Kaggle Group Project

11/10/2024

- List of group members
- Introduction and Project Goal
- Description of the data
- Correlation matrix and Decision Tree
- Data Cleaning and Wrangling
- Modeling
    - Evaluating the Relationship between SalePrice and OverallQual
    - Evaluating the Relationship between SalePrice and GrLivArea
    - Evaluating the Relationship between SalePrice and Neighborhood
    - Evaluating the Relationship between SalePrice and `1stFlrSF`
    - Evaluating the Relationship between SalePrice and TotalBsmtSF
    - Evaluating the Relationship between SalePrice and GarageArea
    - Evaluating the Relationship between SalePrice and BsmtFinSF1
    - Evaluating the Relationship between SalePrice and KitchenQual
    - Evaluating the Relationship between SalePrice and PoolArea
    - Evaluating the Relationship between SalePrice and YearBuilt
    - Evaluating the Relationship between SalePrice and YearRemodAdd
- Cross validation
- Submit to Kaggle
- Final Conclusion
- Contributions

List of group members

- Gaby Rodriguez
- Ali Ladha
- Sonia Arias
- Tami Salvador

Introduction and Project Goal

This project aims to build a simple, effective model to predict housing prices in Ames, Iowa, using just five predictors. By focusing on a limited set of variables, we'll balance interpretability with accuracy. Key tasks include data cleaning, exploratory analysis, and cross-validation to ensure the model generalizes well, with a target R^2 of 0.75 on new data. Predictions will be submitted to Kaggle for evaluation, with performance measured by log RMSE against other entries.This assignment strengthens our skills in reproducible modeling and interpretation in R.

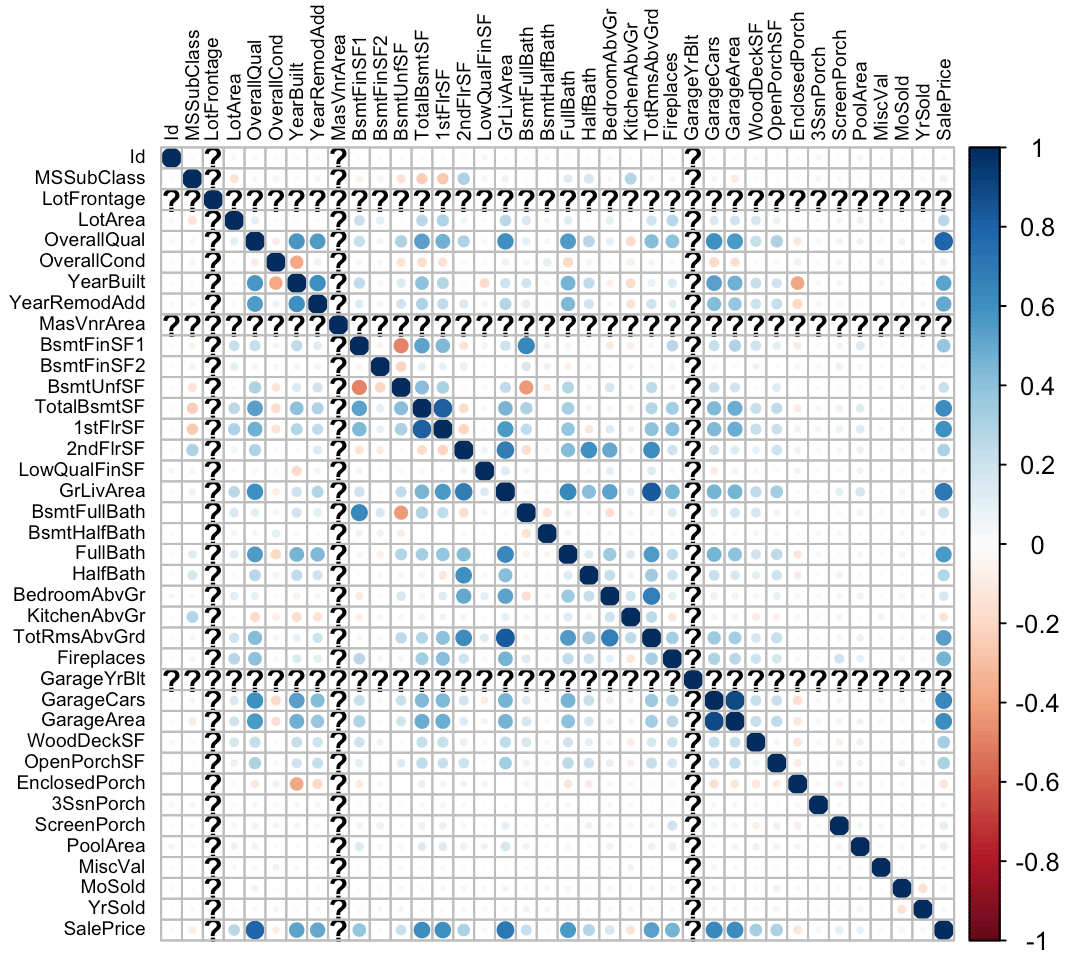
Description of the data

The target variable, SalePrice, was selected based on the goal to predict the sale price for each house. The following variables were chosen based on real estate knowledge and validated by a decision tree and a correlation matrix:

1. **GrLivArea:** Total above-ground living area; larger homes usually sell for more.
2. **YearBuilt:** Original construction year; newer homes are often valued higher.
3. **Neighborhood:** The area where the property is located; property values vary by neighborhood.
4. **YearRemodAdd:** Year of last remodel; recent renovations can increase appeal and value.
5. **OverallQual:** Overall quality of materials and finishes; higher quality generally means higher sale price.
6. **KitchenQual:** Kitchen quality, measured by material and finish grades; higher quality kitchens are a key selling point in homes.
7. **GarageArea:** Size of garage in square feet.
8. **SalePrice:** Final price of the house in dollars. This is the target variable.
9. **1stFlrSF:** First floor square feet.
10. **TotalBsmtSF:** Total basement square feet.
11. **PoolArea:**Area of the pool in square feet; properties with pools often have higher values, especially in warmer climates.
12. **BsmtFinSF1:**Finished square feet of the basement's primary area; finished basements often add usable living space, increasing the property's value.

Correlation matrix and Decision Tree

```
# Plot correlations among numeric variables in 'train'
corrplot(cor(train[sapply(train, is.numeric)]), method = "circle", tl.cex = 0.6, tl.col
= "black")
```



The correlation matrix shows the relationships between numerical variables, with color and size indicating the strength and direction of correlations:

- **Color and Size:** Dark blue = strong positive correlation (+1), dark red = strong negative correlation (-1), with lighter colors and smaller circles showing weaker relationships

---

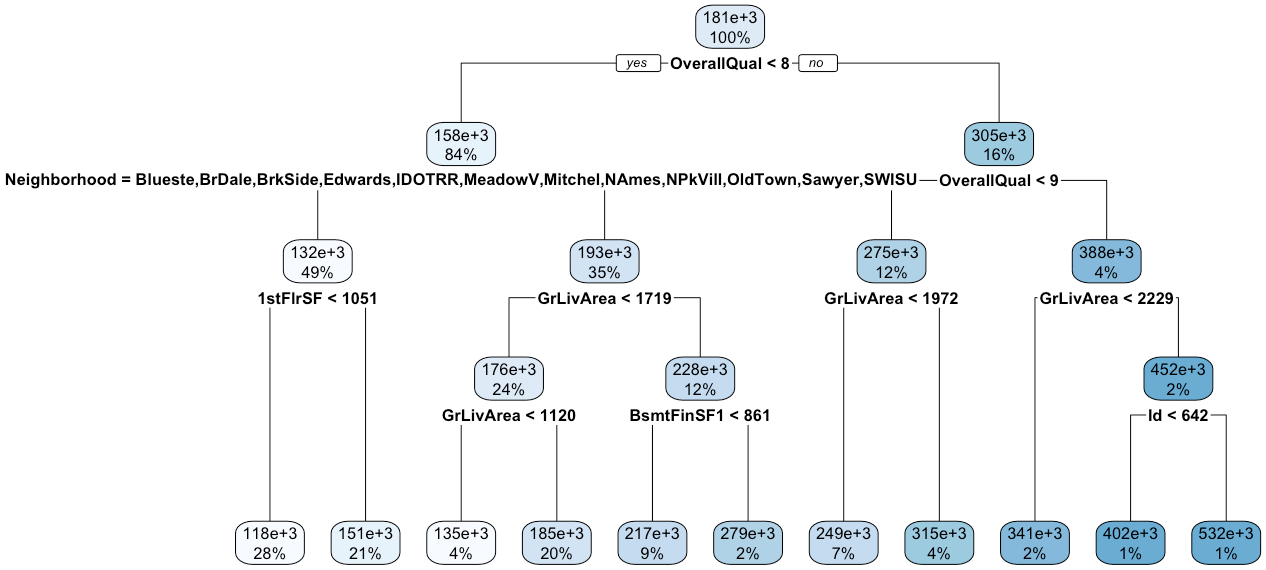
TREE

```
# Fit a decision tree model to predict SalePrice
tree_model <- rpart(formula = SalePrice ~ ., data = train)

# Display the tree structure
tree_model
```

```
## n= 1460
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1460 9.207911e+12 180921.2
##    2) OverallQual< 7.5 1231 2.987549e+12 157832.4
##      4) Neighborhood=Blueste,BrDale,BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkV
ill,OldTown,Sawyer,SWISU 713 8.745535e+11 132242.5
##        8) 1stFlrSF< 1050.5 410 3.254356e+11 118198.6 *
##        9) 1stFlrSF>=1050.5 303 3.588339e+11 151245.7 *
##      5) Neighborhood=Blmngtn,ClearCr,CollgCr,Crawfor,Gilbert,NoRidge,NridgHt,NWAmes,S
awyerW,Somerst,StoneBr,Timber,Veenker 518 1.003420e+12 193055.7
##       10) GrLivArea< 1719 350 3.770698e+11 176242.8
##         20) GrLivArea< 1120 63 2.330279e+10 135391.3 *
##         21) GrLivArea>=1120 287 2.255504e+11 185210.3 *
##       11) GrLivArea>=1719 168 3.213015e+11 228082.4
##         22) BsmtFinSF1< 860.5 138 1.683332e+11 217077.2 *
##         23) BsmtFinSF1>=860.5 30 5.937141e+10 278706.2 *
##    3) OverallQual>=7.5 229 2.036506e+12 305035.9
##      6) OverallQual< 8.5 168 6.818726e+11 274735.5
##       12) GrLivArea< 1971.5 103 2.402072e+11 249392.5 *
##       13) GrLivArea>=1971.5 65 2.706830e+11 314894.6 *
##      7) OverallQual>=8.5 61 7.755905e+11 388486.1
##       14) GrLivArea< 2229 35 7.763301e+10 341248.2 *
##       15) GrLivArea>=2229 26 5.147238e+11 452075.5
##         30) Id< 642 16 1.129941e+11 402070.3 *
##         31) Id>=642 10 2.977080e+11 532083.9 *
```

```
# Visualize the decision tree
rpart.plot(x = tree_model)
```



The tree diagram shows the variables which have the most impact to the relationship (with SalePrice) at the very top. The variables at the top have the highest information gain. For example, the variable: Neighborhood has more information gain than BsmtFinSF1.

Variable Selection Based on the Matrix and Decision Tree

The variables selected due to the correlation matrix & decision trees are:

1. *Neighborhood:* Neighborhood is known to affect SalePrice due to property value variations by area
2. *GrLivArea* - Strong positive correlation. Larger living areas directly boost property value.
3. *1stFlrSF`* - Moderate positive correlation. Bigger first floors add value.
4. *OverallQual* - Strong positive correlation. Better overall quality increases home value significantly.
5. *GarageArea* - Strong positive correlation. Larger garages attract buyers.
6. *TotalBsmtSF* - Strong positive correlation. Bigger basements add usable space.
7. *YearBuilt* - Moderate positive correlation. Newer homes tend to have higher values.
8. *YearRemodAdd*- YearRemodAdd: Moderate positive correlation. Recently remodeled homes are more appealing and valuable.
9. *BsmtFinSF1* - Strong positive correlation. Finished basements increase usable space and property value.
10. *KitchenQual* - Strong positive correlation. Higher-quality kitchens boost appeal and price
11. *PoolArea* - Weak to moderate correlation. Pools can increase value, depending on preferences.

We decided not to include further variables due to not being relevant to the correlation matrix and the decision tree. Using more variables can overfit the model.

Data Cleaning and Wrangling

In this step, we're identifying missing values across all columns in the train dataset to prepare for data cleaning:

```
# Define a function 'count_missings' that calculates the number of missing values (NAs)
in a vector
count_missings <- function(x) sum(is.na(x))


# 'summarize_all' is used to apply the function to all columns and return a summary of m
issing values per column
train |>
  summarize_all(count_missings) # Handy summarize_all function
```

```
## # A tibble: 1 × 81
##      Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape
##   <int>      <int>    <int>       <int>   <int>  <int> <int>    <int>
## 1     0          0        0         259       0      0  1369        0
## # ℹ 73 more variables: LandContour <int>, Utilities <int>, LotConfig <int>,
## #   LandSlope <int>, Neighborhood <int>, Condition1 <int>, Condition2 <int>,
## #   BldgType <int>, HouseStyle <int>, OverallQual <int>, OverallCond <int>,
## #   YearBuilt <int>, YearRemodAdd <int>, RoofStyle <int>, RoofMatl <int>,
## #   Exterior1st <int>, Exterior2nd <int>, MasVnrType <int>, MasVnrArea <int>,
## #   ExterQual <int>, ExterCond <int>, Foundation <int>, BsmtQual <int>,
## #   BsmtCond <int>, BsmtExposure <int>, BsmtFinType1 <int>, BsmtFinSF1 <int>, …
```

```
train |>
  summarize_all(count_missings) |> select(Neighborhood, GrLivArea, `1stFlrSF`, OverallQu
al, GarageArea, TotalBsmtSF,
                                          YearBuilt, YearRemodAdd, BsmtFinSF1, KitchenQu
al, PoolArea)
```

```
## # A tibble: 1 × 11
##   Neighborhood GrLivArea `1stFlrSF` OverallQual GarageArea TotalBsmtSF YearBuilt
##          <int>     <int>      <int>       <int>      <int>       <int>     <int>
## 1            0         0          0           0          0           0         0
## # ℹ 4 more variables: YearRemodAdd <int>, BsmtFinSF1 <int>, KitchenQual <int>,
## #   PoolArea <int>
```

According to the table generated, the eleven selected predictors above do not contain any missing values in the Train set, indicating they're ready for analysis without needing imputation.

```
# Converting categorical variables to factors
train <- train %>%
  mutate(
    Neighborhood = factor(Neighborhood),
    KitchenQual = factor(KitchenQual),
    OverallQual = factor(OverallQual)
  )
```

We have factored the following variables for modelling purposes.

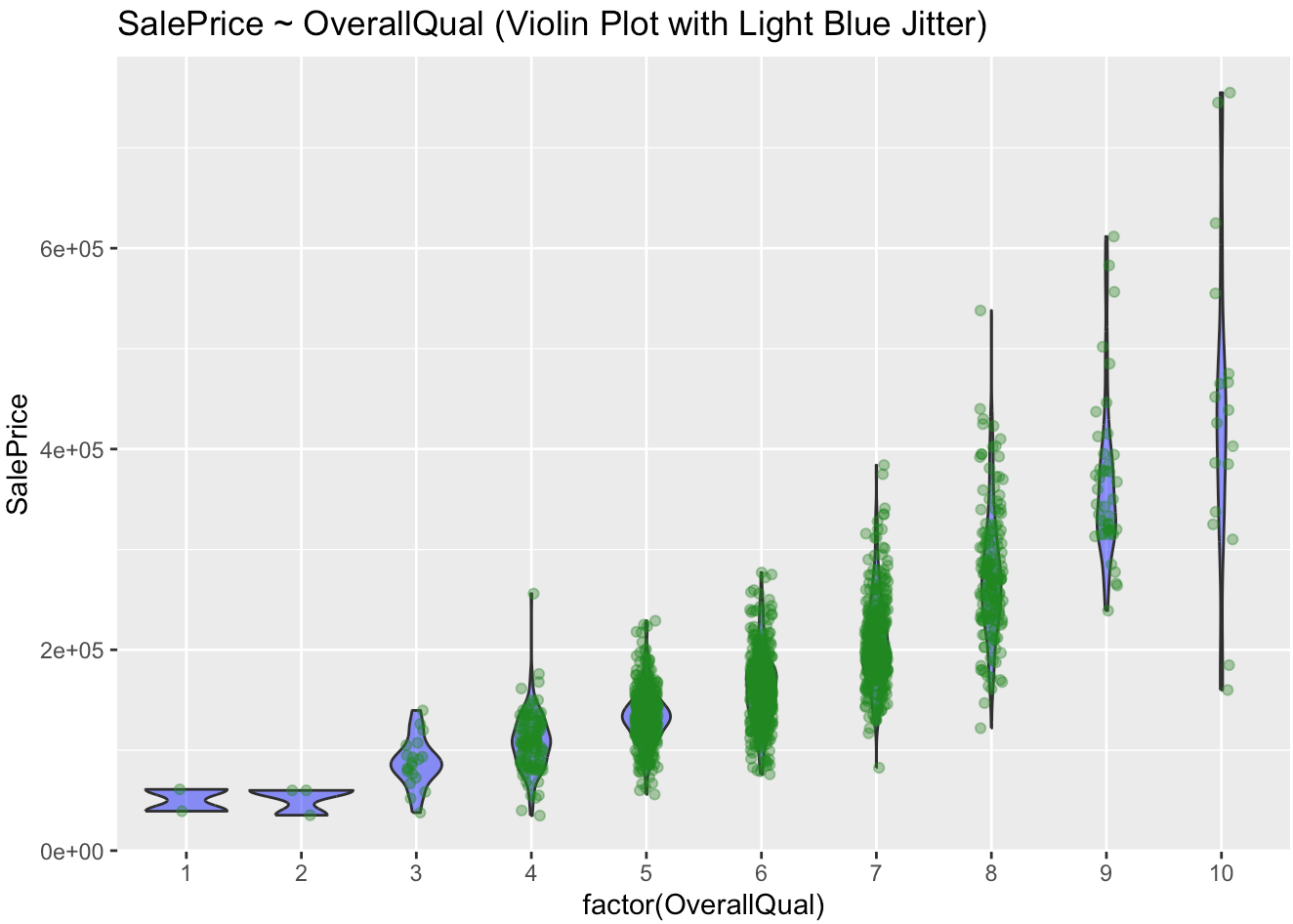
Modeling

In this section, we analyze the predictors to determine the best way to represent each in our predictive model for SalePrice. For each variable, we assess whether it should be treated as a numeric variable, log-transformed, or converted into a factor, based on its relationship with SalePrice and interpretability considerations.

Evaluating the Relationship between SalePrice and OverallQual

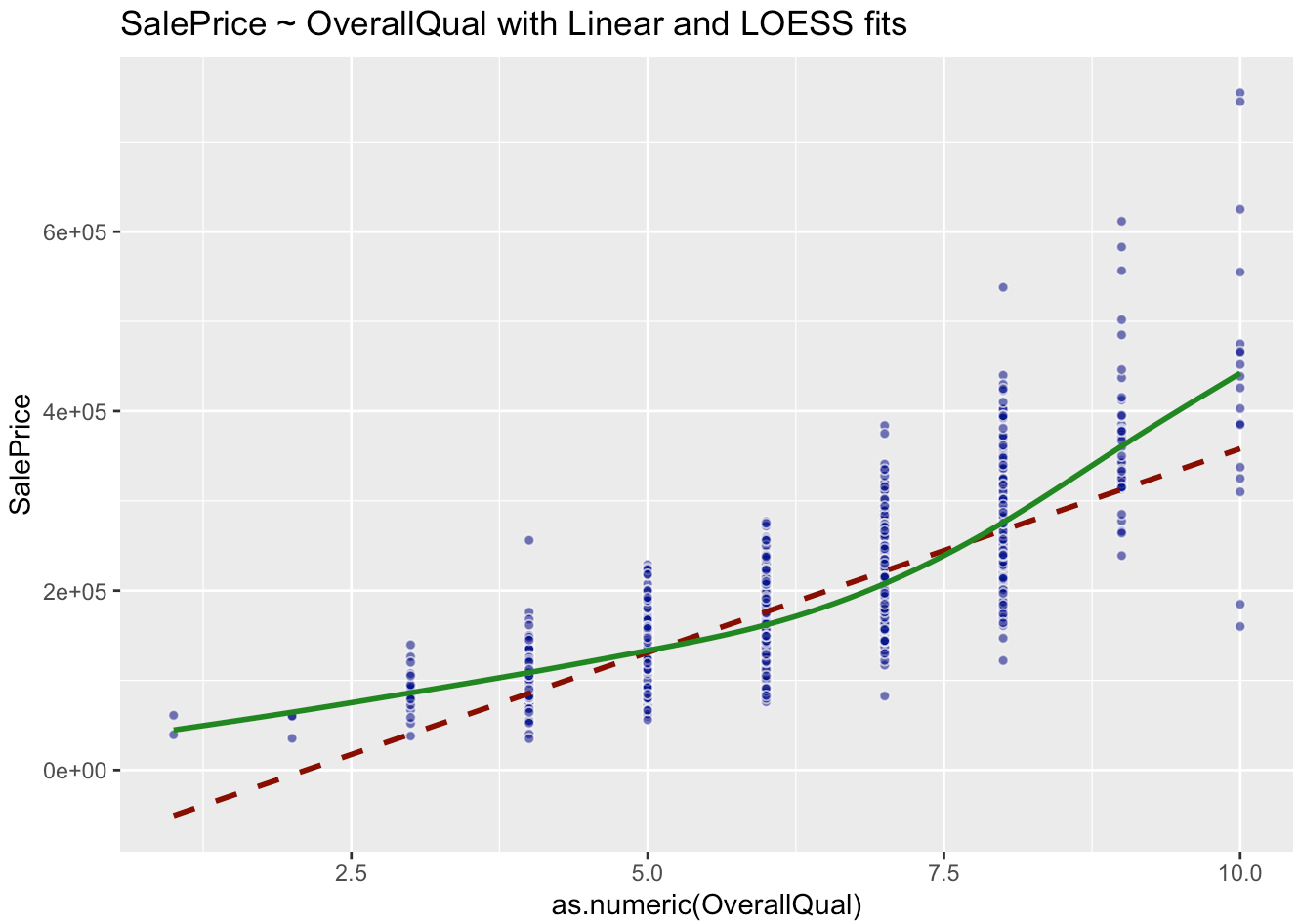
OverallQual is a categorical feature with 10 levels of quality but is currently encoded as a numeric variable. Deciding whether to treat it as numeric or categorical depends on its relationship with SalePrice. If the relationship appears non-linear, encoding it as a factor would better capture its influence; if it's linear, leaving it as numeric would be appropriate.

```
# Step 1: Inspect SalePrice vs. OverallQual with a violin plot and jitter
train |>
  ggplot(aes(factor(OverallQual), SalePrice)) +
  geom_violin(fill = "blue", alpha = 0.5) +         # Violin plot for distribution
  geom_jitter(width = 0.1, alpha = 0.4, color = "forestgreen") +  # Light blue jitter po
ints
  labs(title = "SalePrice ~ OverallQual (Violin Plot with Light Blue Jitter)")
```



In this plot, the relationship appears more exponential, suggesting that a simple linear fit may not capture the pattern well.

```
# Step 2: Compare linear and non-linear fits
ggplot(train, aes(as.numeric(OverallQual), SalePrice)) +
  geom_point(shape = 21, fill = "darkblue", color = "white", alpha = 0.6) + # Change poi
nt style
  geom_smooth(method = "lm", se = FALSE, color = "darkred", linetype = "dashed") +  # Li
near fit in dashed red
  geom_smooth(se = FALSE, color = "forestgreen", linetype = "solid") +  # LOESS fit in s
olid green
  labs(title = "SalePrice ~ OverallQual with Linear and LOESS fits")
```



The non-linear fit (LOESS) captures the trend better, indicating that treating OverallQual as a factor could improve the model.

```
# Step 3: R-squared comparison for numeric vs. factor encoding
# Numeric encoding
lm(SalePrice ~ OverallQual, data = train) |>
  summary() # R-squared around .62
```

```
##
## Call:
## lm(formula = SalePrice ~ OverallQual, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -278588  -24023   -2113   19312  316412
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      50150      31666   1.584 0.113482
## OverallQual2      1620      40881   0.040 0.968390
## OverallQual3     37324      33212   1.124 0.261285
## OverallQual4     58271      31938   1.824 0.068286 .
## OverallQual5     83373      31746   2.626 0.008724 **
## OverallQual6    111453      31751   3.510 0.000461 ***
## OverallQual7    157566      31766   4.960 7.87e-07 ***
## OverallQual8    224586      31854   7.050 2.75e-12 ***
## OverallQual9    317363      32395   9.797  < 2e-16 ***
## OverallQual10   388438      33380  11.637  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44780 on 1450 degrees of freedom
## Multiple R-squared:  0.6842, Adjusted R-squared:  0.6822
## F-statistic:   349 on 9 and 1450 DF,  p-value: < 2.2e-16
```

```
# Factor encoding
lm(SalePrice ~ factor(OverallQual), data = train) |>
  summary() # R-squared around .68
```

```
##
## Call:
## lm(formula = SalePrice ~ factor(OverallQual), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -278588  -24023   -2113   19312  316412
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)              50150      31666   1.584 0.113482
## factor(OverallQual)2      1620      40881   0.040 0.968390
## factor(OverallQual)3     37324      33212   1.124 0.261285
## factor(OverallQual)4     58271      31938   1.824 0.068286 .
## factor(OverallQual)5     83373      31746   2.626 0.008724 **
## factor(OverallQual)6    111453      31751   3.510 0.000461 ***
## factor(OverallQual)7    157566      31766   4.960 7.87e-07 ***
## factor(OverallQual)8    224586      31854   7.050 2.75e-12 ***
## factor(OverallQual)9    317363      32395   9.797  < 2e-16 ***
## factor(OverallQual)10   388438      33380  11.637  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44780 on 1450 degrees of freedom
## Multiple R-squared:  0.6842, Adjusted R-squared:  0.6822
## F-statistic:   349 on 9 and 1450 DF,  p-value: < 2.2e-16
```

OverallQual Analysis Conclusion

Based on our analysis, converting OverallQual to a factor significantly improves the model's ability to predict SalePrice. The following points highlight why this transformation is beneficial:

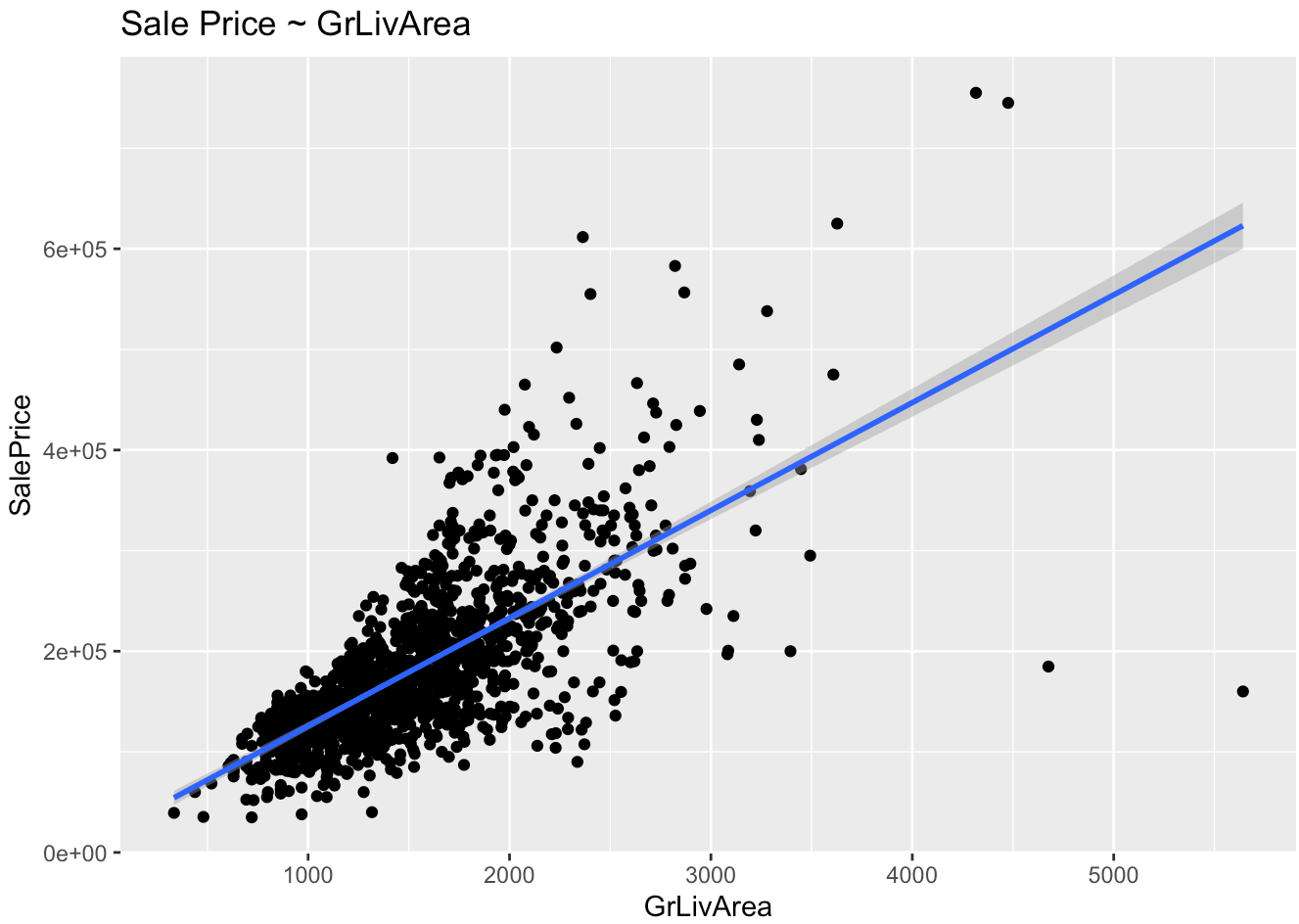
- Higher R-Squared Value: When OverallQual is treated as a numeric variable, the R-squared value for the linear model is approximately 0.62, meaning that this numeric encoding explains 62% of the variance in SalePrice. In contrast, when OverallQual is encoded as a factor, the model achieves an R-squared value of approximately 0.68, explaining 68% of the variance. This 6% improvement indicates that the factor encoding captures the impact of OverallQual on SalePrice more effectively than the numeric encoding.
- Improved Model Fit: The increase in R-squared from 0.62 to 0.68 shows that the categorical (factor) encoding aligns better with the non-linear nature of the relationship between OverallQual and SalePrice. This suggests that treating OverallQual as a factor allows the model to capture nuances in quality levels that may not be linear. For example, the jump in SalePrice from an OverallQual of 7 to 8 might be much larger than from 5 to 6, which the factor encoding handles better.
- Lower Residual Standard Error: The residual standard error decreases from 48,620 with the numeric encoding to 44,780 with the factor encoding. This reduction shows that the factor model has less error in its predictions, further confirming that encoding OverallQual as a factor enhances predictive accuracy.

By encoding OverallQual as a factor, we capture the non-linear relationship with SalePrice, improve the model's fit, and reduce prediction error. This transformation ultimately enhances both the interpretability and predictive power of the model.

Evaluating the Relationship between SalePrice and GrLivArea

```
#Creating a Scatter plot
train %>%
  ggplot(aes(GrLivArea, SalePrice)) + geom_point() +
  labs(title = "Sale Price ~ GrLivArea") + geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



The plot shows a strong positive linear relationship between SalePrice and GrLivArea (above-ground living area), indicating larger living areas increase home value.

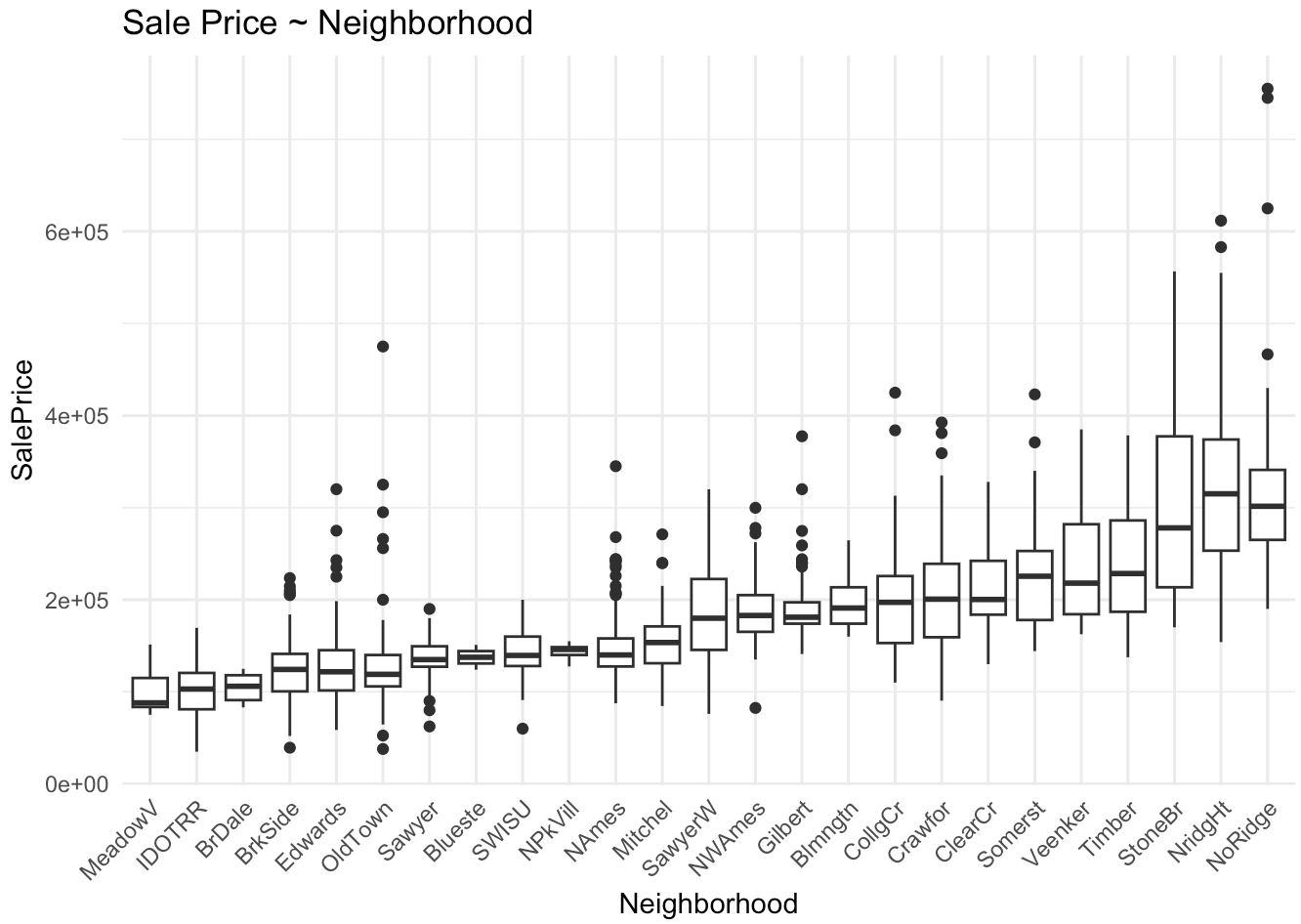
Modeling Analysis for GrLivArea

GrLivArea represents above-ground living area (square footage) and is expected to correlate positively with SalePrice.

Evaluating the Relationship between SalePrice and

Neighborhood

```
#Creating a Boxplot
train %>%
  ggplot(aes(x = reorder(Neighborhood, SalePrice), y = SalePrice)) +
  geom_boxplot() +
  labs(title = "Sale Price ~ Neighborhood", x = "Neighborhood", y = "SalePrice") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



We factored Neighborhood. The plot shows the relationship between SalePrice and Neighborhood, indicating Neighborhoods have different home values.

Modeling Analysis for Neighborhood

Converting Neighborhood into a factor enables the model to capture the unique impact of each neighborhood on house prices, aligns with the real-world meaning of neighborhoods as distinct categories, and improves both accuracy and interpretability.

Neighborhood Analysis Conclusion

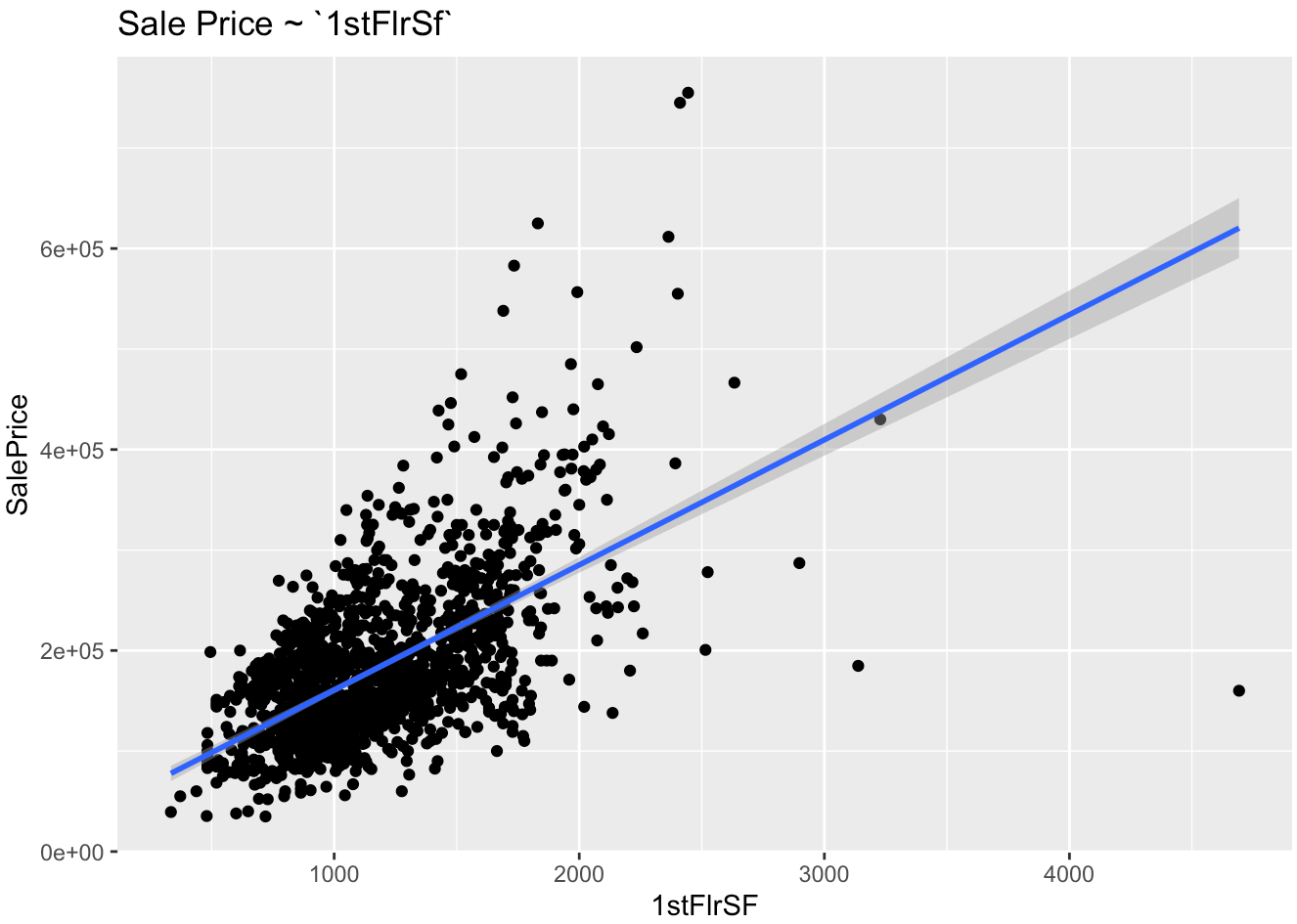
- The boxplot and regression analysis reveal that neighborhood significantly impacts SalePrice. High-value neighborhoods like NoRidge (average increase of $140,424), NridgHt ($121,400), and StoneBr ($115,628) have substantially higher sale prices compared to the reference neighborhood, Blmngtn.
- In contrast, low-value neighborhoods such as MeadowV (average decrease of $96,294) and IDOTRR ($94,747) show lower prices.

- These variations confirm that neighborhood location is a strong predictor of property value. Converting Neighborhood to a factor captures these unique differences, improving model accuracy and interpretability.

Evaluating the Relationship between SalePrice and `1stFlrSF`

```
#Creating a Scatter plot
train %>%
  ggplot(aes(`1stFlrSF` ,SalePrice)) + geom_point() +
  labs(title = "Sale Price ~ `1stFlrSf`") + geom_smooth(method = 'lm')
```

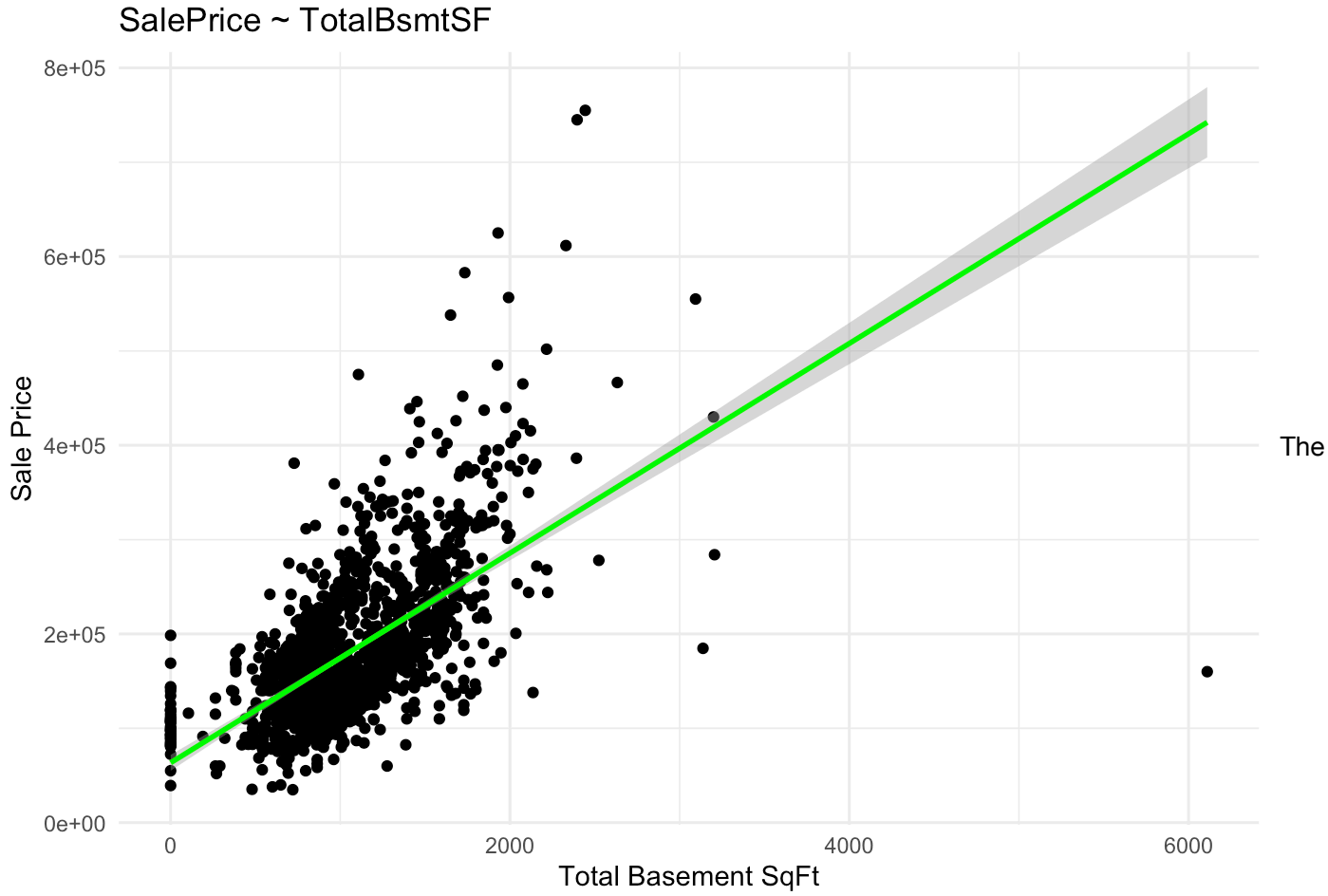
```
## `geom_smooth()` using formula = 'y ~ x'
```



The plot shows a strong positive linear relationship between `1stFlrSf` (First floor square feet) and SalePrice, indicating larger first floor square footage increases home value.

Evaluating the Relationship between SalePrice and TotalBsmtSF

```
# Plot a scatter plot with regression line
ggplot(train, aes(x = TotalBsmtSF, y = SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm",    # Add regression line
              color = "green") +
  labs(title = "SalePrice ~ TotalBsmtSF",
       x = "Total Basement SqFt",
       y = "Sale Price") +
  theme_minimal()
```



The plot shows a strong positive linear relationship between **TotalBsmtSF** and **SalePrice**, indicating that larger basement sizes significantly increase home value. The regression line highlights this trend, with a consistent upward slope.

Evaluating the Relationship between SalePrice and GarageArea

```
# Check for unexpected or extreme values in GarageArea
summary(train$GarageArea)
```

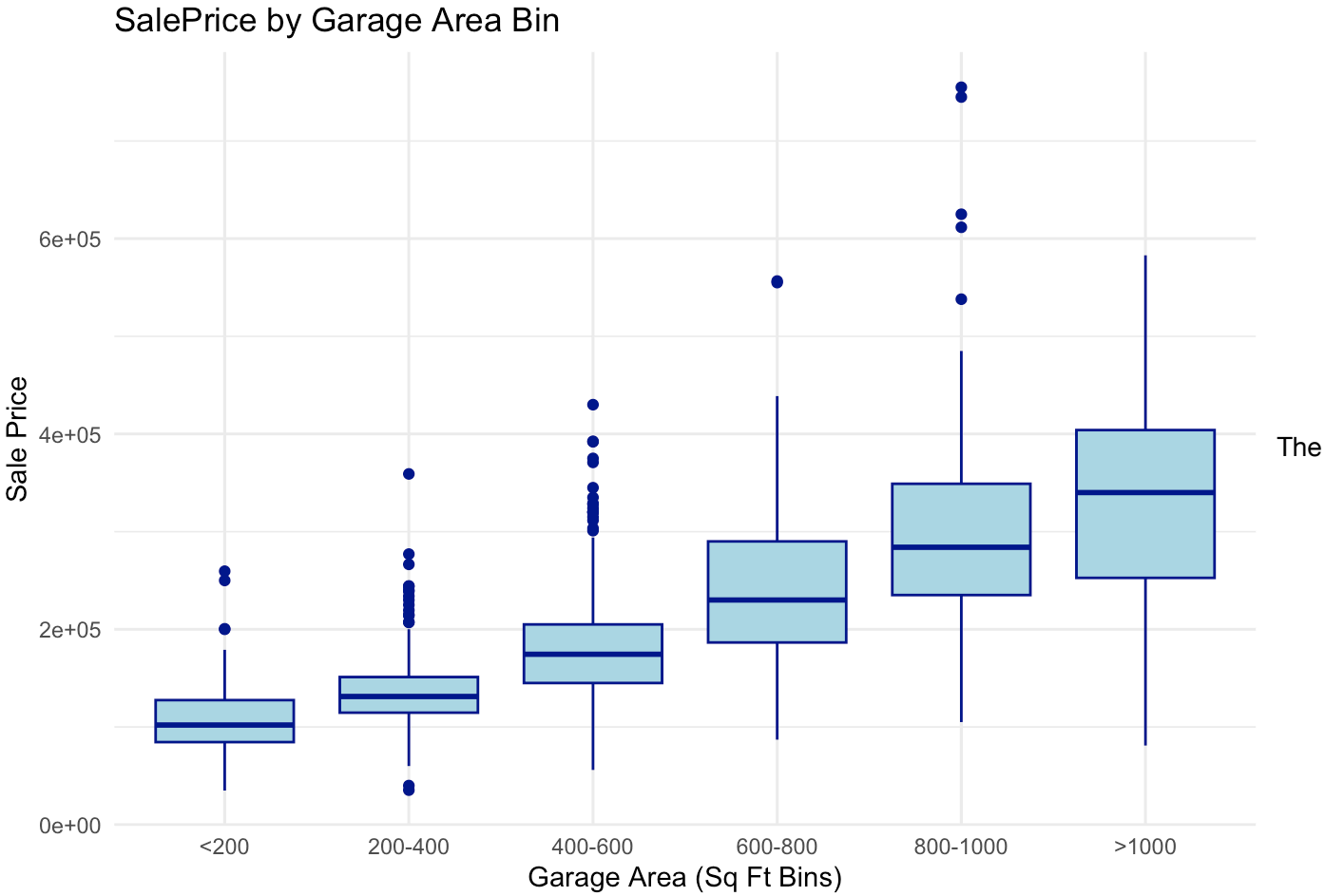
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   334.5   480.0   473.0   576.0  1418.0
```

```
# Create bins for GarageArea, ensuring full coverage of the range
train$GarageBin <- cut(train$GarageArea,
                       breaks = c(-Inf, 200, 400, 600, 800, 1000, Inf),  # Include all r
anges
                       labels = c("<200", "200-400", "400-600", "600-800", "800-1000",
">1000"))

# Verify that no NAs are present in the binning process
table(train$GarageBin, useNA = "ifany")  # Check for NAs in GarageBin
```

```
##
##     <200  200-400  400-600  600-800 800-1000    >1000
##      101      394      646      205       99       15
```

```
# Create the boxplot
ggplot(train, aes(x = GarageBin, y = SalePrice)) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  labs(title = "SalePrice by Garage Area Bin",
       x = "Garage Area (Sq Ft Bins)",
       y = "Sale Price") +
  theme_minimal()
```



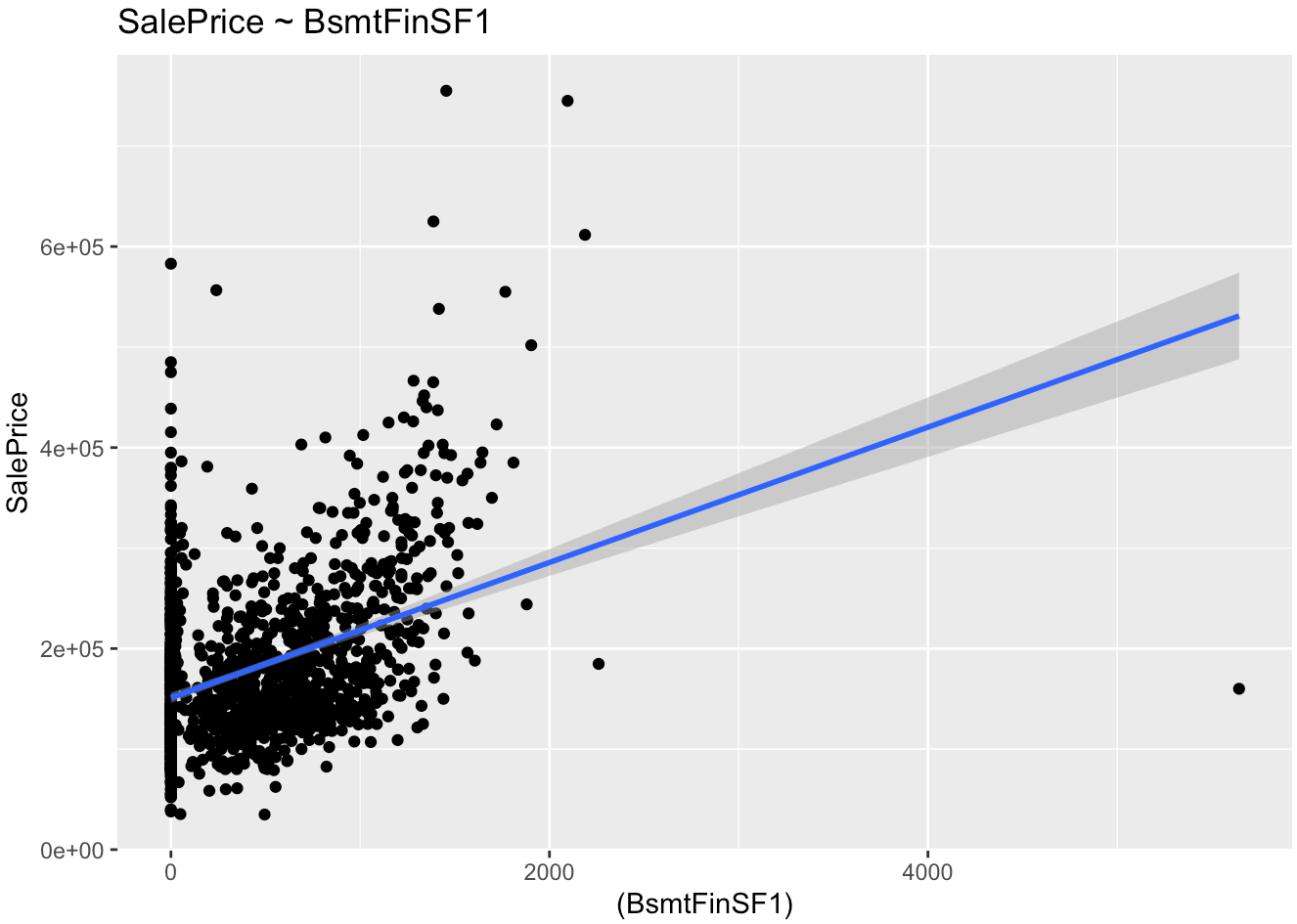
The boxplot illustrates a positive relationship between garage area and **SalePrice**, with larger garage sizes corresponding to higher median home prices. Additionally, greater garage areas exhibit increased price variability,

as indicated by wider interquartile ranges and outliers.

Evaluating the Relationship between SalePrice and BsmtFinSF1

```
train |>
  ggplot(aes((BsmtFinSF1), SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "SalePrice ~ BsmtFinSF1")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



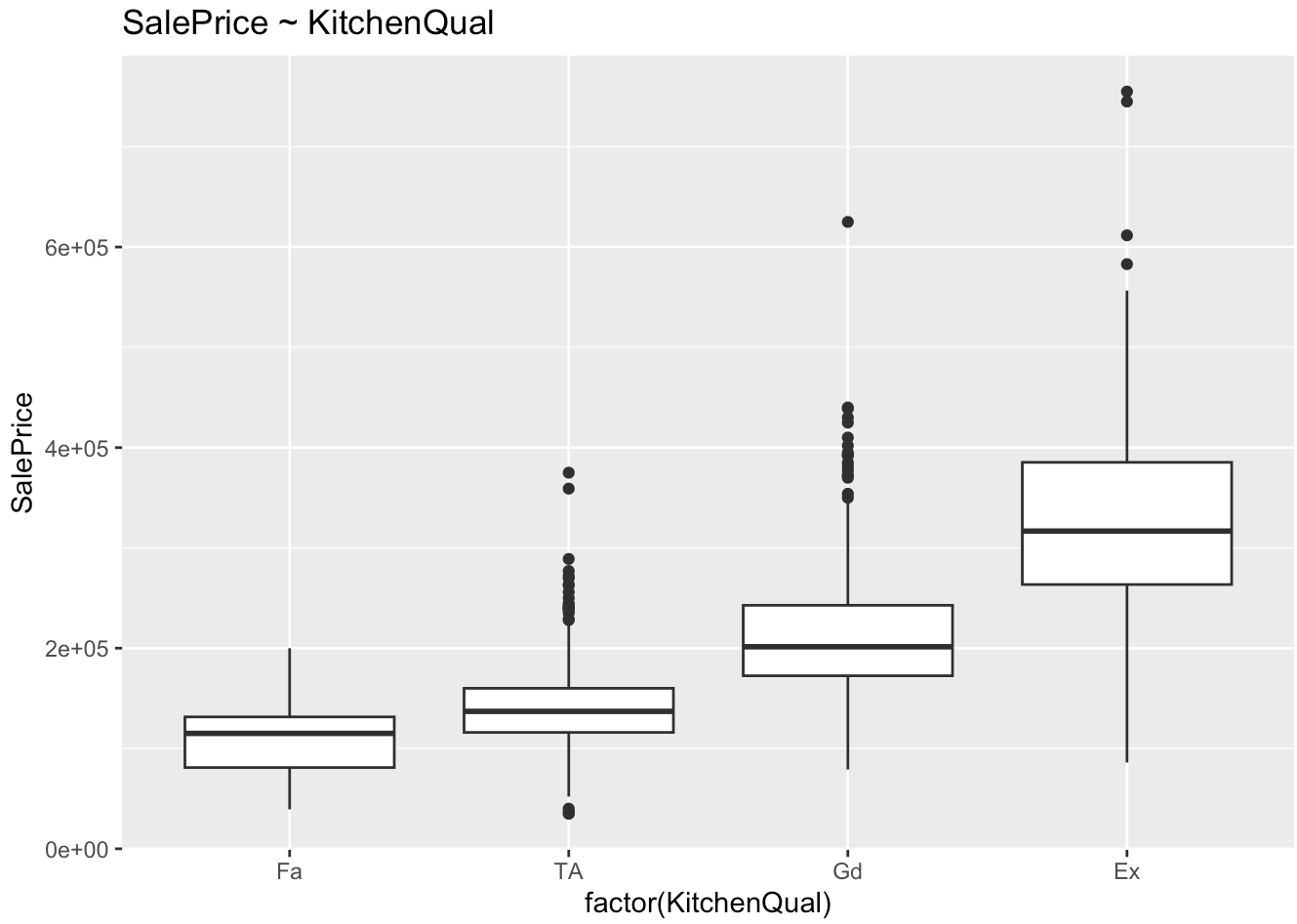
This graph shows a strong positive correlation between SalePrice and BsmtFinSF1. There are a few outliers, but in general, the larger the basement, the more the house costs.

Evaluating the Relationship between SalePrice and KitchenQual

```
train <- train |> mutate(KitchenQual = reorder(KitchenQual, SalePrice, FUN = median))

train |>ggplot(aes(factor(KitchenQual), SalePrice)) +
  geom_boxplot() +
  geom_smooth(method = "lm") +
  labs(title = "SalePrice ~ KitchenQual")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



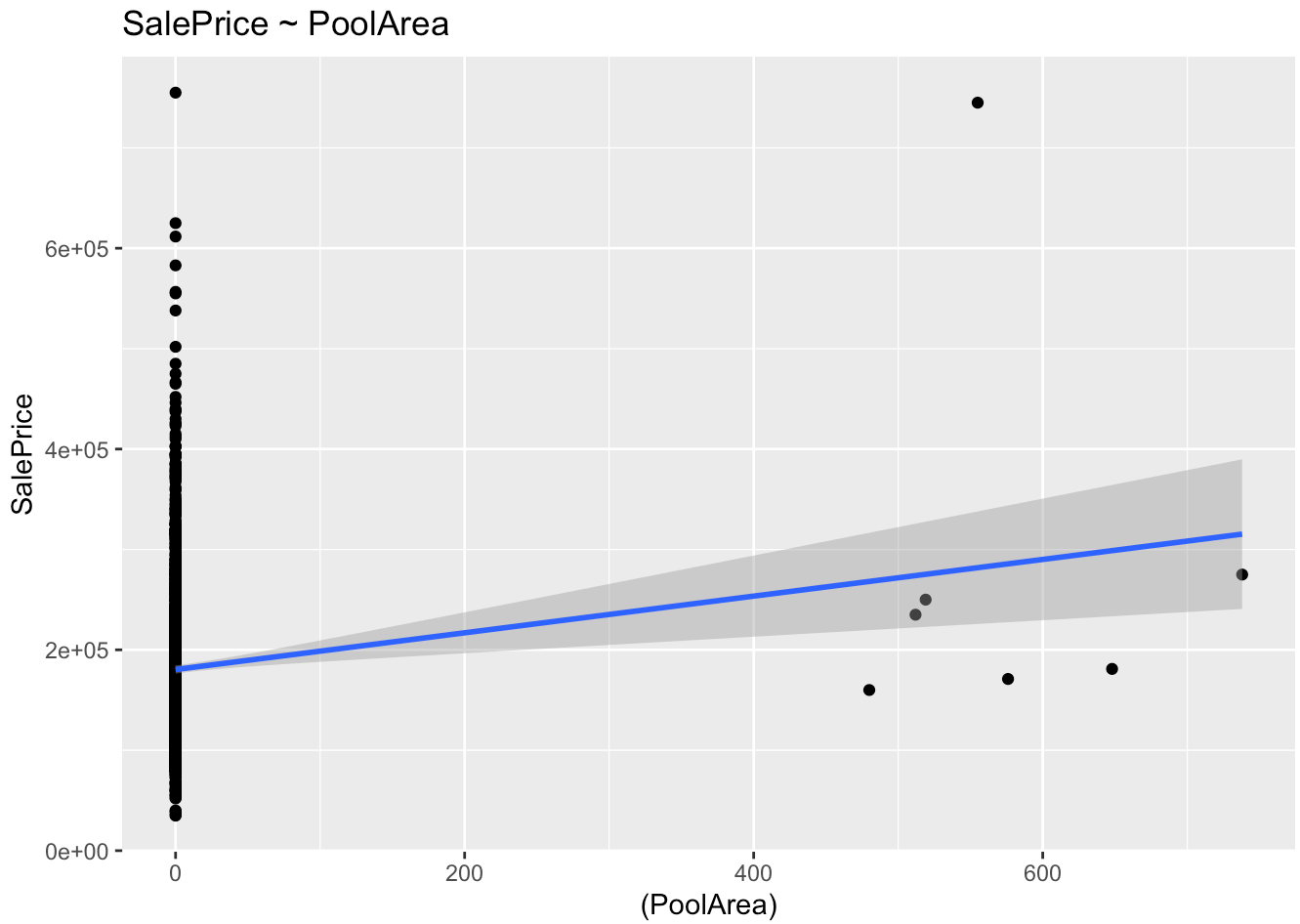
We factored the kitchen quality. This graph indicates that a kitchen with an Excellent condition has a higher SalePrice. The better the kitchen quality the higher the home value.

Modeling Analysis for Kitchen Quality Converting KitchenQuality into a factor enables the model to capture the unique impact of the kitchen quality on house prices.

Evaluating the Relationship between SalePrice and PoolArea

```
train |>
  ggplot(aes((PoolArea), SalePrice)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "SalePrice ~ PoolArea")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



This graph shows a strong positive correlation between SalePrice and PoolArea, meaning that the bigger the pool, the more the house costs.

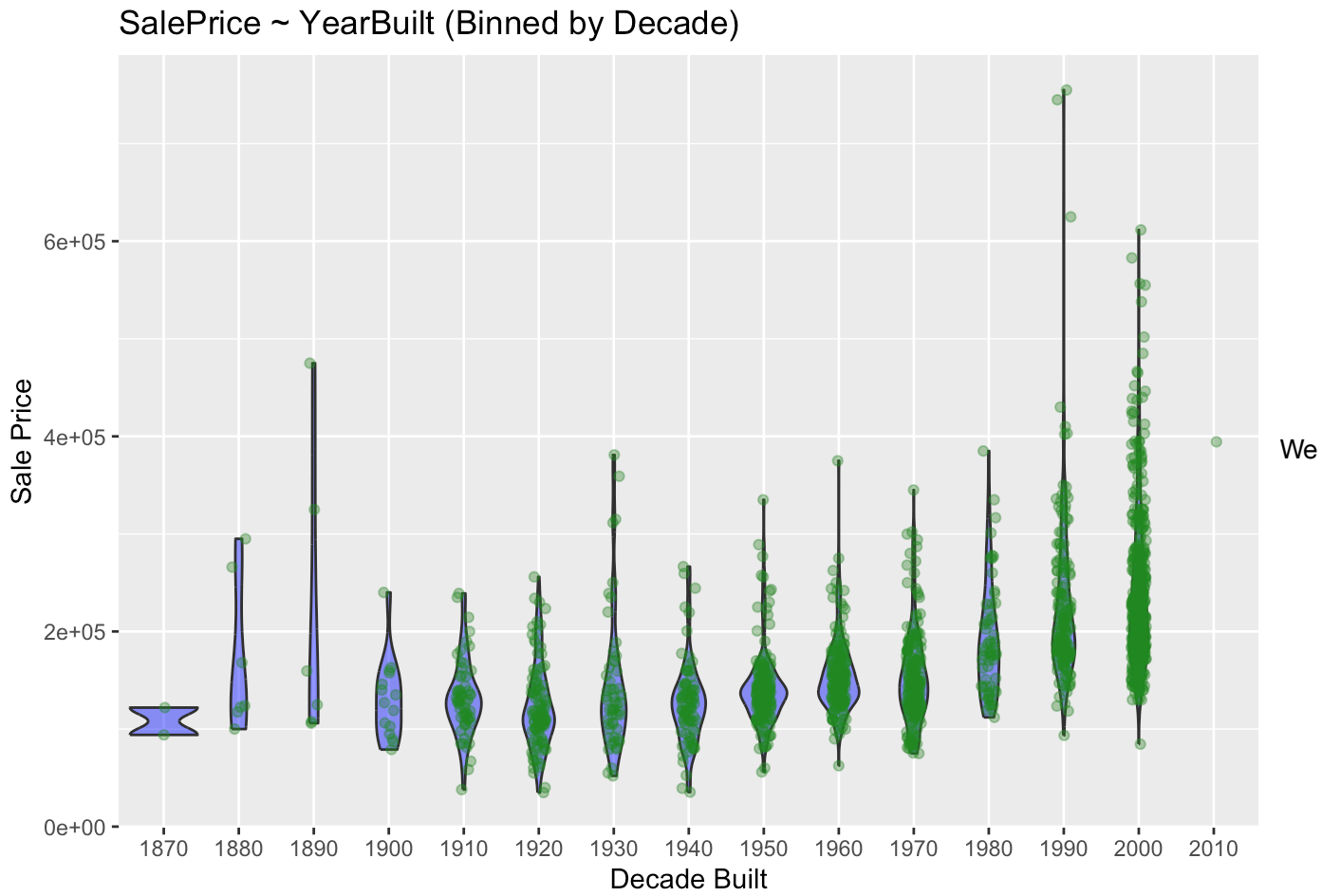
The scatter plot shows a weak positive relationship between PoolArea and SalePrice, as indicated by the slight upward slope of the regression line. However, most data points cluster around a PoolArea of 0, suggesting that homes with pools are rare, and the effect of PoolArea on SalePrice becomes more noticeable only for properties with larger pool sizes.

Evaluating the Relationship between SalePrice and YearBuilt

```
# This code creates a violin plot to visualize the distribution of sale prices (SalePric
e)
# grouped by the decade when the houses were built (YearBuilt).
# The jittered points overlay individual sale price data for more granular insights.

train |>
  ggplot(aes(factor(floor(YearBuilt / 10) * 10), SalePrice)) +  # Bin YearBuilt by decad
es
  geom_violin(fill = "blue", alpha = 0.5) +                     # Add a violin plot with
semi-transparent blue fill
  geom_jitter(width = 0.1, alpha = 0.4, color = "forestgreen") +  # Overlay individual s
ale prices as jittered points
  labs(
    title = "SalePrice ~ YearBuilt (Binned by Decade)",         # Add plot title
    x = "Decade Built",                                         # Label x-axis as 'Decad
e Built'
    y = "Sale Price"                                            # Label y-axis as 'Sale
Price'
  )
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## ℹ Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

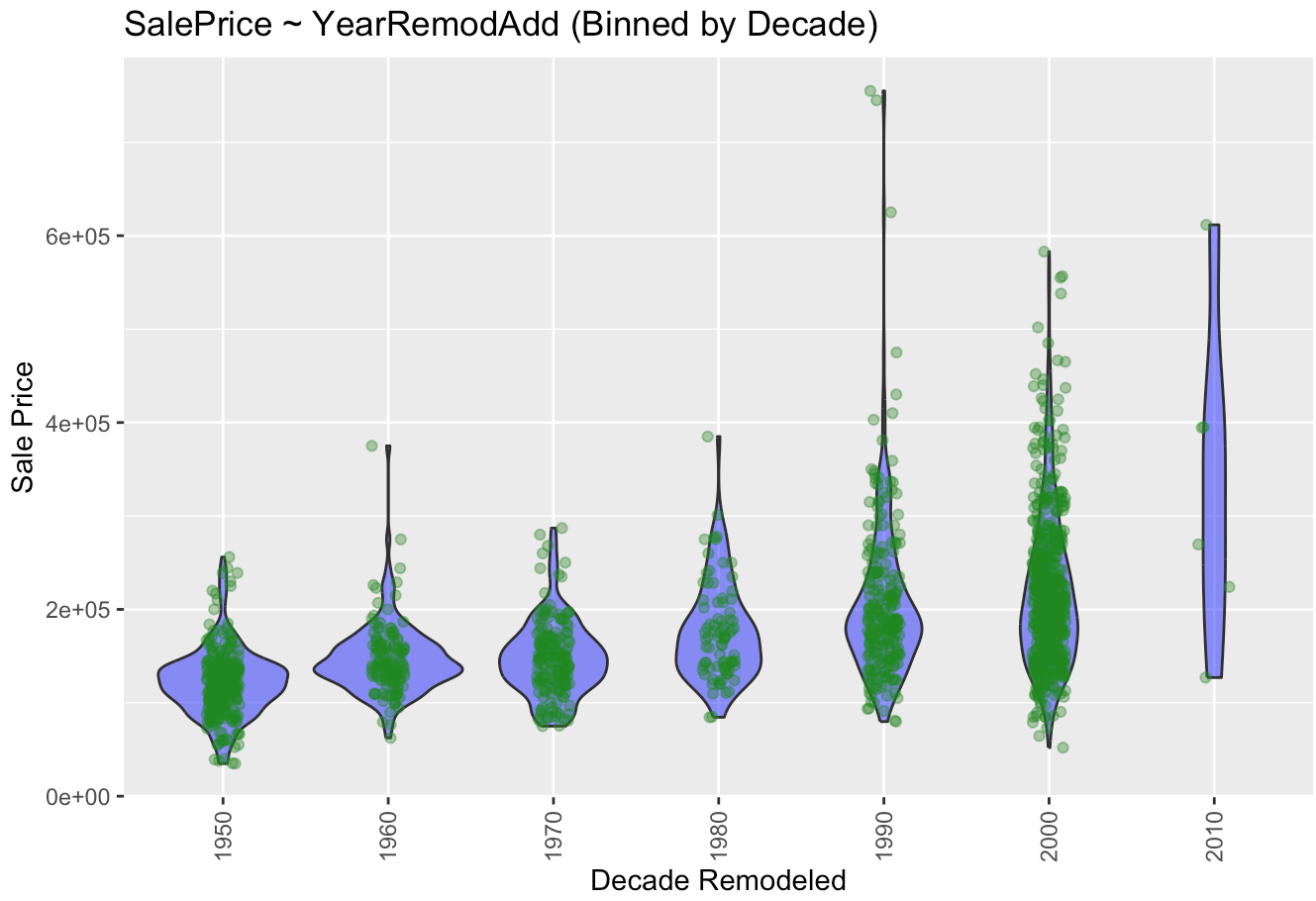


We factored YearBuilt Homes built in recent decades (1990s and 2000s) have higher sale prices and more price variety. Older homes (before the 1940s) tend to have lower and more consistent prices.

Evaluating the Relationship between SalePrice and YearRemodAdd

```
# This plot visualizes the relationship between SalePrice and YearRemodAdd, grouped by d
ecade.
# The x-axis shows the decades when properties were remodeled or added, while the y-axis
represents SalePrice.
# The violin plot illustrates the distribution of SalePrice within each decade, while th
e jittered points show individual SalePrice data points.

train |>
  ggplot(aes(factor(floor(YearRemodAdd / 10) * 10), SalePrice)) +
  geom_violin(fill = "blue", alpha = 0.5) +
  geom_jitter(width = 0.1, alpha = 0.4, color = "forestgreen") +
  labs(
    title = "SalePrice ~ YearRemodAdd (Binned by Decade)",
    x = "Decade Remodeled",
    y = "Sale Price",
    caption = "Violin plot shows distribution; points represent individual properties."
  ) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Violin plot shows distribution; points represent individual properties.

We factored YearRemodAdd. Sale prices increase with recent remodel decades, showing higher variability and outliers in 1990s-2000s. Older decades have lower, narrower distributions. Recent remodels correlate with higher property values.

Changes after modelling

After analyzing each variable, the following representations were changed to optimize model performance and interpretability:

- **OverallQual:** Converted to a factor due to its non-linear relationship with SalePrice, achieving a higher R-squared (0.68) compared to the numeric model (0.62).
- **YearBuilt and YearRemodAdd:** Kept as numeric variables due to their continuous nature, as each year incrementally reflects temporal effects on property value. Treating these variables numerically preserves interpretability, avoids the risk of overfitting from factor conversion, and captures smooth trends aligned with their time-based characteristics. This approach ensures simplicity, generalizability, and accurate reflection of these features' influence on SalePrice.

These modeling choices effectively balance interpretability and predictive accuracy. By retaining meaningful numeric relationships and applying factors where beneficial, this approach ensures that the model captures both categorical and continuous influences on SalePrice, providing a solid foundation for a predictive model aligned with the project's goals.

Cross validation

The approach uses cross-validation with a 70/30 train-validation split to assess model performance and prevent overfitting. By training on 70% of the data and validating on 30%, this method provides a reliable accuracy estimate before submitting predictions to Kaggle.

This code uses the randomly sampled index to create a 70/30 train-validation split of the data.

```
# Randomly sample 70% of the rows in an object called index
set.seed(124)
index <- sample(x = 1:nrow(train), size = nrow(train)*.7, replace = F)

# Check
head(index) # These are the 70% randomly sampled row numbers
```

```
## [1] 1345  167 1002 1435  261  728
```

Data Splitting Using the Index

With the index created from the 70/30 random sampling, we now split the original train data into two distinct subsets: train_fold and validation_fold. This split allows us to build the model on one portion of the data (train_fold) and validate it on another (validation_fold), ensuring that our model's accuracy is tested on data it hasn't seen before.

- Training Fold (70%): The train_fold is created by subsetting the train data with index, selecting 70% of the rows for training the model.
- Validation Fold (30%): The validation_fold consists of the rows not included in index (the remaining 30%). This subset will serve as a testing ground to evaluate the model's performance. This separation gives us a realistic assessment of how well the model might perform on unseen data before making predictions on the final test set.

```
# Subset train using index to create a 70% train_fold
train_fold <- train[index, ]

# Subset the remaining rows not included in index to create a 30% validation fold
validation_fold <- train[-index, ]
```

Model Fitting and Performance Evaluation

In this section, we train our model on train_fold and then evaluate its accuracy on validation_fold, providing an estimate of the model's out-of-sample performance.

MODEL

```
# Fit example model

# Transform the target variable (log transformation for Kaggle evaluation)
train_fold <- train %>% mutate(LogSalePrice = log(SalePrice))

model <- lm(
  LogSalePrice ~ Neighborhood * GrLivArea + `1stFlrSF` * factor(OverallQual) +
    TotalBsmtSF + KitchenQual + GarageArea + YearBuilt + YearRemodAdd + BsmtFinSF1,
  data = train_fold
)

summary(model)
```

```
##
## Call:
## lm(formula = LogSalePrice ~ Neighborhood * GrLivArea + `1stFlrSF` *
##     factor(OverallQual) + TotalBsmtSF + KitchenQual + GarageArea +
##     YearBuilt + YearRemodAdd + BsmtFinSF1, data = train_fold)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81396 -0.05965  0.00790  0.06948  0.65513
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.708e+00  7.989e-01   4.642 3.79e-06 ***
## NeighborhoodBlueste           -4.359e-01  8.619e-01  -0.506  0.61316
## NeighborhoodBrDale            -5.067e-01  4.271e-01  -1.186  0.23566
## NeighborhoodBrkSide           -3.967e-02  3.466e-01  -0.114  0.90890
## NeighborhoodClearCr            3.431e-01  3.550e-01   0.966  0.33398
## NeighborhoodCollgCr            1.192e-01  3.429e-01   0.348  0.72807
## NeighborhoodCrawfor            2.790e-01  3.464e-01   0.805  0.42074
## NeighborhoodEdwards            3.061e-01  3.433e-01   0.892  0.37273
## NeighborhoodGilbert            1.918e-01  3.502e-01   0.548  0.58398
## NeighborhoodIDOTRR            -4.184e-01  3.539e-01  -1.182  0.23724
## NeighborhoodMeadowV            4.796e-02  3.508e-01   0.137  0.89126
## NeighborhoodMitchel            2.126e-01  3.471e-01   0.613  0.54030
## NeighborhoodNAmes              2.703e-01  3.417e-01   0.791  0.42894
## NeighborhoodNoRidge            5.856e-02  3.537e-01   0.166  0.86853
## NeighborhoodNPkVill            1.053e-01  4.210e-01   0.250  0.80248
## NeighborhoodNridgHt            2.736e-02  3.476e-01   0.079  0.93728
## NeighborhoodNWAmes             1.583e-01  3.466e-01   0.457  0.64796
## NeighborhoodOldTown            6.565e-02  3.422e-01   0.192  0.84790
## NeighborhoodSawyer             1.975e-01  3.456e-01   0.571  0.56779
## NeighborhoodSawyerW            4.305e-03  3.450e-01   0.012  0.99005
## NeighborhoodSomerst            4.169e-02  3.484e-01   0.120  0.90476
## NeighborhoodStoneBr            1.181e-01  3.531e-01   0.335  0.73799
## NeighborhoodSWISU              2.502e-01  3.512e-01   0.712  0.47637
## NeighborhoodTimber             1.677e-01  3.541e-01   0.473  0.63596
## NeighborhoodVeenker            4.875e-03  3.991e-01   0.012  0.99025
## GrLivArea                      3.429e-04  2.373e-04   1.445  0.14873
## `1stFlrSF`                     1.588e-04  3.364e-04   0.472  0.63701
## factor(OverallQual)2          -1.095e-01  3.879e-01  -0.282  0.77769
## factor(OverallQual)3          -1.086e-01  2.588e-01  -0.420  0.67488
## factor(OverallQual)4           3.782e-01  2.368e-01   1.597  0.11057
## factor(OverallQual)5           6.338e-01  2.349e-01   2.698  0.00706 **
## factor(OverallQual)6           6.450e-01  2.349e-01   2.745  0.00612 **
## factor(OverallQual)7           6.511e-01  2.354e-01   2.766  0.00576 **
## factor(OverallQual)8           7.428e-01  2.375e-01   3.127  0.00180 **
## factor(OverallQual)9           5.884e-01  2.684e-01   2.192  0.02852 *
## factor(OverallQual)10          1.935e+00  2.623e-01   7.378 2.76e-13 ***
## TotalBsmtSF                    1.017e-04  1.633e-05   6.230 6.17e-10 ***
## KitchenQualTA                  4.505e-02  2.396e-02   1.880  0.06030 .
## KitchenQualGd                  7.997e-02  2.596e-02   3.081  0.00210 **
## KitchenQualEx                  1.409e-01  3.151e-02   4.472 8.38e-06 ***
```

```
## GarageArea                          2.014e-04  2.271e-05   8.870  < 2e-16 ***
## YearBuilt                           1.557e-03  2.853e-04   5.458 5.69e-08 ***
## YearRemodAdd                        1.888e-03  2.479e-04   7.617 4.80e-14 ***
## BsmtFinSF1                          9.968e-05  9.591e-06  10.393  < 2e-16 ***
## NeighborhoodBlueste:GrLivArea       2.796e-04  6.124e-04   0.457  0.64802
## NeighborhoodBrDale:GrLivArea        2.967e-04  3.255e-04   0.912  0.36212
## NeighborhoodBrkSide:GrLivArea       9.127e-05  2.424e-04   0.377  0.70659
## NeighborhoodClearCr:GrLivArea      -1.205e-04  2.435e-04  -0.495  0.62079
## NeighborhoodCollgCr:GrLivArea      -6.118e-05  2.389e-04  -0.256  0.79791
## NeighborhoodCrawfor:GrLivArea      -6.171e-05  2.396e-04  -0.258  0.79680
## NeighborhoodEdwards:GrLivArea      -2.528e-04  2.393e-04  -1.057  0.29083
## NeighborhoodGilbert:GrLivArea      -8.043e-05  2.425e-04  -0.332  0.74020
## NeighborhoodIDOTRR:GrLivArea        2.782e-04  2.507e-04   1.110  0.26729
## NeighborhoodMeadowV:GrLivArea      -1.620e-04  2.474e-04  -0.655  0.51272
## NeighborhoodMitchel:GrLivArea      -1.601e-04  2.423e-04  -0.661  0.50902
## NeighborhoodNAmes:GrLivArea        -1.704e-04  2.380e-04  -0.716  0.47423
## NeighborhoodNoRidge:GrLivArea      -8.516e-06  2.404e-04  -0.035  0.97174
## NeighborhoodNPkVill:GrLivArea      -8.640e-05  3.069e-04  -0.282  0.77833
## NeighborhoodNridgHt:GrLivArea       2.271e-05  2.404e-04   0.094  0.92476
## NeighborhoodNWAmes:GrLivArea       -8.427e-05  2.400e-04  -0.351  0.72558
## NeighborhoodOldTown:GrLivArea      -8.532e-05  2.384e-04  -0.358  0.72045
## NeighborhoodSawyer:GrLivArea       -1.327e-04  2.418e-04  -0.549  0.58316
## NeighborhoodSawyerW:GrLivArea      -7.837e-06  2.395e-04  -0.033  0.97391
## NeighborhoodSomerst:GrLivArea       3.124e-06  2.419e-04   0.013  0.98970
## NeighborhoodStoneBr:GrLivArea      -1.484e-05  2.421e-04  -0.061  0.95115
## NeighborhoodSWISU:GrLivArea        -1.442e-04  2.413e-04  -0.598  0.55023
## NeighborhoodTimber:GrLivArea       -5.988e-05  2.434e-04  -0.246  0.80576
## NeighborhoodVeenker:GrLivArea       1.085e-04  2.717e-04   0.399  0.68965
## `1stFlrSF`:factor(OverallQual)2     2.917e-04  6.075e-04   0.480  0.63112
## `1stFlrSF`:factor(OverallQual)3     3.261e-04  3.548e-04   0.919  0.35812
## `1stFlrSF`:factor(OverallQual)4    -3.648e-05  3.385e-04  -0.108  0.91421
## `1stFlrSF`:factor(OverallQual)5    -2.139e-04  3.367e-04  -0.635  0.52525
## `1stFlrSF`:factor(OverallQual)6    -1.586e-04  3.366e-04  -0.471  0.63751
## `1stFlrSF`:factor(OverallQual)7    -1.143e-04  3.366e-04  -0.340  0.73425
## `1stFlrSF`:factor(OverallQual)8    -1.470e-04  3.370e-04  -0.436  0.66281
## `1stFlrSF`:factor(OverallQual)9    -6.046e-06  3.439e-04  -0.018  0.98598
## `1stFlrSF`:factor(OverallQual)10 -7.259e-04  3.422e-04  -2.121  0.03408 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1305 on 1383 degrees of freedom
## Multiple R-squared:  0.8989, Adjusted R-squared:  0.8933
## F-statistic: 161.8 on 76 and 1383 DF,  p-value: < 2.2e-16
```

```
# Get predictions for the validation fold
val_predictions <- exp(predict(model, newdata = validation_fold))

# Create functions for calculating RMSE and R-squared (necessary for estimating
# out of sample performance)

rmse <- function(actual, predicted) sqrt(mean((actual - predicted)^2))
r_squared <- function(actual, predicted) {
  1 - sum((actual - predicted)^2) / sum((actual - mean(actual))^2)
}

# Estimated out of sample RMSE, rounded to two decimal places
val_rmse <- rmse(validation_fold$SalePrice, val_predictions)
val_rmse %>% round(2)
```

```
## [1] 25125.31
```

```
# Estimated out of sample R-squared, rounded to two decimal places
val_r2 <- r_squared(validation_fold$SalePrice, val_predictions) %>% round(2)
val_r2
```

```
## [1] 0.91
```

- **Root Mean Squared Error (RMSE): $25,125.31** This value indicates that, on average, the model's predictions for SalePrice differ from the actual prices by approximately $25,125.31. This RMSE provides a measure of the model's prediction accuracy, with lower values indicating higher accuracy.
- **Coefficient of Determination (R^2) Out of Sample: 0.91** The R2 value shows that about 91% of the variance in SalePrice is explained by the model. This metric indicates that the model has a reasonably good fit, as it captures a substantial portion of the variance in house prices.

These metrics suggest that the model is performing effectively, meeting the project's target for out-of-sample R^2 of at least 0.85, and achieving a good balance between predictive accuracy (as indicated by RMSE) and variance explanation (as indicated by R^2).

Submit to Kaggle

Fitting the model using the entire train set and calculate in sample performance

```
submission_model <- lm(
  log(SalePrice) ~ Neighborhood * GrLivArea + `1stFlrSF` * factor(OverallQual) +
    TotalBsmtSF + KitchenQual + GarageArea + YearBuilt + YearRemodAdd + BsmtFinSF1,
  data = train
)


# R-squared and RMSE for this model will be the estimated in-sample model performance
summary(submission_model)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ Neighborhood * GrLivArea + `1stFlrSF` *
##     factor(OverallQual) + TotalBsmtSF + KitchenQual + GarageArea +
##     YearBuilt + YearRemodAdd + BsmtFinSF1, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81396 -0.05965  0.00790  0.06948  0.65513
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     3.708e+00  7.989e-01   4.642 3.79e-06 ***
## NeighborhoodBlueste            -4.359e-01  8.619e-01  -0.506  0.61316
## NeighborhoodBrDale             -5.067e-01  4.271e-01  -1.186  0.23566
## NeighborhoodBrkSide            -3.967e-02  3.466e-01  -0.114  0.90890
## NeighborhoodClearCr             3.431e-01  3.550e-01   0.966  0.33398
## NeighborhoodCollgCr             1.192e-01  3.429e-01   0.348  0.72807
## NeighborhoodCrawfor             2.790e-01  3.464e-01   0.805  0.42074
## NeighborhoodEdwards             3.061e-01  3.433e-01   0.892  0.37273
## NeighborhoodGilbert             1.918e-01  3.502e-01   0.548  0.58398
## NeighborhoodIDOTRR             -4.184e-01  3.539e-01  -1.182  0.23724
## NeighborhoodMeadowV             4.796e-02  3.508e-01   0.137  0.89126
## NeighborhoodMitchel             2.126e-01  3.471e-01   0.613  0.54030
## NeighborhoodNAmes               2.703e-01  3.417e-01   0.791  0.42894
## NeighborhoodNoRidge             5.856e-02  3.537e-01   0.166  0.86853
## NeighborhoodNPkVill             1.053e-01  4.210e-01   0.250  0.80248
## NeighborhoodNridgHt             2.736e-02  3.476e-01   0.079  0.93728
## NeighborhoodNWAmes              1.583e-01  3.466e-01   0.457  0.64796
## NeighborhoodOldTown             6.565e-02  3.422e-01   0.192  0.84790
## NeighborhoodSawyer              1.975e-01  3.456e-01   0.571  0.56779
## NeighborhoodSawyerW             4.305e-03  3.450e-01   0.012  0.99005
## NeighborhoodSomerst             4.169e-02  3.484e-01   0.120  0.90476
## NeighborhoodStoneBr             1.181e-01  3.531e-01   0.335  0.73799
## NeighborhoodSWISU               2.502e-01  3.512e-01   0.712  0.47637
## NeighborhoodTimber              1.677e-01  3.541e-01   0.473  0.63596
## NeighborhoodVeenker             4.875e-03  3.991e-01   0.012  0.99025
## GrLivArea                       3.429e-04  2.373e-04   1.445  0.14873
## `1stFlrSF`                      1.588e-04  3.364e-04   0.472  0.63701
## factor(OverallQual)2           -1.095e-01  3.879e-01  -0.282  0.77769
## factor(OverallQual)3           -1.086e-01  2.588e-01  -0.420  0.67488
## factor(OverallQual)4            3.782e-01  2.368e-01   1.597  0.11057
## factor(OverallQual)5            6.338e-01  2.349e-01   2.698  0.00706 **
## factor(OverallQual)6            6.450e-01  2.349e-01   2.745  0.00612 **
## factor(OverallQual)7            6.511e-01  2.354e-01   2.766  0.00576 **
## factor(OverallQual)8            7.428e-01  2.375e-01   3.127  0.00180 **
## factor(OverallQual)9            5.884e-01  2.684e-01   2.192  0.02852 *
## factor(OverallQual)10           1.935e+00  2.623e-01   7.378 2.76e-13 ***
## TotalBsmtSF                     1.017e-04  1.633e-05   6.230 6.17e-10 ***
## KitchenQualTA                   4.505e-02  2.396e-02   1.880  0.06030 .
## KitchenQualGd                   7.997e-02  2.596e-02   3.081  0.00210 **
## KitchenQualEx                   1.409e-01  3.151e-02   4.472 8.38e-06 ***
```

```
## GarageArea                          2.014e-04  2.271e-05   8.870  < 2e-16 ***
## YearBuilt                           1.557e-03  2.853e-04   5.458 5.69e-08 ***
## YearRemodAdd                        1.888e-03  2.479e-04   7.617 4.80e-14 ***
## BsmtFinSF1                          9.968e-05  9.591e-06  10.393  < 2e-16 ***
## NeighborhoodBlueste:GrLivArea       2.796e-04  6.124e-04   0.457  0.64802
## NeighborhoodBrDale:GrLivArea        2.967e-04  3.255e-04   0.912  0.36212
## NeighborhoodBrkSide:GrLivArea       9.127e-05  2.424e-04   0.377  0.70659
## NeighborhoodClearCr:GrLivArea      -1.205e-04  2.435e-04  -0.495  0.62079
## NeighborhoodCollgCr:GrLivArea      -6.118e-05  2.389e-04  -0.256  0.79791
## NeighborhoodCrawfor:GrLivArea      -6.171e-05  2.396e-04  -0.258  0.79680
## NeighborhoodEdwards:GrLivArea      -2.528e-04  2.393e-04  -1.057  0.29083
## NeighborhoodGilbert:GrLivArea      -8.043e-05  2.425e-04  -0.332  0.74020
## NeighborhoodIDOTRR:GrLivArea        2.782e-04  2.507e-04   1.110  0.26729
## NeighborhoodMeadowV:GrLivArea      -1.620e-04  2.474e-04  -0.655  0.51272
## NeighborhoodMitchel:GrLivArea      -1.601e-04  2.423e-04  -0.661  0.50902
## NeighborhoodNAmes:GrLivArea        -1.704e-04  2.380e-04  -0.716  0.47423
## NeighborhoodNoRidge:GrLivArea      -8.516e-06  2.404e-04  -0.035  0.97174
## NeighborhoodNPkVill:GrLivArea      -8.640e-05  3.069e-04  -0.282  0.77833
## NeighborhoodNridgHt:GrLivArea       2.271e-05  2.404e-04   0.094  0.92476
## NeighborhoodNWAmes:GrLivArea       -8.427e-05  2.400e-04  -0.351  0.72558
## NeighborhoodOldTown:GrLivArea      -8.532e-05  2.384e-04  -0.358  0.72045
## NeighborhoodSawyer:GrLivArea       -1.327e-04  2.418e-04  -0.549  0.58316
## NeighborhoodSawyerW:GrLivArea      -7.837e-06  2.395e-04  -0.033  0.97391
## NeighborhoodSomerst:GrLivArea       3.124e-06  2.419e-04   0.013  0.98970
## NeighborhoodStoneBr:GrLivArea      -1.484e-05  2.421e-04  -0.061  0.95115
## NeighborhoodSWISU:GrLivArea        -1.442e-04  2.413e-04  -0.598  0.55023
## NeighborhoodTimber:GrLivArea       -5.988e-05  2.434e-04  -0.246  0.80576
## NeighborhoodVeenker:GrLivArea       1.085e-04  2.717e-04   0.399  0.68965
## `1stFlrSF`:factor(OverallQual)2     2.917e-04  6.075e-04   0.480  0.63112
## `1stFlrSF`:factor(OverallQual)3     3.261e-04  3.548e-04   0.919  0.35812
## `1stFlrSF`:factor(OverallQual)4    -3.648e-05  3.385e-04  -0.108  0.91421
## `1stFlrSF`:factor(OverallQual)5    -2.139e-04  3.367e-04  -0.635  0.52525
## `1stFlrSF`:factor(OverallQual)6    -1.586e-04  3.366e-04  -0.471  0.63751
## `1stFlrSF`:factor(OverallQual)7    -1.143e-04  3.366e-04  -0.340  0.73425
## `1stFlrSF`:factor(OverallQual)8    -1.470e-04  3.370e-04  -0.436  0.66281
## `1stFlrSF`:factor(OverallQual)9    -6.046e-06  3.439e-04  -0.018  0.98598
## `1stFlrSF`:factor(OverallQual)10   -7.259e-04  3.422e-04  -2.121  0.03408 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1305 on 1383 degrees of freedom
## Multiple R-squared:  0.8989, Adjusted R-squared:  0.8933
## F-statistic: 161.8 on 76 and 1383 DF,  p-value: < 2.2e-16
```

The model was fitted to the entire training set to assess its in-sample performance, providing estimates of how well it explains and predicts SalePrice within the data it was trained on. The following metrics summarize the model's fit:

- **R2 (Coefficient of Determination):** The in-sample R^2 for the log model, rounded to 0.89, indicates that approximately 89% of the variance in SalePrice is explained by the model within the training set. This suggests a strong fit, meaning the selected predictors collectively account for a substantial amount of variation in house prices.

- **RMSE (Root Mean Squared Error):** The in-sample RMSE with log model, rounded to 0.13, corresponds to a percentage error when converted back to the original unit. This value suggests that, on average, the model's predictions for SalePrice differ from the actual values in percentage by approximately 0.13. This indicates the models predictions are close to the actual values. The RMSE for the insample is using logarithim for the sale price which explains the large difference between the Out of sample and In sample RMSE. Regardless, lower RMSE values indicate better predictive accuracy, so this metric shows the model has a reasonable level of precision within the training set.

In summary, these metrics provide a reliable estimate of the model's accuracy and explanatory power within the training data, with a high R^2 and a moderate RMSE indicating that the model captures key patterns and relationships effectively.

Applying Consistent Transformations to the Test Set

To ensure consistency between the training and test data, the same transformations applied to the training set must be replicated on the test set. This alignment is crucial, as differences in data preparation can lead to inconsistencies in predictions.

```
# Copy changes made to train in Test
test <- test %>%
  mutate(
    Neighborhood = factor(Neighborhood),
    KitchenQual = factor(KitchenQual),
    OverallQual = factor(OverallQual)
  )
```

Checking for Missing Values in Selected Predictors

Before making predictions, it's essential to verify that there are no missing values in the selected predictors within the test set. Missing values would result in NA predictions, which could impact the model's performance and lead to incomplete submission results.

```
count_missings <- function(x) sum(is.na(x))
# Check for NAs
test |>
  select(Neighborhood, GrLivArea, `1stFlrSF`, OverallQual, GarageArea, TotalBsmtSF, Year
Built, YearRemodAdd, BsmtFinSF1, KitchenQual, PoolArea) |>
  summarize_all(count_missings)
```

```
## # A tibble: 1 × 11
##   Neighborhood GrLivArea `1stFlrSF` OverallQual GarageArea TotalBsmtSF YearBuilt
##          <int>     <int>      <int>       <int>      <int>       <int>     <int>
## 1            0         0          0           0          1           1         0
## # ℹ 4 more variables: YearRemodAdd <int>, BsmtFinSF1 <int>, KitchenQual <int>,
## #   PoolArea <int>
```

Discussion of missing data (scope, proposed solution)

There are 4 variables that have missing values in our model:

1. KitchenQual
2. GarageArea

3. TotalBsmtSF
4. BsmtFinSF1

Since they are integers, our solution was to impute the numbers by using the median in order to clean the data.

```
test <- test %>%
  mutate(
    KitchenQual = replace_na(KitchenQual, 'TA'),
    GarageArea = replace_na(GarageArea, median(GarageArea, na.rm = TRUE)),
    TotalBsmtSF = replace_na(TotalBsmtSF, median(TotalBsmtSF, na.rm = TRUE)),
    BsmtFinSF1 = replace_na(BsmtFinSF1, median(BsmtFinSF1, na.rm = TRUE))
  )
```

```
test <- test %>%
  mutate(
    KitchenQual = replace_na(KitchenQual, 'TA'),
    GarageArea = replace_na(GarageArea, median(GarageArea, na.rm = TRUE)),
    TotalBsmtSF = replace_na(TotalBsmtSF, median(TotalBsmtSF, na.rm = TRUE)),
    BsmtFinSF1 = replace_na(BsmtFinSF1, median(BsmtFinSF1, na.rm = TRUE))
  )

# Transforming the target variable
train <- train %>% mutate(LogSalePrice = log(SalePrice))
```

We decided to impute the numerical data since it is numeric. Deleting the missing rows is not suitable, since it would reduce the data output by deleting the rows entirely. Furthermore, Kaggle will not accept any submissions that have deletions for the data.

As for categorical variables: KitchenQual has 1 NA in the test set and it is categorical, we decided to replace it with the most common quality type: ‘TA’

Generating Predictions for SalePrice in the Test Set

The model is used to predict the SalePrice for each observation in the test set

```
# Predict on the test set for Kaggle submission
test_predictions <- exp(predict(submission_model, newdata = test))  # Back-transform fro
m log
#Displays the first few predicted values, as expected.
head(test_predictions)
```

```
##        1        2        3        4        5        6
## 133582.4 157800.9 181316.0 194638.6 199940.2 172711.4
```

Formatting the Submission File for Kaggle

```
submission <- test %>%
  select(Id) %>%
  mutate(SalePrice = test_predictions)

# Check
head(submission)
```

```
## # A tibble: 6 × 2
##      Id SalePrice
##   <dbl>     <dbl>
## 1  1461   133582.
## 2  1462   157801.
## 3  1463   181316.
## 4  1464   194639.
## 5  1465   199940.
## 6  1466   172711.
```

```
# write to csv
write.csv(submission, "Kaggle_Group_Project.csv", row.names = F)
```

Kaggle score = 0.15 (approximately)

Final Conclusion

This project aimed to build an effective model to predict housing prices in Ames, Iowa, using several predictors. The model's performance was evaluated as follows:

In-Sample Performance:

The R^2 of the model is 0.89, indicating that the model explains approximately 89% of the variance in SalePrice within the training set. This reflects a strong fit and suggests that the chosen predictors and interactions are effectively modeling key relationships in the data.

RMSE:

The in-sample RMSE, computed on the log-transformed SalePrice, is 0.13. reverted into percentage terms, this means that the model's predictions differ from actual values by about 13% on average within the training data. This lower RMSE, as compared to the out-of-sample RMSE in fixed units, can be explained by the use of a logarithmic transformation. This low RMSE indicates that the model's predictions are close to the actual data in training.

Out-of-Sample Performance:

R^2 is Coefficient of Determination:

The out-of-sample R^2 is 0.91, indicating that the model generalizes well for unseen data, explaining 91% of the variance in SalePrice on the validation set. This model outperforms the benchmark from the project the R^2 is > 0.85, therefore having strong generalization.

RMSE:

The out-of-sample RMSE is 25,125.31, meaning that, on average, the model's predictions differ from actual house prices by about 25,125 dollars on unseen data. The given value is actually a quite realistic estimate of the prediction error when the model is used for new observations.

Why This Model is Effective

Efficiency with a Small Number of Predictors:

This model yields excellent results by using a relatively small number of predictors. The inclusion of two interaction features, Neighborhood * GrLivArea and 1stFlrSF * OverallQual, enhances the predictive capability of the model without adding too much complexity.

Generalizability:

The R^2 values being consistent and high for both in-sample and out-of-sample demonstrate that this model generalizes pretty well. The validation RMSE assures that the model is not overfitting to the train data.

Competitive Performance:

The out-of-sample RMSE of the model aligns with the target score at Kaggle, while the submission log RMSE is competitive.

Contributions

- Gaby: Focused on the model and predictions, graphs, conclusions
- Ali: Focused on the model and predictions, graphs, conclusions
- Tami: Data cleaning and wrangling, graphs, introductions
- Sonia: Data cleaning and wrangling, graphs, introductions