Hubert Nguyen (hubnguyen@ucdavis.edu),
Tyler Guo, (tylguo@ucdavis.edu)
Andy Yang (anyang@ucdavis.edu),
Freddie Kiessling (ftkiessling@ucdavis.edu)

Final Project: Data Analysis on Ames Dataset

Background:

Dataset:

The Ames Housing dataset is a rich dataset with 79 explanatory variables describing (but not limited to) the physical features of residential homes in Ames, Iowa, between 2006 and 2010.
The dataset is available on Kaggle, which is a platform for predictive modeling and analytics competitions: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

Background:

The housing market is complex, influenced by a multitude of factors ranging from the physical attributes of a property to the economic climate. Ames, Iowa, serves as an interesting case study due to its unique mix of urban and rural properties. The goal is to dissect these influences and construct a predictive model that could serve stakeholders such as homeowners, buyers, and real estate agents in making informed decisions. The data being worked with will be relating to characteristics and/or factors that contribute to housing prices. There are 79 explanatory variables that contribute to the cost of housing, each contributing a certain and different amount. The 'SalePrice' variable will be the values indicating the property's sale price and will be the variable we are trying to predict. The data being downloaded will consist of a train.csv (training set) and a test.csv (test set) to have more accurate results.

Goals:

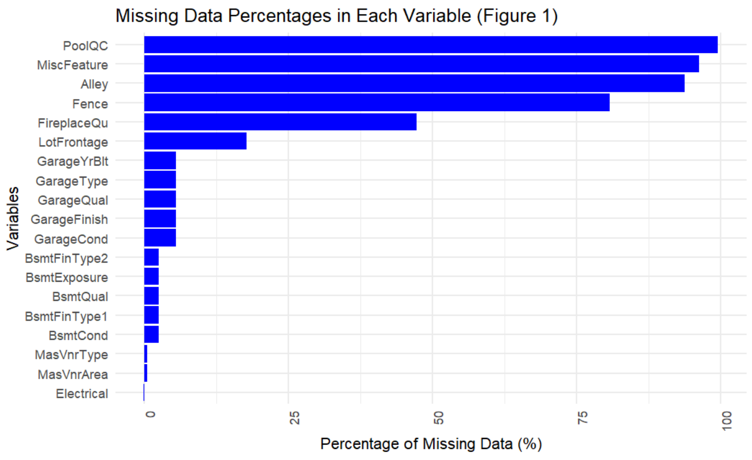
We will be building different linear and logistic regression models based on a multitude of predictors, such as kitchen quality, exterior quality, etc., using a variety of statistical learning techniques, and conducting a comprehensive and nuanced examination of the data through deep analysis. Before we can do this, we must pre-process and clean the original data. We will be focusing on 3 linear models using linear regression techniques and machine learning techniques. Techniques that will be used will include model selection (employing a forward selection algorithm), using AIC based on forward & backward selection, k-fold cross validation (both linear and logistic regression), diagnostics for linear regression, examining and analyzing summary statistics for models (RSME, R-squared, MAE, RMSE SD, etc), performing Shapiro-Wilk Test, Residual vs. Fitted & Breusch-Pagan Test, Hosmer-Lemeshow test, McNemar's Test, backward selection algorithm, and diagnostics regarding the performance of logistic regression models (accuracy, sensitivity, specificity, precision, F1 score, ROC, AIC). Before we build these models, we must deduce which variables are most influential and best indicate changes within the housing prices. After building these models, we will create visualizations and analyze them as well. We are primarily using the base plotting package on R, ggplot2, gridExtra, corrplot, car, pROC for plotting and tidyr, tidyverse, dplyr, caret, ResourceSelection, for data cleaning and data analysis. There will be explanations throughout the report tying to our research question and answering it as accurately as possible. In addition, we will check for the assumptions of linear regression: linearity, homoscedasticity, independence, etc. The rest of the report will go more in depth of each of these techniques/methodologies.

Research Questions:

- What are the key factors/variables that affect housing prices in Ames, Iowa?
- What are valuable statistics that can be pulled from the dataset relating to SalePrice?

Data Preprocessing and Exploration:

Before doing any analysis, we first need to clean and process the original data. The overall goal of this section is to simplify the dataset through reduction and transformation of the variables so that we can focus on the more significant predictor variables in our analysis. The first step of this process is handling the NA/missing values in the data set See Figure 1 for a graphic displaying such missing values. The methods used to deal with such values included replacing the NA variables with other values depending on the column we are looking at, examples include replacing with “0” or “None”, other methods include converting the entire column into a binary variable adjusted based off the specific needs of the column or simply deleting the column from the dataset completely.



After handling the NA values, the remaining data was parsed in two sections, numerical and categorical. For categorical variables, the summarize function was used to calculate the mean sale price for each of the categories of each variable. This was used in the determination process of transforming individual variables. Below is a summary of what was done to each variable. See the code appendix for more details regarding the process.

Numerical Variables:

Variables Removed or Altered:

- ID Column: Removed (considered irrelevant).
- Pool Variables: PoolArea, PoolQC removed; new binary variable HasPool created.
- Lot Frontage: LotFrontage removed.
- Date Variables: MoSold removed; new variables AgeAtSale and YearsSinceRemodel added.
- Masonry Veneer Area: NA values in MasVnrArea set to 0.
- Basement and Above Grade Square Footage: BsmtUnfSF, BsmtFinSF1, BsmtFinSF2, X1stFlrSF, X2ndFlrSF removed.
- Bathrooms: BsmtFullBath, BsmtHalfBath, FullBath, HalfBath combined into BsmtBaths and TotalBaths.
- Outside Square Footage: WoodDeckSF, OpenPorchSF, EnclosedPorch, X3SsnPorch, ScreenPorch combined into TotalOutSideSF.

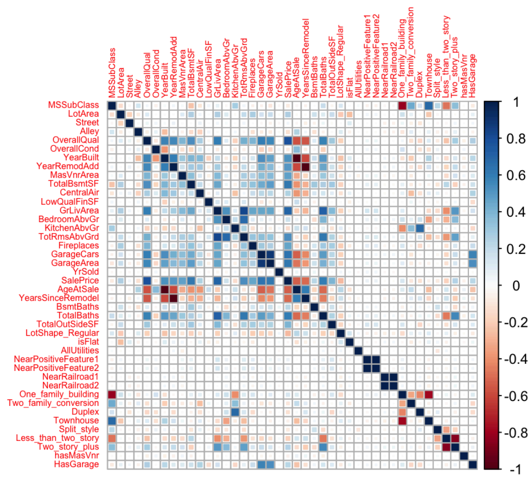
Categorical Variables:

Variables Removed or Altered:

- Street: Converted to a binary variable.
- Alley: Converted to a binary variable.
- Lot Variables: LotShape and LandContour replaced with new binary variables LotShape_Regular and isFlat.
- Utilities: Utilities replaced with binary variable AllUtilities.
- Electrical: Electrical removed.
- Condition Proximity: Condition1 and Condition2 removed; new binary variables NearPositiveFeature1, NearPositiveFeature2, NearRailroad1, NearRailroad2 created.
- Building Type: BldgType replaced with binary variables for different building types.
- House Style: HouseStyle replaced with binary variables for different styles.
- Roof, Exterior, Foundation, and Masonry Veneer: RoofStyle, RoofMatl, Exterior1st, Exterior2nd, Foundation, MasVnrType removed or transformed; new binary variable hasMasVnr created.

Unchanged Variables:

- MSSubClass, LotArea, OverallQual, OverallCond, TotalBsmtSF, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageCars, GarageArea, LotConfig, MSZoning, LandSlope, Neighborhoods, ExteriorCond, ExteriorQual, HeatingQC, KitchenQual.



Now having prepared our dataset, we employ tools such as the correlation matrix from the corrplot package (pictured right) to steer our focus to the most significant predictors. In addition to the matrix we also applied the cor() function to our training set to find the specific correlation coefficients. The variables identified with the highest correlation coefficients to Sale Price were: Overall quality of the house (OverallQual), above grade living area square feet (GrLivArea), size of garage in car capacity (GarageCars), size of garage in square feet (GarageArea), total square feet of basement area (TotalBsmtSF), total number of bathrooms (TotalBaths), total rooms above grade (excluding bathrooms) (TotRmsAbvGrd), age of the house at sale (AgeAtSale), original construction date (YearBuilt), years since remodeling (YearsSinceRemodel), remodeling date (YearRemodAdd) and masonry veneer area in square feet (MasVnrArea). These variables, showing a strong linear relationship with the Sale Price, were selected as initial candidates for our regression model.

Linear Regression:

Model Selection:

To identify the most predictive model, we employed a forward selection algorithm based on the Akaike Information Criterion (AIC). This method systematically adds variables to the model, starting with the most strongly correlated one, and continues to add variables that provide a significant improvement to the model's predictive power, as indicated by a lower AIC value. The goal is to find a model that has the right balance of complexity and goodness of fit. Too simple, and it won't capture important patterns; too complex, and it may not generalize well.

Rationale for Using AIC in Model Selection:

We incorporated AIC as it provides a good balance by penalizing the complexity of the model while rewarding its goodness of fit. Essentially, AIC evaluates both the explanatory power of the model and its simplicity. There is also a penalty that increases with the number of parameters, thus discouraging needless complexity. By balancing model complexity against the quality of the model fit, AIC serves us as a useful guide for our model selection.

Algorithm Explanation:

The algorithm starts with the simplest model, SalePrice ~ 1, which means only the intercept is included, with no predictors. Various predictors are tested for addition, and the one that lowers the AIC the most (OverallQual) is added to the model. In each step, the process considers adding another predictor from our list of top variables (based on preliminary analysis highest-correlation coefficients). The change in AIC is then reported for each potential addition. The predictor that results in the largest drop in AIC (and hence the lowest AIC) is added to the model. This process continues, adding one variable at a time, and re-evaluating the AIC. The process stops when adding any more predictors does not meaningfully lower the AIC. In our case, the final model includes OverallQual, GrLivArea, KitchenQual, GarageCars, TotalBsmtSF, and ExterQual. The stepwise process tried adding GarageArea, but it resulted in an AIC of 30680, which is higher than the AIC without it (30678.74), so it was not included in the final model. By the start of the process, we had an AIC of 32946.74 and by the end of the process we reduced this number to 30678.74 The final model selected by this process is considered the "best" among the models evaluated, based on the AIC criterion. It includes the predictors that the stepwise process determined to be most valuable in explaining the variance in SalePrice while avoiding overfitting.

Cross-Validation:

To assess the model's robustness and guard against overfitting, we performed k-fold cross-validation, with k = 10. This technique divides the data into 'k' subsets and iteratively uses one subset for validation while the remaining subsets are used for training the model. This process ensures that the model's performance is tested across all data points and provides a more generalizable measure of its predictive accuracy.

Linear Regression Models for K-fold (top 3 models):

To further investigate our model selection we performed the k-fold (with k=10) on the top 3 linear regression models (based on our AIC algorithm). We included our best model: (**Step: AIC=30678.74)**: **Model 1**: SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + ExterQual. This model has the lowest AIC value of 30678.74, making it the best model among those evaluated. The **Second Best Model (Step: AIC=30715.39)**: **Model 2**: SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF. The second-best model includes all the variables from the best model except ExterQual. Its AIC value is slightly higher at 30715.39. **Third Best Model (Step: AIC=30812.04)**: **Model 3**: SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars. The third-best model excludes TotalBsmtSF and ExterQual from the best model. It has an AIC value of 30812.04. Sometimes the model with the best AIC might not have the best performance on unseen data. AIC is a good indicator of model quality, but it's based on the training data and doesn't guarantee the best predictive accuracy. Comparing the top models can give us a more rounded view of how they might perform in the real world.

K-fold CV Results:

We performed k-fold cross-validation on our best model(s) as it helps us understand how well the model generalizes to new, unseen data.

	Model 1	Model 2	Model 3
RMSE	36465.02	36873.85	38051.26
R-squared	0.7925	0.7872	0.7721
MAE	23136.1	23642.17	25503.92
RMSE SD	8420.544	8124.157	5860.588
R-squared SD	0.0850	0.0820	0.0601
MAE SD	2011.868	2176.002	2239.503

Interpretation:

- Model 1 has the lowest RMSE and the highest R-squared, indicating it performs the best in terms of both error and the proportion of variance explained. Its MAE is also the lowest, suggesting better average prediction accuracy.
- Model 2 shows slightly higher RMSE and MAE, and slightly lower R-squared, but these differences are relatively small.
- Model 3 has the highest RMSE and MAE and the lowest R-squared, indicating it performs less effectively compared to the other two models.

Conclusion

- Model 1 appears to be the best model among the three based on these metrics. It offers a better balance of model complexity and predictive accuracy.

- While Model 2 is close in performance, Model 1's slightly better metrics across the board make it a preferable choice.

- Model 3, being the simplest, has the lowest performance and might be too simplistic for the complexity inherent in the data.

The results from our k-fold cross-validation align well with the outcomes of the AIC-based forward selection process. This congruence between AIC selection and cross-validation performance is often expected, but not guaranteed, as these methods assess model quality from slightly different perspectives: AIC (Akaike Information Criterion) focuses on the trade-off between the goodness of fit of the model and its complexity. Cross-validation assesses a model's ability to generalize to an independent dataset. It provides a more direct measurement of how well a model performs on unseen data. Metrics like RMSE, R-squared, and MAE give a practical sense of prediction error and variance explanation.

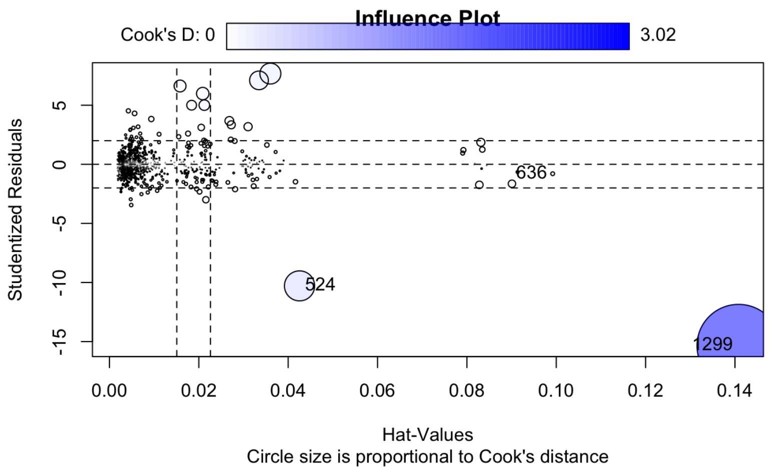
The results of both tests match. This reassures our confidence in our best model (SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + ExterQual) as both AIC-based selection and cross-validation point towards the same model as being optimal. It suggests that the model not only balances complexity and fit well (as per AIC) but also generalizes effectively to new data (as indicated by cross-validation).

Diagnostics:

Post-model selection, we conducted diagnostic tests to validate the assumptions of linear regression. This included examining the normality of residuals, checking for homoscedasticity (constant variance of residuals), and looking for any influential outliers. Ensuring these assumptions hold is crucial for the validity of the model's predictions.

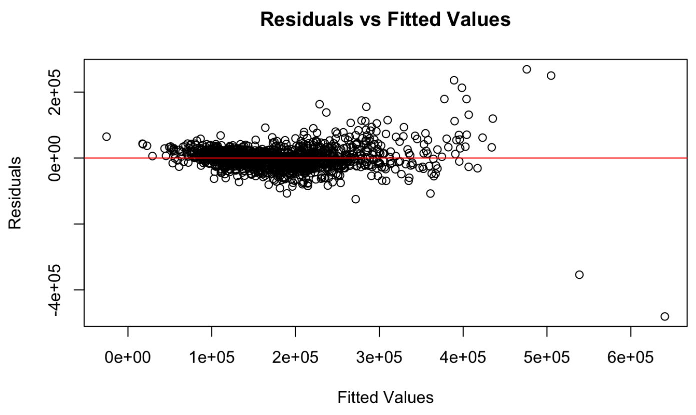
To test our multicollinearity in our multiple regression multiples, we applied a Variance Inflation Factor (VIF). OverallQual: VIF = 3.123781, Adjusted VIF = 1.767422, GrLivArea: VIF = 1.656540, Adjusted VIF = 1.287066, KitchenQual: VIF = 3.746110, Adjusted VIF = 1.246226, GarageCars: VIF = 1.730260, Adjusted VIF = 1.315394, TotalBsmtSF: VIF = 1.557468, Adjusted VIF = 1.247986, ExterQual: VIF = 4.276028, Adjusted VIF = 1.274012. A VIF value greater than 10 is typically considered to indicate high multicollinearity. However, more conservative thresholds like 5 are often used to ensure more stringent checks against multicollinearity. The highest VIF is for ExterQual, but it's still below the threshold, suggesting that while there might be some correlation with other variables, it is not to a degree that would warrant major concerns about multicollinearity in our model.

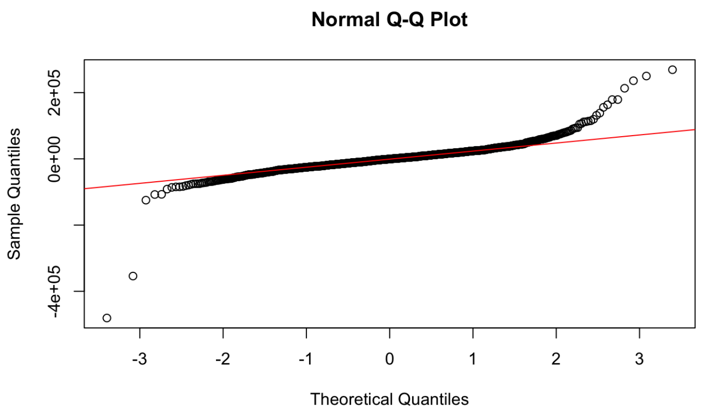
In our Influence Plot (Outliers and High-Leverage Points): Specific points (e.g., 524, 636, 1299) have been identified with significant studentized residuals or high Cook's distances. Interpretation: These points may be outliers or have high leverage, potentially influencing the model disproportionately. It's worth investigating these points to see if they are data entry errors, require transformation, or are valid extreme values.



Normality of Residuals:

We performed a Shapiro-Wilk Test Result and observed a QQ-plot observation. Our results: W = 0.80807, p-value < 2.2e-16, indicate a significant deviation from normality. In other words, the distribution of residuals does not align well with what would be expected if they were normally distributed. This specific pattern in the Q-Q plot suggests that the residuals have a heavier upper tail than a normal distribution would. This means there are more extreme values (larger residuals) than you would expect under normality. Given this violation, we might consider further applying data transformations like log or square root to the response variable or predictors might help achieve normality. We also would need to reevaluate how critical the assumption of normality is for our specific analysis and decisions based on the model. For large sample sizes, the Central Limit Theorem can sometimes mitigate concerns about non-normality.





Homoscedasticity Assumption:

We performed a Residual vs. Fitted & Breusch-Pagan Test. Our Breusch-Pagan Test Result shows a BP = 398.38, df = 10, p-value < 2.2e-16. Interpretation: The very low p-value in the Breusch-Pagan test indicates that there is significant heteroscedasticity in the model. This means that the assumption of constant variance in the residuals is violated. The residual vs fitted plot confirms this, the spread of residuals (the width of the scatter plot points) increases or decreases as you move along the fitted values, it suggests a pattern of heteroscedasticity, meaning there are potential inconsistencies in the variance of the residuals as it is not constant across the range of fitted values. For addressing non-normality, transformations such as logarithmic or square root transformations could be applied to the response variable or predictors. This can often help in aligning the distribution of residuals more closely with a normal distribution. Similarly, for heteroscedasticity, transformations or adopting a different variance structure in the model (like using Generalized Least Squares) can be effective. However, it is essential to consider the context and purpose of the model. In large datasets, such as in our case, the Central Limit Theorem suggests that the impact of non-normality on the regression estimates may be less critical, especially for inferential statistics like hypothesis tests and confidence intervals. Furthermore, in predictive modeling, the primary concern is often the accuracy of predictions rather than strict adherence to assumptions. In such cases, the model's ability to reliably predict new observations can be more important than meeting all classical assumptions.

Practical Implications for Our Housing Prediction Model:

In the context of our housing prediction model, the primary goal is to predict sale prices accurately. The non-normality and heteroscedasticity issues, while important, might not critically impair the model's predictive performance. It would be beneficial to test the model's predictions against an independent validation dataset to assess its real-world applicability. If the model demonstrates strong predictive performance, then the violation of these assumptions may be less concerning. Nonetheless, it is always valuable to explore data transformations or alternative modeling approaches that might better align with the assumptions, as this could potentially enhance the model's predictive accuracy and interpretability.

Logistic Regression:

- Model Selection (backwards algorithm, AIC)
- Cross validation
- Model Diagnostics

Model Selection:

Logistic Regression with Backward Selection:

In the logistic regression analysis, our goal was to predict the category of house prices – high or low – based on various predictors. We categorized the sale prices into 'High' and 'Low' based on the median sale price. This binary categorization was then encoded numerically with '1' representing 'High' and '0' representing 'Low'.

Original Model Specification:

The initial full model included an extensive range of predictors: Overall Quality (OverallQual), Above Ground Living Area (GrLivArea), External Quality (ExterQual), Kitchen Quality (KitchenQual), Number of Cars that can fit in Garage (GarageCars), Garage Area (GarageArea), Total Basement Area (TotalBsmtSF), Neighborhood, Total Number of Bathrooms (TotalBaths), Basement Quality (BsmtQual), Total Rooms Above Ground (TotRmsAbvGrd), and Age at Sale (AgeAtSale). This comprehensive model was designed to capture a wide spectrum of factors that could potentially influence the price category of a house.

Backward Selection Algorithm:

To refine the model, we employed a backward selection process using the Akaike Information Criterion (AIC) as a guiding metric. This approach began with the full model, including all the predictors. The stepAIC function from the MASS package was then used to systematically remove the least significant predictor at each step. After each removal, the model's AIC was recalculated to see if it improved. The process continued iteratively, removing one variable at a time, until no further improvement in AIC was observed. The aim was to arrive at a simpler model that still adequately explains the data without unnecessary complexity.

The backward selection process helps in identifying the most significant variables for the prediction task while eliminating those that do not contribute meaningfully to the model. This results in a more parsimonious model, which typically offers better interpretability and generalizability.

In contrast to the forward selection algorithm, we now look at the last 3 models the algorithm produces to determine the best 3 models for further analysis. **The Final Model (Step with the Lowest AIC)**: This model has the combination of predictors that results in the lowest AIC, suggesting it's the best balance of model complexity and goodness of fit. The last model model: PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual had the AIC: 663.94. **Model Just Before the Last Predictor Was Removed**: This model includes all predictors from the final model plus the last predictor that was removed. The last predictor removed: TotRmsAbvGrd (Based on the step before the final model). Model: PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual + TotRmsAbvGrd. This model had the AIC: 665.04 **Model Before the Second-Last Predictor Was Removed**: This model includes all predictors from the second model plus the second-last predictor that was removed. The second-last predictor removed: AgeAtSale (Based on the step before the second model). Model: PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual + TotRmsAbvGrd + AgeAtSale. This model had the AIC: 667.01.

K-Fold Cross-Validation for Logistic Regression Models:

Process:

To further evaluate the performance of our logistic regression models, we conducted 10-fold cross-validation. This technique involves dividing the dataset into ten subsets (folds). In each round of validation, one subset is used as the test set, and the remaining nine are combined to form the training set. The model is trained on the training set and then validated on the test set. This process is repeated ten times, with each subset serving as the test set once. The results from these ten rounds are then averaged to provide a comprehensive assessment of the model's performance.

The approach and metrics used in the k-fold cross-validation for logistic regression differ from those used in linear regression. Our choice of metrics was guided by the nature of the predictive task. Logistic regression's binary outcomes require metrics that assess classification accuracy (ROC AUC, Sensitivity, Specificity), while linear regression's continuous outcomes necessitate metrics that measure error and variance explanation (RMSE, R-squared). Using the same metrics for both models would be inappropriate due to their differing objectives and types of outcome variables.

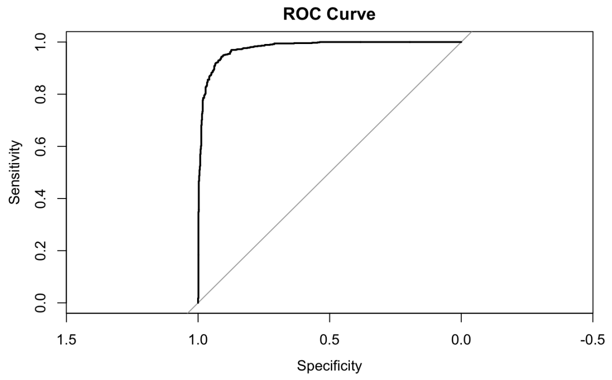
K-Fold Results for Logistic Regression:

We applied this cross-validation method to the top three models derived from our backward selection process. The performance metrics focused on were the Receiver Operating Characteristic (ROC) Area Under Curve (AUC), Sensitivity, and Specificity. These metrics provide a balanced view of model performance, especially for binary classification problems like ours.

Model 1 (OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual): ROC: 0.9671 (SD: 0.0131), Sensitivity: 0.9207 (SD: 0.0288), Specificity: 0.9176 (SD: 0.0273)

Model 2 (Adds TotRmsAbvGrd to Model 1): ROC: 0.9670 (SD: 0.0121), Sensitivity: 0.9181 (SD: 0.0287), Specificity: 0.9149 (SD: 0.0375)

Model 3 (Adds AgeAtSale to Model 2): ROC: 0.9676 (SD: 0.0181), Sensitivity: 0.9166 (SD: 0.0419), Specificity: 0.9149 (SD: 0.0321)



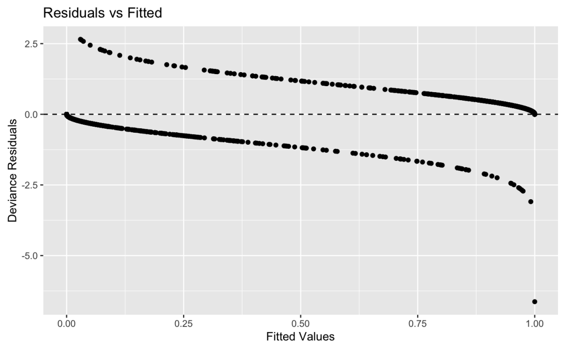
Interpretation:

The ROC values across all three models are consistently high, around 0.967, which indicates excellent model performance in distinguishing between high and low sale prices. The Sensitivity and Specificity values are also comparable across models, with Model 1 showing a slightly higher Sensitivity. This indicates that Model 1 is marginally more effective at correctly identifying houses in the 'High' price category. The ROC curve is a graphical representation that plots the True Positive Rate (Sensitivity) against the False Positive Rate (1 - Specificity) at various threshold settings. A ROC value of 0.967 means that the models have an excellent ability to discriminate between the two categories ('High' and 'Low' sale prices) across a wide range of thresholds.

Diagnostics:

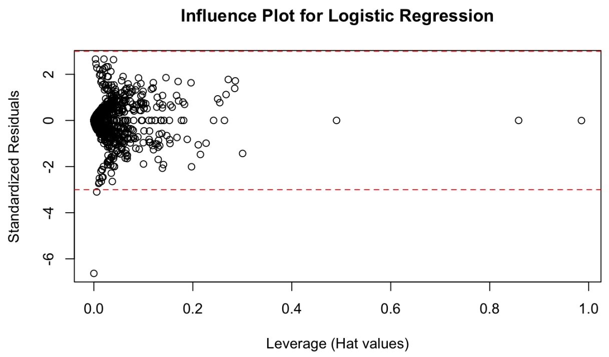
The diagnostics for logistic regression are fundamentally different from those for linear regression. This stems from the intrinsic nature of logistic regression, which deals with binary outcomes and probability predictions, as opposed to the continuous outcomes in linear regression. Unlike linear regression, where residuals are expected to be normally distributed, logistic regression does not operate under this assumption. Model Performance Metrics: The performance of logistic regression models is typically assessed using classification-specific metrics like accuracy, sensitivity (recall), specificity, precision, and F1 score, all derived from the confusion matrix. Additionally, the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are crucial for evaluating the model's ability to distinguish between binary outcomes.

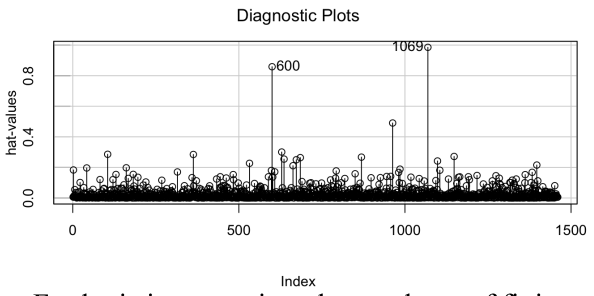
In the evaluation of the logistic regression model, the confusion matrix revealed that 676 instances were correctly predicted as 'Low', while 674 instances were correctly identified as 'High', demonstrating a high level of accuracy in classification. Misclassifications included 56 instances predicted as 'High' but actually 'Low', and 54 instances predicted as 'Low' but actually 'High'. The model achieved an accuracy of 92.47%, significantly higher than the no information rate of 50.14%, as indicated by a p-value of less than 2e-16. This suggests that the model's predictions are substantially better than random guessing. The Kappa statistic of 0.8493 further confirms strong agreement between predicted and actual classifications, beyond what would be expected by chance. Sensitivity and specificity values were both over 92%, indicating the model's robustness in correctly identifying both 'Low' and 'High' price categories. The balanced accuracy, averaging sensitivity and specificity, stood at 92.47%, confirming the model's consistent performance across both classes. Moreover, the McNemar's Test P-Value of 0.924 suggests balanced misclassification in both directions. This combination of high accuracy, balanced sensitivity, and specificity, along with a strong Kappa value, underscores the model's efficacy in predicting house price categories. The ROC curve analysis reinforced these findings. The curve closely approached the top left corner, reflecting the model's excellent ability to differentiate between 'High' and 'Low' price categories. The AUC value, a crucial summary measure of the ROC curve, indicated a strong model performance in classification tasks. This amalgamation of high accuracy, balanced sensitivity, and specificity, coupled with a strong Kappa value and an impressive ROC curve, collectively underscore the efficacy of our logistic regression model in predicting house price categories.

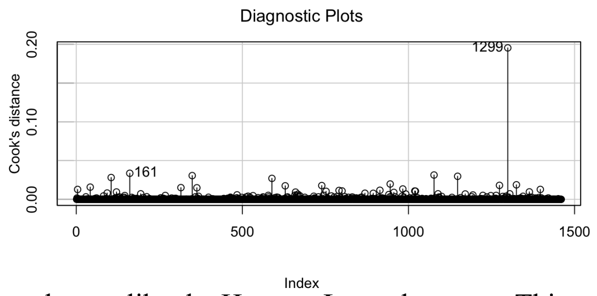


Influence and Leverage:

Similar to linear regression, logistic regression diagnostics also involve identifying influential observations that might disproportionately affect the model. Influence plots can highlight such data points. In the diagnostic plot, the presence of only one value significantly below the -3 residual line and a few points with high leverage (far right) but not coinciding with large residuals suggests a generally robust model. The lack of points combining both high leverage and large residuals indicates that there are no extreme cases significantly distorting the model's overall fit. This observation supports the model's reliability, as it seems to be neither overly influenced by outliers nor by unusual predictor values.







For logistic regression, the goodness of fit is evaluated through tests like the Hosmer-Lemeshow test. This test checks whether the observed event rates match expected event rates in subgroups of the model population.

Conclusion:

Based on our finalized models (linear and logistic regression), we were able to effectively answer our research questions:

- What are the key factors/variables that affect housing prices in Ames, Iowa?
- What are valuable statistics that can be pulled from the dataset relating to SalePrice?

Linear model: Using k-fold cross-validation in evaluating the top three models, emphasized the superiority of model 1 (SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + ExterQual) based on the 3. It exhibited the lowest Root Mean Squared Error (RMSE), highest R-squared, and minimal Mean Absolute Error (MAE), indicating its superior predictive accuracy and balance in complexity. Aligning with the AIC-based model selection, the cross-validation outcomes affirmed Model 1's excellence, reinforcing its efficacy in explaining Sale Price variance and generalizing to new data. From our investigation, focusing on identifying crucial variables significantly correlated with the Sale Price of houses, Overall Quality, Living Area, Kitchen Quality, Garage, Basement Size, and Exterior Material Quality, among others, this demonstrates a strong linear relationship with Sale Price.

Logistic regression: The logistic regression analysis aimed to predict house price categories (high or low) based on various predictors, using a backward selection process guided by the Akaike Information Criterion (AIC) for model refinement. The final model, PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual, demonstrated the best balance of complexity and fit. To assess model performance, 10-fold cross-validation was applied to the top three models derived from the backward selection process. Metrics like Receiver Operating Characteristic (ROC) Area Under Curve (AUC), Sensitivity, and Specificity highlighted excellent model performance in distinguishing between high and low sale prices. Model 1, specifically, displayed marginally higher effectiveness in correctly identifying houses in the 'High' price category. The diagnostic tests for logistic regression indicated a high level of accuracy in classification, with an accuracy rate of 92.47%. Sensitivity and specificity values exceeded 92%, affirming the model's robustness in classifying both 'Low' and 'High' price categories. Metrics like Kappa statistics, balanced accuracy, and McNemar's Test further emphasized the model's consistent performance across both classes. The ROC curve analysis also indicated the model's exceptional ability to differentiate between price categories. Furthermore, influence plots and diagnostics for logistic regression suggested a generally robust model, not affected by outliers or unusual predictor values.

In conclusion, our analysis identified important variables influencing housing prices in Ames, Iowa, leading to the development of strong models for pricing classification and prediction. These models, validated through cross-validation and diagnostic assessments, offer useful insights for stakeholders, enabling more informed decision-making in the housing market. Our research successfully addressed our initial queries, shedding light on the critical variables shaping housing prices in Ames, Iowa.

Appendix

~~~
---
title: "Housing Price Prediction Model"
output:
  pdf_document: default
  html_document: default
date: "2023-12-04"
---
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```{r}
# load libraries
library(tidyverse)
library(ggplot2)
library(gridExtra)
library(dplyr)
library(tidyr)
library(corrplot)
library(caret)
```
```{r}
# load datasets
orig_train = read.csv("train.csv")
orig_test = read.csv("test.csv")
train = read.csv("train.csv")
test = read.csv("test.csv")
```
```{r}
head(train)
```
# Data Cleaning/Pre-processing
As one of our goals is determining most significant predictors (not necessarily determining predicting the sale price of the home), in the data cleaning section I will be removing/combining many of the minor variables in order to help simplify the data set.
```{r}
# Function counting missing values
missing =  sapply(train, function(x) {
  sum(is.na(x)) / length(x) * 100
})
missing_df = data.frame(Missing_Percentages = missing)
missing_df = missing_df %>%
  arrange(desc(Missing_Percentages))
missing_df = missing_df[missing_df$Missing_Percentages > 0, , drop = F]
missing_df # percentage of columns with NA values
```
### Dealing with Numerical Variables
```{r}
# Removing id column as irrelevant
train = train[,-1]
~~~

~~~~
```
```{r}
# PoolArea and PoolQC
# Add variable to check for pool or not due to large amount of missing values
train$HasPool <- ifelse(train$PoolArea > 0 | !is.na(train$PoolQC), 1, 0)
train$HasPool <- factor(train$HasPool)
# Removing others due to missing variables
train$PoolArea <- NULL
train$PoolQC <- NULL
```
```{r}
# LotFrontage
# Due to unknown meaning of NA, removing:
train$LotFrontage <- NULL
```
```{r}
# Date variables
# Removing MoSold as YrSold is sufficient
train$MoSold <- NULL
# Creating a new variable for the age of the house at the time of the sale
train$AgeAtSale <- train$YrSold - train$YearBuilt
# Creating a new variable for the age of the house since its last remodel at the time of the sale
train$YearsSinceRemodel <- train$YrSold - train$YearRemodAdd
# Removing this as there are missing values
train$GarageYrBlt = NULL
```
```{r}
# Masonry Veneer Area
# Dealing with NA values, setting to 0
train$MasVnrArea[is.na(train$MasVnrArea)] <- 0
```
```{r}
# Basement Square Footage and Above Grade Square Footage
# Only keeping total square feet for basement, removing extra variables
# If conducting a more in-depth analysis, it might be better to keep these
train$BsmtUnfSF = NULL
train$BsmtFinSF1 = NULL
train$BsmtFinSF2 =NULL
# Keeping total square feet for above grade, as 1st and 2nd floor square feet are components to total
# Keeping low quality square feet as it seems like an interesting statistic
train$X1stFlrSF = NULL
train$X2ndFlrSF = NULL
```
```{r}
# Basement Bathrooms, combining into one variable to simplify the data
train$BsmtBaths <- train$BsmtFullBath + train$BsmtHalfBath * 0.5
train$BsmtFullBath <- NULL
train$BsmtHalfBath <- NULL
```
```{r}
~~~~

~~~
# Above grade bathrooms, also combining to simplify
train$TotalBaths <- train$FullBath + train$HalfBath * 0.5
train$FullBath <- NULL
train$HalfBath <- NULL
```
```{r}
# Outside square footage
# Combining all into one variable
train$TotalOutSideSF = train$WoodDeckSF + train$OpenPorchSF + train$EnclosedPorch +
train$X3SsnPorch + train$ScreenPorch
train$WoodDeckSF = NULL
train$OpenPorchSF = NULL
train$EnclosedPorch = NULL
train$X3SsnPorch = NULL
train$ScreenPorch = NULL
```
### Dealing with Categorical Variables
```{r}
# MSZoning, leaving as is
price <- summarize(group_by(train, MSZoning),
        mean(SalePrice, na.rm=T))
```
```{r}
summarize(group_by(train, Street),
        mean(SalePrice, na.rm=T))
# Updated Street to binary, changed to 1 if road access is paved, 0 if not
train$Street <- ifelse(train$Street == "Pave", 1, 0)
summarize(group_by(train, Alley),
        mean(SalePrice, na.rm=T))
# Updated Alley to binary , 1 if there is alley access to the property, 0 if not
train$Alley <- ifelse(is.na(train$Alley), 0, 1)
```
```{r}
# Lot Variables
summarize(group_by(train, LotConfig),
        mean(SalePrice, na.rm=T))
# Creating new binary variable, 1 if lot shape is regular, 0 if not
train$LotShape_Regular <- ifelse(train$LotShape == "Reg", 1, 0)
train$LotShape = NULL
# Creating new binary variable, 1 if land is flat, 0 if not
train$isFlat <- ifelse(train$LandContour == "Lvl", 1, 0)
train$LandContour <- NULL
```
```{r}
# Utilities, 1 for has all the utilities (sewer, electricity, etc.), 0 if the property is missing any of these
summarize(group_by(train, Utilities),
        mean(SalePrice, na.rm=T))
train$AllUtilities <- ifelse(train$Utilities == "AllPub", 1, 0)
train$Utilities = NULL
# Electricity, excluding due to unknown variable meanings
~~~

~~~
train$Electrical = NULL
```
```{r}
# Land Slope
summarize(group_by(train, LandSlope),
        mean(SalePrice, na.rm=T))
# Unexpected correlation, will keep all separate
```
```{r}
# Condition Proximity
summarize(group_by(train, Condition1),
        mean(SalePrice, na.rm=T))
# Creating binary variable, 1 if property is located near a positive feature, 0 otherwise
train$NearPositiveFeature1 <- ifelse(train$Condition1 %in% c("PosA", "PosN"), 1, 0)
# Indicates the presence of a second positive feature
train$NearPositiveFeature2 <- ifelse(train$Condition1 %in% c("PosA", "PosN"), 1, 0)
# Creating binary variable, 1 if property is located near a railroad, 0 otherwise
train$NearRailroad1 <- ifelse(train$Condition1 %in% c("RRNn", "RRAn", "RRNe", "RRAe"), 1, 0)
# Indicates the presence of a second railroad
train$NearRailroad2 <- ifelse(train$Condition1 %in% c("RRNn", "RRAn", "RRNe", "RRAe"), 1, 0)

# Initial variables, not needed as they have been condensed into the above variables
train$Condition1 <- NULL
train$Condition2 <- NULL
```
```{r}
# Building Type
summarize(group_by(train, BldgType),
        mean(SalePrice, na.rm=T))
# Combining townhouse variable into one and splitting into four different binary variables indicating the
type of building
train$One_family_building <- ifelse(train$BldgType == "1Fam", 1, 0)
train$Two_family_conversion <- ifelse(train$BldgType == "2fmCon", 1, 0)
train$Duplex <- ifelse(train$BldgType == "Duplex", 1, 0)
train$Townhouse <- ifelse(train$BldgType %in% c("TwnhsE", "Twnhs"), 1, 0)
# Removing initial variable
train$BldgType <- NULL
```
```{r}
# House Style
summarize(group_by(train, HouseStyle),
        mean(SalePrice, na.rm=T))

# Splitting HouseStyle into three different binary variables
train$Split_style <- ifelse(train$HouseStyle %in% c("SLvl", "SFoyer"), 1, 0)
train$Less_than_two_story <- ifelse(train$HouseStyle %in% c("1Story", "1.5Fin", "1.5Unf"), 1, 0)
train$Two_story_plus <- ifelse(train$HouseStyle %in% c("2Story", "2.5Fin", "2.5Unf"), 1, 0)
# Removing initial variable
train$HouseStyle <- NULL
```
~~~

~~~
```{r}
# Roof + Exterior + Masonry Veneer + Foundation
summarize(group_by(train, RoofStyle),
        mean(SalePrice, na.rm=T))
# Leaving out
train$RoofStyle <- NULL
summarize(group_by(train, RoofMatl),
        mean(SalePrice, na.rm=T))
train$RoofMatl <- NULL
# Same with Exterior + Foundation
train$Exterior1st <- NULL
train$Exterior2nd <- NULL
train$Foundation <- NULL
# Leaving exterior condition and quality
summarize(group_by(train, MasVnrType),
        mean(SalePrice, na.rm=T))
train$hasMasVnr <- ifelse(is.na(train$MasVnrType), 0, 1)
train$MasVnrType <- NULL
~~~
```{r}
# Basement
# Leaving quality and condition the same, cleaning NA values
train$BsmtQual = ifelse(is.na(train$BsmtQual), "None", train$BsmtQual)
train$BsmtCond = ifelse(is.na(train$BsmtCond), "None", train$BsmtCond)
# Exposure
# Leaving exposure the same, cleaning NA values
train$BsmtExposure = ifelse(is.na(train$BsmtExposure), "No", train$BsmtExposure)
# Finished Types, updating to binary, 1 for seemingly good qualities, 0 for seemingly bad qualities
train$BsmtFinType1[train$BsmtFinType1 %in% c("GLQ", "ALQ", "Rec")] <- 1
train$BsmtFinType1[train$BsmtFinType1 %in% c("BLQ", "LwQ", "Unf")] <- 0
train$BsmtFinType1[is.na(train$BsmtFinType1)] <- 0
# Indicates presence of a second feature
train$BsmtFinType2[train$BsmtFinType2 %in% c("GLQ", "ALQ", "Rec")] <- 1
train$BsmtFinType2[train$BsmtFinType2 %in% c("BLQ", "LwQ", "Unf")] <- 0
train$BsmtFinType2[is.na(train$BsmtFinType2)] <- 0
```
```{r}
# Heating + CentralAir
# Leaving out heating types as unsure of quality
train$Heating <- NULL
# Updating CentralAir to binary,
train$CentralAir <- ifelse(train$CentralAir == "Y", 1, 0)
```
```{r}
# Functionality
# Due to unclear descriptions in data description, leaving variable out of analysis
train$Function <- NULL
```
```{r}
# Fireplaces
```

~~~
# Cleaning NA values
train$FireplaceQu = ifelse(is.na(train$FireplaceQu), "None", train$FireplaceQu)
```
```{r}
# Garage
# Cleaning NA values
train$GarageQual = ifelse(is.na(train$GarageQual), "None", train$GarageQual)
train$GarageCond = ifelse(is.na(train$GarageCond), "None", train$GarageCond)
# Creating new binary variable, 1 if there is a garage on the property, 0 if there is no garage
train$HasGarage <- ifelse(is.na(train$GarageType), 0, 1)
# Removing unnecessary variables
train$GarageType <- NULL
train$GarageFinish <- NULL
```
```{r}
# Paved driveway
# Updating paved driveway to binary, 1 for a paved driveway, 0 for unpaved
train$PavedDrive[train$PavedDrive == "Y"] <- 1
train$PavedDrive[!train$PavedDrive != "Y"] <- 0
train$PavedDrive[is.na(train$PavedDrive)] <- 0
```
```{r}
# Fence
# Updated to binary, 1 if there is a fence, 0 if there isn't
train$Fence[train$Fence %in% c("GdPrv", "MnPrv", "GdWo","MnWw")] <- 1
train$Fence[is.na(train$Fence)] <- 0
```

```{r}
# MiscFeature and MiscVal variables
# Average value of miscellaneous feature
mean(train$MiscVal[train$MiscVal != 0])
# Due to the value being relatively modest compared to the average final sale price of the home, we will
exclude it
train$MiscFeature <- NULL
train$MiscVal <- NULL
```
```{r}
# Sale Type + Sale Condition
# Unsure of quality of each type, excluding
train$SaleType = NULL
train$SaleCondition = NULL
```
Done with pre-processing
```{r}
# Double checking for missing values
missing =  sapply(train, function(x) {
  sum(is.na(x)) / length(x) * 100
})
missing_df = data.frame(Missing_Percentages = missing)
~~~

~~~
missing_df = missing_df %>%
  arrange(desc(Missing_Percentages))
missing_df = missing_df[missing_df$Missing_Percentages > 0, , drop = F]
missing_df # percentage of columns with NA values
```
```{r}
# Correlation Matrix for numeric variables
numeric_vars <- sapply(train, is.numeric)
cor_matrix <- cor(train[, numeric_vars], use = "pairwise.complete.obs")
corrplot(cor_matrix, method = "square",
         tl.cex = 0.5,
         tl.srt = 90,
         addrect = 5)
```
```{r}
# Correlation Matrix using all original data for reference (delete later)
numeric_vars <- sapply(orig_train, is.numeric)
cor_matrix <- cor(orig_train[, numeric_vars], use = "pairwise.complete.obs")
corrplot(cor_matrix, method = "square",
         tl.cex = 0.5,
         tl.srt = 45,
         addrect = 5)
```
As we made many changes to the training data all the steps must also be applied to the test data:
```{r}
test = test[,-1]
test$HasPool <- ifelse(test$PoolArea > 0 | !is.na(test$PoolQC), 1, 0)
test$PoolArea <- NULL
test$PoolQC <- NULL
test$LotFrontage <- NULL
test$MoSold <- NULL
test$AgeAtSale <- test$YrSold - test$YearBuilt
test$YearsSinceRemodel <- test$YrSold - test$YearRemodAdd
test$GarageYrBlt = NULL
test$MasVnrArea[is.na(test$MasVnrArea)] <- 0
test$BsmtUnfSF = NULL
test$BsmtFinSF1 = NULL
test$BsmtFinSF2 =NULL
test$X1stFlrSF = NULL
test$X2ndFlrSF = NULL
test$BsmtBaths <- test$BsmtFullBath + test$BsmtHalfBath * 0.5
test$BsmtFullBath <- NULL
test$BsmtHalfBath <- NULL
test$TotalBaths <- test$FullBath + test$HalfBath * 0.5
test$FullBath <- NULL
test$HalfBath <- NULL
test$TotalOutSideSF = test$WoodDeckSF + test$OpenPorchSF + test$EnclosedPorch + test$X3SsnPorch
+ test$ScreenPorch
test$WoodDeckSF = NULL
test$OpenPorchSF = NULL
~~~

```
test$EnclosedPorch = NULL
test$X3SsnPorch = NULL
test$ScreenPorch = NULL
test$Street <- ifelse(test$Street == "Pave", 1, 0)
test$Alley <- ifelse(is.na(test$Alley), 0, 1)
test$LotShape_Regular <- ifelse(test$LotShape == "Reg", 1, 0)
test$LotShape = NULL
test$isFlat <- ifelse(test$LandContour == "Lvl", 1, 0)
test$LandContour <- NULL
test$AllUtilities <- ifelse(test$Utilities == "AllPub", 1, 0)
test$Utilities = NULL
test$Electrical = NULL
test$NearPositiveFeature1 <- ifelse(test$Condition1 %in% c("PosA", "PosN"), 1, 0)
test$NearPositiveFeature2 <- ifelse(test$Condition1 %in% c("PosA", "PosN"), 1, 0)
test$NearRailroad1 <- ifelse(test$Condition1 %in% c("RRNn", "RRAn", "RRNe", "RRAe"), 1, 0)
test$NearRailroad2 <- ifelse(test$Condition1 %in% c("RRNn", "RRAn", "RRNe", "RRAe"), 1, 0)
test$Condition1 <- NULL
test$Condition2 <- NULL
test$One_family_building <- ifelse(test$BldgType == "1Fam", 1, 0)
test$Two_family_conversion <- ifelse(test$BldgType == "2fmCon", 1, 0)
test$Duplex <- ifelse(test$BldgType == "Duplex", 1, 0)
test$Townhouse <- ifelse(test$BldgType %in% c("TwnhsE", "Twnhs"), 1, 0)
test$BldgType <- NULL
test$Split_style <- ifelse(test$HouseStyle %in% c("SLvl", "SFoyer"), 1, 0)
test$Less_than_two_story <- ifelse(test$HouseStyle %in% c("1Story", "1.5Fin", "1.5Unf"), 1, 0)
test$Two_story_plus <- ifelse(test$HouseStyle %in% c("2Story", "2.5Fin", "2.5Unf"), 1, 0)
test$HouseStyle <- NULL
test$RoofStyle <- NULL
test$RoofMatl <- NULL
test$Exterior1st <- NULL
test$Exterior2nd <- NULL
test$Foundation <- NULL
test$hasMasVnr <- ifelse(is.na(test$MasVnrType), 0, 1)
test$MasVnrType <- NULL
test$BsmtQual = ifelse(is.na(test$BsmtQual), "None", test$BsmtQual)
test$BsmtCond = ifelse(is.na(test$BsmtCond), "None", test$BsmtCond)
test$BsmtExposure = ifelse(is.na(test$BsmtExposure), "No", test$BsmtExposure)
test$BsmtFinType1[test$BsmtFinType1 %in% c("GLQ", "ALQ", "Rec")] <- 1
test$BsmtFinType1[test$BsmtFinType1 %in% c("BLQ", "LwQ", "Unf")] <- 0
test$BsmtFinType1[is.na(test$BsmtFinType1)] <- 0
test$BsmtFinType2[test$BsmtFinType2 %in% c("GLQ", "ALQ", "Rec")] <- 1
test$BsmtFinType2[test$BsmtFinType2 %in% c("BLQ", "LwQ", "Unf")] <- 0
test$BsmtFinType2[is.na(test$BsmtFinType2)] <- 0
test$Heating <- NULL
test$CentralAir <- ifelse(test$CentralAir == "Y", 1, 0)
test$Function <- NULL
test$FireplaceQu = ifelse(is.na(test$FireplaceQu), "None", test$FireplaceQu)
test$GarageQual = ifelse(is.na(test$GarageQual), "None", test$GarageQual)
test$GarageCond = ifelse(is.na(test$GarageCond), "None", test$GarageCond)
test$HasGarage <- ifelse(is.na(test$GarageType), 0, 1)
```

~~~~
test$GarageType <- NULL
test$GarageFinish <- NULL
test$PavedDrive[test$PavedDrive == "Y"] <- 1
test$PavedDrive[!test$PavedDrive != "Y"] <- 0
test$PavedDrive[is.na(test$PavedDrive)] <- 0
test$Fence[test$Fence %in% c("GdPrv", "MnPrv", "GdWo","MnWw")] <- 1
test$Fence[is.na(test$Fence)] <- 0
mean(test$MiscVal[test$MiscVal != 0])
test$MiscFeature <- NULL
test$MiscVal <- NULL
test$SaleType = NULL
test$SaleCondition = NULL
```
12 Highest predictors:
```{r}
correlations <- cor(train[sapply(train, is.numeric)], train$SalePrice)
sorted_correlations <- sort(abs(correlations[,1]), decreasing = TRUE)
top_12_predictors <- names(sorted_correlations)[2:13]
print(top_12_predictors)
sorted_correlations[1:13]
```
```{r}
base_model <- lm(SalePrice ~ 1, data = train)
library(MASS)
best_model <- stepAIC(base_model, scope = list(lower = base_model, upper = ~ OverallQual +
GrLivArea + ExterQual + KitchenQual + GarageCars + GarageArea + TotalBsmtSF), direction =
"forward")
```
K-fold CV (10 Folds): - on the top 3 models:
```{r}
library(caret)
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
model1_formula <- SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF +
ExterQual
model2_formula <- SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars + TotalBsmtSF
model3_formula <- SalePrice ~ OverallQual + GrLivArea + KitchenQual + GarageCars
set.seed(123)  # For reproducibility
model1_cv <- train(model1_formula, data = train, method = "lm", trControl = fitControl)
set.seed(123)
model2_cv <- train(model2_formula, data = train, method = "lm", trControl = fitControl)
set.seed(123)
model3_cv <- train(model3_formula, data = train, method = "lm", trControl = fitControl)
model1_cv$results
model2_cv$results
model3_cv$results
```
Diagnostics:
```{r}
model1 <- lm(model1_formula, data = train)
# Residuals vs Fitted Values
~~~~

```
plot(model1$fitted.values, residuals(model1),
    xlab = "Fitted Values",
    ylab = "Residuals",
    main = "Residuals vs Fitted Values")
abline(h = 0, col = "red")
# Normal Q-Q Plot
qqnorm(residuals(model1))
qqline(residuals(model1), col = "red")
# Shapiro-Wilk Test
shapiro.test(residuals(model1))
# install.packages("lmtest")
library(lmtest)
# Breusch-Pagan Test
bptest(model1)
# install.packages("lmtest")
library(lmtest)
# Breusch-Pagan Test
bptest(model1)
# install.packages("car")
library(car)
# Variance Inflation Factor
vif(model1)
# install.packages("car")
library(car)
# Plot for Outliers and Leverage
influencePlot(model1, id.method = "identify", main = "Influence Plot", sub = "Circle size is proportional
to Cook's distance")
```
Shapiro-Wilk Normality Test
* Result: W = 0.80807, p-value < 2.2e-16
* Interpretation: This test checks whether the residuals of your model are normally distributed. A significant p-value (in this case, less than 2.2e-16) suggests that the residuals are not normally distributed. This is a violation of one of the key assumptions of linear regression.

Breusch-Pagan Test
* Result: BP = 398.38, df = 10, p-value < 2.2e-16
* Interpretation: This test assesses homoscedasticity, meaning whether the residuals have constant variance across the range of predictors. The very low p-value here indicates that the residuals do not have constant variance (heteroscedasticity). This is another violation of linear regression assumptions.

Variance Inflation Factor (VIF)
* Result: VIF values are provided for each predictor.
* Interpretation: VIF values greater than 5 or 10 can indicate problematic multicollinearity. In your case, the VIF values seem to be within an acceptable range, suggesting multicollinearity is not a major concern for this model.

Influence Plot (Outliers and High-Leverage Points)
* Result: Specific points (e.g., 524, 636, 1299) have been identified with significant studentized residuals or high Cook's distances.
* Interpretation: These points may be outliers or have high leverage, potentially influencing the model disproportionately. It's worth investigating these points to see if they are data entry errors, require transformation, or are valid extreme values.

Residuals vs Fitted Plot Observation

* Observation: If the spread of residuals (the width of the scatter plot points) increases or decreases as you move along the fitted values, it suggests a pattern of heteroscedasticity, meaning the variance of the residuals is not constant across the range of fitted values.
* Interpretation: A fan or funnel-like shape in this plot is a typical sign of heteroscedasticity.

Breusch-Pagan Test Result
* Result: BP = 398.38, df = 10, p-value < 2.2e-16
* Interpretation: The very low p-value in the Breusch-Pagan test indicates that there is significant heteroscedasticity in the model. This means that the assumption of constant variance in the residuals is violated.

Shapiro-Wilk Test Result
* Result: W = 0.80807, p-value < 2.2e-16
* Interpretation: The very small p-value indicates a significant deviation from normality. In other words, the distribution of residuals does not align well with what would be expected if they were normally distributed.

Q-Q Plot Observation
* Observation: An increase in the upper tail deviating from the line.

Logistic Regression (using a backwards selection algorithm):

~~~
```{r}
# Categorizing Sale Price into High and Low based on the median
median_price <- median(train$SalePrice)
train$PriceCategory <- ifelse(train$SalePrice > median_price, "High", "Low")
train$PriceCategoryBinary <- ifelse(train$PriceCategory == "High", 1, 0)
# Assuming you have already loaded the MASS package for stepAIC
library(MASS)
# Fit the full logistic regression model with all the specified predictors
full_log_model <- glm(PriceCategoryBinary ~ OverallQual + GrLivArea + ExterQual + KitchenQual +
GarageCars + GarageArea + TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual + TotRmsAbvGrd
+ AgeAtSale, data = train, family = binomial)
# Apply backward elimination to find the most optimal model
optimal_log_model <- stepAIC(full_log_model, direction = "backward")
# View the summary of the optimal model
summary(optimal_log_model)
```
~~~

~~~
```{r}
library(caret)
# Ensure PriceCategoryBinary is a factor with valid level names
train$PriceCategoryBinary <- factor(train$PriceCategoryBinary, levels = c(0, 1), labels = c("Low",
"High"))
# Check the levels to ensure they are now valid R variable names
levels(train$PriceCategoryBinary)
model1_formula <- PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars +
TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual
model2_formula <- PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars +
TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual + TotRmsAbvGrd
model3_formula <- PriceCategoryBinary ~ OverallQual + GrLivArea + KitchenQual + GarageCars +
TotalBsmtSF + Neighborhood + TotalBaths + BsmtQual + TotRmsAbvGrd + AgeAtSale
# Define control parameters for k-fold cross-validation
fitControl <- trainControl(method = "cv",
                           number = 10,
                           classProbs = TRUE, # important for AUC
~~~

~~~
                    summaryFunction = twoClassSummary) # Use AUC, Sensitivity, etc.
# Define the metric to optimize
metric <- "ROC"  # You can also use "Acc" for Accuracy
# Model Formulas
# (Assuming you have defined model1_formula, model2_formula, model3_formula)
# K-fold cross-validation for each model
cv_model1 <- train(model1_formula, data = train, method = "glm",
              family = "binomial", metric = metric, trControl = fitControl)
cv_model2 <- train(model2_formula, data = train, method = "glm",
              family = "binomial", metric = metric, trControl = fitControl)
cv_model3 <- train(model3_formula, data = train, method = "glm",
              family = "binomial", metric = metric, trControl = fitControl)
# View the results of cross-validation for each model
cv_model1$results
cv_model2$results
cv_model3$results
```
Diagnostics:
```{r}
# Predict on training data
predicted_classes <- predict(cv_model1, newdata = train, type = "raw")
confusionMatrix(predicted_classes, train$PriceCategoryBinary)
```
```{r}
library(pROC)
# Predict probabilities
predicted_probs <- predict(cv_model1, newdata = train, type = "prob")
roc_response <- roc(response = train$PriceCategoryBinary, predictor = predicted_probs[, "High"])
# Plot ROC Curve
plot(roc_response, main = "ROC Curve")
```
```{r}
# Calculate residuals
log_model1 <- glm(model1_formula, data = train, family = "binomial")
residuals_df <- data.frame(Fitted = fitted(log_model1), Residuals = residuals(log_model1, type =
"deviance"))
# Residuals vs Fitted Plot
ggplot(residuals_df, aes(x = Fitted, y = Residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Residuals vs Fitted", x = "Fitted Values", y = "Deviance Residuals")
```
```{r}
library(car)
# Load the car package
library(car)
# Assuming log_model1 is your logistic regression model object
# Replace 'log_model1' with your actual model name
# Influence Index Plot for Cook's Distance
influenceIndexPlot(log_model1, vars = "Cook")
~~~

~~~
# Influence Index Plot for Leverage (hat values)
influenceIndexPlot(log_model1, vars = "hat")
```
```{r}
# Assuming 'log_model' is your fitted logistic regression model
# and 'train' is your dataset
# Step 1: Compute Influence Measures
influence_measures <- broom::augment(log_model1, train)
# Step 2: Plotting Influence Measures
# Plot Standardized Residuals
plot(influence_measures$.hat, influence_measures$.std.resid,
     xlab = "Leverage (Hat values)",
     ylab = "Standardized Residuals",
     main = "Influence Plot for Logistic Regression")
abline(h = c(-3, 3), col = "red", lty = 2)  # Adding reference lines for residuals
# Identifying points with high leverage or large residuals
identify(influence_measures$.hat, influence_measures$.std.resid, labels =
row.names(influence_measures))
```
```{r}
library(ResourceSelection)
predicted_probs <- predict(log_model1, newdata = train, type = "response")
# Conduct the Hosmer-Lemeshow test
hoslem.test(x = train$PriceCategoryBinary, y = predicted_probs, g = 10)
```
~~~