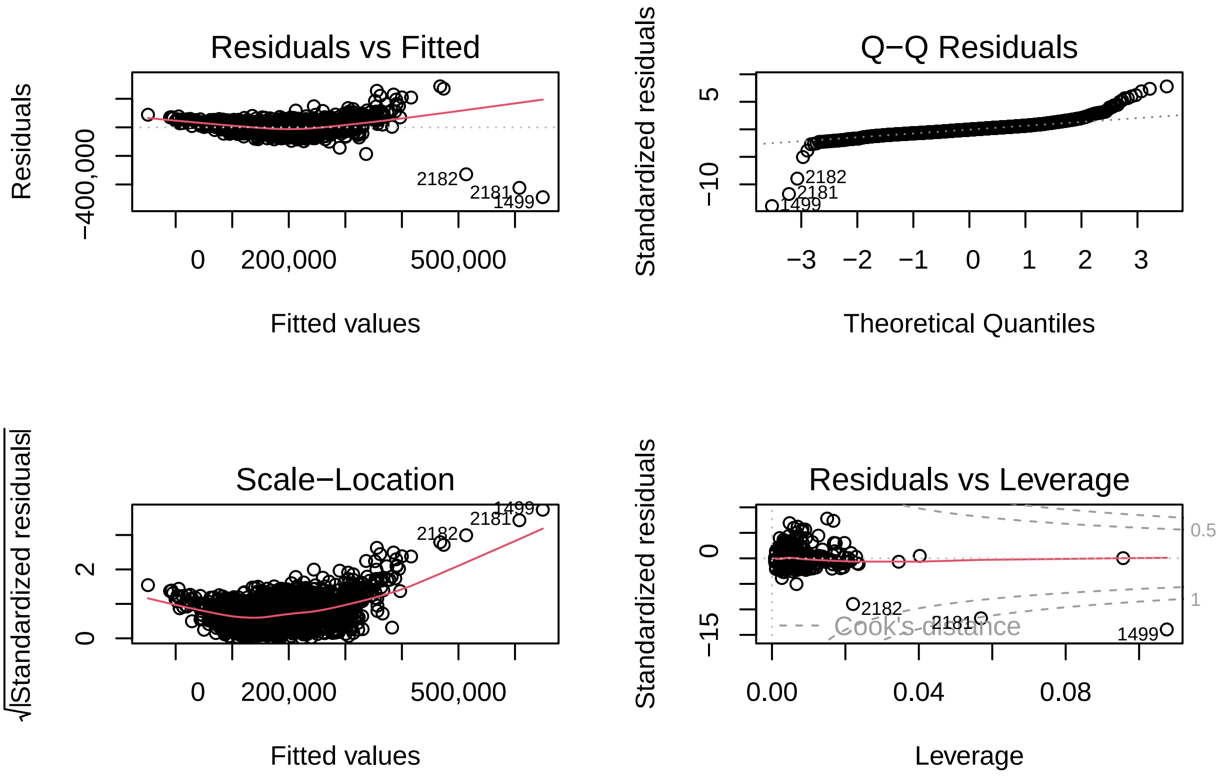
Despite the generally normal-looking distribution, we cannot derive a clear conclusion on the normality of the residuals distribution due to the moderate skewness.

We will now plot the residuals to take a closer look.

```
options(scipen = 999)
par(mfrow = c(2, 2))

plot(lm_model, which = 1, xaxt = "n", yaxt = "n")
axis(1, at = axTicks(1), labels = format(axTicks(1), big.mark = ","))
axis(2, at = axTicks(2), labels = format(axTicks(2), big.mark = ","))

plot(lm_model, which = 2, xaxt = "n", yaxt = "n")
axis(1, at = axTicks(1), labels = format(axTicks(1), big.mark = ","))
axis(2, at = axTicks(2), labels = format(axTicks(2), big.mark = ","))

plot(lm_model, which = 3, xaxt = "n", yaxt = "n")
axis(1, at = axTicks(1), labels = format(axTicks(1), big.mark = ","))
axis(2, at = axTicks(2), labels = format(axTicks(2), big.mark = ","))
```

```r
plot(lm_model, which = 5, xaxt = "n", yaxt = "n")
axis(1, at = axTicks(1), labels = format(axTicks(1), big.mark = ","))
axis(2, at = axTicks(2), labels = format(axTicks(2), big.mark = ","))
```

Residuals Plots



In each plot, we can derive some useful information about the residual normality of the data.

- **Residuals vs Fitted Plot**: The residuals show a slightly curved pattern, especially at low and high fitted values. This indicates possible non-linearity. A transformation (e.g., log of *SalePrice*) might help.
- **Q–Q Residuals Plot**: Residuals deviate strongly from the straight line around the tails. This indicates non-normality and possibly heavy-tailed errors. Outliers (like obs 2182, 2181, 1499) likely contributed to this deviation.
- **Scale–Location Plot**: The spread of residuals increases with fitted values. This suggests heteroscedasticity - that is, variance of errors is not constant. Again, a log transformation of *SalePrice* may stabilize the variance.
- **Residuals vs Leverage Plot**: A few points (e.g., 2182, 2181, 1499) have both high residuals and higher leverage. Therefore, these points could be highly influential to the model. All other points remain relatively low-leverage.

Together, these four plots suggests that the residuals of our model could be non-linear. We now proceed to conduct a normality test to examine our preliminary conclusion of the non-normality of the model's residual.

```
shapiro_result <- shapiro.test(lm_model$residuals)

shapiro_df <- data.frame(
  "Statistic" = signif(shapiro_result$statistic, 5),
  "p_Value" = format(shapiro_result$p.value,
                     scientific = TRUE, digits = 5)
)

kable(shapiro_df, caption = "Shapiro-Wilk Normality Test for lm Model Residuals")
```

Residual Normality Test

Table 5: Shapiro-Wilk Normality Test for lm Model Residuals

	Statistic	p_Value
W	0.81843	1.6297e-45

The Shapiro-Wilk normality test shows a p-value $\approx 1.63 \times 10^{-45} < 0.001$. This extremely small p-value indicates strong evidence against normality. We conclude that the residuals are not normally distributed, which aligns with the patterns seen in the diagnostic plots.

In conjunction with the results from the previous tests, we determined that the model's residuals are not normally distributed. For a more holistic review, we continue to check the model performance using our *test* dataset.

```
pred <- predict(lm_model, newdata = test_data)

rmse <- sqrt(mean((test_data$SalePrice - pred)^2))
mae  <- mean(abs(test_data$SalePrice - pred))
r2   <- cor(test_data$SalePrice, pred)^2

metrics_df <- data.frame(
  "RMSE" = signif(rmse, 4),
  "MAE" = signif(mae, 4),
  "R-squared" = signif(r2, 4)
)

kable(metrics_df, caption = "lm Model Performance on Test Dataset")
```

Table 6: lm Model Performance on Test Dataset

RMSE	MAE	R.squared
34660	24330	0.8252

For model performance, the RMSE is 34,660, meaning predictions are off by about $35k on average when squared errors are considered. The MAE is 24,330, so the average absolute prediction error is about $24k. The R^2 is approximately 0.83, which means the model explains about 83% of the variation in sale prices.

Overall, the model shows relatively strong predictive power but violates the assumptions of normality and constant variance. Therefore, an improvement on the model fitness such as a log transformation of *SalePrice* should be considered.

Log transformation of the lm model

We now conduct a log transformation of *SalePrice* to improve the model fitness.

```r
train_data$LogSalePrice <- log(train_data$SalePrice)
test_data$LogSalePrice  <- log(test_data$SalePrice)

lm_model_log <- lm(LogSalePrice ~ Overall.Qual + Gr.Liv.Area + Garage.Cars +
                     Garage.Area + Total.Bsmt.SF + X1st.Flr.SF + Year.Built,
                   data = train_data)

lm_model_log_summary <- summary(lm_model_log)

coefs_log <- as.data.frame(lm_model_log_summary$coefficients)
colnames(coefs)[4] <- "Pr"

rsq <- lm_model_log_summary$r.squared
adj_rsq <- lm_model_log_summary$adj.r.squared
sigma <- lm_model_log_summary$sigma
f_stat <- lm_model_log_summary$fstatistic[1]
df1 <- lm_model_log_summary$fstatistic[2]
df2 <- lm_model_log_summary$fstatistic[3]

f_statistic_log <- lm_model_log_summary$fstatistic
p_log <- pf(f_statistic_log[1], f_statistic_log[2], f_statistic[3], lower.tail = FALSE)

overview_log <- data.frame(
  `R-squared` = rsq,
  `Adjusted R-squared` = adj_rsq,
  `Residual Std. Error` = sigma,
  `F-statistic` = f_stat,
  `df1 (num)` = df1,
  `df2 (den)` = df2,
  `p-value` = signif(p_log,5)
)

kable(coefs_log, caption = "Log lm Model Coefficients", digits = 5, escape = FALSE)
```

Table 7: Log lm Model Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	6.11260	0.29160	20.96194	0.00000
Overall.Qual	0.11010	0.00391	28.13238	0.00000
Gr.Liv.Area	0.00022	0.00001	22.91756	0.00000
Garage.Cars	0.06377	0.01087	5.86861	0.00000
Garage.Area	0.00006	0.00004	1.73209	0.08339
Total.Bsmt.SF	0.00009	0.00001	6.65939	0.00000
X1st.Flr.SF	0.00005	0.00002	2.94896	0.00322
Year.Built	0.00234	0.00015	15.28692	0.00000

```r
kable(overview_log, caption = "Log lm Model Fit Statistics", digits = 5, escape = FALSE)
```

Table 8: Log lm Model Fit Statistics

	R.squared	Adjusted.R.squared	Residual.Std..Error	F.statistic	df1..num.	df2..den.	p.value
value	0.82458	0.82405	0.167	1567.312	7	2334	0

The model uses the log of *SalePrice* as the response, which stabilizes variance and reduces skewness. Several predictors remain highly significant. *Overall.Qual*, *Gr.Liv.Area*, *Garage.Cars*, *X1st.Flr.SF*, and *Year.Built* all have very small p-values (≈ 0), indicating their strong influence on housing prices even after transformation. The predictor *Garage.Area* is no longer significant (p $\approx$ 0.08), suggesting it does not provide explanatory power when other variables are included.

The R^2 has increased to 0.825, with an adjusted R^2 of 0.824, both higher than in the lm model before the transformation. This means about 82% of the variation in log house prices is explained by the predictors. The F-statistic is also highly significant (p $\approx$ 0), suggesting that the overall log lm model is strongly predictive.

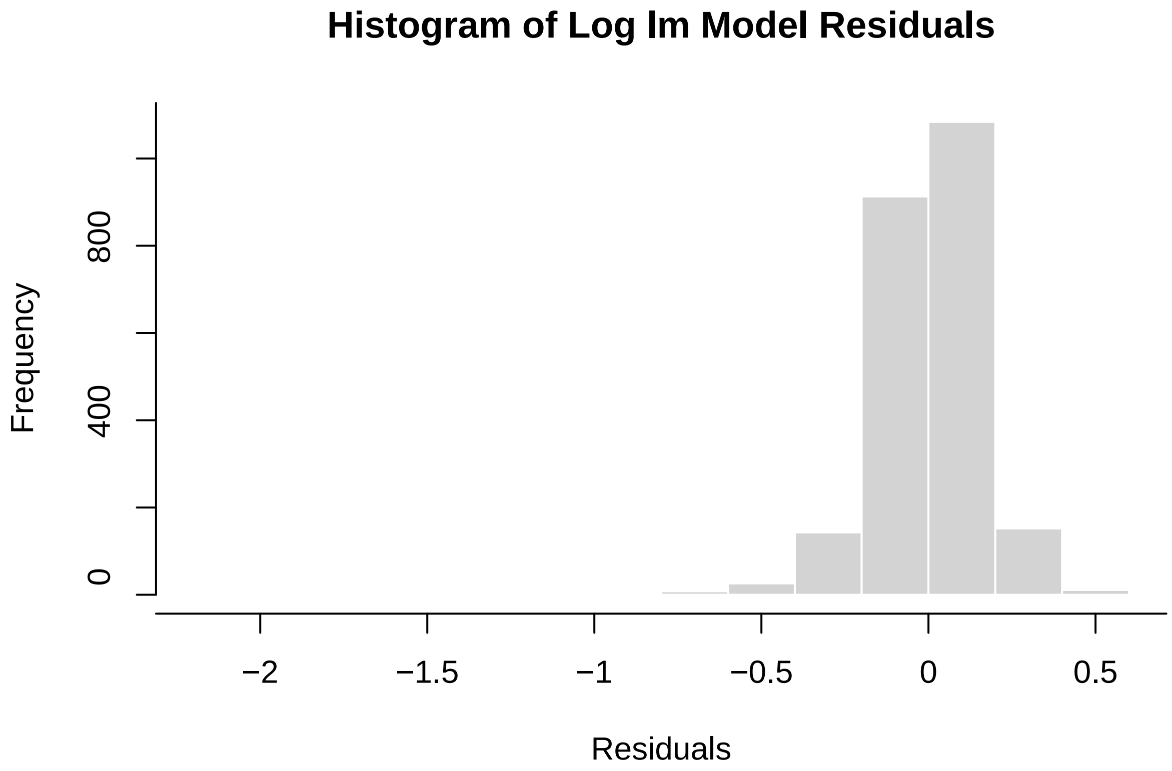
Finally, the reduction in residual standard error (≈ 0.167) indicates that predictions are more tightly centered around observed values compared to the raw-scale model.

To ensure the validity of our log lm model, we now conduct the normality check for the model residuals.

Log Linear Model Assumptions Check

```r
# Histogram of the Residuals
hist(
  lm_model_log$residuals,
  main = "Histogram of Log lm Model Residuals",
  xlab = "Residuals",
  col = "lightgray",
  border = "white",
  axes = FALSE
)

x_ticks <- pretty(lm_model_log$residuals)
axis(1, at = x_ticks, labels = x_ticks)

y_ticks <- pretty(hist(lm_model_log$residuals, plot = FALSE)$counts)
axis(2, at = y_ticks, labels = format(y_ticks, big.mark=","))
```

Normality of the Data

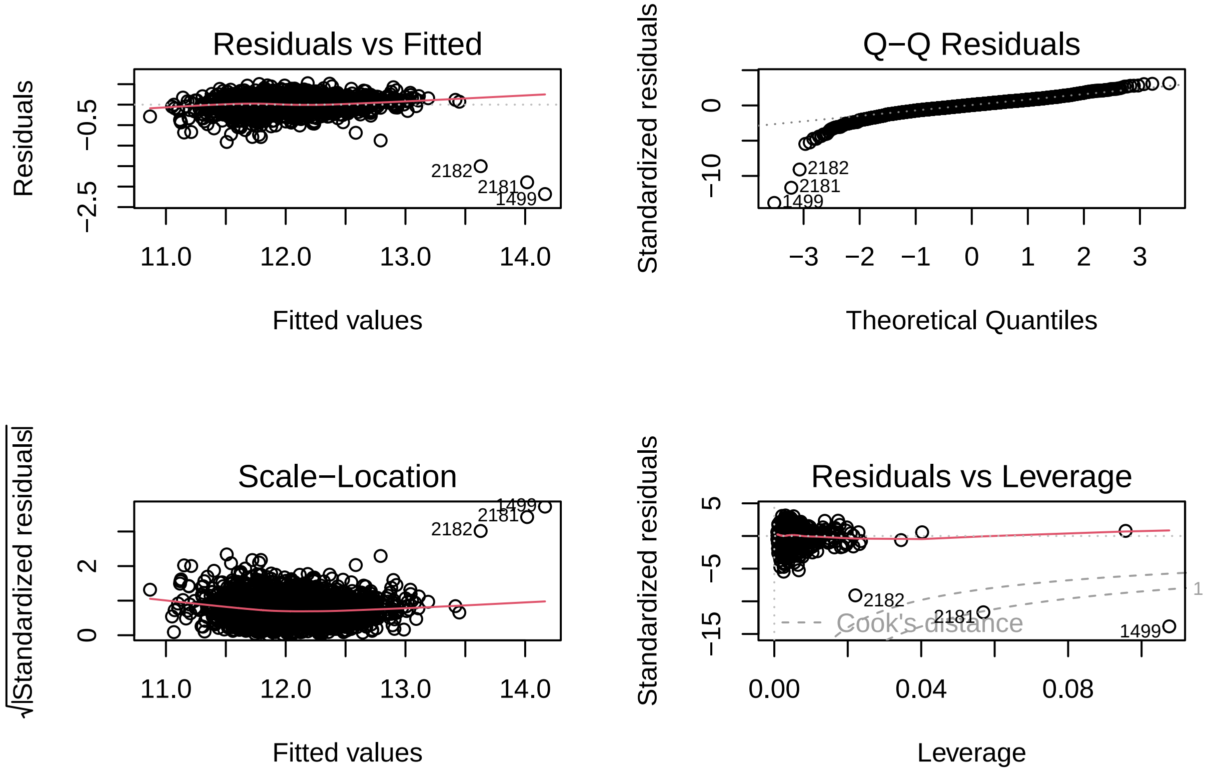


The residuals distribution of our model appears to be relatively normal with moderate skewness to the left. Specifically, the distribution is centered around 0, with the lower tail located around -1.0 and upper tail located around 0.75. Again, despite the generally normal-looking distribution, we cannot derive a clear conclusion on the normality of the residuals distribution due to the moderate skewness.

However, in this case, we are more inclined to conclude that the residuals of the log lm model are normal than the lm model since the skewness in this case is less severe. To further examine the normality, we proceed to plot the residuals.

```r
options(scipen = 999)
par(mfrow=c(2,2))
plot(lm_model_log)
```

Residuals Plots



For the log-transformed model, we can also derive some meaningful information from the residual plots.

- **Residuals vs Fitted Plot**: The residuals are centered around zero with less obvious curvature compared to the lm model. This indicates that the log transformation has improved linearity. However, at high fitted values, we can still observe values that are distanced from the pattern, suggesting potential or even light heteroscedasticity.
- **Q–Q Residuals Plot**: The residuals still deviate from the diagonal line in the tails. While the log transformation reduced skewness, some chance of non-normality persists, especially with extreme outliers.
- **Scale–Location Plot**: The spread of residuals looks more even across fitted values than before. The log transformation has reduced the fluctuation of variance, but a few high-leverage points still stand out.
- **Residuals vs Leverage Plot**: Most points fall well within Cook's distance lines, showing that no single observation dominates the model. The few cases (e.g., 2182, 1499, etc) are still more influential, but not overly so, compared to the lm model.

Overall, the log transformation meaningfully improved model assumptions, especially linearity and variance stability. Normality of residuals is not perfect, but the model is now more robust and interpretable than before.

We now move forward to further examine the normality assumption.

```
shapiro_result_log <- shapiro.test(lm_model_log$residuals)
```

```r
shapiro_df_log <- data.frame(
  "Statistic" = signif(shapiro_result_log$statistic, 5),
  "p_Value" = signif(shapiro_result_log$p.value, 5)
)

kable(shapiro_df_log, caption = "Shapiro-Wilk Normality Test for Log lm Model Residuals")
```

Residual Normality Test

Table 9: Shapiro-Wilk Normality Test for Log lm Model Residuals

	Statistic	p_Value
W	0.87676	0

The Shapiro–Wilk normality test concluded a *p*-value of approximately 0. This extremely small *p*-value indicates strong evidence against normality.

Based on the observation on the three assumptions tested, we conclude that the residuals are not normally distributed.

For a more holistic perspective, we continue to check the model performance using our *test* dataset.

```r
log_predictions <- predict(lm_model_log, newdata=test_data)
predictions_back <- exp(log_predictions)

rmse_log <- sqrt(mean((test_data$SalePrice - predictions_back)^2))
mae_log  <- mean(abs(test_data$SalePrice - predictions_back))
r2_log   <- cor(test_data$SalePrice, predictions_back)^2

metrics_df_log <- data.frame(
  "RMSE" = signif(rmse_log, 4),
  "MAE" = signif(mae_log, 4),
  "R-squared" = signif(r2_log, 4)
)

kable(metrics_df_log, caption = "Log lm Model Performance on Test Dataset")
```

Table 10: Log lm Model Performance on Test Dataset

RMSE	MAE	R.squared
30050	20350	0.8761

For model performance, the RMSE is 30,050, meaning predictions are off by about \$30k on average when squared errors are considered. The MAE is 20,350, so the average absolute prediction error is about \$20k. The R^2 is around 0.876, which means the model explains about 88% of the variation in sale prices.

The log-transformed model provides a clear improvement over the original linear model. After applying the log transformation to *SalePrice*, the model explains about 88% of the variation in housing prices, compared to 83% before transformation. Prediction errors also decreased, with RMSE reduced from roughly \$35k to \$30k and MAE from about \$24k to \$20k. These results indicate that the log transformation not only improved model fit but also enhanced predictive accuracy.

However, diagnostic checks still suggest some deviations from normality and potential heteroscedasticity, meaning that while performance improved, certain model assumptions remain imperfect. Overall, the log-transformed model is a stronger and more reliable model for predicting housing prices.

Regularized regression methods - Ridge and Lasso

We've found and improved the linear regression model. Nonetheless, after conducting diagnostic check on the log-transformed (improved) model, we still detects potential heteroscedasticity. This is one of the drawbacks of linear regression models while the benefits is that they are simple and highly interpretable - prone to overfitting and sometimes could have poor performance due to violation(s) in assumptions.

Regularized regression, in this case, is another approach we can try. Regularized regression adds penalty to either the sum of squared coefficients (Ridge Regression / L2-norm, where all coefficients will be shrunk towards 0) or the sum of the absolute values of the coefficients (Lasso Regression / L1-norm, where some coefficients will be forced to become 0). Specifically, it solves a penalized least squares problem:

$$min_{\beta_0,\beta}\frac{1}{2n}\|y-\beta_0 1-X\beta\|+\lambda(\alpha\|\beta\|_1+\frac{1-\alpha}{2}\|\beta\|_2^2)$$

There are several key components of the above equation that requires attention:

- X is the design matrix, which we will need to create as *glmnet* package won't create the matrix by default like what *lm()* function will do;
- y is the response vector;
- λ is a coefficient that controls the penalty strength;
- α is a definitional coefficient, where the regression is lasso when $\alpha = 1$, ridge when $\alpha = 0$, and elastic when $0 < \alpha < 1$.

With the knowledge in mind, we now proceed to find and introduce the regularized regression model using both Ridge and Lasso's method, starting by creating the model matrices.

```r
x_train <- model.matrix(LogSalePrice ~ Overall.Qual + Gr.Liv.Area + Garage.Cars +
                          Garage.Area + Total.Bsmt.SF + X1st.Flr.SF + Year.Built,
                        data = train_data)[, -1]  # remove intercept column
y_train <- train_data$LogSalePrice

# glmnet package will add the intercept column by default
# Therefore, we don't need the intercept column created by model.matrix

x_test <- model.matrix(LogSalePrice ~ Overall.Qual + Gr.Liv.Area + Garage.Cars +
                         Garage.Area + Total.Bsmt.SF + X1st.Flr.SF + Year.Built,
                       data = test_data)[, -1]  # remove intercept column
y_test <- test_data$LogSalePrice
```

We will use cross-validation to choose the best penalty parameter λ for both Ridge and Lasso regression models. Worth mentioning, we can obtain two λ values via cross-validation method, specifically *lambda.min* and *lambda.1se*. When λ = *lambda.min*, the model's predictive accuracy is maximized ("best fit") since *lambda.min* is obtained when the cross-validation error is minimized. When λ = *lambda.1se*, the model is nearly as accurate as when λ = *lambda.min*, but possibly with fewer predictors ("simpler model").

In terms of real-world housing price prediction, both *lambda.min* and *lambda.1se* are very meaningful. Particularly, the model obtained using *lambda.min* is best for prediction due to its optimized fitness. The model obtained using *lambda.1se* is best for identifying key factor(s) that influence housing prices due to its minimized amount of predictors exist. We will find both λ values for both models.

We start with finding the two λ values for Ridge regression model.

```r
set.seed(20251016)
# Ridge Regression (alpha = 0)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0, standardize = TRUE)
lambda_min_ridge <- cv_ridge$lambda.min
lambda_1se_ridge <- cv_ridge$lambda.1se

# Ridge predictions
ridge_pred <- predict(cv_ridge, s = lambda_min_ridge, newx = x_test)
ridge_rmse <- sqrt(mean((y_test - ridge_pred)^2))
ridge_mae  <- mean(abs(y_test - ridge_pred))
ridge_r2   <- cor(y_test, ridge_pred)^2

ridge_results <- data.frame(
  Model = "Ridge",
  Lambda_Min = signif(lambda_min_ridge, 4),
  Lambda_1se = signif(lambda_1se_ridge, 4),
  RMSE = signif(ridge_rmse, 4),
  MAE = signif(ridge_mae, 4),
  R2 = signif(ridge_r2, 4)
)

names(ridge_results) <- c("Model", "Lambda_Min", "Lambda_1se", "RMSE", "MAE", "R2")
kable(ridge_results)
```

Model	Lambda_Min	Lambda_1se	RMSE	MAE	R2
Ridge	0.03297	0.176	0.1939	0.1221	0.8226

The Ridge regression model selected a penalty parameter $\lambda_{min} \approx 0.03297$ and $\lambda_{1se} \approx 0.176$. The RMSE is approximately 0.1939, indicating the average squared prediction error is moderate when scaled. The MAE is 0.1221, meaning the average absolute error is slightly above 0.12 in the response scale. The R^2 is approximately 0.8226, so the Ridge model explains about 82.26% of the variation in house prices. This suggests strong predictive performance while keeping most predictors in the model.

With the λ values obtained, we can now look at the corresponding models.

```r
ridge_coef_min <- coef(cv_ridge, s="lambda.min")
ridge_coef_1se <- coef(cv_ridge, s="lambda.1se")

kable(as.matrix(ridge_coef_min), digit = 6, caption = "Ridge Regression Coefficients (Lambda_Min)")
```

Table 11: Ridge Regression Coefficients (Lambda_Min)

	lambda.min
(Intercept)	6.218260
Overall.Qual	0.100591
Gr.Liv.Area	0.000206
Garage.Cars	0.060964
Garage.Area	0.000106
Total.Bsmt.SF	0.000092
X1st.Flr.SF	0.000062

	lambda.min
Year.Built	0.002312

```r
kable(as.matrix(ridge_coef_1se), digit = 6, caption = "Ridge Regression Coefficients (Lambda_1se)")
```

Table 12: Ridge Regression Coefficients (Lambda_1se)

	lambda.1se
(Intercept)	6.845501
Overall.Qual	0.077077
Gr.Liv.Area	0.000166
Garage.Cars	0.062400
Garage.Area	0.000163
Total.Bsmt.SF	0.000095
X1st.Flr.SF	0.000085
Year.Built	0.002066

We proceed to find the two λ values for Lasso regression model.

```r
set.seed(20251016)
# Lasso Regression (alpha = 1)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, standardize = TRUE)
lambda_min_lasso <- cv_lasso$lambda.min
lambda_1se_lasso <- cv_lasso$lambda.1se

# Lasso predictions
lasso_pred <- predict(cv_lasso, s = lambda_min_lasso, newx = x_test)
lasso_rmse <- sqrt(mean((y_test - lasso_pred)^2))
lasso_mae  <- mean(abs(y_test - lasso_pred))
lasso_r2   <- cor(y_test, lasso_pred)^2

lasso_results <- data.frame(
  Model = "Lasso",
  Lambda_Min = signif(lambda_min_lasso, 4),
  Lambda_1se = signif(lambda_1se_lasso, 4),
  RMSE = signif(lasso_rmse, 4),
  MAE = signif(lasso_mae, 4),
  R2 = signif(lasso_r2, 4)
)

names(lasso_results) <- c("Model", "Lambda_Min", "Lambda_1se", "RMSE", "MAE", "R2")
kable(lasso_results)
```

Model	Lambda_Min	Lambda_1se	RMSE	MAE	R2
Lasso	0.001801	0.0388	0.1916	0.1219	0.8235

The Lasso regression model selected a penalty parameter $\lambda_{min} \approx 0.001801$ and $\lambda_{1se} \approx 0.0388$. The RMSE is approximately 0.1916 and the MAE is approximately 0.1219, very similar to Ridge regression. The R^2 is 0.8235, meaning the model explains about 82.35% of the variation.

With the λ values obtained, we can now look at the corresponding models.

```
lasso_coef_min <- coef(cv_lasso, s="lambda.min")
lasso_coef_1se <- coef(cv_lasso, s="lambda.1se")

kable(as.matrix(lasso_coef_min), digit = 6, caption = "Lasso Regression Coefficients (Lambda_Min)")
```

Table 13: Lasso Regression Coefficients (Lambda_Min)

	lambda.min
(Intercept)	6.192871
Overall.Qual	0.110190
Gr.Liv.Area	0.000217
Garage.Cars	0.063895
Garage.Area	0.000062
Total.Bsmt.SF	0.000093
X1st.Flr.SF	0.000046
Year.Built	0.002303

```
kable(as.matrix(lasso_coef_1se), digit = 6, caption = "Lasso Regression Coefficients (Lambda_1se)")
```

Table 14: Lasso Regression Coefficients (Lambda_1se)

	lambda.1se
(Intercept)	7.772332
Overall.Qual	0.111707
Gr.Liv.Area	0.000179
Garage.Cars	0.060982
Garage.Area	0.000026
Total.Bsmt.SF	0.000073
X1st.Flr.SF	0.000026
Year.Built	0.001561

The comparison of Ridge and Lasso regression shows that both models achieve very similar predictive performance, with RMSE values around 0.19, MAE values near 0.12, and R^2 values slightly over 0.82. This indicates that both approaches explain more than 82% of the variation in housing prices, which demonstrates strong predictive accuracy. Ridge regression, on the one hand, emphasize stability by retaining all predictors in the model and shrinking them towards zero. Lasso regression, on the other hand, has the additional benefit of shrinking some coefficients to exactly zero, which improves interpretability by placing greater emphasis on the most influential predictors.

In the context of real estate, these results suggest that while many housing features contribute to price, only a subset (such as quality, living area, and year built) are consistently influential. Ridge is useful when the goal is to leverage all available predictors for robust predictions, whereas Lasso is more practical when stakeholders, such as buyers, sellers, or policymakers, need clear identification of the key drivers of housing prices.

Bayesian Linear Regression

The two general types of regression models we've derived, frequentist and regularized, mainly provides point estimates $\hat{\beta}$ and standard errors. Although not as commonly analyzed, the uncertainty of these two regression models are summarized by confidence intervals.

Another approach to analyze and therefore control the uncertainty is, instead of confidence intervals, credible interval. Bayesian regression is a type of regression that analyze the estimators with credible intervals. It starts with a prior on β, combines it with the likelihood from the dataset, and returns a posterior distribution for each β. We can then summarize the distriution by its mean, standard deviation, and credible interval.

We will then approach the question of interest with Bayesian regression methods, starting with defining the Bayesian regression formula using the *bf()* function, specifying the priors, and fit the model with the *brm()* function .

We defined the Bayesian regression formula using the *LogSalePrice* response variable as the linearity, variance, and possible risk for skewness has been improved compared to the non-transformed model. Additionally, before fitting the model with the *brm()* function, we picked the priors deliberately.

- For the slope coefficient, we set the prior to be *normal(0,1)*. This keeps the coefficients near 0 unless data support larger effects.
- For the intercept, we set the prior to be *normal(0,5)*. This is because on a general level, the log-price value can vary more. We certainly can set the standard deviation value of the normal distribution to a smaller value, but it is not very meaningful nor pragmatic.
- For the residual standard error, we set the prior to be *student_t(3, 0, 2.5)*. The student's t-distribution function helps catering the model to heavier tails and outliers while centering around a certain distribution.

We then proceed to derive and observe the posterior summaries and diagnostics of the model, setting the credible interval to be 95%.

```r
## Coefficients (posterior means and 95% credible intervals)
fixef_tbl <- as.data.frame(fixef(bayes_mod, probs = c(.025, .975)))
fixef_tbl$Term <- rownames(fixef_tbl)
fixef_tbl <- fixef_tbl[, c("Term", "Estimate", "Est.Error", "Q2.5", "Q97.5")]
kable(fixef_tbl, digits = 4, caption = "Bayesian Coefficients (posterior means & 95% CrI)")
```

Table 15: Bayesian Coefficients (posterior means & 95% CrI)

	Term	Estimate	Est.Error	Q2.5	Q97.5
Intercept	Intercept	6.1139	0.2823	5.5563	6.6664
Overall.Qual	Overall.Qual	0.1101	0.0038	0.1023	0.1172
Gr.Liv.Area	Gr.Liv.Area	0.0002	0.0000	0.0002	0.0002
Garage.Cars	Garage.Cars	0.0638	0.0112	0.0426	0.0860
Garage.Area	Garage.Area	0.0001	0.0000	0.0000	0.0001
Total.Bsmt.SF	Total.Bsmt.SF	0.0001	0.0000	0.0001	0.0001
X1st.Flr.SF	X1st.Flr.SF	0.0000	0.0000	0.0000	0.0001
Year.Built	Year.Built	0.0023	0.0001	0.0021	0.0026

The table shows the posterior mean estimates of each predictor along with their credible intervals.

- **Overall.Qual (0.1101, CrI [0.1023, 0.1172]**) and **Gr.Liv.Area (0.0002, CrI [0.0002, 0.0002])** stand out with very tight intervals, suggesting strong, precise positive associations with housing prices.
- **Garage.Cars** also has a meaningful positive effect **(0.0638, CrI [0.0426, 0.0860]**).
- Other predictors, like *Garage.Area* and *Total.Bsmt.SF*, have near-zero effects with narrow *CrIs*, suggesting little additional explanatory power once other predictors are included.

This indicates that house quality, living area, and car capacity in the garage are the strongest predictors in the Bayesian model.

We will now conduct two diagnostics check on the Bayesian model, specifically on the model convergence and the posterior predictive ability. We start with checking the model convergence.

```r
## Overall fit
rhat_vals  <- rhat(bayes_mod)
neff_vals  <- neff_ratio(bayes_mod)
r2_bayes   <- bayes_R2(bayes_mod)         # Bayesian R^2 (posterior mean & CrI)

kable(data.frame(
  Metric = c("max R-hat", "median n_eff ratio", "Bayesian R^2 (mean)"),
  Value  = c(signif(max(rhat_vals), 4),
             signif(median(neff_vals), 4),
             signif(mean(r2_bayes), 4))
), caption = "Convergence & Fit")
```

Table 16: Convergence & Fit

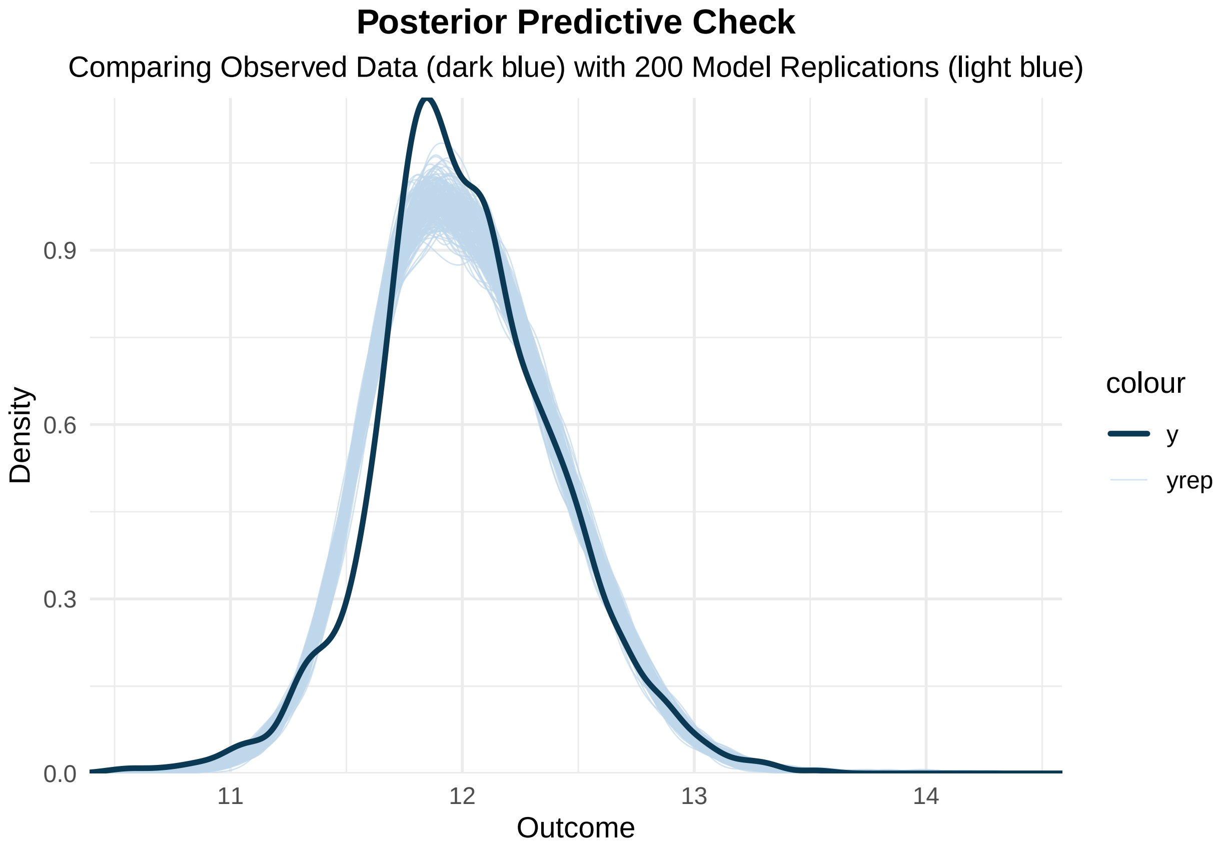
Metric	Value
max R-hat	1.0110
median n_eff ratio	0.2758
Bayesian R^2 (mean)	0.6188

The convergence diagnostics are acceptable.

- The **R-hat** is 1.0110, close to 1, suggesting chains have mostly converged, though slightly above the ideal value.
- The **n_eff ratio** is low (0.2758), which could indicate some inefficiency in sampling and suggests that more iterations or tuning might be needed for stability.
- The **Bayesian** R^2 is 0.6188, meaning the Bayesian model explains about 61% of the variation in log-transformed housing prices. This is lower than the frequentist and regularized regression models, which were closer to 82-89%.

We then proceed to conduct the posterior predictive check.

```r
pp_check(bayes_mod, ndraws = 200) +
  scale_color_manual(values = c("y" = "#0b3954", "yrep" = "#bfd7ea")) +
  scale_fill_manual(values = c("y" = "#0b3954", "yrep" = "#bfd7ea")) +
  theme_minimal() +
  labs(
    title = "Posterior Predictive Check",
    subtitle = "Comparing Observed Data (dark blue) with 200 Model Replications (light blue)",
    x = "Outcome",
    y = "Density"
  ) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5)
  )
```



The posterior predictive density closely matches the observed density of sale prices (on the log scale). The overlap between observed (dark line) and replicated data (light blue lines) suggests that the Bayesian model captures the overall distribution well, though there may be slight underfitting in the tails (especially the lower tail).

At the end, we are also interested in examining the model's performance, based on benchmarks we have used before, such as RMSE, MAE, and R-squared.

```r
## Posterior predictive means on the log scale
pred_log_draws <- posterior_epred(bayes_mod, newdata = test_data)  # (draws x n)
pred_log_mean  <- colMeans(pred_log_draws)

## Back-transform to original dollars
pred_dollar <- exp(pred_log_mean)

## Metrics on original scale
rmse_brm <- sqrt(mean((test_data$SalePrice - pred_dollar)^2))
mae_brm  <- mean(abs(test_data$SalePrice - pred_dollar))
r2_brm   <- cor(test_data$SalePrice, pred_dollar)^2

kable(data.frame(
  RMSE = signif(rmse_brm, 4),
  MAE  = signif(mae_brm, 4),
  R2   = signif(r2_brm, 4)
), caption = "Bayesian model performance on test set (back-transformed)")
```

Table 17: Bayesian model performance on test set (back-transformed)

RMSE	MAE	R2
30040	20350	0.8761

When transformed back to the dollar scale:

- **RMSE** is 30,040, indicating predictions are off by around $30k on average (in squared error sense).
- **MAE** is 20,350, meaning average absolute error is about $20k.
- R^2 is 0.8761, showing the model explains about 87% of variation in housing prices on the original scale.

This performance is consistent with the best results from frequentist and regularized regressions, showing the Bayesian model is competitive in predictive accuracy.

The Bayesian regression confirmed that house quality, living area, and garage capacity are the most influential predictors of housing prices, with narrow credible intervals reflecting high certainty in their effects. The posterior predictive checks show that the model replicates the observed data distribution well, particularly after log transformation.

Although the Bayesian R^2 on the log scale was lower ($\approx$ 62%), the back-transformed performance metrics (RMSE $\approx$ \$30.0k, MAE $\approx$ \$20.3k, R^2 $\approx$ 0.87%) demonstrate strong predictive accuracy comparable to frequentist and penalized regression models.

From a real-world perspective, the Bayesian approach adds interpretability by quantifying uncertainty through credible intervals, which can be particularly useful for housing market stakeholders (e.g., policymakers, business-level buyers, personal-level buyers, etc). Particularly, by conducting comparative analysis using the upper and lower bounds of the coefficients' credible intervals, the stakeholders are able to obtain a general view of the best and worst outcome of their investments. This allows for not only point predictions but also an assessment of confidence in those predictions, making it a powerful complement to traditional methods.

Machine Learning - Random Forest

While our previous models, including the frequentist, regularized, and Bayesian regressions, relied on explicit assumptions about linear relationships and error distributions, machine learning methods like Random Forests take a fundamentally different approach. Instead of estimating coefficients for predictors, Random Forests use an ensemble of decision trees to learn complex, nonlinear relationships between predictors and the target variable.

A Random Forest model works by constructing many individual trees on bootstrapped samples of the data, each using a random subset of predictors at each split. The final prediction is then obtained by averaging the predictions from all trees, effectively reducing variance and improving robustness. This process not only captures nonlinear interactions but also minimizes overfitting compared to a single decision tree.

In this section, we will apply the Random Forest algorithm to predict housing prices using the same predictors as before. We also evaluate its performance through both out-of-bag (OOB) error and held-out test data, while examining the relative importance of each predictor variable. The results will help us understand how well a tree-based model performs compared to the regression models previously discussed, particularly in its ability to model complex relationships and interactions.

```r
set.seed(20251028)

vars <- c("SalePrice","Overall.Qual","Gr.Liv.Area","Garage.Cars",
          "Garage.Area","Total.Bsmt.SF","X1st.Flr.SF","Year.Built")

rf_train <- na.omit(train_data[, vars])
rf_test  <- na.omit(test_data[,  vars])

p <- ncol(rf_train) - 1L
rf_model <- ranger(
  dependent.variable.name = "SalePrice",
  data       = rf_train,
  num.trees  = 1000,
  mtry       = max(1L, floor(sqrt(p))),
  min.node.size = 5,
  sample.fraction = 0.8,
  importance = "permutation",
  seed       = 20250829
)

oob_rmse <- sqrt(rf_model$prediction.error)

rf_pred <- predict(rf_model, data = rf_test)$predictions

rf_rmse <- sqrt(mean((rf_test$SalePrice - rf_pred)^2))
rf_mae  <- mean(abs(rf_test$SalePrice - rf_pred))
rf_r2   <- cor(rf_test$SalePrice, rf_pred)^2

kable(data.frame(
  Model     = "Random Forest (baseline)",
  OOB_RMSE  = signif(oob_rmse, 4),
  Test_RMSE = signif(rf_rmse, 4),
  Test_MAE  = signif(rf_mae, 4),
  Test_R2   = signif(rf_r2, 4)
), caption = "Random Forest performance (OOB and held-out test)")
```

Table 18: Random Forest performance (OOB and held-out test)

Model	OOB_RMSE	Test_RMSE	Test_MAE	Test_R2
Random Forest (baseline)	28720	28100	18950	0.8893

The Random Forest performance table provides an overview for the model's prediction ability.

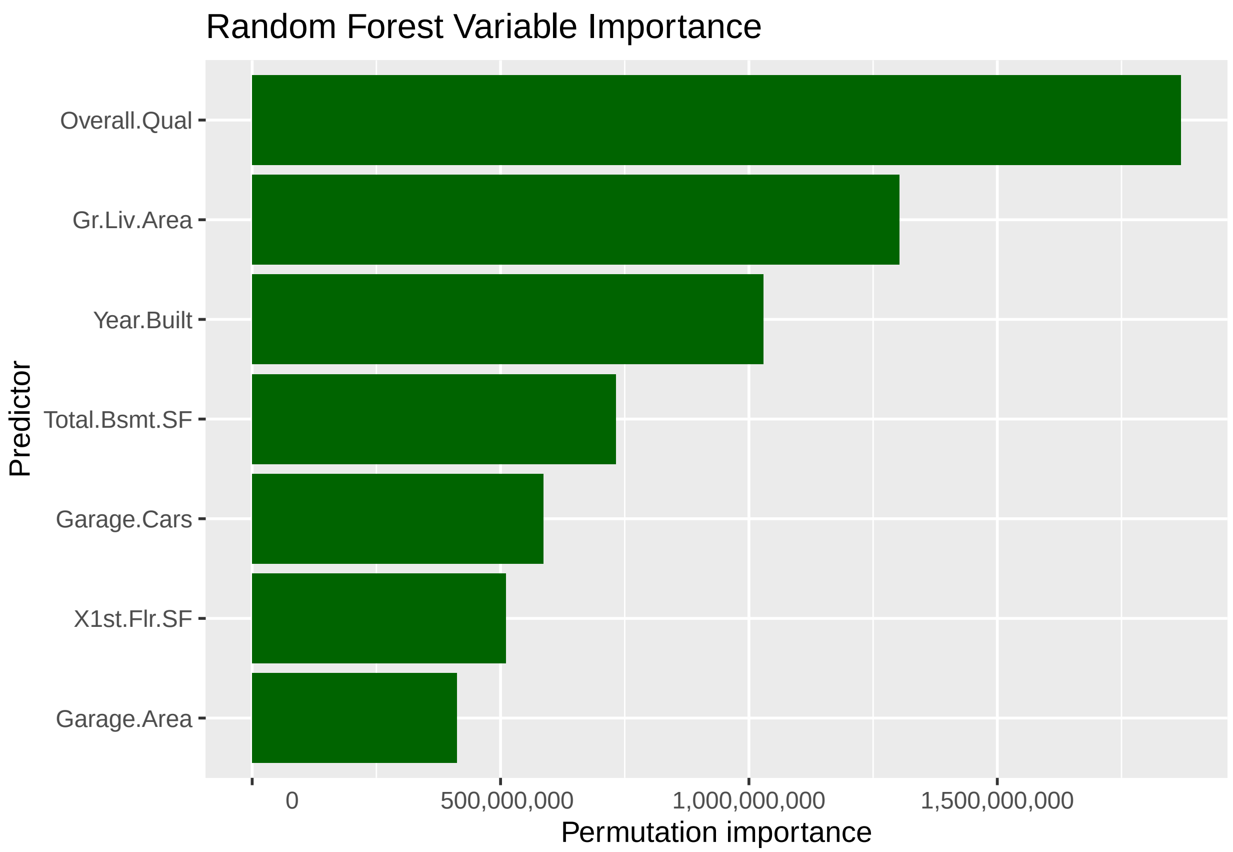
- The **OOB RMSE** is 28,720. This means that when the Random Forest predicts house prices on training data it has not seen (due to its bootstrap sampling), the average prediction error is about $28,720. In other words, the model's typical squared-error-based miss is around $29k on houses during training validation.
- The **Test RMSE** is 28,100, fairly close to the **OOB RMSE**. This number means that when predicting on the test dataset, the average squared-error miss is about $28,100. The closeness between OOB and Test RMSE shows that the Random Forest is not overfitting and its performance is consistent both internally and externally.
- The **Test MAE** is 18,950, meaning that on average, the Random Forest is off by about $18,950 per house when predicting sale prices. This number directly tells us how much the model's predictions deviate from actual prices in dollars.

- The **Test** R^2 is 0.8893, meaning the model explains about 89% of the variation in sale prices. It suggest that almost 9 out of 10 dollars of variability in housing prices can be explained by the predictors we used.

In real-world housing price prediction, being able to predict sale prices within about $19k on average is a very useful result. While not perfect, the model captures almost all the main drivers of house prices and produces realistic predictions. It suggests Random Forest captures hidden patterns in the data (such as interactions between square footage, build year, and garage features) that linear methods might miss.

We then move forward to examine the coefficients importance derived from the random forest model. It is worth mentioning that, unlike traditional models such as linear regression or Bayesian regression, random forest does not produce exact coefficients. Instead, it produce variable importance score for each predictor based on how influential each predictor was in improving model accuracy across the trees.

```r
options(scipen = 999)

imp_df <- data.frame(
  Variable    = names(rf_model$variable.importance),
  Importance  = as.numeric(rf_model$variable.importance)
)

ggplot(imp_df, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_col(fill = "darkgreen") +
  coord_flip() +
  labs(
    title = "Random Forest Variable Importance",
    x     = "Predictor",
    y     = "Permutation importance") +
  scale_y_continuous(
    labels = function(x) format(x, big.mark = ",", scientific = FALSE)
  )
```



The variable importance figure conveys meaningful information about the predictors.

- **Overall.Qual** (quality rating) is the most influential feature. Its importance value is far higher than any other predictor, showing that a home's construction and finish quality are the single strongest factor in determining price.
- **Gr.Liv.Area** (above-ground living area) is the second most important. This aligns with the idea that larger houses typically sell for more.
- **Year.Built** comes next, showing that newer homes tend to command higher prices, reflecting both modern amenities and less wear.
- **Total.Bsmt.SF** (basement size) and **X1st.Flr.SF** (first floor size) also matter, though less than quality and overall living area. These features still contribute significantly to how buyers value houses.
- **Garage.Cars** (number of cars the garage fits) and **Garage.Area** (garage square footage) are the least important among our predictors, but they still play a role. Buyers do value garage space, but not nearly as much as size and quality.

This ranking makes intuitive sense. Buyers are most influenced by the quality and size of the living space, while secondary factors like garage space add value but are not decisive.

The Random Forest model predicts house prices with strong accuracy. On average, predictions are off by about $18,950 per house, and the model captures about 89% of the variation in prices. This is better performance than the linear models we saw earlier.

In practical housing terms, the model confirms what we expect: quality, size, and age of the house are the biggest price drivers, while features like garages still matter but less so. This shows that Random Forest not only predicts prices well but also provides insights into what matters most to buyers in real-world markets.

Machine Learning - Gradient Boosting (XGBoost)

Following the Random Forest model, which builds multiple independent trees and averages their results, we now turn to another powerful technique: Gradient Boosting. Unlike Random Forests that reduce variance through parallel averaging, Gradient Boosting minimizes bias by sequentially improving weak models in a sense that each new tree is trained to correct the errors made by the previous ones.

One of the most widely used implementations of Gradient Boosting is XGBoost (Extreme Gradient Boosting). Known for its efficiency, scalability, and strong predictive performance, XGBoost incorporates advanced regularization techniques, optimized tree construction, and parallel computation, making it a standard tool in modern data science industries and real-world predictive tasks.

In this section, we apply XGBoost to predict housing prices using the same predictors as before. Unlike *lm* or *ranger*, XGBoost expects a matrix (not data frame). Therefore, we first transform the data structure of the train data to matrix.

```r
predictors <- c("Overall.Qual", "Gr.Liv.Area", "Garage.Cars",
                "Garage.Area", "Total.Bsmt.SF", "X1st.Flr.SF", "Year.Built")

x_train <- as.matrix(train_data[, predictors])
y_train <- train_data$SalePrice

x_test  <- as.matrix(test_data[, predictors])
y_test  <- test_data$SalePrice

dtrain <- xgb.DMatrix(data = x_train, label = y_train)
dtest  <- xgb.DMatrix(data = x_test,  label = y_test)
```

Now that we've set up the train and test data set for the gradient boosting model, we move forward to fit the model with the training dataset and evaluate its performance based on the test dataset.

```r
set.seed(20251112)

xgb_model <- xgboost(
  data = dtrain,
  nrounds = 500,
  max_depth = 6,
  eta = 0.05,
  subsample = 0.8,
  colsample_bytree = 0.8,
  objective = "reg:squarederror",
  verbose = 0
)

xgb_pred <- predict(xgb_model, newdata = dtest)

xgb_rmse <- sqrt(mean((y_test - xgb_pred)^2))
xgb_mae  <- mean(abs(y_test - xgb_pred))
xgb_r2   <- cor(y_test, xgb_pred)^2

xgb_results <- data.frame(
  Model = "XGBoost",
  RMSE = signif(xgb_rmse, 4),
  MAE = signif(xgb_mae, 4),
  R2 = signif(xgb_r2, 4)
)

names(xgb_results) <- c("Model", "RMSE", "MAE", "R2")
kable(xgb_results)
```

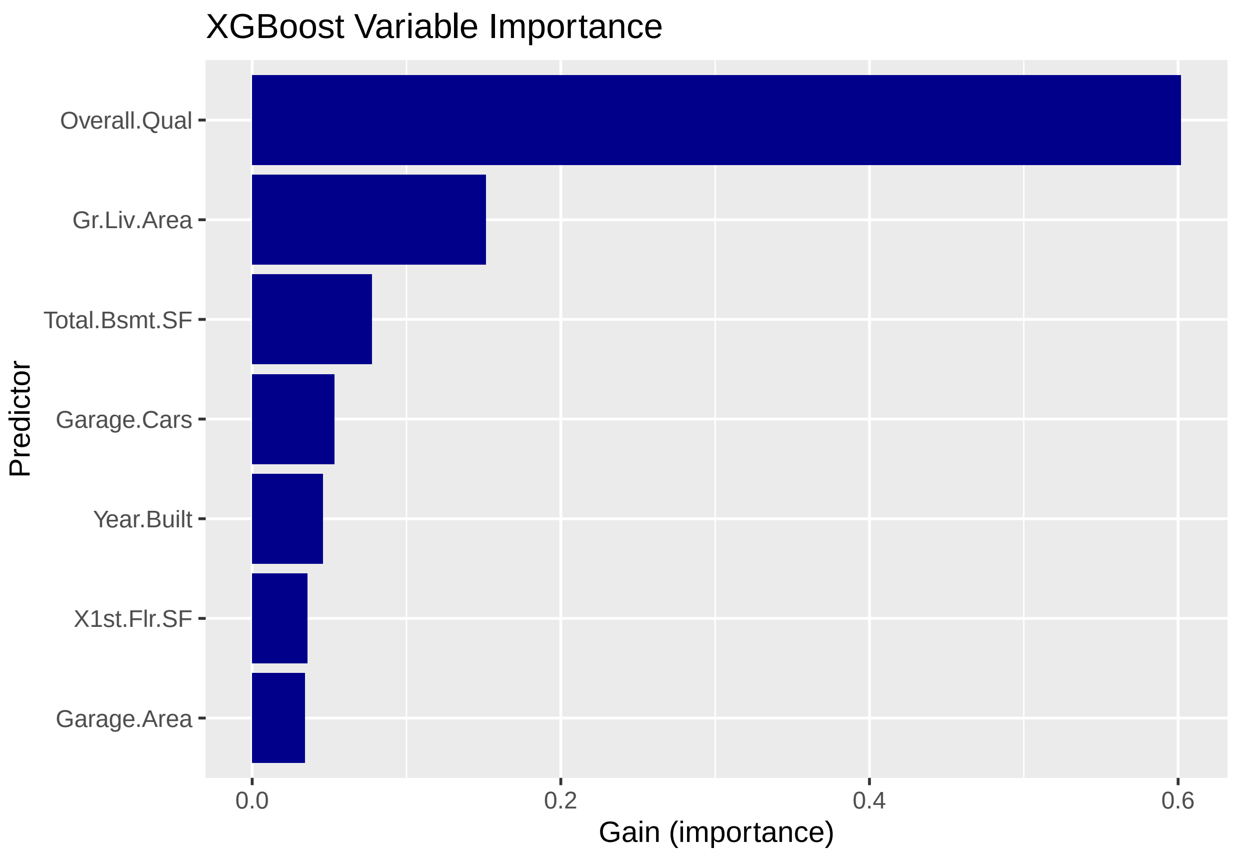
Model	RMSE	MAE	R2
XGBoost	27850	19090	0.8863

The XGBoost performance table provides an overview for the model's prediction ability.

- The **RMSE** is 27,850, which means on average, the squared prediction errors translate to about $27.9k difference between predicted and actual house prices.
- The **MAE** is 19,090, showing that on average, the model's predictions differ from the actual house prices by about $19k.
- The R^2 is 0.8863, meaning the model explains about 89% of the variation in housing prices in the test set. This is a high proportion, showing that the chosen predictors capture most of the important variation in sale prices.

Similar to random forest, XGBoost produce variable importance score for each predictor based on how influential each predictor was in relation to the response variable.

```
importance_matrix <- xgb.importance(model = xgb_model)

imp_df <- data.frame(
  Variable = importance_matrix$Feature,
  Importance = importance_matrix$Gain
)

ggplot(imp_df, aes(x = reorder(Variable, Importance), y = Importance)) +
  geom_col(fill = "darkblue") +
  coord_flip() +
  labs(
    title = "XGBoost Variable Importance",
    x = "Predictor",
    y = "Gain (importance)"
)
```



The variable importance plot highlights which predictors contributed the most to reducing prediction error in the XGBoost model.

- **Overall.Qual** is by far the most important predictor, accounting for over half of the explanatory power. This makes sense as better-built houses with higher quality scores tend to sell for more.
- **Gr.Liv.Area** is the second most important factor, reflecting the common relationship that larger houses command higher prices.
- **Total.Bsmt.SF** and **Garage.Cars** also play substantial roles, suggesting that newer homes and those with more basement garage capacity tend to be valued more highly.
- **Year.Built** and **X1st.Flr.SF** add additional predictive power, while **Garage. Area** has the least relative importance, though it still contributes.

This ranking shows that quality and size are the dominant drivers of price, while features like garage size or basement area matter, but to a smaller degree.

The XGBoost model delivered strong predictive performance, with an RMSE of about $27.9k, an MAE of $19k, and an R^2 of nearly 0.89, showing that it explains most of the variation in house prices. The models's error are slightly larger than the random forest model's performance, but both model achieved comparable explainatory power.

From the variable importance results, overall quality and living area size clearly dominate as the strongest predictors of sale price, while features like construction year, garage capacity, and basement size also contribute meaningfully. This aligns well with real-world housing markets, where buyers pay premiums for high-quality, larger, and newer homes.

Overall, the XGBoost stage again confirms the value of advanced machine learning models in housing price prediction: they not only improve accuracy but also provide insights into which features matter most.

Discussion

As we have now completed the implementation and evaluation of all five modeling approaches, including classical linear regression, regularized regression, Bayesian regression, random forest, and XGBoost, we move to a broader examination of how these models compare. While each method was analyzed individually in earlier sections, the discussion section provides an opportunity to step back and evaluate the models collectively, specifically on their predictive accuracy, interpretability, and practical relevance for housing price prediction.

Cross-Model Comparison

To understand the relative strengths and weaknesses of each method, we summarize and compare their predictive performance on the same test set. The comparison table below highlights key performance metrics, including RMSE, MAE, and R^2, providing a holistic view of how linear, Bayesian, and machine learning models differ in accuracy and stability in this case.

```r
model_compare <- data.frame(
  Model = c(
    "Linear (log, back-transformed)",
    "Ridge",
    "Lasso",
    "Bayesian (log, back-transformed)",
    "Random Forest",
    "XGBoost"
  ),
  RMSE = signif(c(
    rmse_log,
    ridge_rmse,
    lasso_rmse,
    rmse_brm,
    rf_rmse,
    xgb_rmse
  ), 4),
  MAE = signif(c(
    mae_log,
    ridge_mae,
    lasso_mae,
    mae_brm,
    rf_mae,
    xgb_mae
  ), 4),
  R2 = signif(c(
    r2_log,
    ridge_r2,
    lasso_r2,
    r2_brm,
    rf_r2,
    xgb_r2
  ), 4)
)

kable(
  model_compare,
  caption = "Comparison of model performance on the test set (RMSE, MAE, R-squared)"
)
```

Table 19: Comparison of model performance on the test set (RMSE, MAE, R-squared)

Model	RMSE	MAE	R2
Linear (log, back-transformed)	3.005e+04	2.035e+04	0.8761
Ridge	1.939e-01	1.221e-01	0.8226
Lasso	1.916e-01	1.219e-01	0.8235
Bayesian (log, back-transformed)	3.004e+04	2.035e+04	0.8761
Random Forest	2.810e+04	1.895e+04	0.8893
XGBoost	2.785e+04	1.909e+04	0.8863

The comparison shows clear performance distinctions among the models. The tree-based machine learning methods, Random Forest and XGBoost, achieve the strongest predictive power as they achieve the highest R^2. This indicate their strong ability to capture complex nonlinear interactions present in housing data.

The log-transformed linear and Bayesian models perform moderately well but are slightly behind the tree-based machine learning methods in terms of predictive power, reflecting the limitations of linear structure in a dataset with nonlinear relationships.

Ridge and Lasso exhibit lower predictive power, suggesting that while regularization improves stability, these models remain constrained by linear assumptions. Overall, the comparison shows that methods capable of modeling nonlinearities and interactions provide significant advantages in predicting housing prices in the *Ames dataset*.

Having compared the overall predictive performance of all six models, we now turn our attention to the mechanisms that drive each model's behavior. Performance metrics alone do not explain why models differ in accuracy or which predictors exert the most influence. To address this, the we move on to compare how different modeling methods estimate the effects of key housing variables.

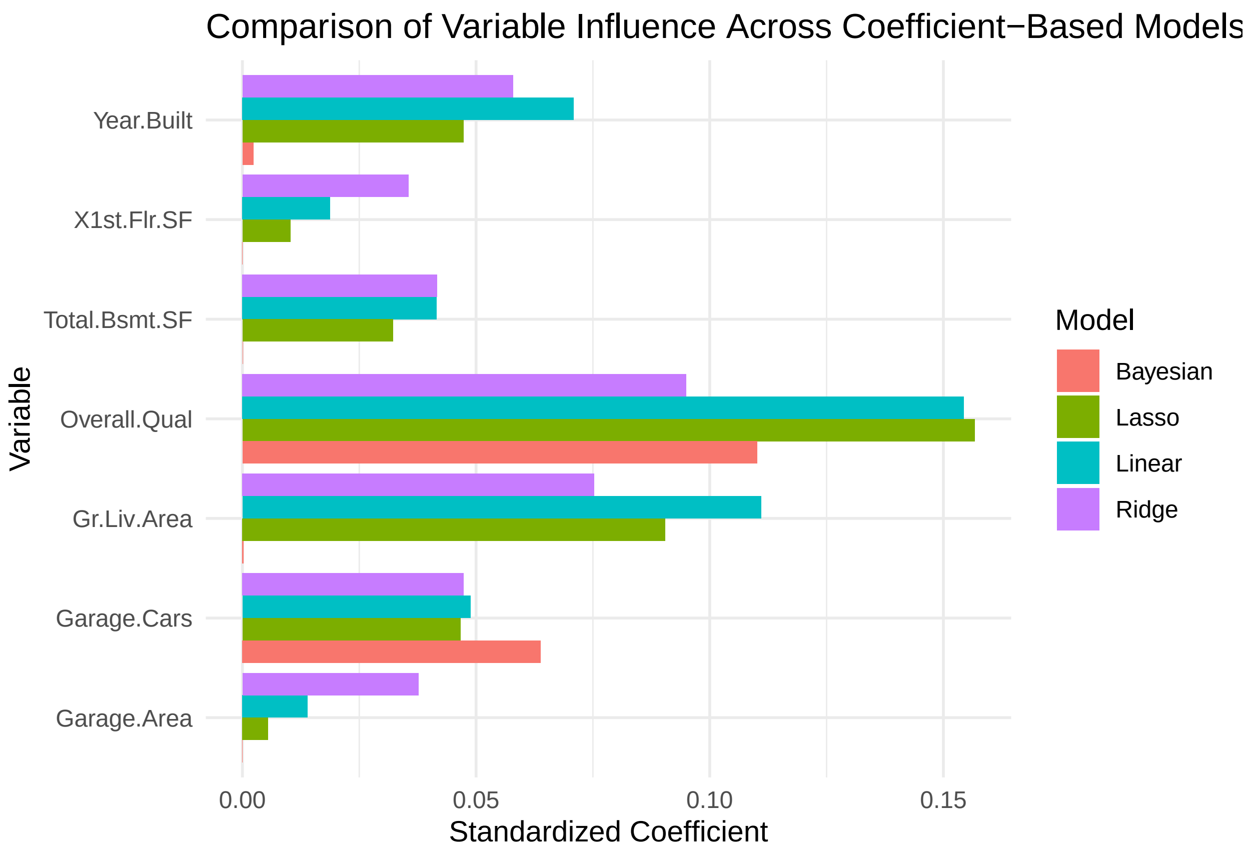
```
coef_vars <- c("Overall.Qual", "Gr.Liv.Area", "Garage.Cars",
               "Garage.Area", "Total.Bsmt.SF", "X1st.Flr.SF", "Year.Built")

train_std <- train_data %>%
  mutate(across(all_of(coef_vars), scale))

lm_std  <- lm(LogSalePrice ~ ., data = train_std[, c("LogSalePrice", coef_vars)])
ridge_std <- cv.glmnet(as.matrix(train_std[, coef_vars]),
                       train_std$LogSalePrice,
                       alpha = 0)
lasso_std <- cv.glmnet(as.matrix(train_std[, coef_vars]),
                       train_std$LogSalePrice,
                       alpha = 1)

lm_coef <- coef(lm_std)[-1]
ridge_coef <- as.vector(coef(ridge_std, s = ridge_std$lambda.1se))[-1]
lasso_coef <- as.vector(coef(lasso_std, s = lasso_std$lambda.1se))[-1]
bayes_coef <- fixef(bayes_mod)[coef_vars, "Estimate"]

coef_compare <- data.frame(
  Variable = coef_vars,
  Linear = lm_coef,
  Ridge = ridge_coef,
  Lasso = lasso_coef,
  Bayesian = bayes_coef,
  row.names = NULL
)

kable(coef_compare, digits = 4,
      caption = "Comparison of coefficient estimates across models")
```

Table 20: Comparison of coefficient estimates across models

Variable	Linear	Ridge	Lasso	Bayesian
Overall.Qual	0.1544	0.0950	0.1567	0.1101
Gr.Liv.Area	0.1110	0.0753	0.0905	0.0002
Garage.Cars	0.0488	0.0474	0.0467	0.0638
Garage.Area	0.0139	0.0376	0.0055	0.0001
Total.Bsmt.SF	0.0415	0.0417	0.0322	0.0001
X1st.Flr.SF	0.0188	0.0355	0.0102	0.0000
Year.Built	0.0709	0.0579	0.0473	0.0023

The coefficient table shows that **Overall.Qual** consistently has the strongest positive association with sale price across all four regression models. Ridge and lasso slightly shrink coefficients compared to the standard linear model, while the Bayesian model produces more conservative estimates overall. Despite these differences, all models agree on the relative ordering of influential predictors.

```
coef_long <- coef_compare %>%
  pivot_longer(cols = -Variable, names_to = "Model", values_to = "Coefficient")

ggplot(coef_long, aes(x = Variable, y = Coefficient, fill = Model)) +
  geom_col(position = "dodge") +
  coord_flip() +
  labs(title = "Comparison of Variable Influence Across Coefficient-Based Models",
       y = "Standardized Coefficient") +
  theme_minimal()
```



Across all coefficient-based models, we observed a consistent pattern that **Overall.Qual** and **Gr.Liv.Area** are the most reliable predictors of sale prices, followed by **Garage.Cars**, **Total.Bsmt.SF**, and **Year.Built**. Although each method adjusts coefficient magnitudes according to its constraints (i.e. penalties for ridge and lasso, prior structure for Bayesian), the core conclusions remain unchanged. This indicates that the main drivers of housing prices are stable across modeling approaches, and differences between models lie more in regularization behavior than in fundamental interpretation.

While coefficient-based models provide insight into how individual predictors influence sale prices under linear assumptions, tree-based methods approach the prediction task from a fundamentally different perspective. Instead of estimating slopes, Random Forest and XGBoost evaluate variables through their contribution to reducing prediction error in a nonlinear, interaction-based environment.

Because these models do not rely on coefficient estimates, comparing variable importance offers a parallel yet equally informative lens for understanding which housing features matter most. We now delve into comparing variable importance for the tree-based methods.

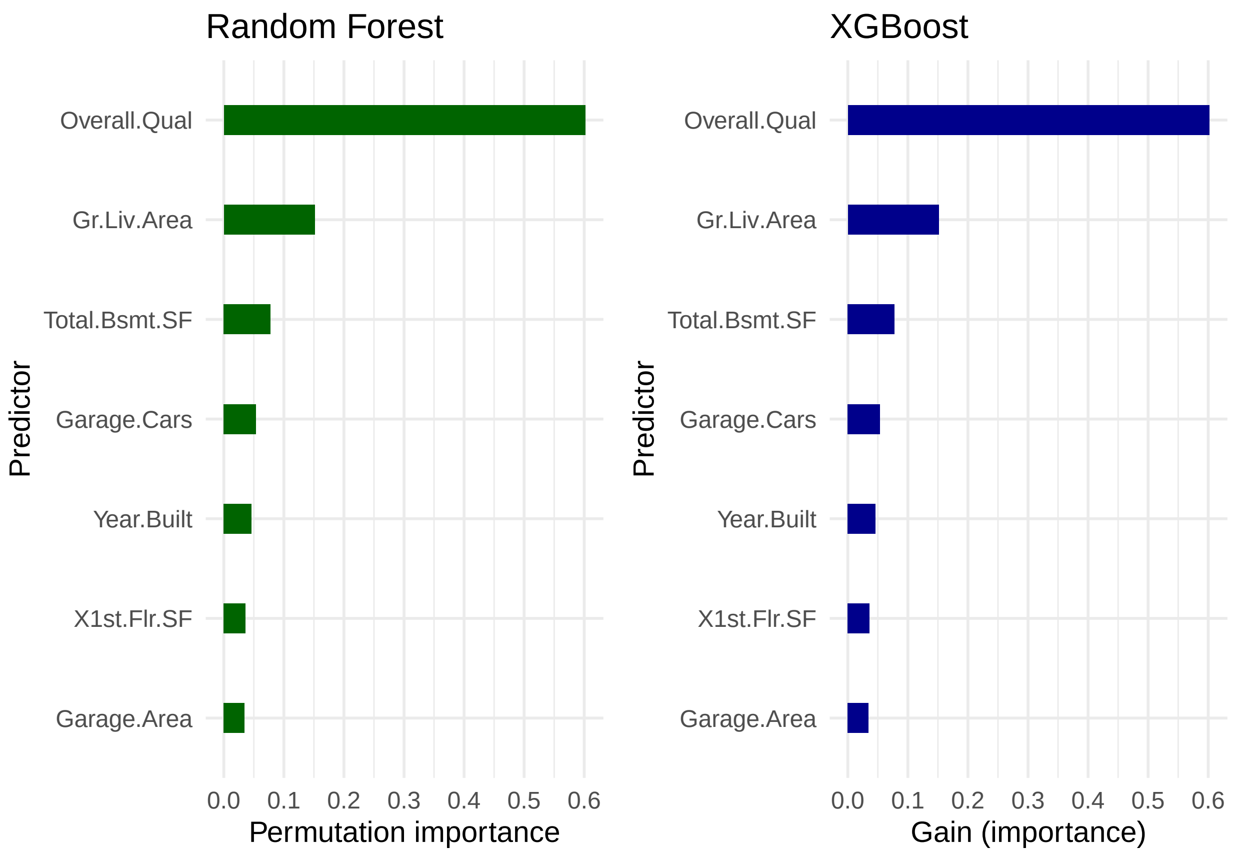
```r
# Variable Coefficients
rf_imp <- data.frame(
  Variable   = names(rf_model$variable.importance),
  RF_Importance = as.numeric(rf_model$variable.importance)
)

xgb_imp_raw <- xgb.importance(model = xgb_model)
xgb_imp <- data.frame(
  Variable = xgb_imp_raw$Feature,
  XGB_Importance = xgb_imp_raw$Gain
)
```

```r
importance_compare <- merge(rf_imp, xgb_imp, by = "Variable", all = TRUE)

kable(importance_compare, digits = 4,
      caption = "Comparison of Variable Importance: Random Forest vs. XGBoost")
```

Table 21: Comparison of Variable Importance: Random Forest vs. XGBoost

Variable	RF_Importance	XGB_Importance
Garage.Area	411990634	0.0342
Garage.Cars	586694049	0.0534
Gr.Liv.Area	1303377055	0.1514
Overall.Qual	1869680199	0.6018
Total.Bsmt.SF	731931788	0.0775
X1st.Flr.SF	510941634	0.0359
Year.Built	1029563001	0.0458

```r
# Variable Importance
plot1 <- ggplot(imp_df, aes(x = reorder(Variable, Importance), y = Importance)) +
                geom_col(fill = "darkgreen", width = 0.3) +
                scale_y_continuous(breaks = seq(0, 1, by = 0.1)) +
                coord_flip() +
                labs(title = "Random Forest",
                     x = "Predictor", y = "Permutation importance") +
                theme_minimal()

plot2 <- ggplot(imp_df, aes(x = reorder(Variable, Importance), y = Importance)) +
                geom_col(fill = "darkblue", width = 0.3) +
                scale_y_continuous(breaks = seq(0, 1, by = 0.1)) +
                coord_flip() +
                labs(
                  title = "XGBoost ",
                  x = "Predictor", y = "Gain (importance)") +
                theme_minimal()

grid.arrange(plot1, plot2, ncol = 2)
```



The Random Forest model identifies **Overall.Qual** as by far the strongest predictor of housing prices, with substantially higher permutation importance than all other variables. The next most influential predictors, **Gr.Liv.Area** and **Total.Bsmt.SF**, show meaningful but much smaller contributions. Variables such as **Garage.Cars**, **Year.Built**, and **Garage.Area** have modest influence, showing that Random Forest places most predictive weight on quality and overall living area.

XGBoost displays a similar ranking of predictors, again placing **Overall.Qual** as the dominant feature with particularly high gain values. **Gr.Liv.Area** appears to be the second-most influential variable, followed by **Total.Bsmt.SF** and **Garage.Cars**. The lower-ranked variables contribute less but still meaningfully to the model's boosting sequence.

Both tree-based methods agree that **Overall.Qual** and **Gr.Liv.Area** are the primary drivers of housing prices. However, their emphasis differs. Specifically, while Random Forest distributes importance more broadly across all predictors due to extensive averaging over bootstrap samples and random splits, XGBoost, designed for sequential error-correction, assigns extreme importance to a few top predictors, producing a more concentrated importance structure. This contrast reflects the difference between bagging (Random Forest) and boosting (XGBoost).

Across all regression and machine-learning approaches, several clear patterns are obvious. Linear, regularized, and Bayesian models all identify **Overall.Qual**, **Gr.Liv.Area**, and **Year.Built** as important predictors, though with different effect sizes due to their modeling assumptions. Tree-based methods confirm these findings while revealing nonlinear relationships more effectively. In terms of predictive accuracy, machine-learning models achieve the lowest RMSE and highest R^2 values, indicating outstanding performance on the test set.

Machine-learning models, especially Random Forest and XGBoost, illustrates that the pricing structure is nonlinear, meaning that the value added by certain features depends on the combination of other property characteristics. In practice, this reflects a real-world housing market where buyers respond to bundles of features rather than individual attributes in isolation. Thus, the results not only identify core drivers of price but also suggest that more flexible predictive tools are better suited for capturing these complex relationships.

More specifically, these findings suggest that housing prices in the *Ames* market are driven most strongly by overall construction quality and the usable living area of the home. This means that improvements that enhance general housing quality, such as renovations, upgrades to finishes, or structural improvements, tend to have the largest impact on market value. Similarly, expansions that increase livable space generally raise prices more reliably than changes in secondary features. The consistent importance of **Year.Built** also indicates that newer homes tend to have higher value due to modern standards in design, materials, and energy efficiency.

Practical Implications

The results of this analysis carry meaningful implications for homeowners, real-estate professionals, and policy analysts. First, the consistency across the findings demonstrates that property quality and living space are the most reliable levers for increasing home value. Renovations focused on structural soundness, kitchen and bathroom upgrades, and high-quality finishes are likely to yield the largest increases in price. Similarly, adding livable space, perhaps through finishing basements, converting attics, or constructing additions, can significantly improve market value as well.

For real-estate agents, the findings support the use of statistical predictive tools in market analysis. Random Forest and XGBoost produce more accurate predictions and can be used to comprehend nuanced interactions among features that traditional regression may not be able to. Given that computational resources are accessible, these models can therefore support more precise price estimates, especially in rapidly changing markets.

From a city-planning or policy-making perspective, the importance of variables such as **Year.Built** suggests that investment in modernization or infrastructure may also influence property values. Policies aimed at encouraging renovations, improving housing stock, or supporting new construction may have measurable impacts on overall market behavior.

Limitations and Future Work

Although the models used in this study provide meaningful insight into the determinants of housing prices, several limitations still remain. First, the analysis relies entirely on the *Ames* dataset, which only reflects the characteristics of a single city and cannot be used to generalize to other housing markets with different economic, demographic, geographic, or even political conditions. The most appropriate method to assess the models' performance, in this case, is to use the test dataset that has been held out in the cross-validation process. Additionally, the modeling process focuses on a selected subset of predictors; many other real-world factors, such as neighborhood desirability, school district quality, and local economic conditions, are not included in this study.

Second, some approaches, particularly linear models, assume additive and linear relationships, which do not fully capture complex market dynamics that often involve interactions between predictors. While ensemble methods address this limitation, they introduce their own challenges in interpretability, making it more difficult to communicate model behavior to non-technical audiences.

Future work could expand the modeling framework by incorporating geospatial variables, economic indicators, or market-specific features to improve predictive power and generalizability. Exploring advanced interpretable machine-learning methods, such as SHAP values or partial-dependence plots, may further enhance understanding of nonlinear effects. Lastly, applying these models to multiple housing markets or using cross-city validation would help determine how broadly the findings extend beyond the *Ames* dataset.

References

c

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2013). *Bayesian Data Analysis (3rd ed.).* CRC Press. Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

Malpezzi, S. (2002). *Hedonic pricing models: A selective and applied review. In Gibb, K. & O'Sullivan, T. (Eds.), Housing Economics and Public Policy* (pp. 67–89). Blackwell Science.