

House Prices and Regressions

Introduction:

The real estate industry is one of the most significant sectors of our economy. The buying and selling of homes form an integral part of this industry, and the value of a home can be determined by several factors. A thorough understanding of these factors can enable real estate agents and investors to make informed decisions regarding the pricing of homes. In this study, we investigated the relationship between the sales price of homes in Ames, Iowa, and the square footage of the living area. Additionally, we explored whether the location of the house in different neighborhoods affected its sales price. Finally, we aimed to develop the most predictive model for sales prices of homes in Ames, Iowa, utilizing forward selection, backward elimination, stepwise selection, and a custom model.

Data Exploration:

To investigate the relationship between the sales price of homes, their living area and location, we first created a dataset by merging the test and train dataset into one. This was done simply to allow for more efficient data wrangling. We also kept in mind the 1459 NAs in the SalePrice column were from the test dataset. Additionally, we added a column to indicate whether the record was from the train or test dataset and verified the successful merge and addition of columns (Diagram 1). All the data marked with test in the Train column were omitted upon cleaning the data. Upon finalizing this dataset, we explored the data for insights into the Ames housing market as well as identified the variables we needed to change, clean and create.

Prior to conducting the study, we visualized which categories had the highest number of NA values and identified whether it would affect the two parameters we were concerned with: SalePrice and GrLivArea. From Diagram 2, we can see that there weren't too many NA values in the dataset, and those that were present were mostly associated with a lack of a certain feature within that respective house. For example, if a house didn't have a pool, then the PoolQC column would have NA for that record.

Analysis 1:

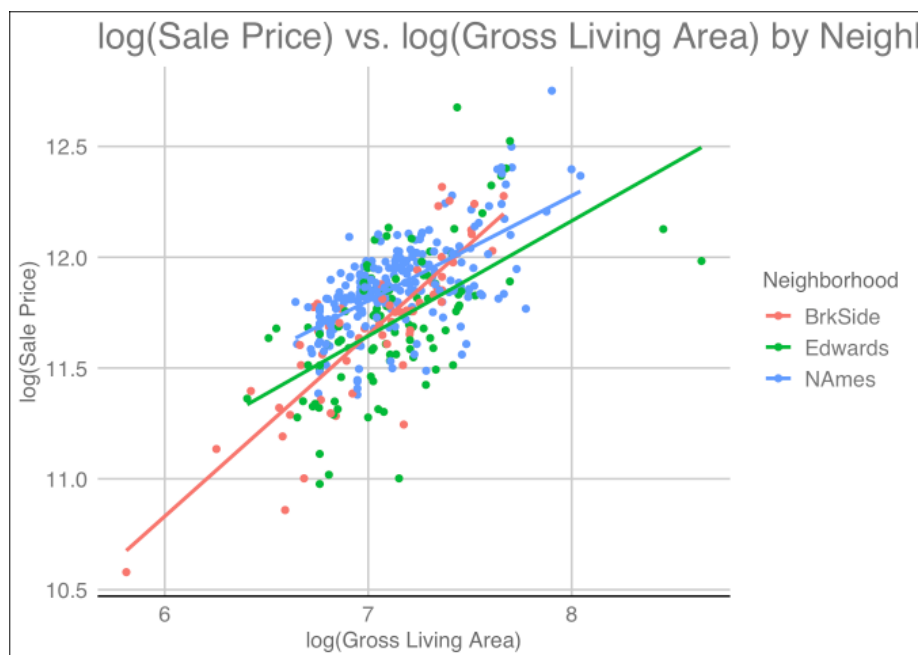
Next, to investigate the relationship between the SalePrice, the GrLivArea and Neighborhood, we plotted a scatterplot of SalePrice vs. GrLivArea and color-coded each point with the respective color associated with the Neighborhood (Diagram 3). Also, note that any records where SalePrice was NA was omitted from the visual plot. From the scatterplot, we can see that there is a clear relationship between SalePrice and GrLivArea. We can also see that the

Neighborhood category appears to have a similar relationship as the relationship between SalePrice and GrLivArea.

Despite the apparent relationship, we decided to perform a log transformation on the categories to see if we could identify a more linear relationship. In Diagram 4, we can see the scatterplot of the log transformed data. Here we can see that the relationship appears to be more linear than when the data wasn't transformed. This allowed us to determine that the log transformed data for SalePrice and GrLivArea would be used moving forward in our analysis.

Now that the data visually represented linearity more clearly, we, at this point, decided to narrow in and analyze the three neighborhoods that Century 21 operates in: NAmes, Edwards and BrkSide. When plotting this scatterplot, we took the relationship between the log of SalePrice and the log of GrLivArea and color-coded the points in respect to the three Neighborhoods. Also, note that the NA rows for SalePrice were omitted. Upon plotting the data, we can see in Diagram 5 that the relationship appears to be linear. With this result, we decided to fit a linear model using this dataset and assess whether it described the SalePrice accurately or not.

Diagram 5:



When conducting this portion of the analysis, we fit the models with and without interaction terms (Diagram 6 & 7). The interaction terms, $\text{logGrLivArea:NeighborhoodEdwards}$ and $\text{logGrLivArea:NeighborhoodNAmes}$, were statistically significant with p-values of 0.0011 and $5.35e-05$ respectively. Also, note that the model that included the interaction terms had a lower Press score of 14.609 versus a Press score of 15.001 for the model without interaction variables. These Press scores indicate that the relationship between the log SalePrice and log GrLivArea is different for each of the three neighborhoods.

Assumption Check:

In addition, we also plotted the studentized residuals to check for any violations of assumptions (Diagram 8, 9, 10 & 11). From these residual plots, we can clearly see there is insufficient evidence to indicate that there has been a violation of linearity, homoscedasticity, or normality. We also created residual plots, QQ plots, and fitted plots to identify if there were any influential points that needed to be investigated further (Diagram 12, 13, 14 & 15). The results of these plots were consistent with the assumptions of the linear model and we concluded that there were no influential points that needed to be investigated.

Comparing Competing Models:

The model that included the flexible slopes per neighborhood (interaction terms) performed better. More importantly, The interaction terms, `logGrLivArea:NeighborhoodEdwards` and `logGrLivArea:NeighborhoodNames`, were strongly significant with p-values of 0.0011 and 5.35e-05 respectively, suggesting that there were real differences in the characteristics for the neighborhood.

Parameters:

The confidence intervals for the model including the interaction terms were conducted with a significance level of 0.05. From the results in Diagram 16, we can say with 95% confidence that when all the independent variables are at zero, the estimated median SalePrice would be between ~\$137 and ~\$992.2. For the `GrLivArea`, we can say with 95% confidence that for every one unit increase in `logGrLivArea`, we can expect the median SalePrice to increase by ~\$1,970 to ~\$2,610. Next, we calculated for the difference in the intercept between the Edwards and Names neighborhoods compared to the BrkSide neighborhood. For each of these comparisons, we can say with 95% confidence, holding all else constant, the median SalePrice in the Edwards neighborhood is expected to increase 2.28 to 28.83 times more for each unit change in `GrLivArea` than BrkSide and the median SalePrice in the Names neighborhood is expected to increase 4.06 and 42.98 times more for each unit change in `GrLivArea` than BrkSide sales prices. The interaction terms represented the estimated change in the median SalePrice associated with a one unit increase in `logGrLivArea` for each neighborhood holding the BrkSide neighborhood as a reference point. We can say with 95% confidence that the effect of `logGrLivArea` on median SalePrice is between 0.62 and 0.89 times smaller in the Edwards neighborhood than BrkSide. To a similar effect, we can say with 95% confidence that the effect of `logGrLivArea` on median SalePrice is between 0.6 and 0.83 times smaller in the Names neighborhood compared to BrkSide.

Analysis 1 Conclusion:

Through the analysis performed on the relationship between SalePrice and GrLivArea, we discovered that there was a significant positive correlation between the sales price of homes and the square footage of the living area. Additionally, we identified that the location of the house in different neighborhoods does have a significant effect on its sales price.

Analysis 2:

In this section, we built four different models using forward selection, backward elimination, stepwise selection, and a custom model. We generated the R^2 , RMSE, the AIC and Kaggle score for each model and determined which model was the best in terms of predicting future sales prices. We also examined residual plots to check whether the assumptions of the linear regression model had been met and addressed any outliers or influential observations through influential point analysis. Finally, we compared competing models based on the R^2 , RMSE, the AIC and Kaggle score and drew conclusions based on our findings.

Data Cleaning:

In order to use a linear regression model, we converted all of the categorical variables into dummy variables. We also removed or imputed the NA values within the dataset by attempting to interpret what the NA value meant for each respective category (Diagram 17, 18 & 19). In most cases, this process created a new category of “none” for the variable. Any remaining values that were NA were imputed to the mean, and the data was split into training and testing sets.

Model Selection:

Once the data was cleaned accordingly, four different predictive models were utilized to determine which of these models would best predict future sale prices. The predictive models used during our analysis were the forward selection, backward elimination, stepwise selection, and a custom model utilizing the top ten parameters from the stepwise results. The question we hoped to answer was whether the increase in performance is worth the cost of performing a forward, backwards or stepwise model. We also wanted to identify if having only ten parameters would be too few to capture the complexity of the data.

Model Evaluation:

First, the forward selection was used, resulting in a R^2 score of 0.942, a RMSE of 0.096, an AIC score of -2504 and a Kaggle score of 0.1398. Next, we used the backward elimination model, and the resulting scores were a R^2 score of 0.944, a RMSE of 0.094, an AIC score of -2493 and a Kaggle score of 0.1458. Next, the stepwise predictive model was used to produce a R^2 score of 0.944, a RMSE of 0.094, an AIC score of -2494 and a Kaggle score of 0.1456. Lastly,

a custom predictive model utilizing the top ten parameters ranked by importance from the stepwise model was performed. The expectation was that this custom model wouldn't perform as well as the others, but each parameter would be more explainable and the fitting would require much less computation. The custom model produced a R^2 score of 0.84, a RMSE of 0.16, an AIC score of -1190 and a Kaggle score of 0.1777 (Diagram 20).

Diagram 20:

Predictive Models	R^2	RMSE	AIC	Kaggle Score
Forward	0.942	0.096	-2504	0.1398
Backward	0.944	0.094	-2493	0.1458
Stepwise	0.944	0.094	-2494	0.1456
Custom	0.84	0.16	-1190	0.1777

We also identified the most important predictor variables for each of the predictive models used during the analysis. When looking at the forward selection model, we can see that the most important predictor variable was roof_matl_cly_tile with an overall importance score of 16.67, followed by overall_cond and ms_zoning_c_all with scores of 12.32 and 12.10 respectively (Diagram 21). When looking at the most important predictor variables for the backward elimination model, we can see that the most important predictor variable was log_gr_liv_area with an overall importance score of 16.98, followed by roof_matl_cly_tile and overall_cond with scores of 11.54 and 11.5 respectively (Diagram 22). The stepwise selection model showed that log_gr_liv_area was the most important predictor variable with a score of 17.00. The second and third were overall_cond and roof_matl_cly_tile with scores of 11.54 and 11.46 respectively (Diagram 23). The custom model's most important predictor variable was overall_qual with a score of 34.06, followed by log_gr_liv_area with a score of 26.71 and bsmt_fin_sfl with a score of 19.63 (Diagram 24). For all of these predictive models, there were many predictor variables that were considered strong predictors and they can be observed in the associated diagrams in the appendix section.

Assumption Check:

Similar to Analysis 1, we also plotted the studentized residuals to check for any violations of assumptions (Diagram 25, 26 & 27). From these residual plots, we can clearly see there is insufficient evidence to indicate that there has been a violation of linearity, homoscedasticity, or normality. We also created residual plots, QQ plots, and fitted plots to identify if there were any

influential points that needed to be investigated further (Diagram 28, 29, 30 & 31). The results of these plots contained several points with a Cook's d of approximately 1, but overall, the results were consistent with the assumptions of the linear model.

Conclusion:

Our findings provided valuable insights into the factors that influenced the sales price of homes in Ames, Iowa. Please keep in mind that our findings were centralized for the Ames housing market and isn't indicative of all housing markets in the United States. We discovered that the size, location and overall condition of the house were the most significant factors that affected its sales price. An interesting observation was made that sales prices were also heavily influenced by the presence of a clay tile roof. Additionally, we provided several predictive models that could be used to estimate future sales price of homes in Ames, Iowa. We concluded from our analysis that the forward selection model should be used when predicting future sales prices of homes in the Ames housing market.

RShiny App:

<https://christianorji.shinyapps.io/house-prices-advanced-regression-techniques/>

Personal Github Websites:

<https://nicksager.github.io/>

<https://jgchung.github.io/>

<https://christianorji.github.io/>

Appendix

Diagram 1:

##	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
##	1	68	RL	65	8458	Pave	<NA>	Reg	Lv1
##	2	28	RL	88	9588	Pave	<NA>	Reg	Lv1
##	3	68	RL	68	11258	Pave	<NA>	IR1	Lv1
##	4	78	RL	68	9558	Pave	<NA>	IR1	Lv1
##	5	68	RL	84	14268	Pave	<NA>	IR1	Lv1
##	6	58	RL	85	14115	Pave	<NA>	IR1	Lv1
##	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2	BldgType		
##	AllPub	Inside	Gt1	CollgCr	Norm	Norm	1Fan		
##	AllPub	FR2	Gt1	Veenker	Feedr	Norm	1Fan		
##	AllPub	Inside	Gt1	CollgCr	Norm	Norm	1Fan		
##	AllPub	Corner	Gt1	Crawfor	Norm	Norm	1Fan		
##	AllPub	FR2	Gt1	NoRidge	Norm	Norm	1Fan		
##	AllPub	Inside	Gt1	Mitchel	Norm	Norm	1Fan		
##	HouseStyle	OverallQual	OverallCond	YearBuilt	YearRemodAdd	RoofStyle	RoofMatl		
##	2Story	7	5	2003	2003	Gable	CompShg		
##	1Story	6	8	1976	1976	Gable	CompShg		
##	2Story	7	5	2001	2002	Gable	CompShg		
##	2Story	7	5	1915	1978	Gable	CompShg		
##	2Story	8	5	2000	2000	Gable	CompShg		
##	1.5Fin	5	5	1993	1995	Gable	CompShg		
##	Exterior1st	Exterior2nd	MasVnrType	MasVnrArea	ExterQual	ExterCond	Foundation		
##	VinylSd	VinylSd	BrkFace	196	Gd	TA	PConc		
##	MetalSd	MetalSd	None	0	TA	TA	CBlock		
##	VinylSd	VinylSd	BrkFace	162	Gd	TA	PConc		
##	Wd Sdng	Wd Sdng	None	0	TA	TA	BrkTil		
##	VinylSd	VinylSd	BrkFace	358	Gd	TA	PConc		
##	VinylSd	VinylSd	None	0	TA	TA	Wood		
##	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinSF1	BsmtFinType2			
##	Gd	TA	No	GLQ	786	Unf			
##	Gd	TA	Gd	ALQ	978	Unf			
##	Gd	TA	Mn	GLQ	486	Unf			
##	TA	Gd	No	ALQ	216	Unf			
##	Gd	TA	Av	GLQ	655	Unf			
##	Gd	TA	No	GLQ	732	Unf			
##	BsmtFinSF2	BsmtFinSF1	TotalBsmtSF	Heating	HeatingQC	CentralAir	Electrical		
##	0	158	856	GasA	Ex	Y	SBrkr		
##	0	284	1262	GasA	Ex	Y	SBrkr		
##	0	434	928	GasA	Ex	Y	SBrkr		
##	0	548	756	GasA	Gd	Y	SBrkr		
##	0	498	1145	GasA	Ex	Y	SBrkr		
##	0	64	796	GasA	Ex	Y	SBrkr		
##	X1stFlrSF	X2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath		
##	856	854	0	1710	1	0	2		
##	1262	0	0	1262	0	1	2		
##	928	866	0	1786	1	0	2		
##	961	756	0	1717	1	0	1		
##	1145	1053	0	2198	1	0	2		
##	796	566	0	1362	1	0	1		
##	HalfBath	BedroomAbvGr	KitchenAbvGr	KitchenQual	TotRmsAbvGrd	Functional			
##	1	3	1	Gd	8	Typ			
##	0	3	1	TA	6	Typ			
##	1	3	1	Gd	6	Typ			
##	0	3	1	Gd	7	Typ			
##	1	4	1	Gd	9	Typ			
##	1	1	1	TA	5	Typ			
##	Fireplaces	FireplaceQu	GarageType	GarageYrBlt	GarageFinish	GarageCars			
##	0	<NA>	Attchd	2003	Rfn	2			
##	1	TA	Attchd	1976	Rfn	2			
##	1	TA	Attchd	2001	Rfn	2			
##	1	Gd	Detchd	1998	Unf	3			
##	1	TA	Attchd	2000	Rfn	3			
##	0	<NA>	Attchd	1993	Unf	2			
##	GarageArea	GarageQual	GarageCond	PavedDrive	WoodDeckSF	OpenPorchSF			
##	548	TA	TA	Y	0	61			
##	468	TA	TA	Y	298	0			
##	688	TA	TA	Y	0	42			
##	642	TA	TA	Y	0	35			
##	836	TA	TA	Y	192	84			
##	488	TA	TA	Y	48	38			
##	EnclosedPorch	X3SsnPorch	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeature		
##	0	0	0	0	<NA>	<NA>	<NA>		
##	0	0	0	0	<NA>	<NA>	<NA>		
##	0	0	0	0	<NA>	<NA>	<NA>		
##	272	0	0	0	<NA>	<NA>	<NA>		
##	0	0	0	0	<NA>	<NA>	<NA>		
##	0	328	0	0	<NA>	MnPrv	Shed		
##	MiscVal	NoSold	YrSold	SaleType	SaleCondition	SalePrice	train		
##	0	2	2008	MD	Normal	208500	1		
##	0	5	2007	MD	Normal	181500	1		
##	0	9	2008	MD	Normal	223500	1		
##	0	2	2006	MD	Abnorml	148800	1		
##	0	12	2008	MD	Normal	250000	1		
##	788	10	2009	MD	Normal	143000	1		

Diagram 2:

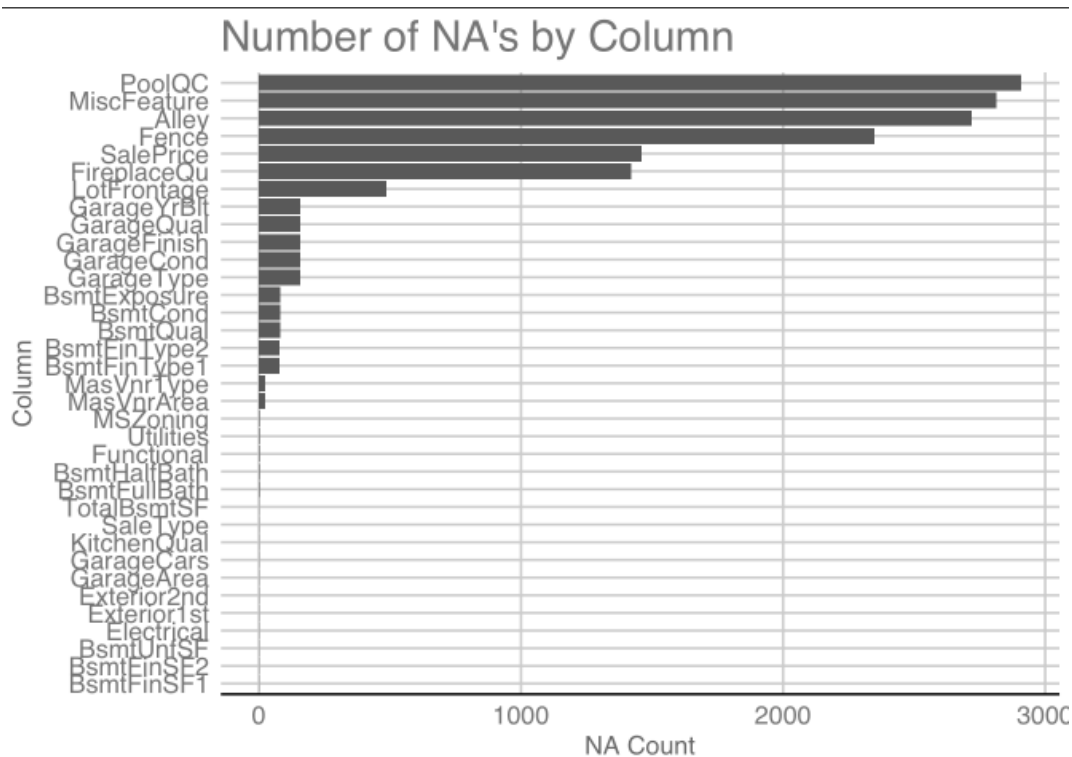


Diagram 3:

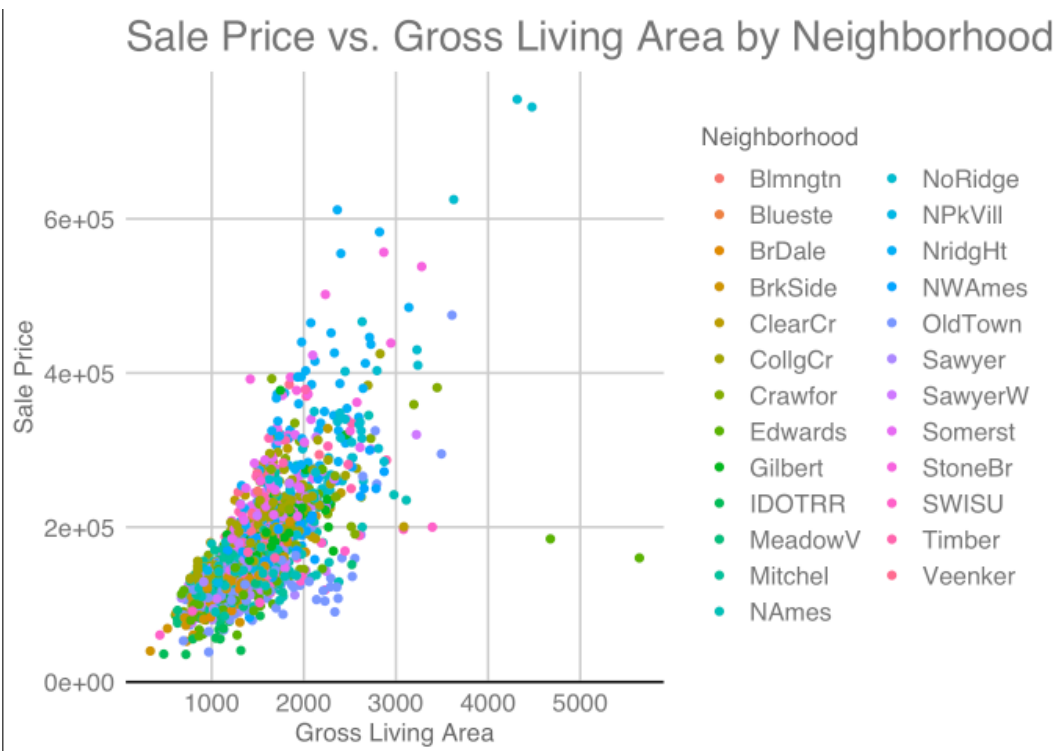


Diagram 4:

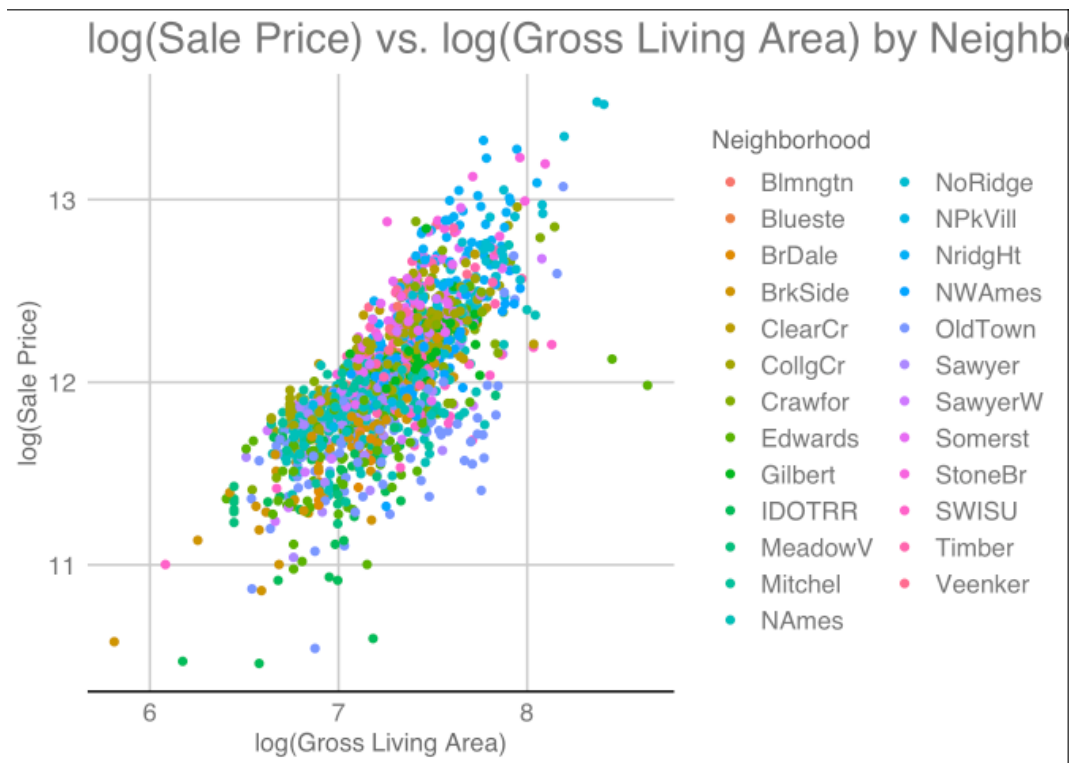


Diagram 5:



Diagram 6:

```
##
## Call:
## lm(formula = logSalePrice ~ logGrLivArea + Neighborhood, data = century21)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72154 -0.10592  0.02469  0.11565  0.79364
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      7.76936    0.22919  33.900 < 2e-16 ***
## logGrLivArea      0.55579    0.03237  17.171 < 2e-16 ***
## NeighborhoodEdwards -0.02044    0.03252  -0.629    0.53
## NeighborhoodNames   0.13279    0.02906   4.569 6.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1961 on 379 degrees of freedom
## Multiple R-squared:  0.4897, Adjusted R-squared:  0.4857
## F-statistic: 121.2 on 3 and 379 DF,  p-value: < 2.2e-16
```

Diagram 7:

```
##
## Call:
## lm(formula = logSalePrice ~ logGrLivArea * Neighborhood, data = century21)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72080 -0.10353  0.02184  0.10586  0.80470
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.91292    0.50459  11.718 < 2e-16 ***
## logGrLivArea      0.81965    0.07163  11.443 < 2e-16 ***
## NeighborhoodEdwards  2.09359    0.64589   3.241  0.0013 **
## NeighborhoodNames   2.57981    0.59988   4.301 2.17e-05 ***
## logGrLivArea:NeighborhoodEdwards -0.29998    0.09122  -3.289  0.0011 **
## logGrLivArea:NeighborhoodNames  -0.34662    0.08482  -4.087 5.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1923 on 377 degrees of freedom
## Multiple R-squared:  0.5121, Adjusted R-squared:  0.5056
## F-statistic: 79.14 on 5 and 377 DF,  p-value: < 2.2e-16
```

Diagram 8:

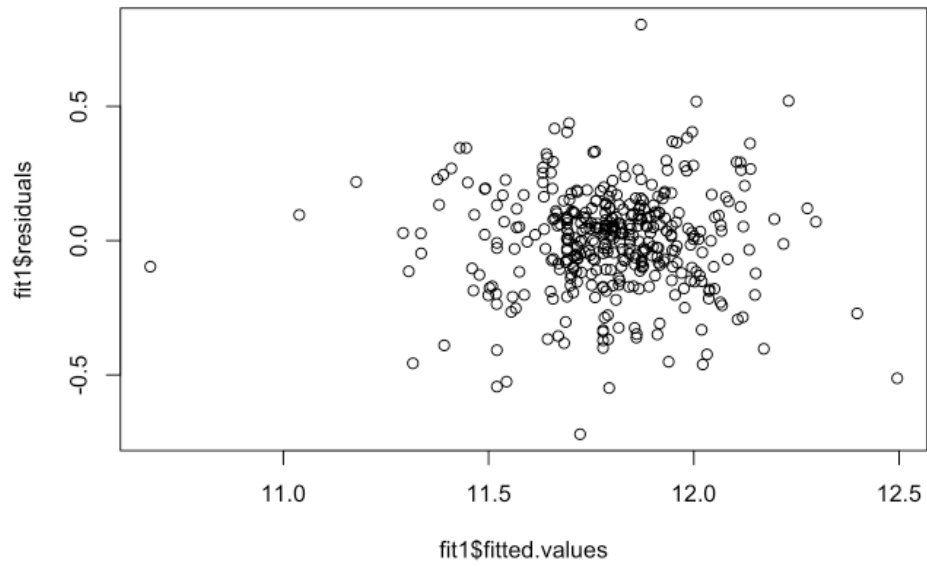


Diagram 9:

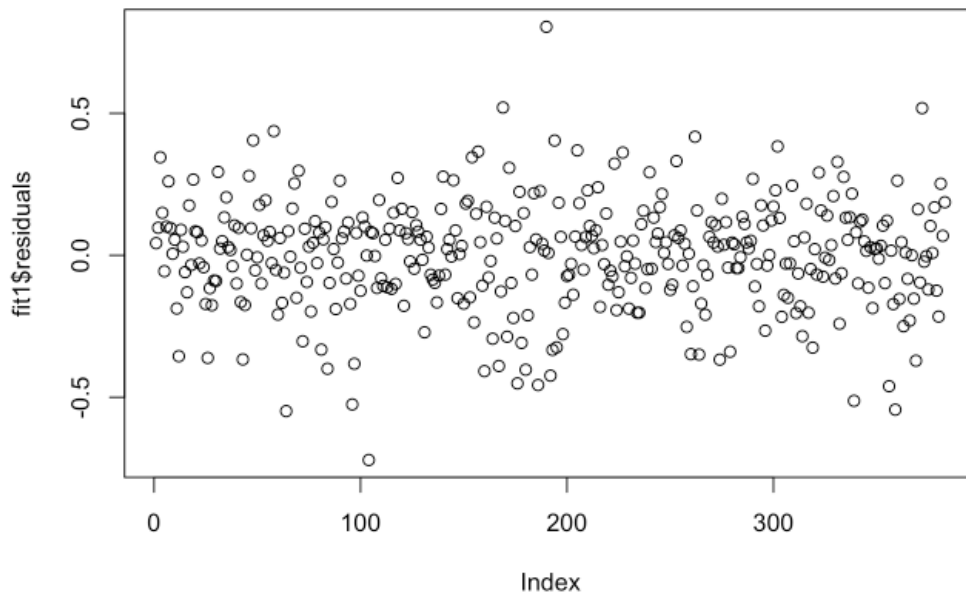


Diagram 10:

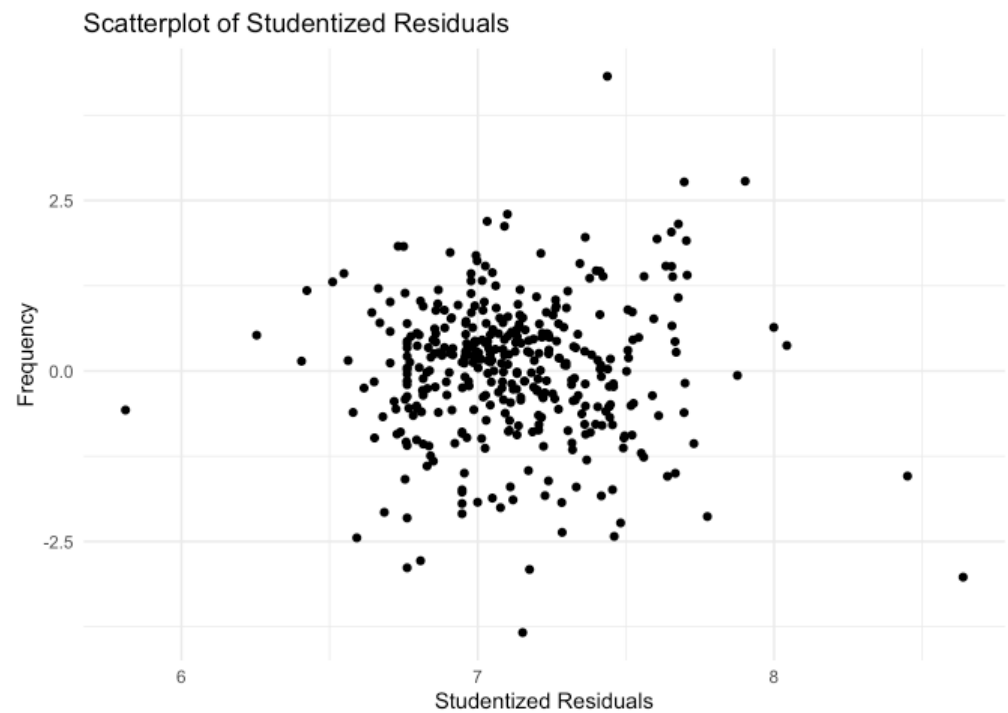


Diagram 11:

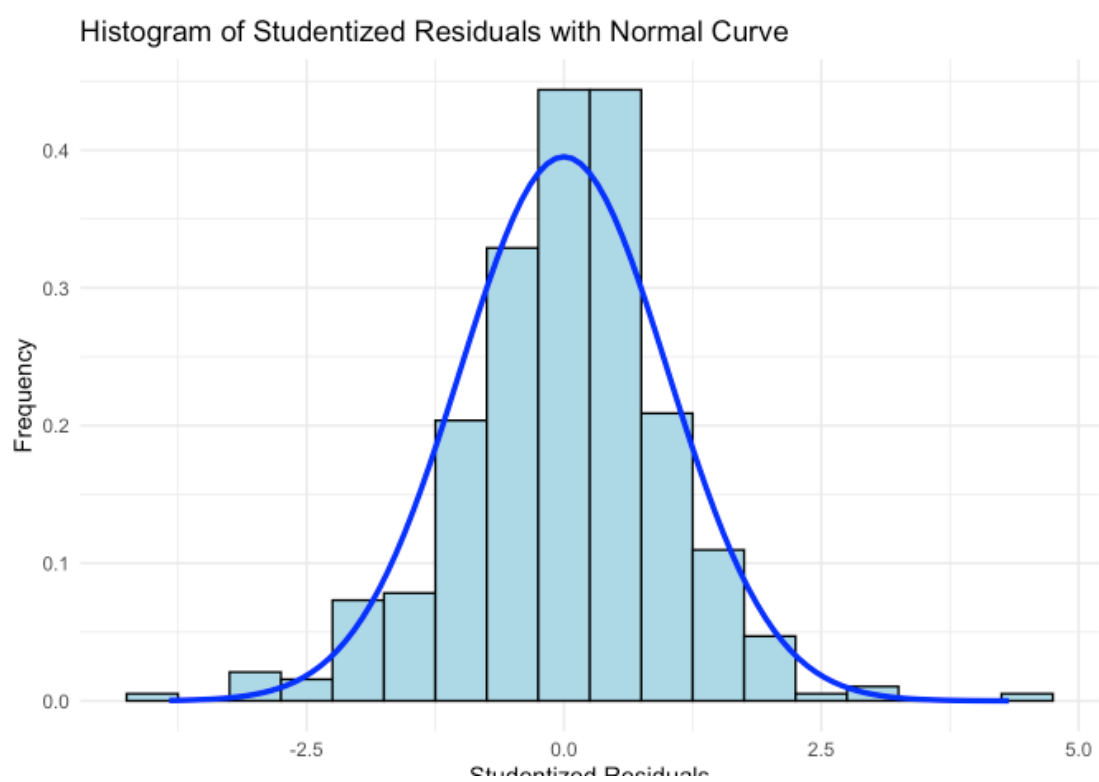


Diagram 12:

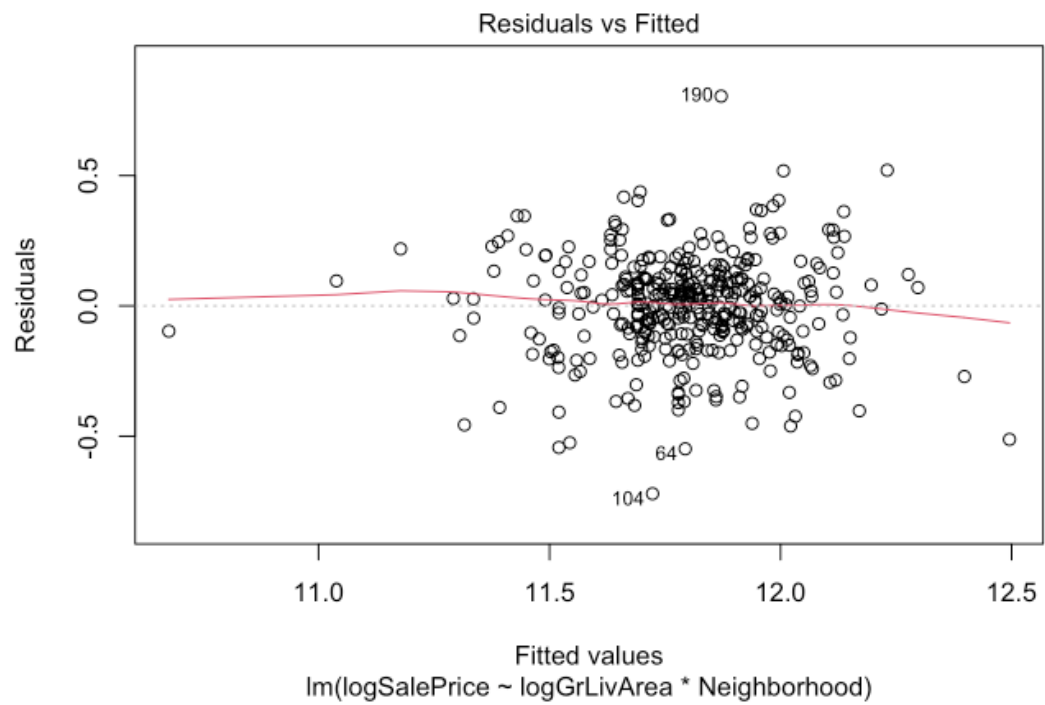


Diagram 13:

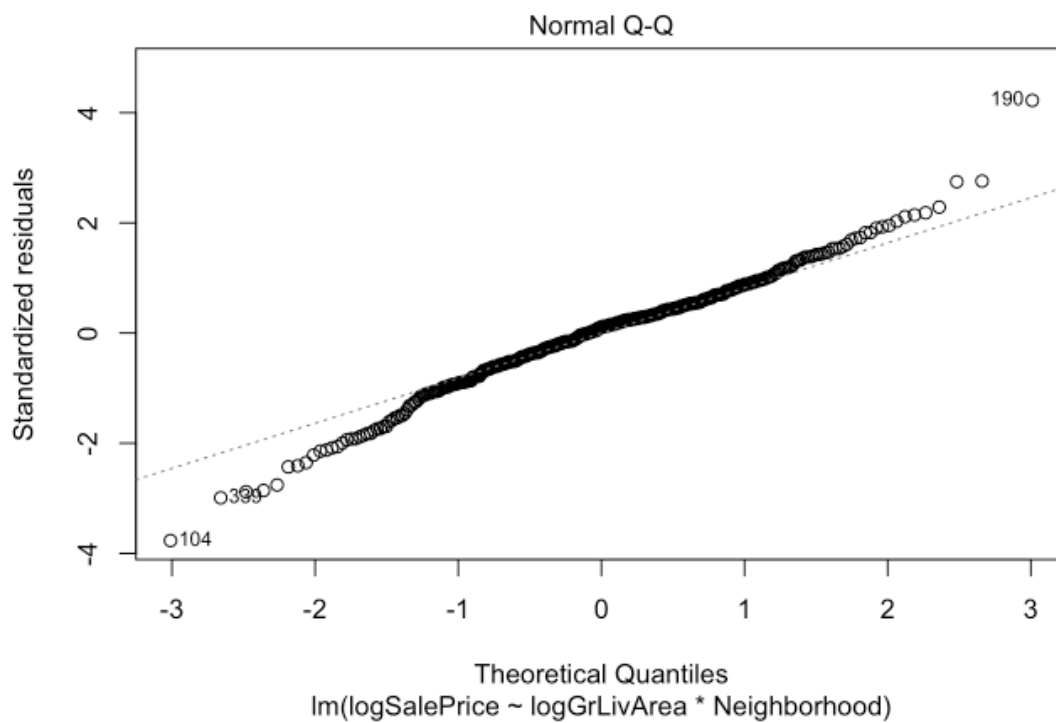


Diagram 14:

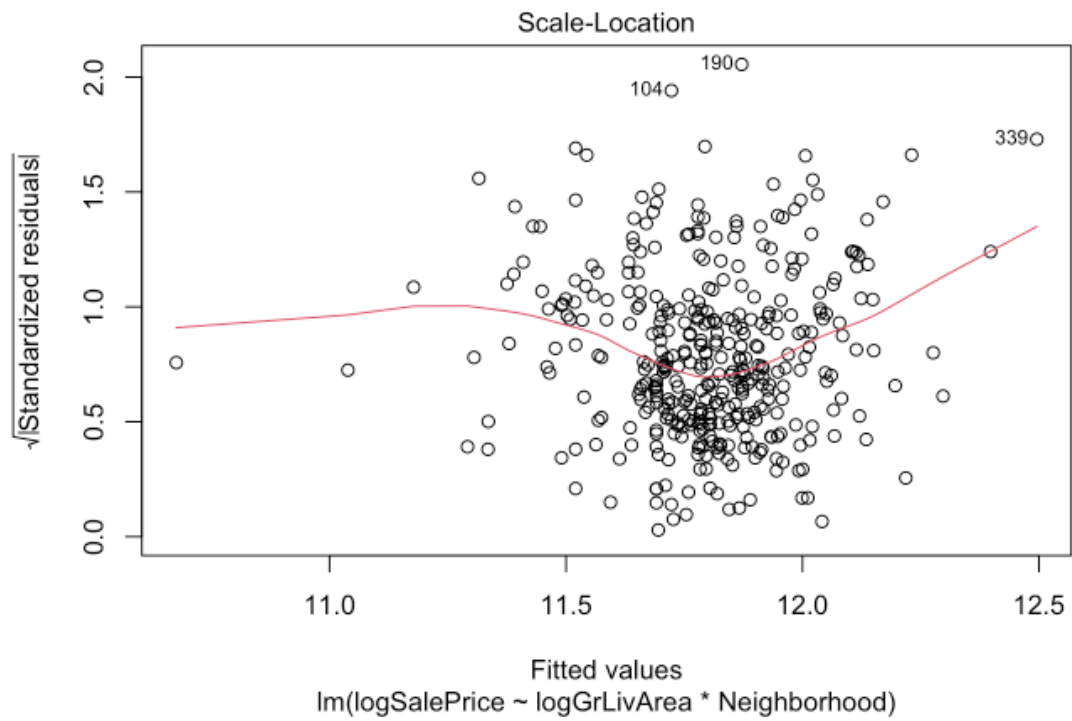


Diagram 15:

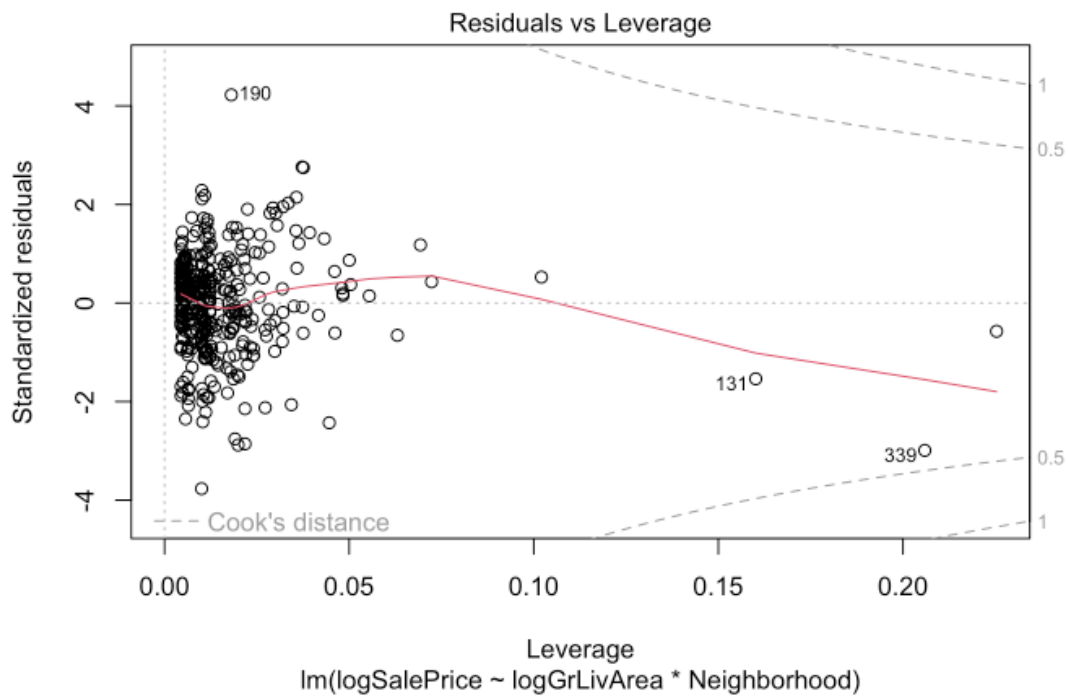


Diagram 16:

```
r$> confint(fit1)
```

	2.5 %	97.5 %
(Intercept)	4.9207572	6.9050843
logGrLivArea	0.6788064	0.9604897
NeighborhoodEdwards	0.8235795	3.3635933
NeighborhoodNames	1.4002744	3.7593394
logGrLivArea:NeighborhoodEdwards	-0.4793353	-0.1206263
logGrLivArea:NeighborhoodNames	-0.5134042	-0.1798447

Diagram 17:

```
# Data Cleaning

# If pool-related variables are NA, assume there is no pool and assign to 0
ames <- ames %>%
  mutate(
    PoolQC = ifelse(is.na(PoolQC), "None", PoolQC),
    PoolArea = ifelse(is.na(PoolArea), 0, PoolArea),
  )

# If garage-related variables are NA, assume there is no garage and assign to 0
ames <- ames %>%
  mutate(
    GarageType = ifelse(is.na(GarageType), "None", GarageType),
    #GarageYrBlt = ifelse(is.na(GarageYrBlt), 0, GarageYrBlt), #These will be changed to the mean because of large year values
    GarageFinish = ifelse(is.na(GarageFinish), "None", GarageFinish),
    GarageCars = ifelse(is.na(GarageCars), 0, GarageCars),
    GarageArea = ifelse(is.na(GarageArea), 0, GarageArea),
    GarageQual = ifelse(is.na(GarageQual), "None", GarageQual),
    GarageCond = ifelse(is.na(GarageCond), "None", GarageCond)
  )

# If Bsmt-related variables are NA, assume there is no Bsmt and assign to 0
ames <- ames %>%
  mutate(
    BsmtQual = ifelse(is.na(BsmtQual), "None", BsmtQual),
    BsmtCond = ifelse(is.na(BsmtCond), "None", BsmtCond),
    BsmtExposure = ifelse(is.na(BsmtExposure), "None", BsmtExposure),
    BsmtFinType1 = ifelse(is.na(BsmtFinType1), "None", BsmtFinType1),
    BsmtFinSF1 = ifelse(is.na(BsmtFinSF1), 0, BsmtFinSF1),
    BsmtFinType2 = ifelse(is.na(BsmtFinType2), "None", BsmtFinType2),
    BsmtFinSF2 = ifelse(is.na(BsmtFinSF2), 0, BsmtFinSF2),
    BsmtUnfSF = ifelse(is.na(BsmtUnfSF), 0, BsmtUnfSF),
    TotalBsmtSF = ifelse(is.na(TotalBsmtSF), 0, TotalBsmtSF)
  )

# If Fence-related variables are NA, assume there is no Fence and assign to 0
ames <- ames %>%
  mutate(
    Fence = ifelse(is.na(Fence), "None", Fence),
  )

# If Misc-related variables are NA, assume there is no Misc features and assign to 0
ames <- ames %>%
  mutate(
    MiscFeature = ifelse(is.na(MiscFeature), "None", MiscFeature),
  )

# If Fireplace-related variables are NA, assume there is no Fireplace and assign to 0
ames <- ames %>%
  mutate(
    FireplaceQu = ifelse(is.na(FireplaceQu), "None", FireplaceQu),
  )

# If ALLEY-related variables are NA, assume there is no ALLEY and assign to 0
ames <- ames %>%
  mutate(
    Alley = ifelse(is.na(Alley), "None", Alley),
  )

# Summarize the amount of remaining NA's by column to check what's left
colSums(is.na(ames))
```

Diagram 18:

```
# Use the dummyVars() function to convert categorical variables into dummy variables
# Then use janitor::clean_names() to clean up the column names
dummy_model <- dummyVars(~ ., data = ames)
ames_dummy <- as.data.frame(predict(dummy_model, newdata = ames))
ames_dummy <- clean_names(ames_dummy)

# Fill in all remaining na values with the mean of the column
ames_dummy <- ames_dummy %>%
  mutate(across(
    c(-sale_price, -log_sale_price),
    ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)
  ))
# Summarize the amount of remaining NA's by column
colSums(is.na(ames_dummy))
```

Diagram 19:

```
# Split the data into training and testing sets
train <- ames_dummy[ames_dummy$train == 1, ]
test <- ames_dummy[ames_dummy$train == 0, ]
```

Diagram 20:

Predictive Models	R ²	RMSE	AIC	Kaggle Score
Forward	0.942	0.096	-2504	0.1398
Backward	0.944	0.094	-2493	0.1458
Stepwise	0.944	0.094	-2494	0.1456
Custom	0.84	0.16	-1190	0.1777

Diagram 21:

```
r$> varImp(fit2$finalModel)%>%  
  filter(Overall > 4) %>%  
  arrange(desc(Overall))
```

	Overall
roof_matl_cly_tile	16.670625
overall_cond	12.316543
ms_zoning_c_all	12.095573
total_bsmt_sf	11.388951
condition2pos_n	9.922932
year_built	9.699077
overall_qual	9.667944
log_gr_liv_area	9.091300
neighborhood_crawfor	7.850502
lot_area	7.018695
ms_zoning_rm	6.353041
neighborhood_stone_br	5.836962
kitchen_qual_ex	5.560925
functional_maj2	5.489575
land_slope_sev	5.452246
neighborhood_nridg_ht	4.886112
bsmt_unf_sf	4.847542
neighborhood_edwards	4.778576

Diagram 22:

```
r$> varImp(fit3$finalModel)%>%  
  filter(Overall > 4) %>%  
  arrange(desc(Overall))
```

	Overall
log_gr_liv_area	16.984408
roof_matl_cly_tile	11.542834
overall_cond	11.497800
condition2pos_n	10.287568
ms_zoning_c_all	9.065580
overall_qual	9.047682
bsmt_fin_sf1	8.842386
lot_area	7.027994
sale_type_new	6.900116
neighborhood_edwards	6.697630
year_built	6.663097
functional_maj2	6.076291
kitchen_qual_ex	5.929819
neighborhood_crawfor	5.731806
bsmt_unf_sf	5.688449
condition1artery	5.647557
bsmt_fin_sf2	5.574467
land_slope_mod	5.512792
sale_condition_normal	5.292479
neighborhood_stone_br	5.180033
neighborhood_meadow_v	5.040680

Diagram 23:

```
r$> varImp(fit4$finalModel) %>%  
  filter(Overall > 4) %>%  
  arrange(desc(Overall))  
Overall  
log_gr_liv_area      16.998839  
overall_cond         11.540682  
roof_matl_cly_tile   11.463697  
condition2pos_n      10.233332  
ms_zoning_c_all       9.061792  
overall_qual         8.923294  
bsmt_fin_sf1         8.736801  
sale_type_new        6.862606  
lot_area             6.806617  
neighborhood_edwards 6.739504  
year_built           6.698790  
functional_maj2      6.103470  
kitchen_qual_ex      5.958244  
neighborhood_crawfor 5.710567  
condition1artery     5.645616  
bsmt_unf_sf          5.556263  
bsmt_fin_sf2         5.514768  
land_slope_mod       5.312905  
sale_condition_normal 5.303034  
neighborhood_stone_br 5.254721  
neighborhood_meadow_v 5.077700  
functional_mod       4.056417
```

Diagram 24:

```
r$> varImp(fit5) %>%  
  filter(Overall > 4) %>%  
  arrange(desc(Overall))  
Overall  
overall_qual         34.059059  
log_gr_liv_area      26.711591  
bsmt_fin_sf1         19.634865  
roof_matl_cly_tile   14.793029  
sale_type_new        10.923877  
lot_area             8.249606  
condition2pos_n      7.929971  
ms_zoning_c_all      7.765308  
overall_cond         6.073940
```

Diagram 25:

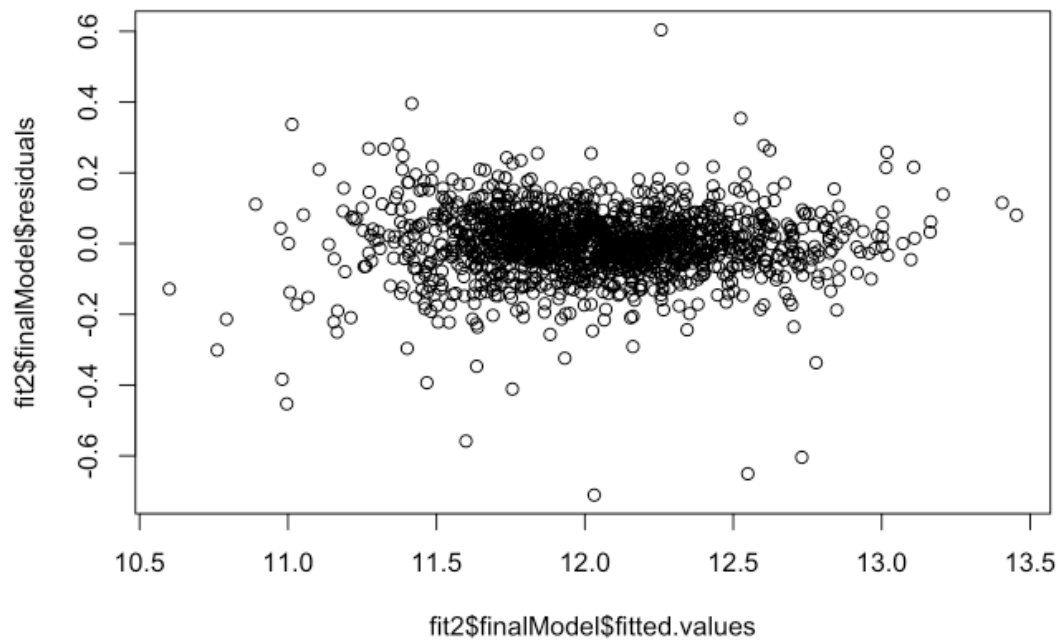


Diagram 26:

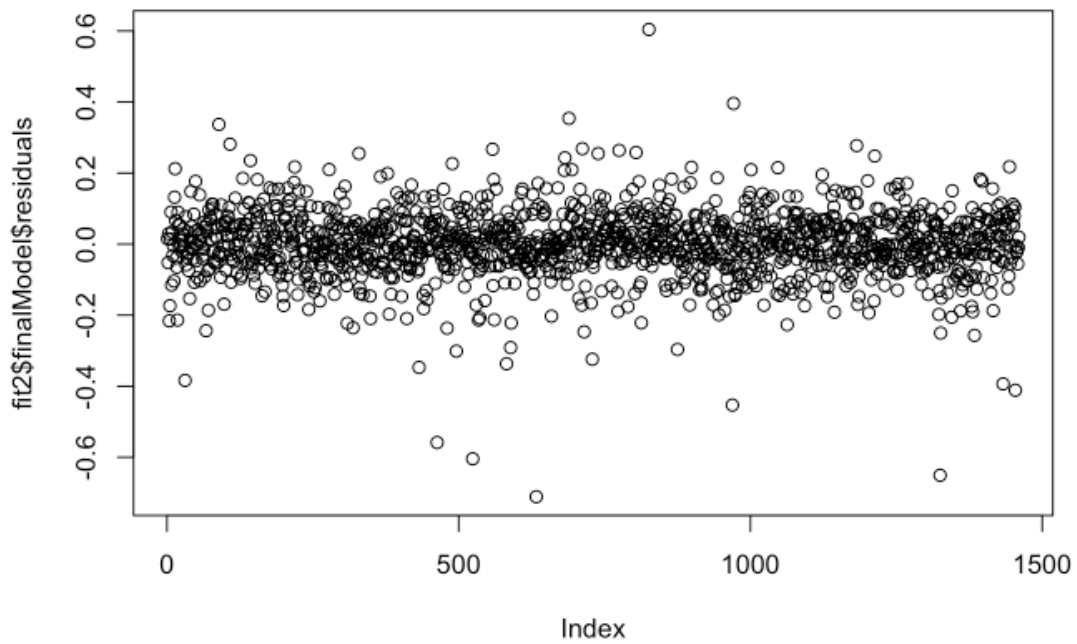


Diagram 27:

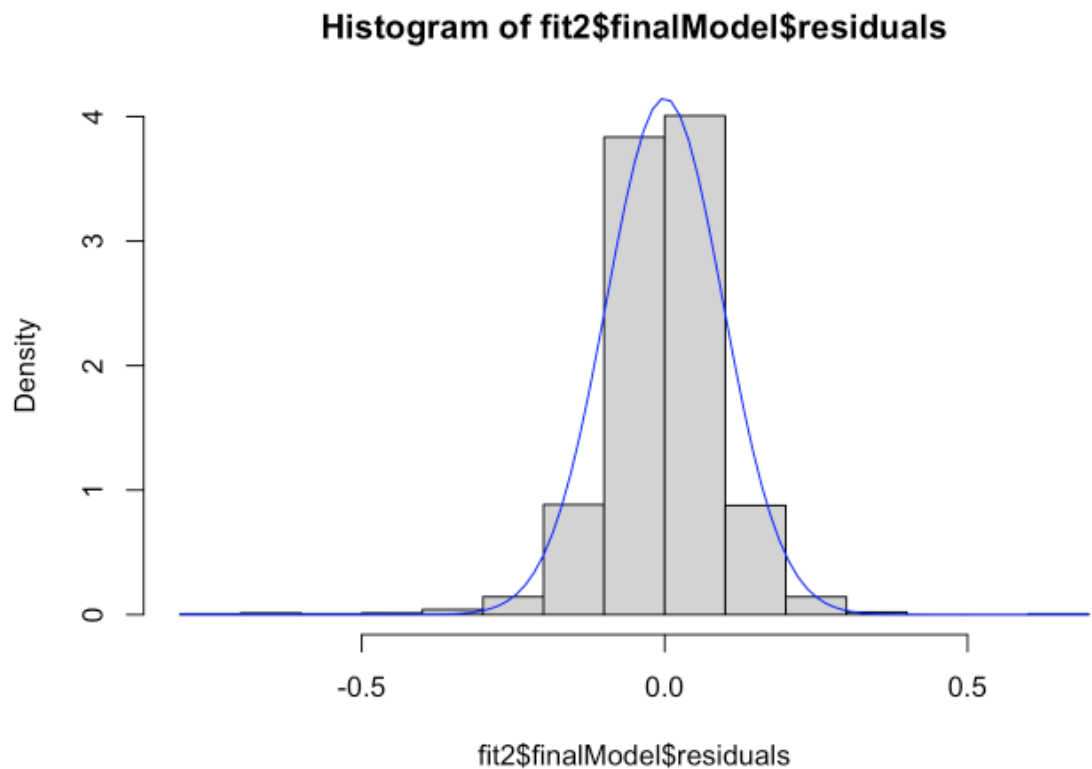


Diagram 28:

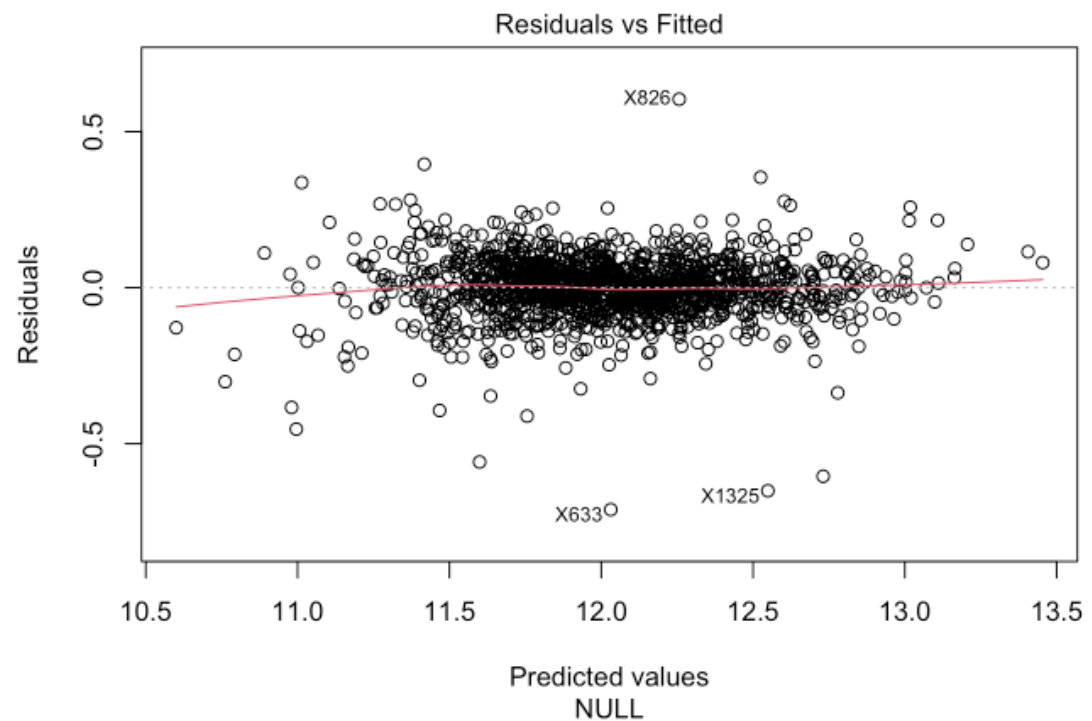


Diagram 29:

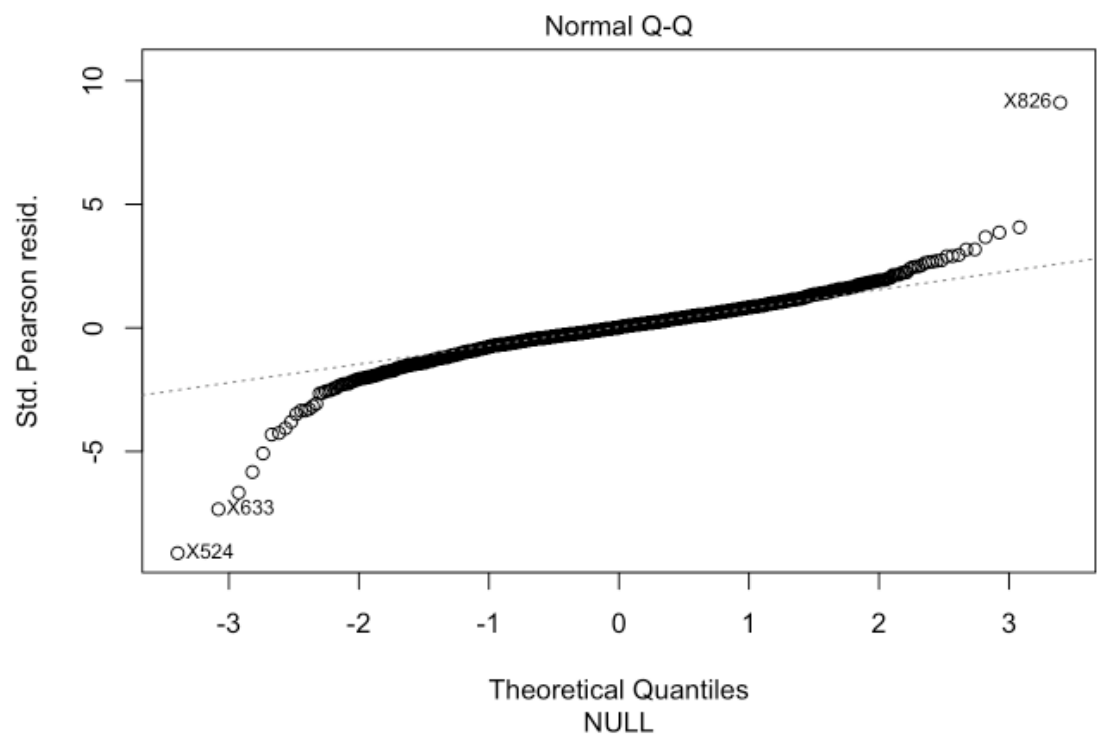


Diagram 30:

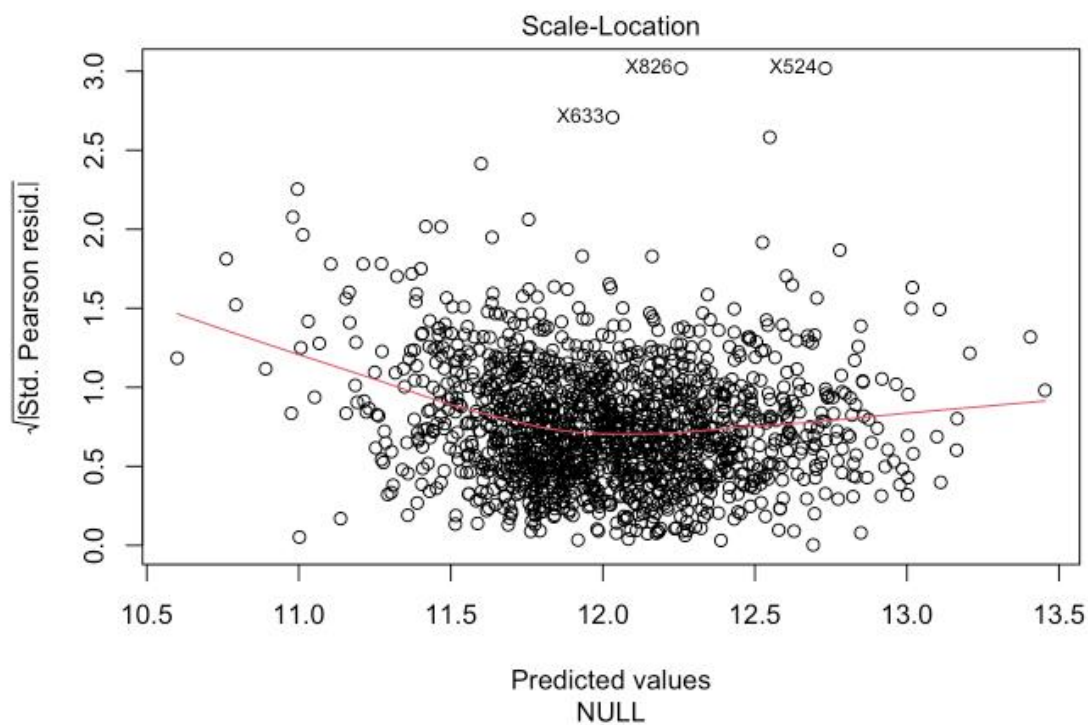
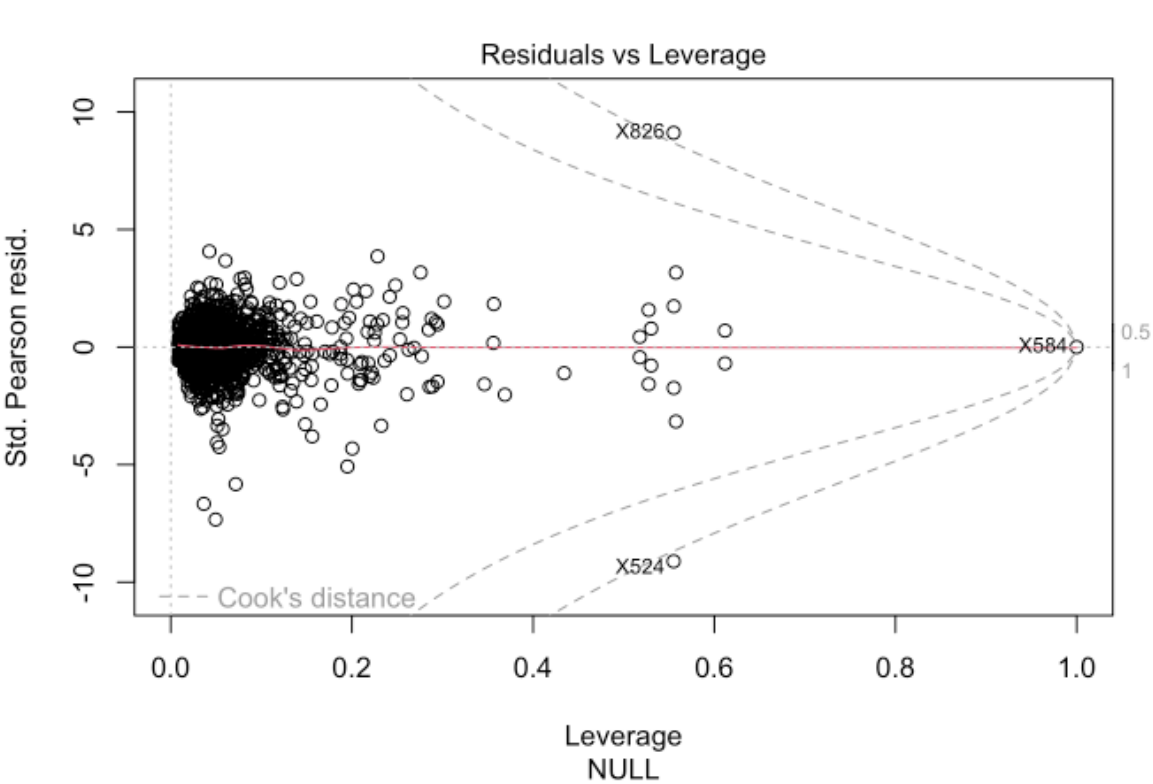


Diagram 31:



RMD:

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)

# Required Libraries
library(tidyverse)
library(knitr)
library(kableExtra)
library(ggthemes)
library(caret)
library(janitor)
library(doParallel)

#library(e1071)
#library(class)
train <- read.csv("Data/train.csv")
test <- read.csv("Data/test.csv")

# Merge the data frames and add a column indicating whether they come from
the train or test set
train$train <- 1
test$SalePrice <- NA
test$train <- 0
ames <- rbind(train, test)

# Verify data frame
head(ames)
str(ames)
summary(ames)
# Summarize NA's by column
ames %>%
  summarise_all(funs(sum(is.na(.)))) %>%
  gather(key = "Column", value = "NA_Count", -1) %>%
  filter(NA_Count > 0) %>%
  ggplot(aes(x = reorder(Column, NA_Count), y = NA_Count)) +
```

```

geom_col() +
coord_flip() +
theme_gdocs() +
labs(title = "Number of NA's by Column", x = "Column", y = "NA Count")

# Create a table of the missing NAs by column
ames %>%
  summarise_all(funs(sum(is.na(.)))) %>%
  gather(key = "Column", value = "NA_Count", -1) %>%
  filter(NA_Count > 0) %>%
  arrange(desc(NA_Count)) %>%
  kable()

# Plot Sale Price vs. Gross Living Area colored by neighborhood, omitting
rows where SalePrice is NA
ames %>%
  filter(!is.na(SalePrice)) %>%
  ggplot(aes(x = GrLivArea, y = SalePrice, color = Neighborhood)) +
  geom_point() +
  theme_gdocs() +
  labs(title = "Sale Price vs. Gross Living Area by Neighborhood", x = "Gross
Living Area", y = "Sale Price")

# Plot log(Sale Price) vs. log(Gross Living Area) colored by neighborhood,
omitting rows where SalePrice is NA
ames %>%
  filter(!is.na(SalePrice)) %>%
  ggplot(aes(x = log(GrLivArea), y = log(SalePrice), color = Neighborhood)) +
  geom_point() +
  theme_gdocs() +
  labs(
    title = "log(Sale Price) vs. log(Gross Living Area) by Neighborhood",
    x = "log(Gross Living Area)",
    y = "log(Sale Price)"
  )

# Create columns for log(SalePrice) and log(GrLivArea)
ames$logSalePrice <- log(ames$SalePrice)
ames$logGrLivArea <- log(ames$GrLivArea)

```



```

PRESS <- function(linear.model) {
  #' calculate the predictive residuals
  pr <- residuals(linear.model) / (1 - lm.influence(linear.model)$hat)
  #' calculate the PRESS
  PRESS <- sum(pr^2)

  return(PRESS)
}

# Function for calculating PRESS
# Tom Hopper
# https://gist.github.com/tomhopper/8c204d978c4a0cbcb8c0
# Plot log(Sale Price) vs. log(Gross Living Area) colored by neighborhood,
# omitting rows where SalePrice is NA for only the neighborhoods of interest
century21 <-
  ames %>%
  filter(!is.na(SalePrice)) %>%
  filter(Neighborhood %in% c("NAmes", "Edwards", "BrkSide"))
century21 %>%
  ggplot(aes(x = logGrLivArea, y = logSalePrice, color = Neighborhood)) +
  geom_point() +
  theme_gdocs() +
  labs(
    title = "log(Sale Price) vs. log(Gross Living Area) by Neighborhood",
    x = "log(Gross Living Area)",
    y = "log(Sale Price)"
  )
# Fit a linear model to the data
fit1x <- lm(logSalePrice ~ logGrLivArea + Neighborhood, data = century21)
summary(fit1x)
PRESS(fit1x)

# Fit a linear model to the data with interaction variables
fit1 <- lm(logSalePrice ~ logGrLivArea * Neighborhood, data = century21)
summary(fit1)

```

```

PRESS(fit1)
confint(fit1) %>% kable()

# Plot the data with the linear model superposed
century21 %>%
  ggplot(aes(x = logGrLivArea, y = logSalePrice, color = Neighborhood)) +
  geom_point() +
  theme_gdocs() +
  labs(
    title = "log(Sale Price) vs. log(Gross Living Area) by Neighborhood",
    x = "log(Gross Living Area)",
    y = "log(Sale Price)"
  ) +
  geom_smooth(
    method = "lm", formula = y ~ x, se = FALSE, linewidth = 1,
    data = data.frame(
      logGrLivArea = century21$logGrLivArea,
      Neighborhood = century21$Neighborhood,
      logSalePrice = predict(fit1)
    )
  )

# # Print parameter estimate table nicely. Not working, needs debugging
# fit1 %>%
#   summary() %>%
#   {cbind(as.data.frame(coef(.)), .[["coefficients"]][, 2:4])} %>%
#   setNames(c("Estimate", "Std. Error", "t-value", "Pr(>|t|)")) %>%
#   rownames_to_column(var = "Term") %>%
#   mutate(Term = ifelse(Term == "(Intercept)", "Intercept", Term)) %>%
#   add_row(Term = "Adjusted R-squared", Estimate = round(.$adj.r.squared,
# 3), Std..Error = NA, `t-value` = NA, `Pr(>|t|)` = NA) %>%
#   kable(digits = 3, align = "c") %>%
#   kable_styling(full_width = FALSE)
# Plot the studentized residuals using base R
plot(fit1$fitted.values, fit1$residuals, type = "p")

```

```

plot(fit1$residuals)

# Decide which of these we like better ggplot or R

# Calculate studentized residuals
stud_res <- rstudent(fit1)
# Create a data frame with the studentized residuals
df <- data.frame(stud_res, logGrLivArea = model.frame(fit1)$logGrLivArea)

# Create a scatterplot of the studentized residuals
ggplot(df, aes(x = logGrLivArea, y = stud_res)) +
  geom_point() +
  labs(title = "Scatterplot of Studentized Residuals",
    x = "Studentized Residuals",
    y = "Frequency") +
  theme_minimal()

# Create histogram with normal curve
ggplot(df, aes(x = stud_res)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5, fill = "lightblue",
    color = "black") +
  stat_function(fun = dnorm, args = list(mean = mean(df$stud_res), sd =
    sd(df$stud_res)), color = "blue", size = 1.2) +
  labs(title = "Histogram of Studentized Residuals with Normal Curve",
    x = "Studentized Residuals",
    y = "Frequency") +
  theme_minimal()

# Plot the residuals vs. the fitted values
fit1 %>%
  plot()

# Data Cleaning

# If pool-related variables are NA, assume there is no pool and assign to 0
ames <- ames %>%
  mutate(

```

```

    PoolQC = ifelse(is.na(PoolQC), "None", PoolQC),
    PoolArea = ifelse(is.na(PoolArea), 0, PoolArea),
  )
# If garage-related variables are NA, assume there is no garage and assign to 0
ames <- ames %>%
  mutate(
    GarageType = ifelse(is.na(GarageType), "None", GarageType),
    #GarageYrBlt = ifelse(is.na(GarageYrBlt), 0, GarageYrBlt), #These will be
    #changed to the mean because of large year values
    GarageFinish = ifelse(is.na(GarageFinish), "None", GarageFinish),
    GarageCars = ifelse(is.na(GarageCars), 0, GarageCars),
    GarageArea = ifelse(is.na(GarageArea), 0, GarageArea),
    GarageQual = ifelse(is.na(GarageQual), "None", GarageQual),
    GarageCond = ifelse(is.na(GarageCond), "None", GarageCond)
  )
# If Bsmt-related variables are NA, assume there is no Bsmt and assign to 0
ames <- ames %>%
  mutate(
    BsmtQual = ifelse(is.na(BsmtQual), "None", BsmtQual),
    BsmtCond = ifelse(is.na(BsmtCond), "None", BsmtCond),
    BsmtExposure = ifelse(is.na(BsmtExposure), "None", BsmtExposure),
    BsmtFinType1 = ifelse(is.na(BsmtFinType1), "None", BsmtFinType1),
    BsmtFinSF1 = ifelse(is.na(BsmtFinSF1), 0, BsmtFinSF1),
    BsmtFinType2 = ifelse(is.na(BsmtFinType2), "None", BsmtFinType2),
    BsmtFinSF2 = ifelse(is.na(BsmtFinSF2), 0, BsmtFinSF2),
    BsmtUnfSF = ifelse(is.na(BsmtUnfSF), 0, BsmtUnfSF),
    TotalBsmtSF = ifelse(is.na(TotalBsmtSF), 0, TotalBsmtSF)
  )
# If Fence-related variables are NA, assume there is no Fence and assign to 0
ames <- ames %>%
  mutate(
    Fence = ifelse(is.na(Fence), "None", Fence),
  )
# If Misc-related variables are NA, assume there is no Misc features and
assign to 0

```

```

ames <- ames %>%
  mutate(
    MiscFeature = ifelse(is.na(MiscFeature), "None", MiscFeature),
  )
# If Fireplace-related variables are NA, assume there is no Fireplace and assign to 0
ames <- ames %>%
  mutate(
    FireplaceQu = ifelse(is.na(FireplaceQu), "None", FireplaceQu),
  )
# If Alley-related variables are NA, assume there is no Alley and assign to 0
ames <- ames %>%
  mutate(
    Alley = ifelse(is.na(Alley), "None", Alley),
  )

# Summarize the amount of remaining NA's by column to check what's left
colSums(is.na(ames))

# Use the dummyVars() function to convert categorical variables into dummy variables
# Then use janitor::clean_names() to clean up the column names
dummy_model <- dummyVars(~ ., data = ames)
ames_dummy <- as.data.frame(predict(dummy_model, newdata = ames))
ames_dummy <- clean_names(ames_dummy)

# Fill in all remaining na values with the mean of the column
ames_dummy <- ames_dummy %>%
  mutate(across(
    c(-sale_price, -log_sale_price),
    ~ ifelse(is.na(.), mean(., na.rm = TRUE), .)
  ))
# Summarize the amount of remaining NA's by column
colSums(is.na(ames_dummy))

```

```

# Split the data into training and testing sets
train <- ames_dummy[ames_dummy$train == 1, ]
test <- ames_dummy[ames_dummy$train == 0, ]
# Forward Selection

# # Testing olsrr method.
# library(olsrr)
# fit2x <- lm(log_sale_price ~ . - sale_price, data = train)
# fit2y <- ols_step_forward_p(fit2x, penter = 0.15)$model
# summary(fit2y)
# defaultSummary(data.frame(pred = predict(fit2y), obs =
train$log_sale_price))
# PRESS(fit2y)

# Check if the model object exists, train if it doesn't
if (file.exists("Models/lm_forwards.rds")) {
  # Load the model object from disk
  fit2 <- readRDS("Models/lm_forwards.rds")
} else {
  # Perform stepwise selection

  # Set up a parallel backend with the number of cores you want to use
  cores <- 8 # Change this to the number of cores you want to use
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  set.seed(137)
  ctrl <- trainControl(
    method = "boot",
    number = 5,
    allowParallel = TRUE
  )
  fit2 <- train(log_sale_price ~ . - sale_price,
    data = train,
    method = "glmStepAIC",

```

```

    trControl = ctrl,
    direction = "forward",
    penter = 0.05 # Not Working.
  )

  # Stop the parallel backend
  stopCluster(cl)

  # Save the model object to disk
  saveRDS(fit2, "Models/lm_forwards.rds")
}

summary(fit2$finalModel)
defaultSummary(data.frame(pred = predict(fit2), obs = train$log_sale_price))
PRESS(fit2$finalModel) #Press not working with caret models
varImp(fit2$finalModel)%>%
  filter(Overall > 4) %>%
  arrange(desc(Overall))

# Output the predictions for the test set to a csv file
# fit2x <- glm(formula = formula(fit2), data = train)
forward_pred <- predict(fit2$finalModel, test)

forward_pred %>%
  data.frame() %>%
  rownames_to_column(var = "id") %>%
  mutate(SalePrice = exp(forward_pred)) %>%
  dplyr::select(id, SalePrice) %>%
  write_csv("Predictions/forward_predictions.csv")
# Backwards Selection

# Check if the model object exists, train if it doesn't
if (file.exists("Models/lm_backwards.rds")) {
  # Load the model object from disk

```

```

fit3 <- readRDS("Models/lm_backwards.rds")
} else {
  # Perform stepwise selection

  # Set up a parallel backend with the number of cores you want to use
  cores <- 8 # Change this to the number of cores you want to use
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  set.seed(137)
  fit3 <- train(log_sale_price ~ . - sale_price,
    data = train,
    method = "glmStepAIC",
    trControl = trainControl(method = "cv", number = 5, allowParallel =
TRUE),
    direction = "backward",
    penter = 0.05 # Not Working.
  )

  # Stop the parallel backend
  stopCluster(cl)

  # Save the model object to disk
  saveRDS(fit3, "Models/lm_backwards.rds")
}

summary(fit3$finalModel)
defaultSummary(data.frame(pred = predict(fit3), obs = train$log_sale_price))
PRESS(fit3$finalModel) # Press not working with caret models
varImp(fit3$finalModel)%>%
  filter(Overall > 4) %>%
  arrange(desc(Overall))

# Output the predictions for the test set to a csv file

```



```

# fit3x <- glm(formula = formula(fit3), data = train)
backward_pred <- predict(fit3$finalModel, newdata = test)

backward_pred %>%
  data.frame() %>%
  rownames_to_column(var = "id") %>%
  mutate(SalePrice = exp(backward_pred)) %>%
  dplyr::select(id, SalePrice) %>%
  write_csv("Predictions/backward_predictions.csv")

# Stepwise Selection

# Check if the model object exists, train if it doesn't
if (file.exists("Models/lm_stepwise.rds")) {
  # Load the model object from disk
  fit4 <- readRDS("Models/lm_stepwise.rds")
} else {
  # Perform stepwise selection

  # Set up a parallel backend with the number of cores you want to use
  cores <- 8 # Change this to the number of cores you want to use
  cl <- makePSOCKcluster(cores)
  registerDoParallel(cl)

  set.seed(137)
  fit4 <- train(log_sale_price ~ . - sale_price,
    data = train,
    method = "glmStepAIC",
    trControl = trainControl(method = "cv", number = 5, allowParallel =
TRUE),
    direction = "both",
    penter = 0.05 # Not Working.
  )

  # Stop the parallel backend
  stopCluster(cl)

```

```

# Save the model object to disk
saveRDS(fit4, "Models/lm_stepwise.rds")
}

summary(fit4$finalModel)
defaultSummary(data.frame(pred = predict(fit4), obs = train$log_sale_price))
PRESS(fit4$finalModel) # Press not working with caret models
varImp(fit4$finalModel) %>%
  filter(Overall > 4) %>%
  arrange(desc(Overall))

# Output the predictions for the test set to a csv file
# fit4x <- glm(formula = formula(fit4), data = train)
stepwise_pred <- predict(fit4$finalModel, newdata = test)

stepwise_pred %>%
  data.frame() %>%
  rownames_to_column(var = "id") %>%
  mutate(SalePrice = exp(stepwise_pred)) %>%
  dplyr::select(id, SalePrice) %>%
  write_csv("Predictions/stepwise_predictions.csv")

# Custom Feature Selection
top10 <- varImp(fit4$finalModel) %>%
  filter(Overall > 4) %>%
  arrange(desc(Overall)) %>%
  head(10) %>%
  rownames()

form <- as.formula(paste("log_sale_price ~", paste(top10, collapse = "+")))
fit5 <- lm(form, data = train)

summary(fit5)

```

```
defaultSummary(data.frame(pred = predict(fit5), obs = train$log_sale_price))
PRESS(fit5) # Press not working with caret models
# varImp(fit5$finalModel) %>%
#   filter(Overall > 4) %>%
#   arrange(desc(Overall))

# Output the predictions for the test set to a csv file
# fit4x <- glm(formula = formula(fit4), data = train)
custom_pred <- predict(fit5, newdata = test)

custom_pred %>%
  data.frame() %>%
  rownames_to_column(var = "id") %>%
  mutate(SalePrice = exp(custom_pred)) %>%
  dplyr::select(id, SalePrice) %>%
  write_csv("Predictions/custom_predictions.csv")
```