House Price Prediction Using Advanced Regression Techniques

Instructor: Ching – Chi Yang

Table of Contents:

- **<u>Introduction:</u>**

Dataset Information:
The dataset used in this study is sourced from the "House Prices: Advanced Regression Techniques" competition hosted on Kaggle.
You can access the dataset through the following link:
https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Research Objective:
The primary objective of this research is to develop predictive models capable of accurately estimating the sale prices of residential properties.

Housing data:

1	GarageTyp	GarageYrB	GarageFini	GarageCar	GarageAre	GarageQu	GarageCo	PavedDriv	WoodDecl	OpenPorcl	EnclosedP	3SsnPorch	ScreenPor	PoolArea	PoolQC	Fence	MiscFeatu	MiscVal	MoSold	YrSold	SaleType	SaleCondit	SalePrice
2	Attchd	2003	RFn	2	548	TA	TA	Y	0	61	0	0	0	0	NA	NA	NA	0	2	2008	WD	Normal	208500
3	Attchd	1976	RFn	2	460	TA	TA	Y	298	0	0	0	0	0	NA	NA	NA	0	5	2007	WD	Normal	181500
4	Attchd	2001	RFn	2	608	TA	TA	Y	0	42	0	0	0	0	NA	NA	NA	0	9	2008	WD	Normal	223500
5	Detchd	1998	Unf	3	642	TA	TA	Y	0	35	272	0	0	0	NA	NA	NA	0	2	2006	WD	Abnorml	140000
6	Attchd	2000	RFn	3	836	TA	TA	Y	192	84	0	0	0	0	NA	NA	NA	0	12	2008	WD	Normal	250000
7	Attchd	1993	Unf	2	480	TA	TA	Y	40	30	0	320	0	0	NA	MnPrv	Shed	700	10	2009	WD	Normal	143000
8	Attchd	2004	RFn	2	636	TA	TA	Y	255	57	0	0	0	0	NA	NA	NA	0	8	2007	WD	Normal	307000
9	Attchd	1973	RFn	2	484	TA	TA	Y	235	204	228	0	0	0	NA	NA	Shed	350	11	2009	WD	Normal	200000
10	Detchd	1931	Unf	2	468	Fa	TA	Y	90	0	205	0	0	0	NA	NA	NA	0	4	2008	WD	Abnorml	129900
11	Attchd	1939	RFn	1	205	Gd	TA	Y	0	4	0	0	0	0	NA	NA	NA	0	1	2008	WD	Normal	118000
12	Detchd	1965	Unf	1	384	TA	TA	Y	0	0	0	0	0	0	NA	NA	NA	0	2	2008	WD	Normal	129500
13	BuiltIn	2005	Fin	3	736	TA	TA	Y	147	21	0	0	0	0	NA	NA	NA	0	7	2006	New	Partial	345000
14	Detchd	1962	Unf	1	352	TA	TA	Y	140	0	0	0	176	0	NA	NA	NA	0	9	2008	WD	Normal	144000
15	Attchd	2006	RFn	3	840	TA	TA	Y	160	33	0	0	0	0	NA	NA	NA	0	8	2007	New	Partial	279500
16	Attchd	1960	RFn	1	352	TA	TA	Y	0	213	176	0	0	0	NA	GdWo	NA	0	5	2008	WD	Normal	157000
17	Detchd	1991	Unf	2	576	TA	TA	Y	48	112	0	0	0	0	NA	GdPrv	NA	0	7	2007	WD	Normal	132000
18	Attchd	1970	Fin	2	480	TA	TA	Y	0	0	0	0	0	0	NA	NA	Shed	700	3	2010	WD	Normal	149000

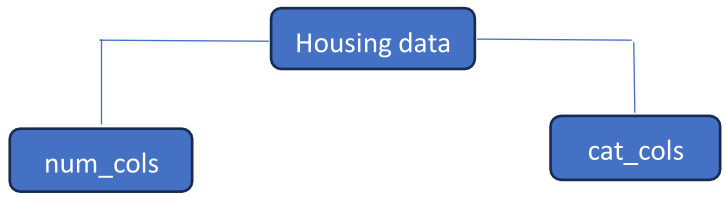
```
> dim(housing)
[1] 1460   80
```

This is how our dataset looks like. It's a combination of both numerical and categorical variables.

<u>STEP 1</u>: Divide and analyze

So if a dataset is either complete numerical or complete categorical, its not that difficult to analyze,
preprocess and implement the models. But in this case we have a mix of both which makes it difficult to run the explorative data analysis. Just by looking at this data we cannot further investigate the in-depth nature of each variable. So we came with a plan - "divide and analyse". So this way we can better

understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.



```
> length(cat_cols)
[1] 43
> names(cat_cols)
 [1] "MSZoning"      "Street"        "Alley"         "LotShape"      "LandContour"   "Utilities"
 [7] "LotConfig"     "LandSlope"     "Neighborhood"  "Condition1"    "Condition2"    "BldgType"
[13] "HouseStyle"    "RoofStyle"     "RoofMatl"      "Exterior1st"   "Exterior2nd"   "MasVnrType"
[19] "ExterQual"     "ExterCond"     "Foundation"    "BsmtQual"      "BsmtCond"      "BsmtExposure"
[25] "BsmtFinType1"  "BsmtFinType2"  "Heating"       "HeatingQC"     "CentralAir"    "Electrical"
[31] "KitchenQual"   "Functional"    "FireplaceQu"   "GarageType"    "GarageFinish"  "GarageQual"
[37] "GarageCond"    "PavedDrive"    "PoolQC"        "Fence"         "MiscFeature"   "SaleType"
[43] "SaleCondition"

> length(num_cols)
[1] 37
> names(num_cols)
 [1] "MSSubClass"    "LotFrontage"   "LotArea"       "OverallQual"   "OverallCond"   "YearBuilt"
 [7] "YearRemodAdd"  "MasVnrArea"    "BsmtFinSF1"    "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"
[13] "X1stFlrSF"     "X2ndFlrSF"     "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"  "BsmtHalfBath"
[19] "FullBath"      "HalfBath"      "BedroomAbvGr"  "KitchenAbvGr"  "TotRmsAbvGrd"  "Fireplaces"
[25] "GarageYrBlt"   "GarageCars"    "GarageArea"    "WoodDeckSF"    "OpenPorchSF"   "EnclosedPorch"
[31] "X3SsnPorch"    "ScreenPorch"   "PoolArea"      "MiscVal"       "MoSold"        "YrSold"
[37] "SalePrice"
```

STEP 2: **Data preprocessing**: clean, transform, and prepare the data for analysis.

we took good care of data preprocessing because it impacts the quality of insights and modelsdeveloped from the data.

```
> missing_values
   MSSubClass   LotFrontage       LotArea   OverallQual   OverallCond     YearBuilt  YearRemodAdd    MasVnrArea
            0           259             0             0             0             0             0             8
   BsmtFinSF1    BsmtFinSF2     BsmtUnfSF   TotalBsmtSF     X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea
            0             0             0             0             0             0             0             0
 BsmtFullBath  BsmtHalfBath      FullBath      HalfBath  BedroomAbvGr  KitchenAbvGr  TotRmsAbvGrd    Fireplaces
            0             0             0             0             0             0             0             0
  GarageYrBlt    GarageCars    GarageArea    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch   ScreenPorch
           81             0             0             0             0             0             0             0
     PoolArea       MiscVal        MoSold        YrSold     SalePrice
            0             0             0             0             0
> #check for any missing values:
> sum(is.na(num_cols))
[1] 348
```

we found that these columns has missing values: MasVnrArea, LotFrontage,

GarageYrBltTreating MasVnrArea, LotFrontage with Mice method is a

reasonable approach.

But, is it appropriate to use mice for treating missing values for GarageYrBlt?

Using Multiple Imputation by Chained Equations (MICE) for imputing missing values in the "GarageYrBlt"variable may not be the most appropriate method, primarily due to the nature of the variable.

"GarageYrBlt" represents the year a garage was built. This variable is typically a discrete numeric variable, as it represents specific years. MICE is commonly used for imputing missing values in continuous or categorical variables. Imputing missing years with MICE may lead to imputed values thatdon't make sense in the context of year-based data.

Instead, for "GarageYrBlt," it's more appropriate to use regression imputations.

#Regression Imputation:

In this approach, we would build a regression model where "GarageYrBlt" is the dependent variable, andother relevant variables (e.g., "YearBuilt," "OverallQual," etc.) are used as independent predictors.

The model is trained using rows where "GarageYrBlt" is not missing.

Once the model is trained, we can use it to predict the missing values of "GarageYrBlt" based on thevalues of the predictor variables in rows where "GarageYrBlt" is missing.

Takes into account the relationships between the variables and can provide more accurate imputations ifthere's a strong relationship between "GarageYrBlt" and the predictors.

#-->how do we know if there's a strong relationship between "GarageYrBlt" and

the predictors.#p-values: Examine the p-values associated with each predictor in

the model summary
(summary(lm_model)). Lower p-values suggest that the predictor variable is statistically significant inexplaining the variation in "GarageYrBlt."

After treating NA’s:

```
> colSums(is.na(num_cols))
   MSSubClass  LotFrontage      LotArea  OverallQual  OverallCond    YearBuilt YearRemodAdd   MasVnrArea
            0            0            0            0            0            0            0            0
   BsmtFinSF1   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF    X1stFlrSF    X2ndFlrSF LowQualFinSF    GrLivArea
            0            0            0            0            0            0            0            0
 BsmtFullBath BsmtHalfBath     FullBath     HalfBath BedroomAbvGr KitchenAbvGr TotRmsAbvGrd   Fireplaces
            0            0            0            0            0            0            0            0
  GarageYrBlt   GarageCars   GarageArea   WoodDeckSF  OpenPorchSF EnclosedPorch   X3SsnPorch  ScreenPorch
            0            0            0            0            0            0            0            0
     PoolArea      MiscVal       MoSold       YrSold    SalePrice
            0            0            0            0            0
> sum(is.na(num_cols))
[1] 0
```

Same way we treated the missing data in cat_cols using **Mode Imputation**.

```
> colSums(is.na(cat_cols))
     MSZoning        Street         Alley     LotShape   LandContour    Utilities    LotConfig    LandSlope
            0             0          1369            0             0            0            0            0
 Neighborhood    Condition1    Condition2     BldgType    HouseStyle    RoofStyle     RoofMatl  Exterior1st
            0             0             0            0             0            0            0            0
  Exterior2nd    MasVnrType     ExterQual    ExterCond    Foundation     BsmtQual     BsmtCond BsmtExposure
            0             8             0            0             0           37           37           38
 BsmtFinType1  BsmtFinType2       Heating    HeatingQC    CentralAir   Electrical  KitchenQual   Functional
           37            38             0            0             0            1            0            0
  FireplaceQu    GarageType  GarageFinish   GarageQual    GarageCond   PavedDrive       PoolQC        Fence
          690            81            81           81            81            0         1453         1179
  MiscFeature      SaleType SaleCondition
         1406             0             0
> sum(is.na(cat_cols))
[1] 6617
```

In our dataset, comprising 1460 rows, we identified several categorical variables with a substantial proportion of missing values. Specifically, 'Alley' has 1369 missing values, 'PoolQC' has 1453, 'Fence' has1179, 'MiscFeature' has 1406, and 'FireplaceQu' has 690. Given that these missing values account for more than 90% of the data for these variables, their inclusion in the predictive model could introduce significant bias and reduce the model's accuracy.

Therefore, to maintain the integrity and predictive power of our model, we decided to exclude these variables from our analysis. While features like 'PoolQC' (Pool Quality) and 'FireplaceQu' (Fireplace
Quality) might have importance in certain contexts for house price prediction, the overwhelming lack of data in our specific dataset renders them unreliable for our modeling purposes. Thus, our decision to
remove these variables is driven by a commitment to model accuracy and

data quality. cat_cols <- subset(cat_cols, select = -c(Alley, PoolQC, Fence,

MiscFeature,FireplaceQu)) Later on we treated the other NA’s with the

Mode imptation.

#Why we used mode imputation here?
Mode Imputation: Replace missing values with the mode (most frequent category) of the respective variable. This is a simple and common method for handling missing categorical data.
#No Assumption of Relationships: Here in our dataset we does not capture any relationships between
the missing categorical variables and other predictors. These variables are missing completely at random and their missingness is not related to the values of other variables, mode imputation can be a
reasonable choice.

After handling NA’s:

```
> colSums(is.na(cat_cols))
     MSZoning        Street      LotShape   LandContour     Utilities     LotConfig     LandSlope  Neighborhood
            0             0             0             0             0             0             0             0
   Condition1    Condition2      BldgType    HouseStyle     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
            0             0             0             0             0             0             0             0
   MasVnrType     ExterQual     ExterCond    Foundation      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1
            0             0             0             0             0             0             0             0
 BsmtFinType2       Heating     HeatingQC    CentralAir    Electrical   KitchenQual    Functional    GarageType
            0             0             0             0             0             0             0             0
 GarageFinish    GarageQual    GarageCond    PavedDrive      SaleType SaleCondition
            0             0             0             0             0             0
> sum(is.na(cat_cols))
[1] 0
```

<u>Encoding:</u>

We need to convert these cleaned categorical columns into numerical format to ensure consistency across the entire dataset.

As we all know that this conversion is essential because when implementing machine learning models, they require all variables to be in numerical form.

So we choose to use label encoding here.

Why Label encoding?

Label encoding is chosen for the following reasons:

Simplicity and Reduced Dimensionality.

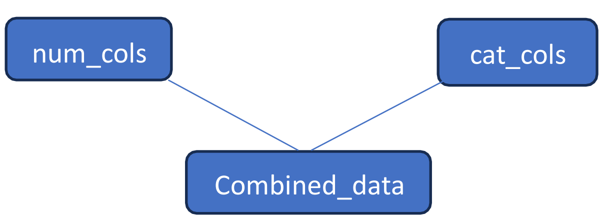
So previously we applied one hot encoding on the data. While one-hot encoding is a powerful technique for handling categorical variables, we opted not to use it in this analysis due to several considerations.
One-hot encoding creates binary variables for each category, leading to a significant increase in
dimensionality. Additionally, one-hot encoding can lead to multicollinearity issues, as the presence of one binary variable implies the absence of others. To maintain dataset efficiency, reduce dimensionality,

and retain ordinal information where applicable, we chose label encoding as a more suitable alternative for our modeling purposes.

After encoding: we can see the structure of the data converted into numerical format.

```
> str(cat_cols)
'data.frame':	1460 obs. of  38 variables:
 $ MSZoning     : num  4 4 4 4 4 4 4 4 5 4 ...
 $ Street       : num  2 2 2 2 2 2 2 2 2 2 ...
 $ LotShape     : num  4 4 1 1 1 1 4 1 4 4 ...
 $ LandContour  : num  4 4 4 4 4 4 4 4 4 4 ...
 $ Utilities    : num  1 1 1 1 1 1 1 1 1 1 ...
 $ LotConfig    : num  5 3 5 1 3 5 5 1 5 1 ...
 $ LandSlope    : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Neighborhood : num  6 25 6 7 14 12 21 17 18 4
 $ Condition1   : num  3 2 3 3 3 3 3 5 1 1 ...
 $ Condition2   : num  3 3 3 3 3 3 3 3 3 1 ...
 $ BldgType     : num  1 1 1 1 1 1 1 1 1 2 ...
 $ HouseStyle   : num  6 3 6 6 6 1 3 6 1 2 ...
 $ RoofStyle    : num  2 2 2 2 2 2 2 2 2 2 ...
 $ RoofMatl     : num  2 2 2 2 2 2 2 2 2 2 ...
 $ Exterior1st  : num  13 9 13 14 13 13 13 7 4 9
 $ Exterior2nd  : num  14 9 14 16 14 14 14 7 16 9
 $ MasVnrType   : num  3 4 3 4 3 4 5 5 4 4 ...
 $ ExterQual    : num  3 4 3 4 3 4 3 4 4 4 ...
 $ ExterCond    : num  5 5 5 5 5 5 5 5 5 5 ...
 $ Foundation   : num  3 2 3 1 3 6 3 2 1 1 ...
 $ BsmtQual     : num  4 4 4 5 4 4 2 4 5 5 ...
 $ BsmtCond     : num  5 5 5 3 5 5 5 5 5 5 ...
 $ BsmtExposure : num  5 3 4 5 2 5 2 4 5 5 ...
```

The data is ready and now we are good to go!!! <u>STEP 3</u>: Combine and Implement.



combined_data <- data.frame(num_cols, cat_cols)

```
> dim(combined_data)
[1] 1460   75
> str(combined_data)
'data.frame':   1460 obs. of  75 variables:
 $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
 $ LotFrontage  : int  65 80 68 60 84 85 75 90 51 50 ...
 $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420
 $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
 $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
 $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
 $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
 $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
```

Model Building and Evaluation:

We split the dataset into training and testing sets to assess the performance of our models and theirability to generalize to new, unseen data.

Linear Regression:

We evaluated the model's performance using several metrics:

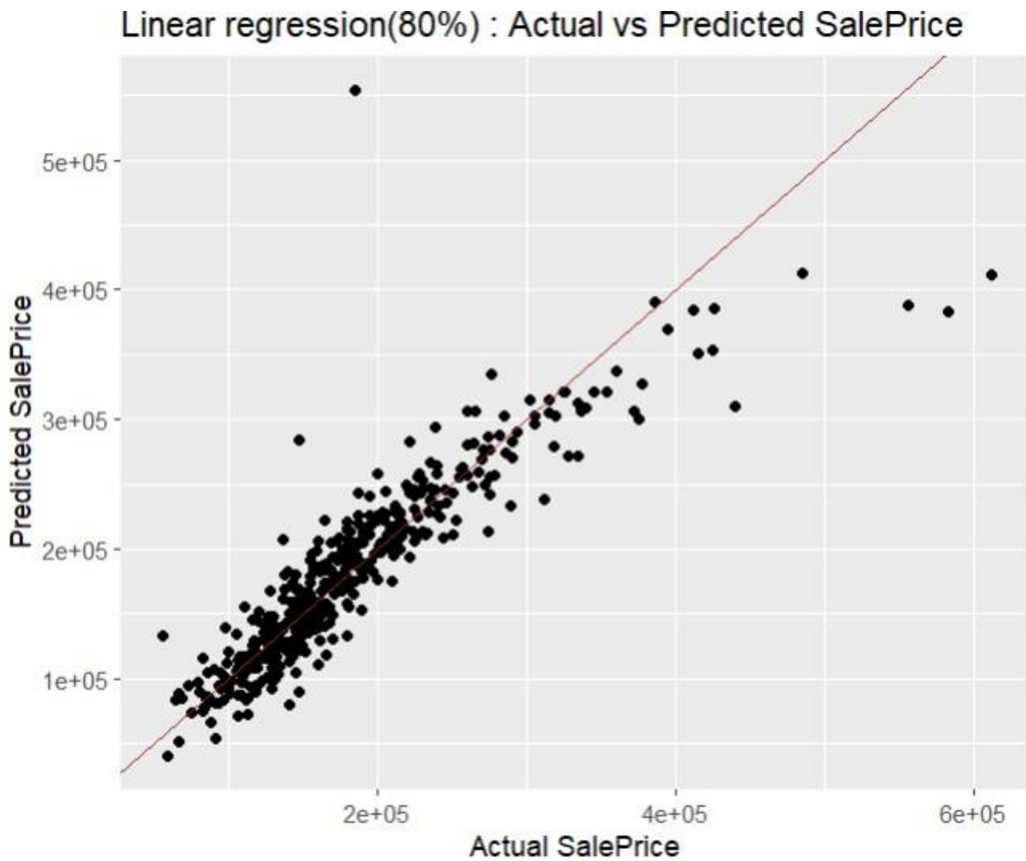
R-squared (R2): 80.31%.
This implies that the linear regression model captures a substantial portion of the variability in houseprices, which is a positive sign.

Root Mean Square Error (RMSE):

34,538.76 Mean Absolute Error

(MAE): 20,536.30

These metrics collectively suggest that our linear regression model performs reasonably well in explaining and predicting house prices based on the selected features. However, further model refinement and exploration of alternative algorithms may lead to potential improvements.



Subset selection and Model application:

We employed the backward selection method to identify the best subset of variables for our model. Thisapproach starts with a model that includes all available predictor variables and iteratively removes the
least significant ones based on a specified criterion,here we used AIC). The final model retains only thevariables that contribute significantly to explaining the target variable, SalePrice.

Selected Variables:
The backward selection process resulted in the following selected variables in the final model:

MSSubCl
ass
LotFronta
ge
LotArea
OverallQu
al
OverallCo
nd
YearBuilt
MasVnrAr
ea
BsmtFinS
F1
BsmtFinS
F2
X1stFlrSF
X2ndFlrS
F
LowQualFi
nSF
BsmtFullBa
th HalfBath
BedroomAb
vGr
KitchenAbv
Gr
Fireplaces
GarageCar
s
GarageAre
a
WoodDeck
SF
ScreenPorc
h YrSold
MSZoning
LotShape
RoofMatl
Exterior2n

d
MasVnrTy
pe
ExterQual
BsmtQu
al
BsmtCo
nd
BsmtExpos
ure
HeatingQC
KitchenQ
ual
Functiona
l
GarageQu
al
SaleCondition

Model Performance Evaluation:

```
Residual standard error: 30870 on 987 degrees of freedom
Multiple R-squared:  0.8568,	Adjusted R-squared:  0.8515
F-statistic:   164 on 36 and 987 DF,  p-value: < 2.2e-16
```

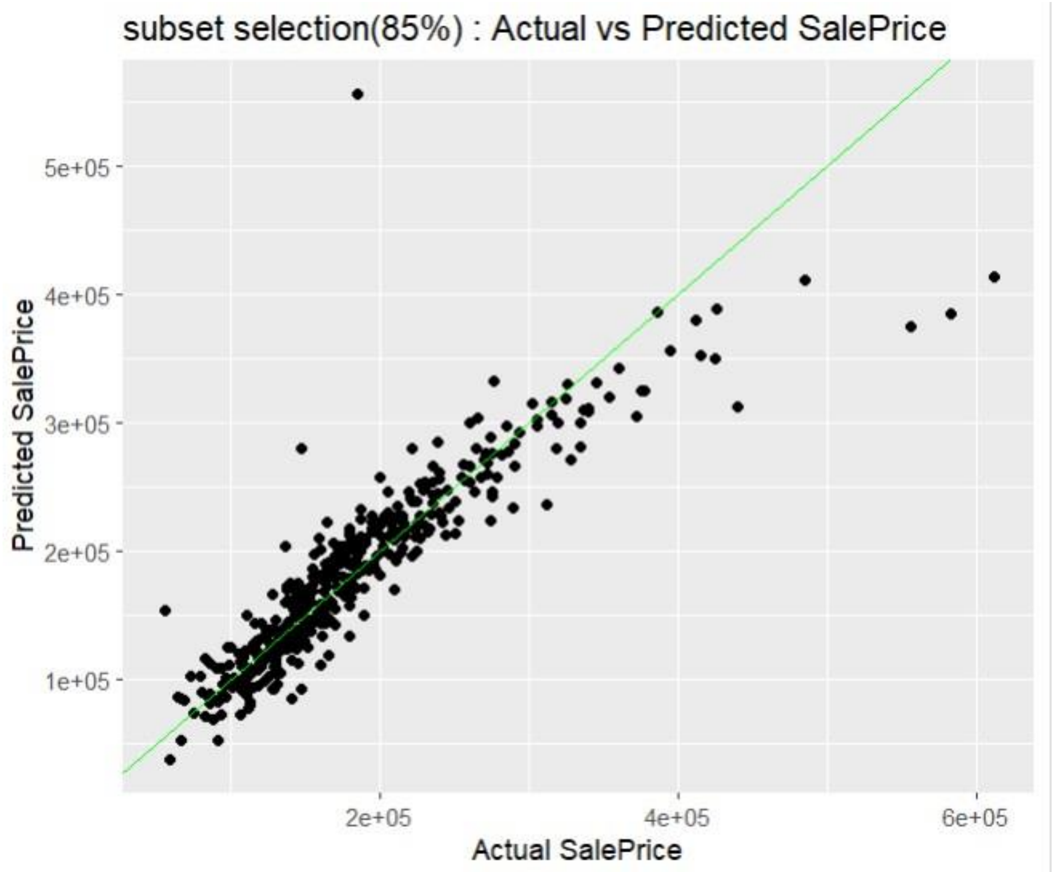
```
> #predictions
> sub_pred <- predict(backward_selection, newdata = test_data)
> # Calculate RMSE
> rmse <- sqrt(mean((sub_pred - test_data$SalePrice)^2))
> # Calculate MAE
> mae <- mean(abs(sub_pred - test_data$SalePrice))
> # Print RMSE and MAE
> print(paste("SUB_RMSE:", rmse))
[1] "SUB_RMSE: 35051.865133641"
> print(paste("SUB_MAE:", mae))
[1] "SUB_MAE: 20438.3975529476"
```

```
Step:  AIC=21207.55
SalePrice ~ MSSubClass + LotFrontage + LotArea + OverallQual + 
    OverallCond + YearBuilt + MasVnrArea + BsmtFinSF1 + BsmtFinSF2 + 
    X1stFlrSF + X2ndFlrSF + LowQualFinSF + BsmtFullBath + HalfBath + 
    BedroomAbvGr + KitchenAbvGr + Fireplaces + GarageCars + GarageArea + 
    WoodDeckSF + ScreenPorch + YrSold + MSZoning + LotShape + 
    RoofMatl + Exterior2nd + MasVnrType + ExterQual + BsmtQual + 
    BsmtCond + BsmtExposure + HeatingQC + KitchenQual + Functional + 
    GarageQual + SaleCondition
```

```
> coefficients_used
  (Intercept)    MSSubClass   LotFrontage       LotArea   OverallQual   OverallCond     YearBuilt    MasVnrArea 
 3.025579e+06 -2.385072e+02 -2.219915e+02  2.663435e-01  1.271604e+04  4.734864e+03  2.097502e+02  3.486439e+01 
   BsmtFinSF1    BsmtFinSF2     X1stFlrSF     X2ndFlrSF  LowQualFinSF  BsmtFullBath      HalfBath  BedroomAbvGr 
 1.009715e+01  1.127004e+01  6.242120e+01  6.256408e+01  3.825973e+01  7.627268e+03 -3.855467e+03 -3.995442e+03 
 KitchenAbvGr    Fireplaces    GarageCars    GarageArea    WoodDeckSF   ScreenPorch        YrSold      MSZoning 
-8.309661e+03  3.567539e+03  1.847958e+04 -2.084111e+01  2.516105e+01  3.375210e+01 -1.691754e+03 -3.327139e+03 
     LotShape      RoofMatl   Exterior2nd    MasVnrType     ExterQual      BsmtQual      BsmtCond  BsmtExposure 
-1.253172e+03  4.264235e+03 -7.420937e+02  4.154786e+03 -1.031649e+04 -5.085643e+03  5.269519e+03 -2.906275e+03 
    HeatingQC   KitchenQual    Functional    GarageQual SaleCondition 
-1.173852e+03 -8.363439e+03  3.875210e+03 -2.392582e+03  2.540978e+03 
```

This is the final model performed by backward selection with an AIC of 21207.55

> length(coefficients_used)[1]
37



Overall, the model appears to perform well in explaining and predicting house prices based on the selected subset of variables.
After applying backward selection and evaluating the subset, we've found it performs exceptionally well in predicting SalePrice. We've decided to use these selected features in subsequent modeling, enhancing simplicity and predictive power. This approach strikes a balance between model complexity and accuracy, promoting efficient house price prediction.

Ridge Regression:

Ridge Regression was chosen for this analysis due to its ability to handle multicollinearity in the dataset and prevent overfitting by adding a penalty term to the linear regression model. This helps in stabilizing the model and improving its generalization.

Tuning Process for Alpha:

To find the optimal value of the regularization parameter alpha (λ) in Ridge Regression, a grid search was conducted over a range of alpha values. Cross-validation was used to get

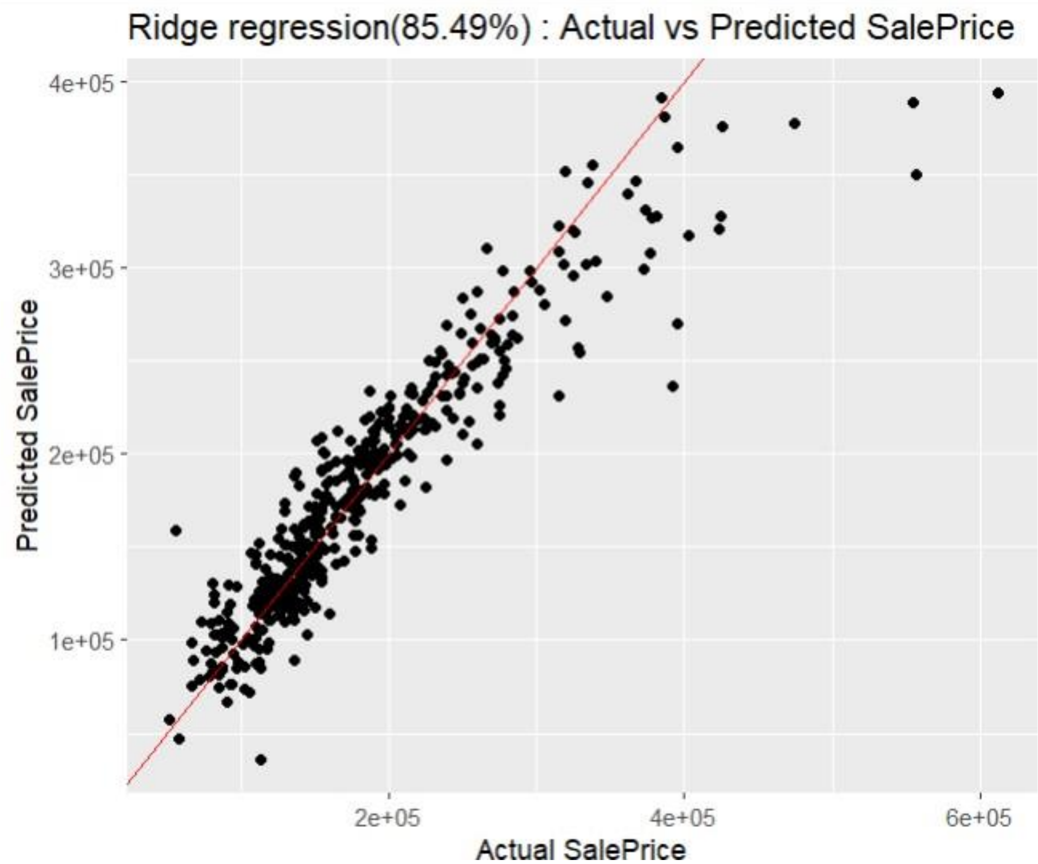
```
> optimal_alpha <- cv$lambda.min
> # Print the optimal alpha
> cat("Optimal Alpha:", optimal_alpha, "\n")
Optimal Alpha: 15199.11
```

the best alpha. The optimal alpha value was determined as 15199.11, which yielded the best cross-validated results.

Model Results and Evaluation Metrics:

After tuning with the optimal alpha, the Ridge Regression model was fitted to the full training dataset, and predictions were made on the test dataset. Here are the evaluation metrics for the Ridge Regression model:

```
> ridge_mae
[1] 958632525
> ridge_rmse
[1] 30961.79
> ridge_R2
[1] 0.8549767
```



Lasso:

We did the same process for Lasso as well.

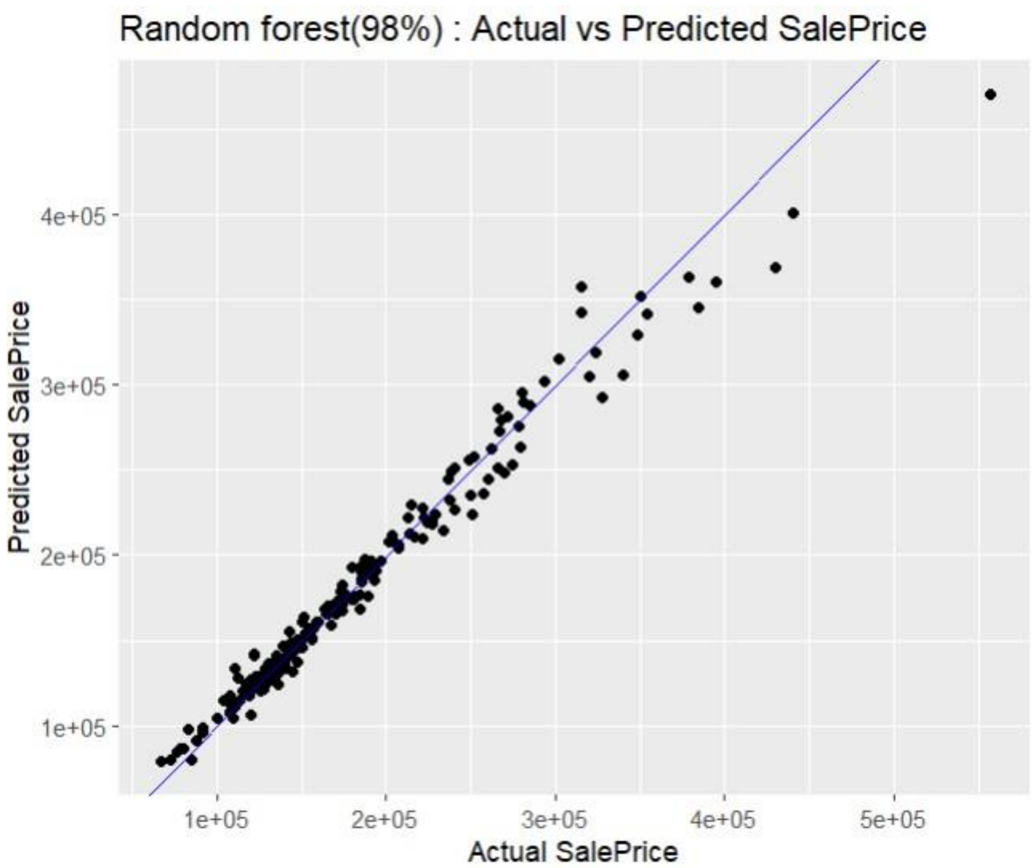
Although both ridge and lasso performed equally, we see a slightest increase in the performance of Lasso

```
> cat("Mean Squared Error (MSE) for Lasso:", mse_lasso, "\n")
Mean Squared Error (MSE) for Lasso: 909094790
> cat("Root Mean Squared Error (RMSE) for Lasso:", rmse_lasso, "\n")
Root Mean Squared Error (RMSE) for Lasso: 30151.2
> cat("R-squared (R^2) for Lasso:", r_squared_lasso, "\n")
R-squared (R^2) for Lasso: 0.8624708
```

Random Forest:

Random Forest is an ensemble learning technique that combines multiple decision trees to improve predictive accuracy and reduce overfitting.

```
> # calculate R-squared
> rsq <- cor(test_data$SalePrice, predictions_rf)^2
> print(paste0("R-squared: ", round(rsq, 2)))
[1] "R-squared: 0.98"
> # Calculate RMSE
> rmse_rf <- RMSE(predictions_rf, test_data$SalePrice)
> print(paste0("RMSE: ", rmse_rf))
[1] "RMSE: 12282.4341346214"
> # Calculate MAE
> mae_rf <- MAE(predictions_rf, test_data$SalePrice)
> print(paste0("MAE: ", mae))
[1] "MAE: 3811.94979680493"
```

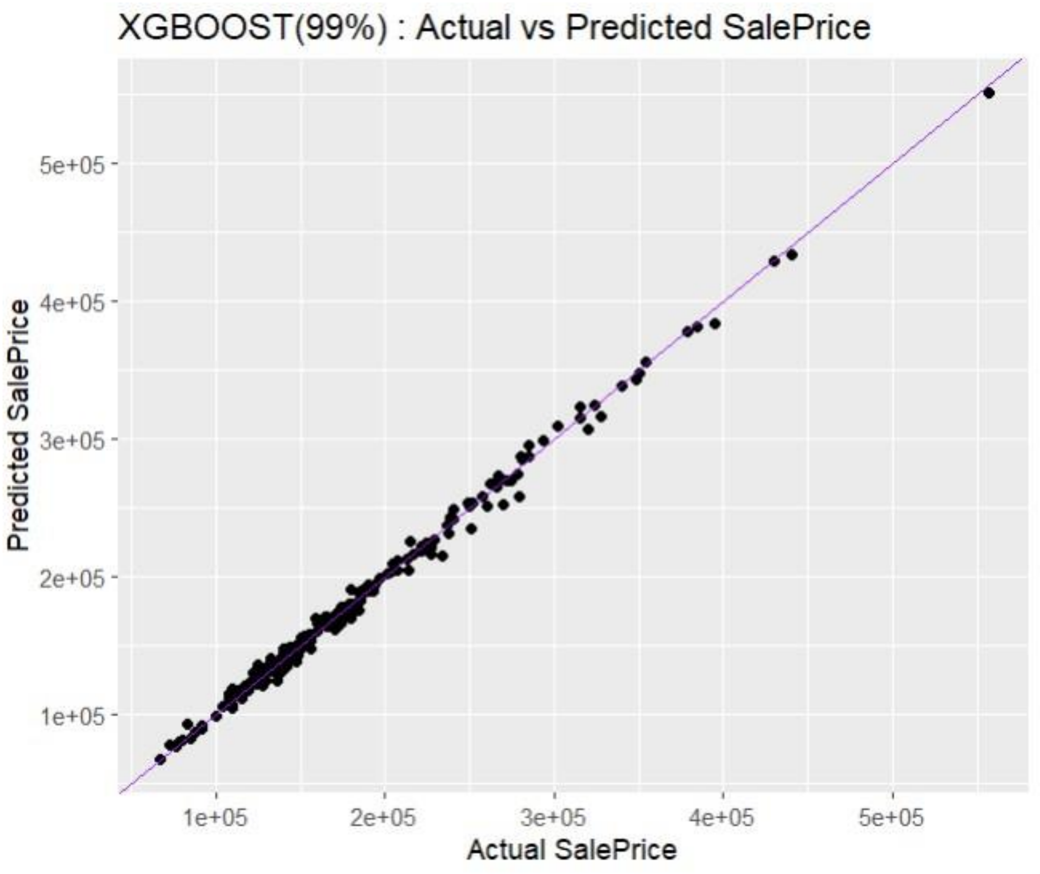


XGBOOST:
In our pursuit of achieving the highest prediction accuracy for our model, we initially implemented a Random Forest model, which performed admirably with an accuracy of 98%. At this point, we contemplated concluding our modeling process, as this level of performance is considered excellent in many statistical learning applications.

However, our commitment to delivering the best results led us to explore more advanced and robust modeling techniques. After careful consideration, we employed the XGBoost algorithm, renowned for its efficiency and effectiveness in regression tasks. We began by defining a tuning grid to explore a range of hyperparameters, including the number of boosting rounds (nrounds), maximum depth of the trees (max_depth), learning rate (eta), and others. These parameters play a crucial role in the model's ability to learn from the data.
We defined a specific range of values for each hyperparameter in our tuning grid, as follows:
Number of Boosting Rounds (nrounds): We experimented with 100, 200, and 300 rounds. This parameter determines the number of times the boosting process is repeated, and higher values can lead to a more complex model.
Maximum Tree Depth (max_depth): The values tested were 3, 4, and 5. This parameter controls the depth of each tree, with deeper trees capturing more complex patterns but also increasing the risk of overfitting.
Learning Rate (eta): We used rates of 0.01, 0.1, and 0.2. The learning rate shrinks the contribution of each tree and can be used to prevent overfitting.
We then applied 5-fold cross-validation, an essential step to ensure that our model's performance is robust and generalizes well to unseen data. This method divides the data into five subsets, using each in turn for validation while training on the remaining four. The caret package streamlined our hyperparameter tuning process, allowing us to efficiently find the optimal parameter combination within our defined grid.
After identifying the best hyperparameters from this tuning process, we trained the final XGBoost model, ensuring it was finely adjusted to our dataset. The final step involved making predictions on the test set and rigorously evaluating the model's performance using the corresponding metrics These metrics provided a comprehensive view of the model's accuracy and error rates, with a lower MSE and RMSE indicating better performance and a higher R-squared signifying a model that closely fits our data. This thorough approach, combining careful parameter tuning and rigorous evaluation, underlines the robustness and reliability of our predictive model, demonstrating its capability in accurately forecasting house prices.
This achievement represented a substantial improvement over the already impressive performance of our Random Forest model.

By embracing XGBoost, we harnessed the full potential to achieve the highest accuracy possible.

```
> cat("Root Mean Squared Error (RMSE):", rmse, "\n")
Root Mean Squared Error (RMSE): 5225.624
> cat("Mean Absolute Error (MAE):", mae, "\n")
Mean Absolute Error (MAE): 3811.95
> cat("R-squared (R^2):", r_squared, "\n")
R-squared (R^2): 0.9950097
```



We randomly selected some indices to compare the actual sale prices with the predicted sale prices generated by our XGBoost model.
This comparison revealed the remarkable accuracy achieved by the XGBoost model in

```
> test_data$SalePrice[c(76, 101, 145, 1, 92)]
[1]  91300 167000 174000 129900 224500
> XG_pred[c(76, 101, 145, 1, 92)]
[1]  91303.17 166973.72 173971.22 129857.92 224457.81
```

predicting housing prices.

The models predictive power and its ability to capture complex patterns in the data is outstanding.

Model Comparision:

Model	R 2	RMSE	MA E
Linear Regression	80.31%	34,538.76	20,536.30
Subset selection	85.68%	35051.86	20438.39
Ridge regression	85.49%	30961.79	20126.54
Lasso	86.24%	30151.2	19723.32
Random forest	98%	12282.43	3811.949
XGBoost	99.50%	5225.624	3811.95

Conclusion:

Our housing price prediction project showcased the power of advanced statistical learning techniques. From Linear Regression to Ridge and Lasso regression, we witnessed significant accuracy improvements. However, it was Random Forest and XGBoost that truly shone, achieving 98% and 99.50% accuracy,
respectively. These ensemble methods, combined with careful feature engineering and cross-validation, offer precise real estate predictions. In summary, our study demonstrates the transformative potential of statistical learning in real estate, providing valuable insights for industry stakeholders.