Modelling Competition: House Prices

Lúa Arconada, Nuria Errasti, Alejandro Macías, Carlos Martin, Iván Samuel Orta

23/12/2023

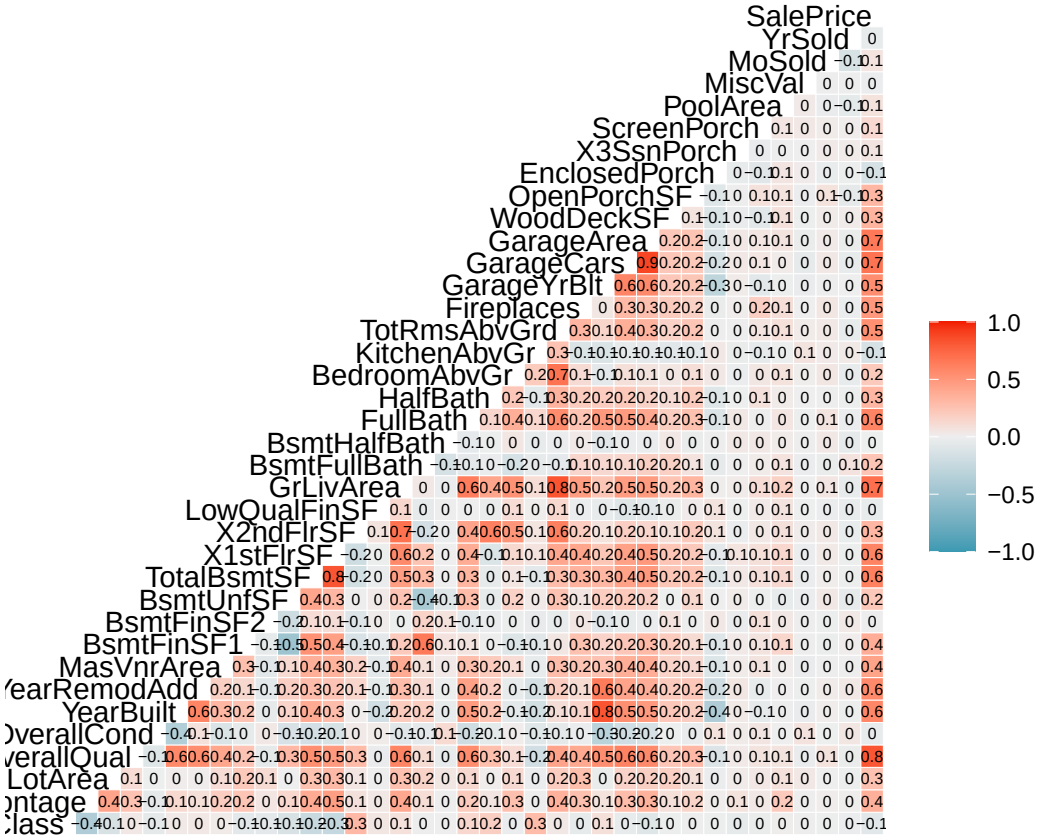
Introduction

The purpose of this report is to outline the process of constructing a predictive model for house prices based on diverse characteristics such as bedrooms, electrical systems, and more. We'll start by detailing the data preprocessing steps, leading to model fitting and prediction. Throughout the report, we'll also explore additional interesting aspects like the most influential variables and characteristics of the most expensive and least expensive houses.

Data preparation and exploratory data analysis

First of all, upon loading the datasets 'data_train' and 'data_test,' we observed that the variable 'Id' serves as a unique identifier for each observation, rendering it irrelevant for our analysis and being deleted. Our target variable, 'SalePrice,' has been transformed using a logarithmic function as requested.

Let's investigate whether there are any discernible patterns that might suggest correlations between variables. We'll calculate the Pearson correlation among the features.



In the graph, we can observe a notable correlation between certain variables and the response variable, indicating potential multicollinearity among them. Notably, OverallQual, GrLivArea, GarageCars, and GarageArea exhibit strong correlations with SalePrice, suggesting their importance in the model.

Additionally, other significant correlations include:

-YearBuilt with GarageYrBlt

-TotalBsmtSF with X1stFlrSF

-GrLivArea with TotRmsAbvGrd and X2ndFlrSF

-BedroomAbvGr with TotRmsAbvGrd

-GarageCars with GarageArea

We've conducted a summary analysis of our datasets and identified numerous issues. Moving forward, the solutions we'll present apply to both the train and test datasets. Let's further explore these problems and the corresponding resolutions.

1- The variable 'Utilities' has only two values among the four available, with one of them represented by only a single observation. Consequently, this variable has been removed as it won't contribute effectively to the model.

2-We encountered several discrete variables with limited values, which could impact prediction accuracy. For instance, houses seldom have more than three basement full bathrooms. Assuming that the precise count beyond three might not significantly influence the price, we categorized these variables. Houses with over three bathrooms were grouped under a new category labeled 'more.' This approach was extended to similar variables like BsmtFullBath, HalfBath, KitchenAbvGr, among others. Additionally, in preparation for future predictions, we converted character variables into factors.

3-We've identified various missing values that require specific treatment as they differ in nature and cannot be uniformly addressed.

Initially, some NA's in certain variables don't represent true missing data but rather indicate a specific category (such as in 'GarageQual' where NA signifies 'No Garage'). To manage these, we transformed the NA category into 'Doesn't have.'

Moving on to 'GarageYrBlt,' the NA values signify instances where a garage was never constructed. In such cases, we plan to categorize the variable into decades, assigning NA's to a new category labeled 'Doesn't have.'

The remaining NA's are legitimate missing values. In our training dataset, these amount to over 200, a substantial count. To address these, we've developed a specialized algorithm inspired by KNN but tailored to suit the unique nature of our dataset.

We've come across a substantial number of variables, totaling 79, and a significant portion of these variables are now categorical (57). This presented an opportunity for us to provide coherent values for the missing entries (NAs) and to correct errors in both train and test data sets. We've devised a program that implements a comparative approach. For each observation featuring an NA within a particular variable, it compares this specific observation with all others lacking an NA in the same variable.

Through this process, it evaluates the similarity of categorical variables among observations. Computes the number of identical categories for each observation concerning other categorical variables. The rationale behind this approach lies in the assumption that if 'closest' observations exhibit numerous identical categories, it's possible that these houses share similar properties. Consequently, their unknown values might also align closely. Therefore, taking the mean for numerical values or mode for categorical values of these similar observations can potentially lead to accurate predictions for the NA values. This approach bears similarity to KNN but capitalizes on the abundance of categorical values available to us. We have considered selecting as closest observations, (taking 'n' as the maximum number of coincidences among all observations, those with 'n', 'n-1', and 'n-2' matches).

Let's proceed to observe the performance of the algorithm:

At first glance, one might assume that considering only matches with n, n-1, and n-2 similarities isn't sufficient. However, upon analysis, we observed that this approach captures an acceptable number of 'close' observations. We illustrate this with the mean of the chosen observation for each predicted NA:

```
## mean number of close observations chosen: 8.836957
```

Additionally, for each observation with an NA to be predicted, our selected close observations consistently exhibit numerous identical categories. Specifically, we observed:

```
## minimum number of coincidences: 33
```

```
##  maximum number of  coincidences: 55
```

```
##  mean number of  coincidences: 44.75362
```

When examining numerical variables for predictions, we compared the potential value range within the train dataset for a specific variable to the range of values among our selected close observations. Our analysis focused on 'LotFrontage' and 'MasVnrArea,' both featuring NAs. What we found was particularly insightful: houses sharing numerous identical categories exhibited notably smaller value ranges. We computed the interval length (from minimum to maximum values) and compared it to analogous intervals composed solely of our closest observations to the NA-filled observation. This analysis revealed that not only the mean length of these intervals but also the largest among them was significantly smaller. This observation suggests that having identical categories may lead to closer values in numerical variables, indicating that our procedure may be more effective than simply taking the mean of all variable values.

```
## length interval of possible values in LotFrontage: 292
```

```
##  mean length interval of possible values of close observations in LotFrontage: 39.00386
```

```
##   length of largest interval of possible values of close observations in LotFrontage: 150
```

```
## length interval of possible values in MasVnrArea: 1600
```

```
##  mean length interval of possible values of close observations in MasVnrArea: 260.875
```

```
##   length of largest interval of possible values of close observations in MasVnrArea: 598
```

A similar method was applied when predicting categorical values using the mode derived from the closest selected observations for each NA. Interestingly, a lot of these observations consistently shared the same category in the variable of interest, even if this category wasn't prevalent when considering the entire dataset.

Hence, our algorithm appears to be an intriguing approach to handling NAs. As a result, we implemented it on both the train and test datasets to address the missing values.

4-We've identified instances with errors in certain observations like 961, 1044, and 1140, where the presence of a pool (PoolQC= Doesn't have) contradicts the recorded pool area (PoolArea>0). Similar inconsistencies occur in other variables across multiple instances. To rectify this, we treated these instances as NA's (both PoolQC and PoolArea in the aforementioned example) and applied our program again to predict and correct these values.

As an example of our program's effectiveness in rectifying errors, we present the updated values of the previously mentioned erroneous instances.

```
data_train[c(961,1044,1140),"PoolQC"]
```

```
## [1] Doesn't have Doesn't have Doesn't have
## Levels: Doesn't have Ex Fa Gd
```

```
data_train[c(961,1044,1140),"PoolArea"]
```

```
## [1] 0 0 0
```

As observed, these instances still lack a pool, yet the PoolArea has been automatically rectified to 0. This exemplifies the potential of our model. It accurately identified that houses with similar categorical values generally do not have a pool, appropriately adjusting both variables.

We proceed similarly with the 'data_test' dataset, as we encountered similar errors.

After this extensive preprocessing, we finally have deeply cleansed data, poised for predicting house prices.

Model fitting, variable selection and predictive power

To initiate the model fitting process, our initial step was to create a basic model using solely the numeric variables:

```
##
## Call:
## lm(formula = formula, data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62986 -0.07231  0.00469  0.08604  0.73518
##
## Coefficients: (2 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.478e+00  5.557e-01  -4.460 8.84e-06 ***
## MiscVal       -4.436e-06  9.683e-06  -0.458  0.64695
## LotFrontage    2.282e-04  2.508e-04   0.910  0.36299
## LotArea        2.041e-06  5.318e-07   3.837  0.00013 ***
## YearBuilt      3.250e-03  2.384e-04  13.635  < 2e-16 ***
## YearRemodAdd   3.659e-03  3.012e-04  12.149  < 2e-16 ***
## MasVnrArea     4.385e-05  3.099e-05   1.415  0.15729
## BsmtFinSF1     1.706e-04  2.158e-05   7.905 5.28e-15 ***
## BsmtFinSF2     1.459e-04  3.564e-05   4.092 4.51e-05 ***
## BsmtUnfSF      1.159e-04  2.115e-05   5.478 5.06e-08 ***
## TotalBsmtSF           NA         NA      NA       NA
## X1stFlrSF      3.093e-04  2.367e-05  13.070  < 2e-16 ***
## X2ndFlrSF      3.154e-04  1.316e-05  23.971  < 2e-16 ***
## LowQualFinSF   1.723e-04  1.011e-04   1.705  0.08849 .
## GrLivArea             NA         NA      NA       NA
## GarageArea     2.903e-04  2.995e-05   9.693  < 2e-16 ***
## WoodDeckSF     1.985e-04  4.152e-05   4.782 1.91e-06 ***
## OpenPorchSF    5.291e-05  7.856e-05   0.674  0.50072
## EnclosedPorch  2.553e-04  8.710e-05   2.931  0.00343 **
## X3SsnPorch     3.028e-04  1.645e-04   1.840  0.06592 .
## ScreenPorch    5.948e-04  8.880e-05   6.698 3.02e-11 ***
## PoolArea      -5.347e-04  1.236e-04  -4.327 1.62e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1828 on 1440 degrees of freedom
## Multiple R-squared:  0.7932, Adjusted R-squared:  0.7905
## F-statistic: 290.7 on 19 and 1440 DF,  p-value: < 2.2e-16
```

As we can see from the summary of the model, there are some coefficients that are returned as NA. This is most likely due to perfect collinearity, which can be confirmed by noting that the problematic variables are in fact the ones that showed high correlations in the correlation plot shown in the begging. The most problematic variables, namely 'TotalBsmtSF' and 'GrLivArea' have been removed from the model.

This same process can be carried out considering only the categorical variables.

The next step involves consolidating all variables into the same model, which might reveal more variables prone to elimination due to similar issues.

Once we did this, it seemed that incorporating all variables introduced new issues with singularity (a similar summary was displayed to see this). Therefore, we removed problematic variables again and updated the model accordingly.

It looks like this model has successfully dealt with singularities in the coefficients by removing problematic variables. However, there might be room for improvement by exploring potential interactions between variables. For this purpose, we've developed the following code:

```
interactions = function(model, df){
  for (var1 in c(names(df))){
    for (var2 in c(names(df))){
      # avoid self-interaction terms (quadratic)
      if (var1 != var2){
        # get the formula of the current model
        current.formula = formula(model)
        # create the formula with the new added interaction
        new.formula = update(current.formula, as.formula(paste(". ~ . + ",
                                                               var1, ":",var2)))
        # new.formula = as.formula(paste(as.character(current.formula), "+", var1, ":",
        #                                var2))
        # creates and refits the model with the new interactions
        new.model = lm(new.formula, data=df)
        # checks if the interaction is significant
        pvalue = anova(new.model)$`Pr(>F)`[2]
        # removes it if not significant
        if ((pvalue > 0.05) | (is.na(pvalue))){
          new.formula = update(formula(new.model), as.formula(paste(". ~ . -",
                                                                    var1, ":",
                                                                    var2)))
          #new.formula = as.formula(paste(as.character(formula(new.model)), "-",
          #                               var1, ":", var2))
          new.model = lm(new.formula, data=df)
          # new_formula = update(formula(new_model), .~. -var
        }
      model = new.model
      }
    }
  }
}
```

The intention behind this code is to introduce potential interactions one by one into the model and evaluate their significance. Despite multiple attempts, integrating these interactions hasn't shown any noticeable improvement in the model's performance. Additionally, given the extensive number of variables, some with numerous categories, exploring interactions computationally becomes exceedingly challenging, almost rendering their inclusion in the model impractical.

To estimate the performance in terms of predictive power of the model, we used cross validation, partitioning the training data set into training and validation partitions and used R function 'lm'.

One challenge we might encounter during cross-validation with the model is when fitting on the training partition. Certain categories within variables might be significantly less common and may appear in the validation partition but not in the training partition. To address this, we've created specific functions to

manage this issue. We use these functions to generate predictions.

```
#### my functions
# find
damaged_categories=function(toy_model,toy_data){
  damaged_levels=c()
  for(i in names(toy_model$xlevels) ){
    if(! length(toy_model$xlevels[[i]])==length( levels(toy_data[,i])) ){
      damaged_levels=c(damaged_levels,i)
    }
  }
  return(damaged_levels)
}
# fix
fix_levels=function(toy_model,toy_data,categories){
  for(i in categories){
    toy_model$xlevels[[i]]=levels(toy_data[,i])
  }
  return(toy_model)
}
```

```
#adjust levels
needed_levels=damaged_categories(modall,performance_train)
modall = fix_levels(modall, performance_train, needed_levels)



##check performance
# RMSE of train train
caret::RMSE(pred = exp(modall$fitted.values), obs = exp(performance_train$SalePrice))
```

```
## [1] 17517.61
```

```
# RMSE of train test
pred = predict.lm(modall, newdata = performance_test)
caret::RMSE(pred = exp(pred), obs = exp(performance_test$SalePrice))
```

```
## [1] 149061.3
```

So we obtained an RMSE on the testing data of 149061.3. Taking into account the high values of the price of houses, it seems to be an acceptable model.

The model could potentially be improved further by utilizing the 'stepAIC' function, which employs the Akaike Information Criterion (AIC) to identify the best model from various possibilities comprising different variable combinations.

```
model.aic = stepAIC(modall, direction="both", trace=0)
```

To compare the original model with the one obtained through the Akaike Information Criterion, we can also use the ANOVA test:

```
anova(model.aic, modall)
```

```
## Analysis of Variance Table
##
## Model 1: SalePrice ~ LotArea + YearBuilt + YearRemodAdd + MasVnrArea +
##     BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF + X2ndFlrSF +
##     LowQualFinSF + GarageArea + WoodDeckSF + EnclosedPorch +
```

```
##     ScreenPorch + PoolArea + MSSubClass + MSZoning + LotShape +
##     LandContour + LotConfig + LandSlope + Neighborhood + Condition1 +
##     Condition2 + OverallQual + OverallCond + RoofMatl + Exterior1st +
##     BsmtQual + BsmtExposure + Heating + HeatingQC + BsmtFullBath +
##     FullBath + KitchenQual + TotRmsAbvGrd + Functional + Fireplaces +
##     GarageType + PoolQC + Fence + YrSold + SaleCondition + CentralAir +
##     HalfBath + KitchenAbvGr + MiscFeature
## Model 2: SalePrice ~ LotFrontage + LotArea + YearBuilt + YearRemodAdd +
##     MasVnrArea + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF +
##     X2ndFlrSF + LowQualFinSF + GarageArea + WoodDeckSF + OpenPorchSF +
##     EnclosedPorch + X3SsnPorch + ScreenPorch + PoolArea + MSSubClass +
##     MSZoning + Alley + LotShape + LandContour + LotConfig + LandSlope +
##     Neighborhood + Condition1 + Condition2 + HouseStyle + OverallQual +
##     OverallCond + RoofStyle + RoofMatl + Exterior1st + MasVnrType +
##     ExterQual + ExterCond + Foundation + BsmtQual + BsmtExposure +
##     BsmtFinType2 + Heating + HeatingQC + BsmtFullBath + FullBath +
##     BedroomAbvGr + KitchenQual + TotRmsAbvGrd + Functional +
##     Fireplaces + GarageType + PavedDrive + PoolQC + Fence + MiscFeature +
##     MoSold + YrSold + SaleType + SaleCondition + Street + CentralAir +
##     BsmtHalfBath + HalfBath + KitchenAbvGr
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    836 8.3423
## 2    773 7.7820 63   0.56037 0.8835 0.7269
```

The test clearly indicates that the model obtained through AIC isn't superior to the original one. Additionally, we experimented with simpler models, like eliminating all observations with missing values and making predictions using all variables, among other combinations. Unfortunately, none of these approaches seemed to enhance the previously obtained RMSE.

Best model

According to both the ANOVA test and the validation RMSE, the most optimal model was generated by addressing missing values through our KNN-based algorithm, utilizing all variables except those prone to causing singularities, and omitting the StepAIC and interactions approaches. Thus, this is the model we used to predict the prices of interest (those of data_test) now using the 100% of train_data. The code that gives our final predictions it then the following:

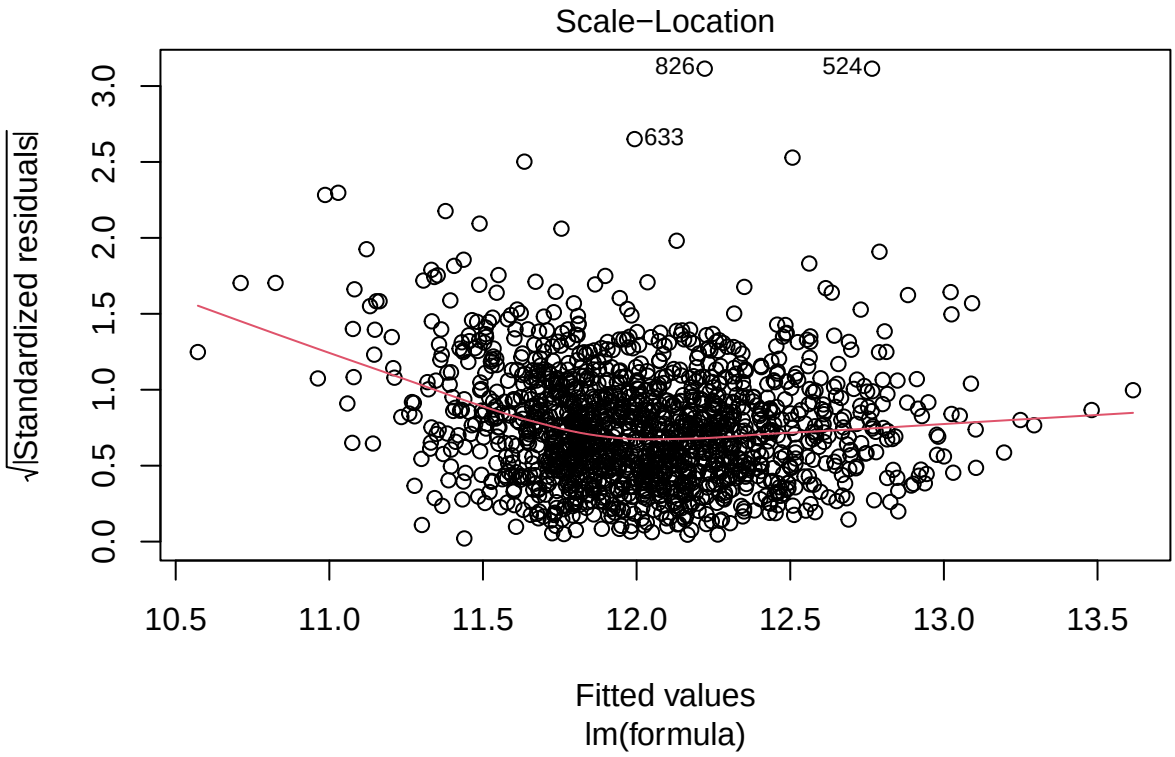
```
formula = SalePrice ~ MiscVal

for (var in numeric){
  formula = update(formula, as.formula(paste(". ~ . +", var)))
}

formula = update(formula, as.formula(paste(". ~ . - MiscVal")))

for (var in categorical){
  formula = update(formula, as.formula(paste(". ~ . + ", var)))
}



for (var in binary){
  formula = update(formula, as.formula(paste(". ~ . + ", var)))
}
```

```
formula = update(formula, as.formula(paste(". ~ . - TotalBsmtSF")))
formula = update(formula, as.formula(paste(". ~ . - GrLivArea")))
formula = update(formula, as.formula(". ~ . - BldgType"))
formula = update(formula, as.formula(". ~ . - Exterior2nd"))
formula = update(formula, as.formula(". ~ . - GarageCars"))
formula = update(formula, as.formula(". ~ . - BsmtCond"))
formula = update(formula, as.formula(". ~ . - BsmtFinType1"))
formula = update(formula, as.formula(". ~ . - Electrical"))
formula = update(formula, as.formula(". ~ . - FireplaceQu"))
formula = update(formula, as.formula(". ~ . - GarageYrBlt"))
formula = update(formula, as.formula(". ~ . - GarageFinish"))
formula = update(formula, as.formula(". ~ . - GarageQual"))
formula = update(formula, as.formula(". ~ . - GarageCond"))

best.model = lm(formula, data=data_train)

#needed_levels=damaged_categories(best.model, data_test)
#best.model = fix_levels(best.model, data_test, needed_levels)

#levels(data_test$Alley) = c("Doesn't have", "1", "2")
levels(data_test$BsmtQual) = levels(data_train$BsmtQual)
levels(data_test$BsmtCond) = levels(data_train$BsmtCond)
levels(data_test$BsmtExposure) = levels(data_train$BsmtExposure)
levels(data_test$BsmtFinType1) = levels(data_train$BsmtFinType1)
levels(data_test$BsmtFinType2) = levels(data_train$BsmtFinType2)
levels(data_test$FireplaceQu) = levels(data_train$FireplaceQu)
levels(data_test$GarageType) = levels(data_train$GarageType)
levels(data_test$GarageFinish) = levels(data_train$GarageFinish)
levels(data_test$GarageCond) = levels(data_train$GarageCond)
levels(data_test$Fence) = levels(data_train$Fence)
levels(data_test$GarageQual) = levels(data_train$GarageQual)
levels(data_test$PoolQC) = levels(data_train$PoolQC)
levels(data_test$MiscFeature) = levels(data_train$MiscFeature)

data_test$MSSubClass[data_test$MSSubClass == 150] = 120

pred = exp(predict.lm(best.model, newdata=data_test))
#the exponential is because we where working with the logarithm of Salesprice
```

Goodness of fit, model assumptions

Linearity.

Residual plots serve as a valuable tool in detecting non-linearity. Any discernible pattern within the residual plot indicates areas for potential model improvement or suggests a deviation from meeting the linearity assumption.

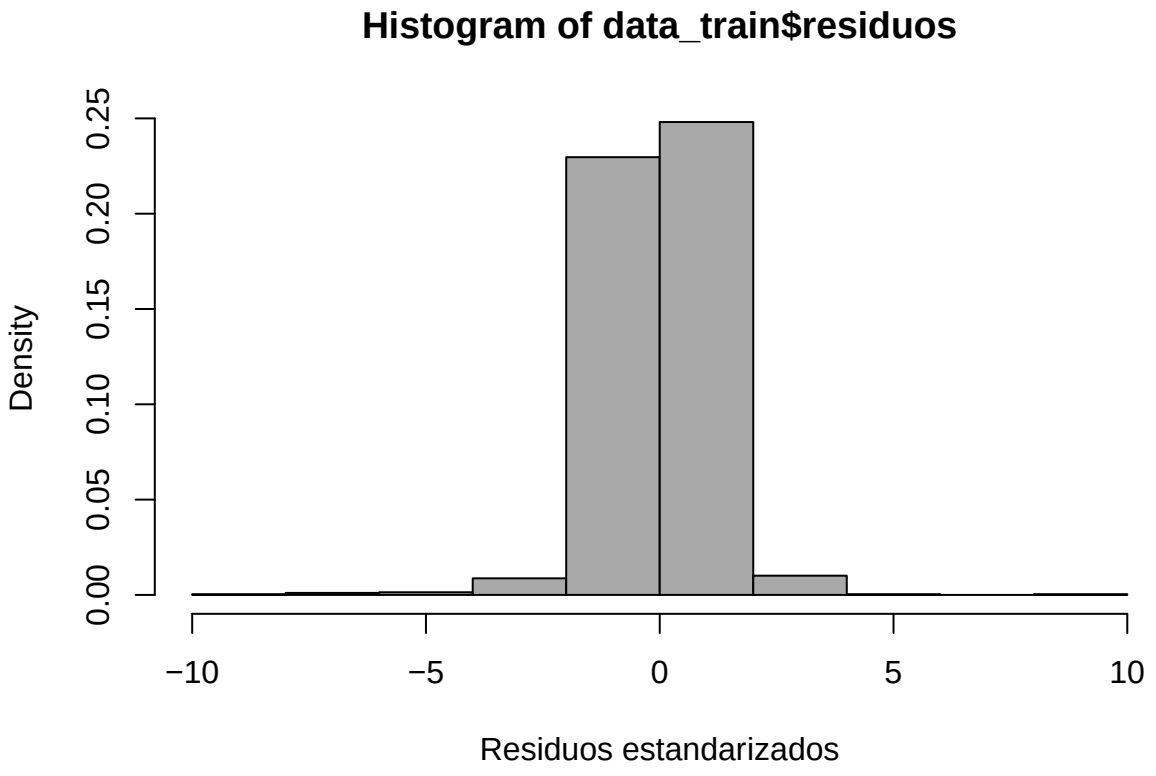


In the plot above we can see the standardized residuals against the fitted values of the model. We can see that the model is not quite perfectly linear. This indicates that the linearity hypothesis between the predictors and the sale price is probably not linear for all of them. This should not come as a surprise when looking at the correlation plot shown at the start of this report, in which some numeric variables are blatantly uncorrelated to the sale price of the house. To attain the linearity condition one option would be to try some transformations on the predictors. Another option would be to give up linearity altogether and attempt to fit a General Additive Model on the data instead.

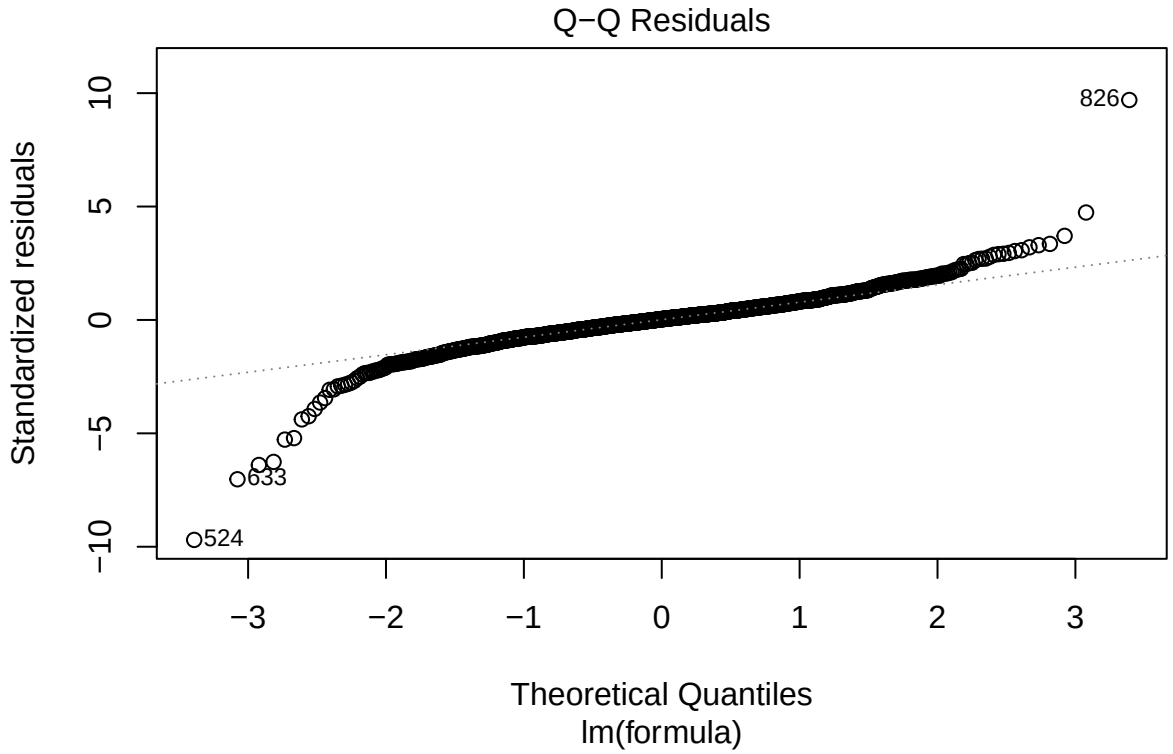
Normality

We'll assess the normality of residuals by generating a QQ-plot and conducting the Shapiro-Wilk test to confirm whether the residuals follow a normal distribution.

```
##
##  Shapiro-Wilk normality test
##
## data:  best.model$residuals
## W = 0.91754, p-value < 2.2e-16
```



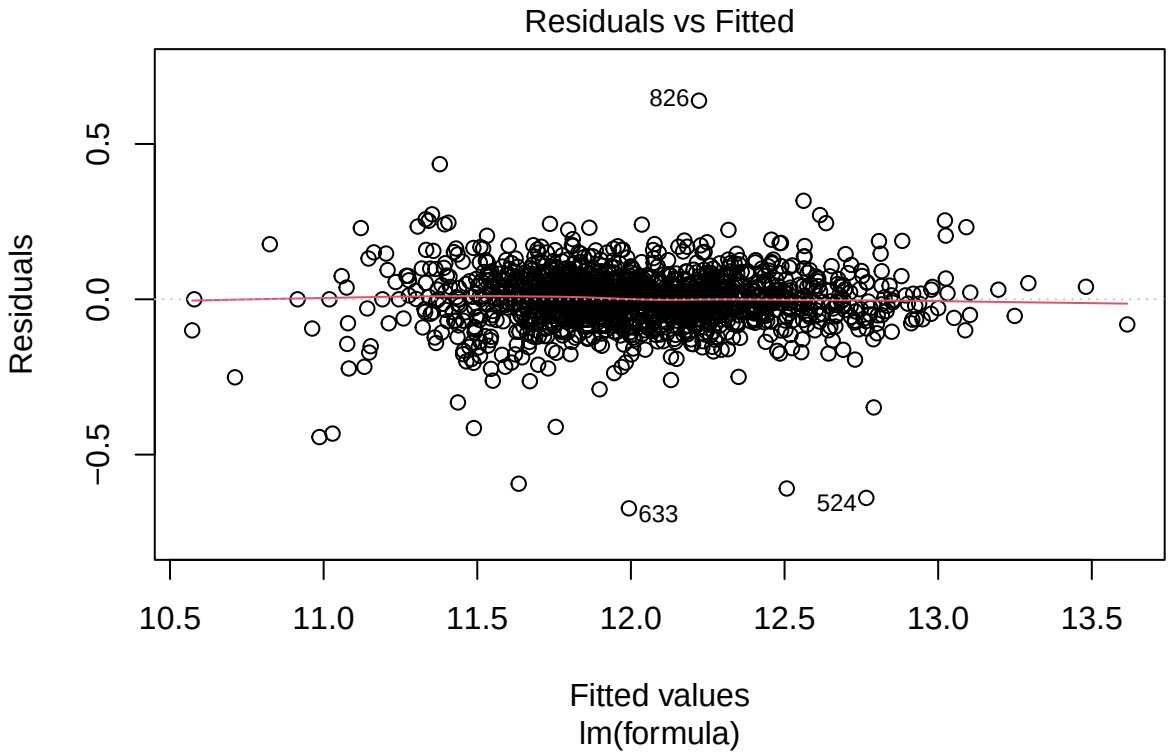
We also plot:



Upon initial observation, the QQ-plot doesn't deviate significantly from a normal distribution, particularly in the intermediate quantiles. However, the highly significant results of the Shapiro-Wilk test unmistakably indicate that the model's residuals do not conform to a normal distribution.

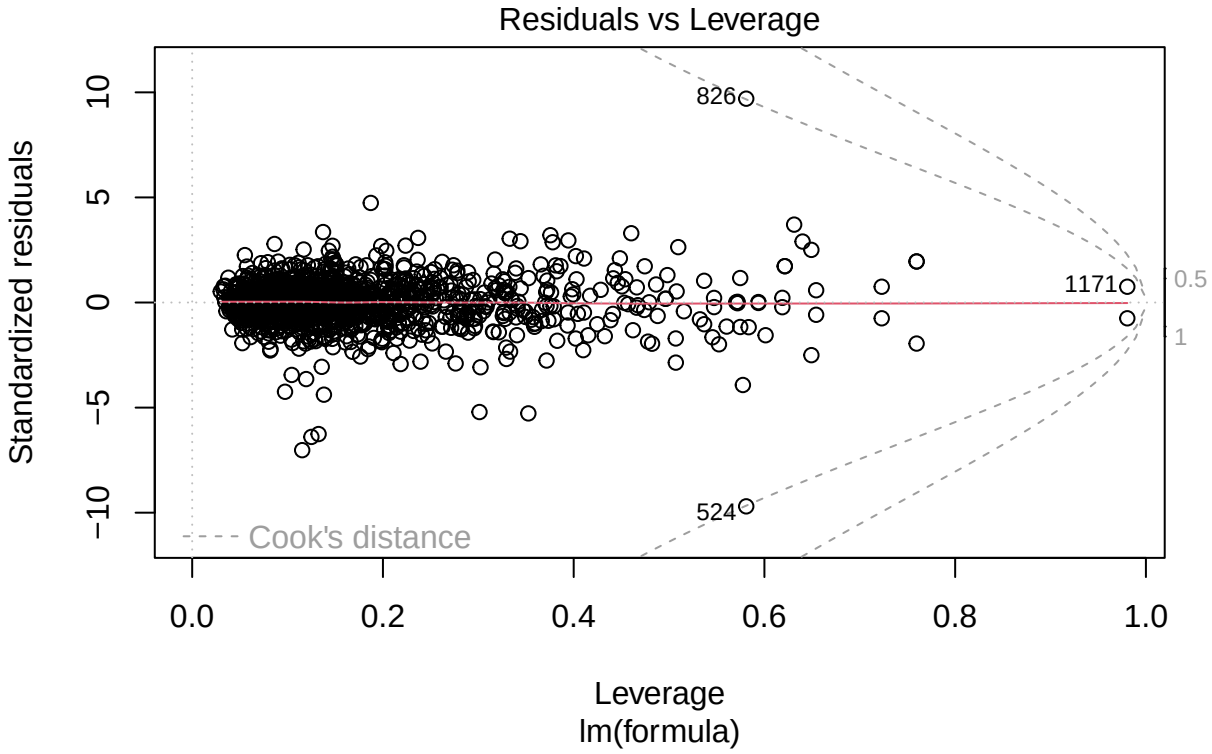
Heterocedasticity.

The variance of the errors has to be constant. Let's check this by plotting the fitted values versus residuals.



When observing the residuals plotted against the fitted values, a notable concentration around the zero line indicates their mean is close to zero. They display a consistent width band, indicating nearly constant variance. Despite this, it's evident there are outliers, notably observations 524, 633, and 826.

Influential observations



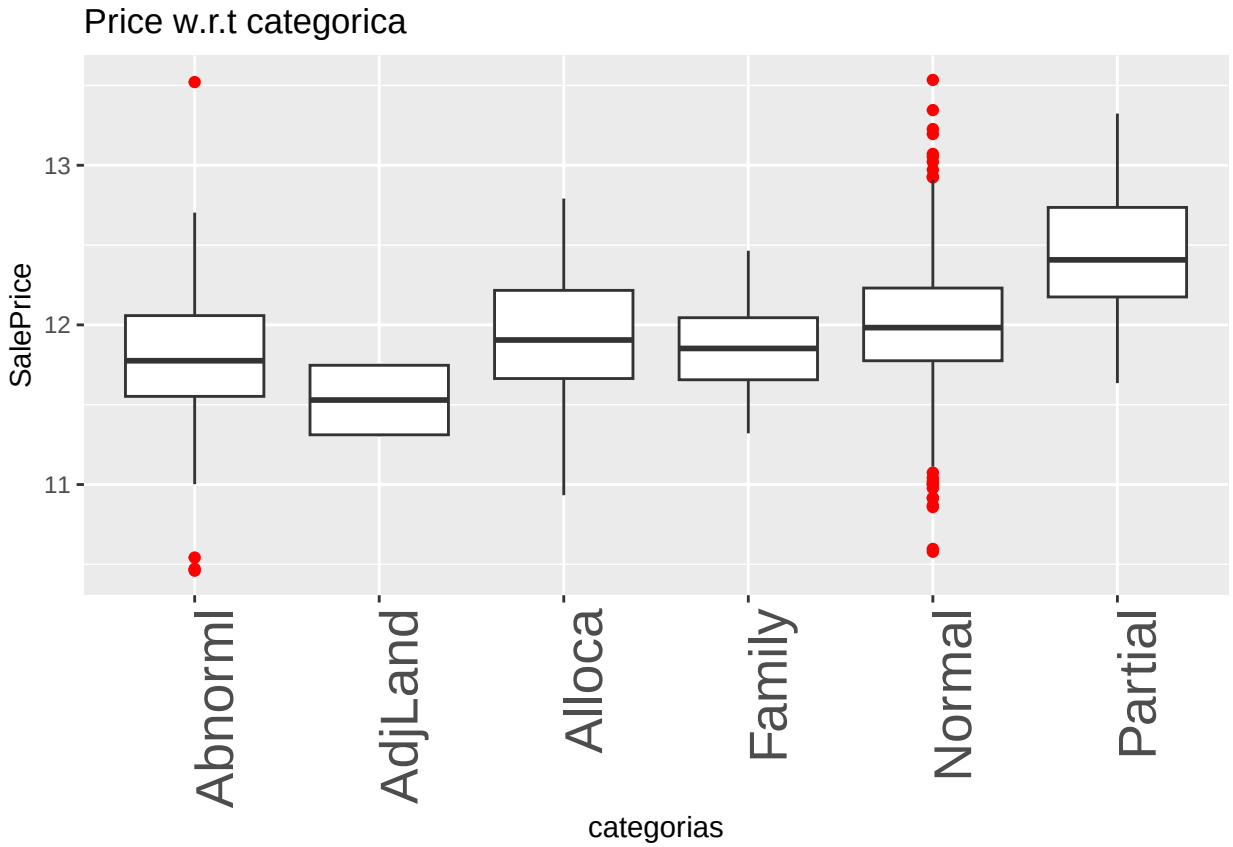
The leverage plot reveals numerous observations with high influence, with 1171 being particularly notable. Additionally, observations 524 and 826, identified as residual outliers, exhibit a leverage around 0.6, indicating moderate influence within the model.

Other questions of interest

With such a rich dataset, abundant in both observations and variables, there are some intriguing aspects worth discussing.

For example, by writing the following code and typing in x each variable of interest, we can extract meaningful information.

```
ggplot(data_train, aes(y = SalePrice, x = SaleCondition)) +scale_x_discrete("categorias") +
  scale_y_continuous("SalePrice") + geom_boxplot(outlier.color="red") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 10))+
  theme(axis.text.x = element_text(size=20))+ggtitle("Price w.r.t categorica")
```



More specifically, we observed some interesting trends: Houses with paved road access tend to be more expensive than those with gravel access. Among the priciest properties are those located in neighborhoods like Northridge, Northridge Heights, and Stone Brook. Properties situated near positive off-site features such as parks or greenbelts also command higher prices. Single-family detached buildings tend to have higher prices compared to other types of structures, especially those that are finished as opposed to unfinished houses. Naturally, homes with superior quality and conditions, including the house itself, basement, exterior materials, and garage, tend to be more expensive. Properties with wood shingle roofs are also on the pricier side. On the flip side, homes lacking features like a garage, fireplaces, basement, central air conditioning, or a pool are generally cheaper. Additionally, larger square footage often correlates with higher prices. The number of bathrooms, rooms, and kitchens in a house also tends to influence its price.

Upon examining the model coefficients, we aimed to identify the variable contributing most significantly to the variability in price (indicated by the largest coefficient). As non-binary categorical variables are associated with several coefficients, we opted to use quantiles in evaluating the model's coefficients instead of simply searching for the maximum value.

```
which(best.model$coefficients > quantile(best.model$coefficients, c(.97)))
```

```
##     (Intercept) RoofMatlCompShg RoofMatlMembran   RoofMatlMetal    RoofMatlRoll
##               1             120             121             122             123
## RoofMatlTar&Grv RoofMatlWdShake RoofMatlWdShngl
##             124             125             126
```

After analyzing coefficients above the 97th quantile, it seems that the variable 'RoofMatl,' signifying the material used in the house's roof, contributes most significantly to the sale price variability.