**MSDS 6371 Project: House Prices-Advanced Regression Techniques**

**By: Rashmi Patel**

**Introduction**

**Data Description**

The project is based on the Competition named House Prices-Advanced regression Techniques from Kaggle. The Ames Housing Dataset is a modernized version of Boston Housing dataset. The Ames Housing Dataset consists of data of house from the year 1872 to 2010 and 81 variables in train data set in which 43 columns are categorical variables and 38 are continuous variables. On the other hand, the test dataset consists of 80 variables in which 43 are categorical variables and 37 are continuous variables.

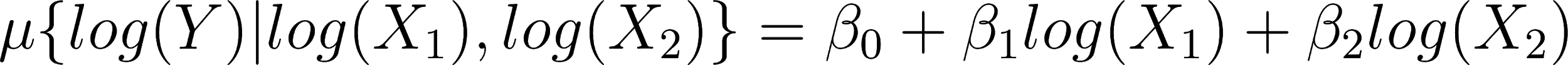
**Analysis Question 1:**

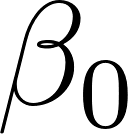
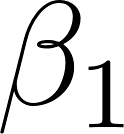
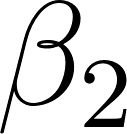
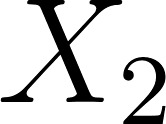
**Restatement of Problem**

The client, Century 21 Ames is a real estate company who has hired me to answer the question that can help them in their business growth. North Ames, Brookside and Edwards are the neighborhood in which the Century 21 Ames sell the houses. The client wants to know the estimate that how the sale price of a house is related to the loving area of the house. Secondly, how the sale price of a house depends on the neighborhood.

**Build and Fit the Model**

Linear regression is a method to determine whether one or more predictor variables explain the dependent variable. In this project we have performed a log-log transformation of the data where both the response and explanatory (predictor) variables are logged and is given by the following

[](https://www.codecogs.com/eqnedit.php?latex=%5Cmu%5C%7Blog(Y)%7Clog(X_1)%2Clog(X_2)%5C%7D%20%3D%20%5Cbeta_0%2B%5Cbeta_1log(X_1)%2B%5Cbeta_2log(X_2)%250)

where [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta_0%250) is the intercept from the linear regression equation, [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta_1%250) and [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta_2%250) are the regression coefficients , and [](https://www.codecogs.com/eqnedit.php?latex=X_1%250) and [](https://www.codecogs.com/eqnedit.php?latex=X_2%250) are explanatory variables.

In this model, we have made the Edwards as reference: We have generated two model (additive and interactive) The interactive model seems to be good model when comparing the results.

* **Additive Model:**

***SalePrice=7.47487+0.595\*(log (GrLivArea)) + 0.014\*(BrkSide)+0.142\*(Names)***

* **Interactive Model:**

***SalePrice=6.923+0.6733\*(log (GrLivArea))-1.010\*(BrkSide)+1.569\*(Names)+ 0.146\* (log (GrLivArea)\* BrkSide)-0.20(log(GrLivArea)\*NAmes)***

**Checking Assumptions**

* + **Residual Plots**: See Appendix for residual plots
  + **Influential point analysis (Cook’s D and Leverage):** See Appendix for residual plots
  + **Make sure to address each assumption**.
    - **Linearity**: See appendix for linear relationship
    - **Normality**: See appendix, as per the histogram the log-transformed data appears to be normally distributed.
    - **Constant Variance**: The model built seems to have equal standard deviation looking at the residual plots (See appendix).
    - **Independence**: The dataset contains the sales of all the houses in Ames, Iowa from the year 1872 to 2010.
    - **Outliers**:

|  |  |
| --- | --- |
|  |  |
| **Before removing Outliers** | **After removing Outliers** |

**Comparing Competing Models**

**Additive Model:**

Adj R2 =0.5002

Internal CV Press =14.45869

**Interactive Model:** Adj R2 =0.5216

Internal CV Press =13.94807

Looking at the Adj-R squared and CV Press, it seems the interactive model is performing better

**Parameters**

**Interactive Model: The better performing model**

**SalePrice=6.923+0.6733\*(log (GrLivArea))-1.010\*(BrkSide)+1.569\*(Names)+ 0.146\* (log (GrLivArea)\* BrkSide)-0.20(log (GrLivArea)\*NAmes)**

**Estimates:** ,,,,

**Interpretation:** Holding all other variables constant, it is estimated that a 10-fold increase in the 100 square footage of the living area is associated with a ( = 4.7130\*100=471.30 dollars) increase in the median sales price of a home in Ames, Iowa (p-value < .0001).

**Confidence Intervals:** A 95% confidence interval for the multiplicative increase is (344.34,644.16) dollars.

**Conclusion**

The interactive model performs the best in comparison to additive model with an adjusted R-squared=0.5216 and RMSE=0.1892. We can say the sale price of houses in Edwards, Brookside and North Ames can be accounted for the living area of a house by 52.79%.

**Analysis Question 2**

**Restatement of Problem**

The next task is to build the most predictive model that performs the best in predicting the sales price of the houses in Ames, Iowa. For this analysis, I must use only forward, backward, stepwise and custom variable selection method for building the model.

**Model Selection**

The model had many outliers, so the outliers was first removed based on the influence statistic.



**Type of Selection**

**Stepwise:** See appendix

**Forward:** See appendix

**Backward:** See appendix

**Custom:** See appendix

**Checking Assumptions**

**Residual Plots:** See Appendix

**Influential point analysis (Cook’s D and Leverage):** See appendix

**Make sure to address each assumption:** See Appendix

**Comparing Competing Models**

|  |  |  |  |
| --- | --- | --- | --- |
| **Predictive Models** | **Adjusted R2** | **CV PRESS** | **Kaggle Score** |
| Forward | **0.9188** | 20.53273 | 0.13985 |
| Backward | .9162 | 19.45764 | **0.13700** |
| Stepwise | .9162 | **19.0847** | 0.13926 |
| CUSTOM | .8568 | 34.379 | 0.13926 |

**Conclusion:**

The Stepwise variable selection model performs the best in predicting the sale price of a house in Ames, Iowa when considering the CV Press Statistic for the best model.

**Appendix**

**---**

**title: "House Prices"**

**author: "Rashmi Patel"**

**date: "4/11/2021"**

**output: word\_document**

**---**

**# Load the libraries**

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

library(Ecdat)

library(boot)

library(DAAG)

library(Metrics)

library(gplots)

library(graphics)

library(corrplot)

library(olsrr)

library(ggpubr)

library(rstatix)

library(tidyverse)

library(visdat)

library(GGally)

library(usmap)

library(mice)

library(VIM)

library(plotly)

library(caret)

library(e1071)

library(class)

library(mapproj)

library(stringr)

library(table1)

library('ggplot2')

library('ggthemes')

library('scales')

library('dplyr')

library('mice')

library('randomForest')

library('data.table')

library('gridExtra')

library('corrplot')

library('GGally')

library('e1071')

library(MASS)

library(data.table)

library(ggplot2)

library(randomForest)

library(dplyr)

library(corrplot)

library(knitr)

library(kableExtra)

library(caret)

library(olsrr)

library(DataExplorer)

library(leaps)

```

**# Read the train.csv dataset from GitHub**

**## Data Description**

**The train.csv dataset fetched from GitHub contains 80 variables and 1460 entries related to the house prices.**

```{r}

house\_train=read.csv("https://raw.githubusercontent.com/RashmiAPatel19/6371-Statistics-Project/main/train.csv",header=TRUE)

head(house\_train)

dim(house\_train)

colnames(house\_train)

```

**# Analysis Question 1: Assume that Century 21 Ames (a real estate company) in Ames Iowa has commissioned you to answer a very important question with respect to their business. Century 21 Ames only sells houses in the NAmes, Edwards and BrkSide neighborhoods and would like to simply get an estimate of how the SalePrice of the house is related to the square footage of the living area of the house (GrLIvArea) and if the SalesPrice (and its relationship to square footage) depends on which neighborhood the house is located in. Build and fit a model that will answer this question, keeping in mind that realtors prefer to talk about living area in increments of 100 sq. ft. Provide your client with the estimate (or estimates if it varies by neighborhood) as well as confidence intervals for any estimate(s) you provide. It turns out that Century 21’s leadership team has a member that has some statistical background. Therefore, make sure and provide evidence that the model assumptions are met and that any suspicious observations (outliers / influential observations) have been identified and addressed. Finally, of course, provide your client with a well written conclusion that quantifies the relationship between living area and sale price with respect to these three neighborhoods.**

```{r}

# Store the datset that has the neighborhood of Brkside, Edwards and NAmes for analysis 1

analysis\_filter=house\_train%>%filter(Neighborhood=="BrkSide"|Neighborhood=="Edwards"|Neighborhood=="NAmes")

unique(analysis\_filter$Neighborhood)

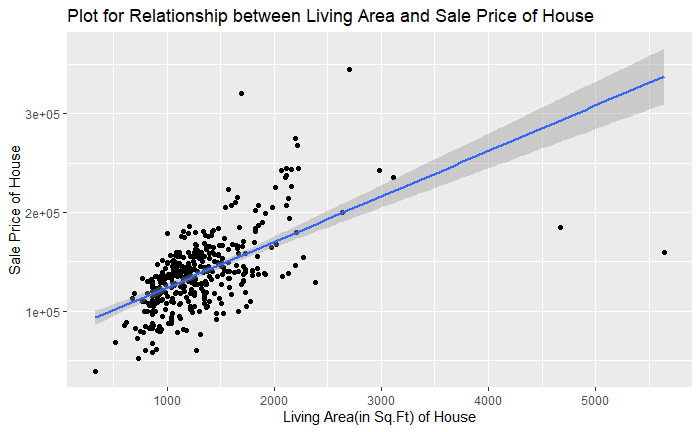
str(analysis\_filter$Neighborhood)

# Plot the relationship between the Living Area and SalePrice to see the linear relationship

analysis\_filter%>%ggplot(aes(x=GrLivArea,y=SalePrice))+geom\_point()+geom\_smooth(method="lm")+

ylab("Sale Price of House")+xlab("Living Area(in Sq.Ft) of House")+

ggtitle("Plot for Relationship between Living Area and Sale Price of House")



# Correlation test between the Liiving Area and Sale Price

## There is enough evidence to suggest that there is correlation between the living area and sale price of house with p-value<0.0001 from correlation test. We are 95% confident that the correlation is between 0.51504 to 0.64731 with an estimate value of 0.58505

cor.test(analysis\_filter$GrLivArea,analysis\_filter$SalePrice)

![Graphical user interface, text

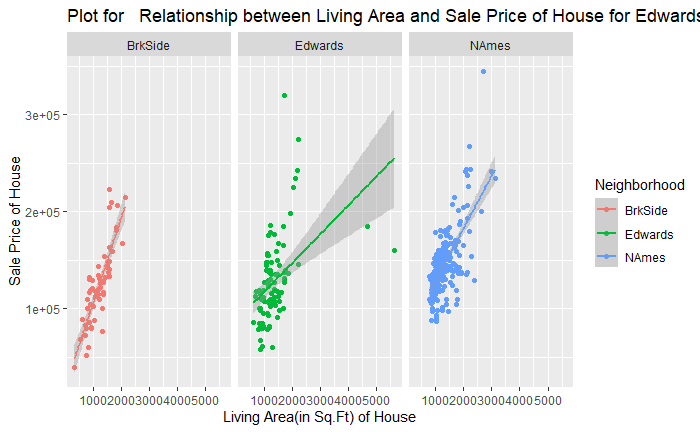
Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDsRXhpZgAATU0AKgAAAAgABAE7AAIAAAALAAAISodpAAQAAAABAAAIVpydAAEAAAAWAAAQzuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEFydGggUGF0ZWwAAAAFkAMAAgAAABQAABCkkAQAAgAAABQAABC4kpEAAgAAAAM5NwAAkpIAAgAAAAM5NwAA6hwABwAACAwAAAiYAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAyMTowNDoxMSAyMjoxODowMQAyMDIxOjA0OjExIDIyOjE4OjAxAAAAQQByAHQAaAAgAFAAYQB0AGUAbAAAAP/hCx1odHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDIxLTA0LTExVDIyOjE4OjAxLjk3MDwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5BcnRoIFBhdGVsPC9yZGY6bGk+PC9yZGY6U2VxPg0KCQkJPC9kYzpjcmVhdG9yPjwvcmRmOkRlc2NyaXB0aW9uPjwvcmRmOlJERj48L3g6eG1wbWV0YT4NCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgPD94cGFja2V0IGVuZD0ndyc/Pv/bAEMABwUFBgUEBwYFBggHBwgKEQsKCQkKFQ8QDBEYFRoZGBUYFxseJyEbHSUdFxgiLiIlKCkrLCsaIC8zLyoyJyorKv/bAEMBBwgICgkKFAsLFCocGBwqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKv/AABEIAKcCBAMBIgACEQEDEQH/xAAfAAABBQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAAAAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgUQpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2gAMAwEAAhEDEQA/APpGiiigAooooAKKKKACiiigAoory7xHBrWq+NPEFno0WtveRW9sLG6t9Wa3tLGRlY75YvNG8ZAJHlSZAwRzSvqkB6jRXnHinXfFbeLrnRfD0WotJZaZFdRGxWy23ErtIP332hgfLBjAxHhvmbJHy1H8QvFOr6Xp91caNPq0WoaZpy3l3bWkdm1rATkr5zTDewJVhiIk4XOASM0k27B5f10/zPS6K4yZte1nxlq2n2XiCTS7aztbSaJYbSKQmR/M3bi6nKHaMgYPAww5BTxXe6xperHU5tQ1C38P26xCT+yltXaJtxMjzrMhYpgpjyjuxn5ehpEqV1c7SiuH1vU9Um1jX/suuSaTDoVjHdRwpFCy3BZXcvKZFJ8vKbcIUPyvz0IqLqXiLX7/AFIxa1caJDbaTaXq29vawvIk0iSMylpEYbflGRjOQMMOQZ5kk32/4P8AkV1S7/8AA/zO8uL60tJreG6uoYZbl/LgSSQK0rYJ2qD944BOB2BqesPSEg8S6D4e1nU7dHu0hjvYiCQIpXhKsRz6SMMHPX1rH8V3usaXqx1ObUNQt/D9usQk/spbV2ibcTI86zIWKYKY8o7sZ+XoauS5ZcrJUlKPMux2lFHWuD8Ranqp8Va1aweI20ey03Ror9fLghY790wYsZFb5MIuRjPTDLzmW0ldlpN7f1rY7yivPre+8S+KNae0t9bl0BToVne+XDZxSPHcSmUHPmKfkG0ZXGTgYZed1/VbzVL3xXqOn2+tz6PBpemw3amCKFjcs7Sgl/NRvkXygMLtPLZPTFTXJv5/gC128vxtb8zsXdY0Z5GCqoyzMcAD1psUsc8KTQSLJFIoZHRsqwPIII6ivNm8Sap4j0x5bnUm8OJb+HLfVJ0EcRjleZZNyyGQEiNfLx8pRvmPzDin2XiS70zQ79WvUt7bT/CVndwGQIBHMyzDdkjnOxBg8cdOaUvcUubp/wAH/IS95xS6/wD2tv8A0o9Jori57/VNZ8VWuiRa5NoyLpMd95lrFC0127MVYDzUdQqYBIC5zIvIHBg1jTdYk+I+kWtv4t1q0t59OuXZIVttu6NoFzhoSCTuJJIOCfl2gkF21t6/hf8AyBa/16f5nd0Vlarq17pjxJaeH9S1cMuWks5LZQh9D5ssfJ9gRXL61qmvyeMXtbDVW0+2QWKm3NtFIczNMsmWOeQFUjBwGUZ3DKlf1+gm7bne0V5xo2r654iFtby69c6WbbRob17iCG3LXkjtIrFvMjZQq+WMhAvL9QMYq2fifxNrtxp1xFqQ0tLi002SS1S1RxuuPOEhy2SOFUrzwQM7hlSbq68vxdhy93f+tLnqNFeUyeKfFOpR6Jp2ntqj3E+jfbXudNjsRLPKH2fMLllTYOCwRc5dcFRwdmC58Ua14mTTLvVZNBKaFa3d1BZwwyyR3LvIHCyOrrt+TBGGzgYI5yrq3N/Wl/8AJj5Xr5f8D/M72ivLNNm1/wASah4NvW8Uatp41PSJpriOzS3EQkUQ8hXiYEtuY/NnH8O0Zz6koKoAWLEDBY4yffimDTW4tFcrr95f3XjDTdAtNYk0WK4tZrozwRxNNOyMq+Wnmq6gAMWb5SeBggZqO+Oq3PiWDw/Dr97p6W+mi6a9jitzNdvv2HO+JkAXALbUXl1xgcUr6J97/he/5MXfy/W3+Z11FedaVrXiHxZcaVDFrDaOl3ojXcr2ttE7NKJQqunmBwFI5wQ3B4IPNaupHXbvxdb6HaeIJbCL+yDPLPBaxNI8wkVQ43qyqDzkYPB4weaL3t53/C9/yE5JX8rfjb/M7Cs7UPEOi6TeW9pqur2Flc3RxBDc3KRvMc4+VWILcnHFcp4t1fX7bXdJ0HS21N3mspbiW50uOzE0roUXAFywjC/OWIALcrggZp3ieXU5/hVYya/brbam0+nG7iQghJftUO4DBI6+hP1NPz8/1sUldpHdUVxnjzVru1Mdpo15rUeoLay3Ri0qKzI8tcDzJGuhtChiBhSGOTwccZcza3rt54KvY/EuqaYNUtC9xDYpB5Qf7PvziSJ85PZiQMcAHmle6v8A11/yZMmotL+un+Z6DdXdtY25nvbiK2hBVTJM4RQWIAGTxySAPcipq5LxzZ3I8P2866nPstp7dZYHt7eSO7zNGMyB4jgjqNhXB57DEutT6jf+MbXQrPV7jRoTYSXhmtY4WlmYSKm0eajqFUNk/LnLLyBkEvrb+tr/AKD6XOoorznStb8QeKb3RLdNYbSorvS57i4ktLaJnlaOdEWSMyB1UMDnkMNrcc4YdT4L1K61XwpbXGoS+fcrJNA8xUKZPLlePcQAACQuTgAZPAFNaq/9b2/QV9bf1tf9TdooooGFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABUSW0EVxLcRwRpNMFEsioAz7emT1OM8ZqWigDO1fw7oniBYhr2j2GpiHJiF7apN5ecZxuBxnA6elNu/Deh39xBcX2i6fczW8ZihkmtUdokxjapIyBgngVp0UAVbLTLDThjT7G2tB5aRYghVPkTIReB0GTgdBmqt14Y0C+1aPVb7Q9NudRiKmO8mtI3mQqcqQ5GRg9OeK1KKOtwM/UPD+jatd211quk2N9cWp3W81zbJI8JyDlSwJXkA8elWvsdr50032aHzbhAkz+WN0ijOAx7gZPB9TU1FHSwDIIIra3jgtokhhiUJHHGoVUUDAAA4AA7VnXXhjQL7Vo9VvtD0251GIqY7ya0jeZCpypDkZGD054rUoo63DpYK5668G6ZqPiybWtWtbTUA1vBFDBc2qyeQ8TyMJFZs4J8zHAGMdea6GigOliJbW3W7e6WCMXEiLG8wQb2VSSFLdSAWbA7ZPrVTU/D+ja3JBJrOk2OoPbEtA13bJKYicZKlgcdB09BWhRQBRvtE0rU7u3utS0yzu7i1z9nmuLdJHhz12kjK5wOlR3HhzRLua3lu9G0+eS1iMNu8tqjGFCMFVJHyqRxgcYrSooDrcztU8PaLrdvFb61pFhqMMJzFHd2ySrGcY+UMCBxxxSX/hvQ9VsILHVNG0+9tLbHkW9xapJHFgYG1WBAwOOO1aVFACKqogRFCqowABgAVA9hZyXBuHtIGmbYTIYwWOzJXnrxuOPTJ9asUUAZd74X0DUobWHUdD027is8/ZkntI3WDOM7AR8vQdPSrjWFm9wZ3tIGmOzMhjBY7MleevG449Mn1qxRQG5mX3hrQtUsIbHU9F068tLc5ht7i0SSOM4xlVIIHB7VZtdMsLEobKytrcpCtupiiVNsS52xjA4UZOB0GTVqigDMvPDOg6jpsGnahomnXVjbkGG1ntI3iiIGBtQjA4JHHrWjHGkUaxxIqIgCqqjAUDoAKdRQBR1XRdK121W21vTLPUoFfesV5brMobBGQGBGcE8+9QXfhbw/f6dbaffaFplzZWgxb201nG8cPGPkUjC8ccVq0UARC0t1uRci3iE6x+UJQg3BM52564zzij7NB9rF15Ef2gJ5Ym2Dfsznbu64zzipaKAKOq6LpWu2q22t6ZZ6lAr71ivLdZlDYIyAwIzgnn3p66VpyabFp6WFqtlDsEdsIVEabSCuFxgYIBHoQKt0UAUdR0PSdXmtpdW0uyvpLR99u9zbpI0LcHKFgdp4HI9BUE3hXw9caPHpNxoWmS6dE5kjs3s42hRiSdwQjaDlm5x3PrWrRQO7IPsFn9iSz+ywfZYwoSDyxsULgqAvQYwMemBVfVtC0jXoEg13SrLUoo23pHeW6TKrYxkBgcHHer9FG+4tiEWdstwlwtvEJo4zEkgQblQkEqD1AyBx04FLb2sFnCIbSCOCIMWCRIFXJJJOB3JJJ9zUtFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVS1jVIdE0W71O6WR4bSJpXWMAsQBk4yQM/jV2vKvGumC7uvFK6hoF5qeqTQr/AGNcQWLy+TD5ShlWYDbGd/mlk3BmBxhsgU1q7A9mz1UHIzUX2mD7WLXzo/tBTzBDvG/ZnG7b1xnjNeWfFCyutU1G4htvD9u93b2SSWd+dDmvLiVgWcrFcJhbcptyNxJYtwvTdoavovho+PbLxBqnhYXdvfafxMNAkuJBPvQq0irEzo23jLgEYI7Ulq18/wAn+dhSbS08v0v9x6RRXlHxRsrrVdTuYrXw/bvd29mkllfnQ5ry4lYFnKxXCYW2KbcjcSWLcL03drqJ02+1Xw1d3OkajdzGV5LO4SCRFsmMLZaZSVKZUlcMD8xAwDQtrjejOiooooAKK8o+KNldarqdzDa+H7d7u3s0ksr86HNeXErBmcrFcJhbcptyNxJYtwvTdoX3hew8S6p4p1DU9EkvY59OtzYi9tXxv8l+VjcfLICQM4DjOOOlRKXLHmD7Sj3PR6xdT8Sf2VrdjYXGk37QX0ywR38ZhMKyMGIUgyCT+E8hCPevMvHVjqus6Fb2p0CKTVI9Dje3vp9Emu7mS42EskcylRaurKDlySWYYHHPf6+JdTg8M3Nnb3LoNTgncNA6PGnlv8zqwDJjIzuAx3rSSs162/GxKleN/Jv8LnUUVwfiNL7R/E0tro25P+ErTyA6YxbXKABpvr5GT6ZgUdW5yvE2h6Zpmt2h0rQzql7Zw21tbWN9oMt3biND8ohuQuy3c7uXZiBtBKjqUunn/X9dynoei3+ofYJLNPsl1c/argQbreLeIcqx3vz8qfLjPqR61crlvEug2lz4i8P6vFpMMuoQ36q96tsGljh8uXgvjITJ9cZNc/41sbQ+LU1C30dtZ1VBAkdrf6DLd25VWJHk3IXZbuSxy7MVG0EqMZKi76edvwQS0+79Wek1iaz4l/srU7bTbXSb/Vr24hecQWRhUpGhVSzNLIi9XUAZJPPHBpt9448KaZeyWep+J9Gs7qI4kguNQijdDjOCpYEda5258KeG/EfxGtNYbQdK1Wxu9Kkka+NnHPFNJ5kQRjJghm2A4Oc4zjimtZJdP+A2Ddk3/W6R2d9qJsYbZzZXc5nnjhKW8Ydot5xufB4Ve55xVDWfEv8AYmo2cFxpF/NbXc0UAvoTCYo5JH2qrBpBJ1xyEI5qj4t0Czvb3R9Tj0mC41G11G2C3a2waaKISZYB8ZVcE55xyan8aWs91YaYtrBJMyavZyOI0LFUWZSzHHQAck9qI6tX/mt8tP8AMmTaul/Lf56/5G611bpdx2rzxrcSozpCXAd1XAYgdSBuXJ7ZHrUtee+MfDegf8J3pmua34Yh1K1e0uIbqSPR2vXaXMPlF1jjdjhVcBiMDpkZGaHxD0e71HxJDLdwxS6I+nCK3E2gXGqfZ59zbmWKF1aJ9pTDkHG0jKn7yvov67/18zSx6jWLr/izTPDcltHqH2uWW5YCOGzs5bl8ZA3FY1JAyQM9zwMniuUt/Btl4g8UTr4osZdXtDoFlAJ723eNZpN0299jfdlAIOfvpvOCMnMvxJaOC40mayOpwatDOptZbSxmuYZBnPlyhAUI3KpwxVh1VhyaqVotLz/UndP0v+Fzr9U8Q6NodvFPrmq2WmRTHEb306wBjjOBvI5x2q9BPFc28c9tKk0Mqh45I2DK6kZBBHBBHeuC1GTUdH8U32t3djMtxe6ZBBZXcWnS6ito6lmlhaOH5wCSDuBCtgZOVANXTfDcfjPXWvfG2g+Yt14dsRJBPBIkMc7NMZAqsflkXcMH76BuCMnLSvf+u/8Al5Atrv8Arb/M9FF1bteNaCeM3KRiRoQ43qhJAYr1wSCM+xqDUtQ/s2CGT7HdXfm3EcGy1i3sm9gu9hnhFzlj2AJrlb7UU0H4nXV5e2Wpvaz6Pbwxy2emXF0pdZpiVJiRsEBlPOOtWvF+i2erRaRqsOkx3N/BqFm0Vy1pmeGLz0Lckb0G3ORxjnNTHVx83+tgn7t/S/4XOsqC5vbWzMIu7mGA3EohhEsgXzHOSEXPVjg8DniuZ8RQahpniJNU0SCWWXU7Y6dKI4i6xygloJnA6IuZAxOOqj0rG8VeHNa1TbpHh3TLNrXS7IJa3N/dPCYrs4KTIPJk8xkCg5yOXYZznCv1/r+v8x9bf1/X+R6NRXnVtoVt418WPfeKNAma1uNAto3t76FxGk3mzb1weN68YbqAcggNzW+I2i3V74iszd28NxoYsGhjSfQp9WSGfdy3lQyBlYpjbIVOMMMjdybfj+F/zt+I7L8vxt/n+B6dRXkfjXTLnU/DumaRPpUesT/2Nsg1bUPD011NNOyhdoAKm0ckKxeRgASP7rEel+H7hrrw3p80i3Cu1um8XMTxSBgMHcrgMDkHqKq2/l/wf8iL7ef/AAP8zRorz74m2barPZac+g2V6kkE3lXl5okupCOQ7VEarGR5JbOfNZgo29+SKmh+H4fEmoaBJ4j0i4u7eHw6sM8epW0gQzrIoZZEkADNlSRuB/vDsalNt/12f+X4+lyT5Vf+t0v1/BnphOBVLRtVg1zRLPVLRZEgvIVmjWUAMFYZGQCRn8a8107TBDb6FD4o0C8v7G3spYLG2FjJN9nnE5CZXGIj5YQJI20KA3zKCc9v4Ctp7P4eaBbXcEtvPFp8KSQzKQ6MEGQwPORTjaUeb+uv+Q1t/XY6CivPvibZtqs9lpz6DZXqSQTeVeXmiS6kI5DtURqsZHkls581mCjb35IqaH4fh8SahoEniPSLi7t4fDqwzx6lbSBDOsihlkSQAM2VJG4H+8OxpJtv+uz/AMvx9LqT5Vf+t0v1/BnplYniLxL/AMI59nkuNJv7q0lljie6tjDsgZ5FjXcHkVzyw+6rV5xqGl3Vx4P0LRL7w3bTbLe5jgn1DQptQ8giTZHCqJt8nKbSJHYKAo69R0d3Hfal8HNJiFrfSXqtpyzRTW8izBo7iLzCVYBuNrHOMEDOcc0LVXXdfr/l/S3qOrS7/wDA/r5M7+iuN8XzXHhvWbXxFplubiW7QaZPbjAErtk2zE9eJSUz2ErE9OMDxT4Y03TdL023uLaTVL60tpWRLrw9NqtpczSMHkkdI1+SQsDhty4Dnhh0Ol/6/r/MOtj0XVNQ/svT2uvsl1ebXRPJtIvMkO5wuQuRwM5PoATVyuN8U6FDrPhLT7u78PW/9q2rWrwxCBZpLI+bGXWNgOAAOSuBhapfEiwsLu4tp7qykv7u3t5BbW1z4fm1WzcsQSWSNfkk+UANuXAY8MOg3Z2FH3lc7+snXvECaH9jjWxu9Rur6Yw29raeXvchGcnMjooAVSclh2HU1TPjHw/o1raW3iLVtI0G+a2jd9Pub+JGhyPugEjIBBAIGDisLXtF8NeNfEnhjV7fTdK8QWjTzR3F6kEd1GY1hk2qzgEbQ/QE4De9U1rYV9LnXXOrSWmhpqE2lX5kYxhrKJElnQuwXkIxU7c5JDEAAnmq3iLxJ/wjkUU82k395asyrLc2ph2wbmCjcHkVjyw+6GrM8ZeGNNuvD1r9m0O1muNOmtxY+XaKz2qCaPIjwMoAq/w44HtVzx3bT3ng66htIJJ5WlgIjiQsxAmQngegBP4VLeo+huTXVvbyQxzzxxPO/lwq7gGRsFtqg9TgE4HYGpa4b4h+HNJ1C/0HVtU0CHVIrO9xeMNN+1y+QYpQF2KjOy+YyHABwee2az/G+mve2fhx9MsIR4bhjkE1jcaBNdxoSqiHdZIUfC4cAYO0kEr3A3pfzKtr8j0msrxD4k0/wxp4u9TM7Kx2pFbQPNJIQCThEBPABJPQdzXDab4Uh1G/8KW2rW82r6ba6ZfI7XWny20RzLD5cckEuSAADtWTP3Aw5AI0fH+naTpnga00+JNTskswkVhLplvNO1uFwu1givlShIw4IP1xTlZK/wDW7RK1R1V14k0fT9Jg1LVtRg0u0uMCOTUm+yZJGQpEu0hsA/KQDx0q3YahZ6pYx3mmXcF5ayjMc9vIJEcZxwwJB5FcXqc8j2Wg6prEutadrMcEmJtK0mS5Uhtm5JIhHLsVtqNtOGGCAwINUvEc2o6x4c0NPEHh2xuHuDK8011oM2oxwMOI/wDRUYvGzqc5ZsJgqSSaHs7ArO3odd4m8Sf8Ixp5vpdJv7+1jUvPLaGHECjHLCSRCev8Iboa2wcjNebwWeozfs+nTJLO8bUoNM+yyW727iQyIApCqRlhxwRkEdCRVv4l6df3x0VlhiudJhlkN9BNpUupRlimI2e2jdWkAOf72CVbHGQS0Bas6zV/EWieH1iOvaxYaYJsiI3t0kPmYxnG4jOMjp61dtrmC8tYrmzmjnt5kEkUsThldSMhgRwQR3FeY6ZouuRXHhFdCvhi1tNRQ3l9odxHFCjSxFIvIaRHjAA2publU71oSeBbWHXvC+l3drLqmm2tpqBuXnjzFJJI8TYkVQEIJLkIRj5eBlc0PTT+upSV7eh2+q6h/ZWk3N99kur37PGZPs9nH5k0uP4UXIyfarYOVBxjPY1x3jTwlpc/w5udPtNEt7ptPsnXTYjbiZ4GC4Xy8gkNwMY54pPHNnHcXeiyavplxq2gxSSfbrSG1a5G8qBE7wqCZEU7uArYJVsfLkD0RPWx2VFeZeLtOS90bw8dB0uOHw3BLN9o0+fw7PNGpIIjLWKmN2UNu/hOCytjA3DrPAlo1h4LsbY3MtysfmBGlsJbIonmNtQQy/OiqMKoOeFBp23DsdDRRRSAK5+Tx74Th199Em8R6ZFqSFVa2kukVtzMVCcnBfI+794ZHHIroK5a8g1HTfiA2rW2kXepW15YQ2bNayQr9nZJZGLOJJEJBEn8O4/KeOmT7SX9bD05W/63OpoorifEHgfWNX1ye9s/EX2OGXbtg8zURtwoHSG+jTtnhB75PJAO2rGl8UWcXii20E218bm434mNqyQgqm/AkbAfIz9zdgjDY4zqWkLW9nDDI/mPHGqM/wA3zEDGfmJP5kn1J61yPiOfWP8AhM9HubHwtql9a6a0xkngmtFWTzIto2B51bgnnIHTjND30J1tc3NX8S2Oi6npen3IlkudUn8mFIk3beMl354UHAz6sBWvXA634U8Tz+Ik1TTtT06RZtQglKXGms0ltDGDhN4uFDKCzHAUElyc12D3eoLr8Nomm79Pe3aSS/8APUeXIGAWPy+pyCTu6DGO9KOq13v+if8An89BvfTa36szdY8YWuj3k0L2F9dRWcazX9zbRoY7KNs4Z9zBm4BJCByAMkDIzHrXjW10a8mgGn31+tpbJd3stmIitrCxYK7BnVmzsc4QMfl6cjOXrWk6zb3niO30rS2vo/EcahbkTxqlpJ5IhYyhiG2bVVhsDkncCBxnI8YeDtR1DWA8GlXV7Nb2UMOjahbXcca2MyNuLXCO48xN6xNjbLwpwoPVx3V/63/4HYbS6f1t/wAH/I7WfxTZW/iW00Rre+NxdMVWU2rrCCIzJ/rGADfKp+5uweDitquK8Rza2fFGhTWvhjUtQi0uV5Zri2mtFSXfAyfIJJ1bhm53AdDjNb2qa1cafqWi2sOn+eNSuWhlYzqrW6iNn3bed/3cEDpnOaCSZNC0+PXn1ny5ZL5o/LEktxJIsanGRGjMVjztXO0DOBnNQeIfE+meG7N3vr20jumgkltrSa5WOS5KLkqgPJ7DgHGa2K4PxPpOsSX/AIkjs9D/ALWj1vS1tYJTNEkcLKsoMcm87tpLggqrcsc7etRNtR90uNnLUNe+KOnaXq1ppVpd6D9tltzPcLqetLZpbfc2oSEc7235AKjhSa7tCTGpYAMRyFOR+B71zdtpN7H4y0y+eHFtBoz2sj71+WQyRELjOeitz04rpa1draef5szi29/L8kFFcV4i8E6xrOuTXtl4h+xQyBQsHmagNuFAPEN9EnbPCD3yea66xge10+3t5ZPNeKJUaTLfMQME/MzNz7sT6k9albFPfQzPEfiNvDlqbubR9QvbKKNpbm5tPJK2yLyWZXkVzxk4RWPB+lRa34th0edYYdMvtTcWxu5hZ+UDBCDje3mSJnPOAu48HjpVHx7Z3up6eLPTdI1K9uWjb7PcW97HDbQyHhTOjyDzEBwSpjkGAflJ4rG8b+F7/XNVtpL/AEe61ZYtP8q0n026jgazumP7x3EkiB42xHhSHHyHKc8zd20/rR/r/wAPro/6/L+uvppZ9VceLbC31jTdO8i+eTUnCRSi1dYlzG8gzIwCnhDwpJBxkCtyuF8QHxEb7wx/xT19qsul3C3N7dWctqkcjG3kjYIJJkbO5weQBjv2rodX1y401tGWLTTM+pXiW0kbzqj26lGZmwM7yu3kKemTnAzWllsu9ib6J+V/zubNZuo+ItE0eTy9W1jT7F/l+W5ukjPzZ28MR12tj12n0rSrznXPtf8AwsbXU07Qjq1zcaBb2ykSRIIt8lx98uwPlkgbtu48D5TUO/QtJNO/9apHo1RJcwSXEsEc8bzQhTLGrgtHnpkdRnBxmsGLw1qS6PplnF4o1OxeztI7eQ2cdswmZVALnzopDk47GoPEGjakdZgvdFVne9tv7Ovpd6o0ce7cs/PUoDKAADkyDoASKa1siI6q7NTUPFfh3SY7eTVde0yyS6Utbtc3kcYmAxkqWI3DkdPUVT8TeIdS0Caxmh0y0u9NuLq3tZZmvWjmjaaZYwVj8plYDcDy696xfHGleIr2D+xdAguotJn057ZTYfZQFkIKhZ/PyRCFI/1SlvvegBt6ppep6p4F0O1j0+WG7t7zTpZraWSPdGsM8bOSVYqcKhPBOfrxThZyV9rr7r6g7pfJ/kjqnvrSOeWGS6hWWGITSRtIAyRnIDkdl+VuenB9Kq6l4g0bR7SG61fV7Cwt5yBFNdXKRJISMgKWIB454rmvEena0nifU7vS9IfUE1LRBYxutxHGkMqtK37zcwYKfMGCqvyDkDrWZrvhjXBqOl39qNVeJdGGnTwaU1iZo2yC2ftSlCjYwdrA5ReCORmm30/r3v8AJfeXJJOy/rb/ADf3HpIORkdKKz/D9kNN8N6bYqt0i21rFEFvHV5gFUDDshKluOSpIz0rJ8UeG7zWJUmiuY723QfPo1+StpccEYYxjdzkZ8wSpxwmeRctHZEK7Wp0cM0VxCstvIksbDKujBlP0IrI1/xE+gI0z6LqV9aRRGa4ubTySsCDqSHkVmwAThFY+2cCtS0Zms4i9ubZtozCSDs9uOPyrmPF76rdX0Gmp4e1DU9EkiL3hsZrZTO2cCFhLKhCY5bHXgZxuBl36FK3U6qGaO4gjmhYPHIodGHcEZBrL13XZ9FQSRaHqWpxBGklksvJxCo9RJIhPGeFDHj6Znuru/gvNNhstK8+2ndlupTOqfY1CEg7f48sAuF6Zz0rF8YyavPNbada6Ff6jpNxG5v3sJ7dJGHAEP72VMKwJJZTnAwOuQS30EttTWufE2h2Wj22q6hq1nZWF2qtBcXc6wo+5dy4Lkckc461pqyugdGDKwyCDkEVwfivQtXn1vSNY0ePVIobexltHtNM+xefDvKMOLkGLbhNrbWB4XGRmtXTvDNzF4W0fT9O1rW9DjsbVYfKxZySkAAASFopFyAMfIQPrxVaa/13/wCB94leyuX9W8UWej6pZWFxbX0kt5NHErxWrGJC5IBaU4TqPugluc7cZNO1nxDHpNzbWcNjd6nf3Su8VnZ+X5hRMbnJkdEVQWUZLDlgBk1ieMBrBGj2mnaHqes/Y7y3upruOW0QOEJ3Ah5Y/nPXhQvPUdKq+KNCk1bWNP1q+8LS63aNp8ltPo8skBeJ2ZHVmV5BE2NpBIYkHBXPJEa2+b+62n46D15vkvz1/A7i2lee1illgktndAzQylS0ZI+6dpK5HTgkehNSVl+GLC70vwtptjqUxmu7e2SOZzIZMsByNx5bHTJ5OMmtSre4lsZ1zoWn3etW2q3Ucst1aj9wHuJDFGcEbxFu2b8MRv27sHGcVo0UUhhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFY2q+LNJ0S+jtdTa7hMjRqJhYTvApdtqhplQxrliB8zDGRnrWzXn/jHwf4j8QapdG3uIZbRhC9oJNVuLZYGjIYo0EaFJQ7L99ySobhTtwx1QG1r2qa1pXiXRFhuLB9M1G9FnJbvaP5yHypH3CXzdvWMDGzv1p13470qy8UHQZrXWGuhF5heHSLmWPG4DhlQ5HP3h8o6E54qxrWkXmrz+H5x5ET6fqC3dwhckY8mRCqHbyd0g6gcA9OlQatpusReLLbWtEt7G7/0NrSaK8u3t9oLq4ZSsUm48EYOO3NKN72fd/lp+P8AVgn8N472X5u/4f1cs6x4u0XQboW+p3Ukcnl+bJ5VtLKII848yVkUiJMg/M5UcNzwcM1DxpoWmalLYXV3IbqHyzLFDayzGJZM7HbYp2qcfePyjuRXO+KvAl3qniK81G2t01G31G0S2ubOXXbvTlATcP8AlgGEisHIKsoxjqdxA6PR9Fm07xBqt8/lLBeRWyRRo5Yp5aMpByB6jB/lRrYUt9CPVfHGgaNfXNne3UxurSNZbiG2s5rh4o2BIdhGjEJwfm6DoTUuq+L9H0Zov7QkulilRZBcxWM8sCqxwC0yIUQf7zDA56VA/h+6bUfEtwJIdmrWscMA3HKlY3U7uOBlh0zXK694C8R6rpcemLdwyWqaVDaxAapcWyW8yIQxMUaYnVyF5kPyjovUMot9f63/AMl94a/n+lvv1Oou/HelWXic6FNa6w10IvMLw6Rcyx4yBwyocjn7w+UdCc8VJq/iI2XiTSNItz5ct3cbZWuLK4Mbx+XI22OZU8vzMoDhm6Bu+Kr3mna+niCw1vT7PTZrj7AbS7tp7+SNIiXV8o4hYvyGHKp2PtWlrWlT6lf6JPA0arp9/wDaZQ5OWXyZY8LgdcyDrjjNVFaK/fX7/wDIa3l6afd/mWk1exe6vrcXAWXTwrXKupXy1ZdwbJGCuAeRkZBHUEVnaj400LShAb26mXzoBc4jtJpDFCf+WkoVD5SdfmfaOD6HEGu+G7nUdbgurKSGOG4iFpqYckNLAHDgLgcn76ckYErHqADjeKvAt3qfie41ayhW+ivbKO0uLSTXLvTVARnIOYAwkDCQgqw4xwfmIpdP6/rXQel/6/rud4CGUFSCCMgjvXJ3njyz0jxZqml6szCK0toJ4ltLSa4mw/mb3ZYwxCDYvzYABOCeRXT2kIt7KCBY1jEcaoEViwXAxgE8n6msX+wbr+39fvt8PlalZwwQjJ3KyCQEtxwPnHTPeiWl7Ex1tc0LPXdMv0u3tLtHSz2mdiCoUMgkDZI5UqwO4ZHXnINZU3xB8NQAF72Y/wCjxXT7LKdvKhkBKyPhPkTCnLNgDuRXJPpCNq+ieGLG+tprv+y49P8AEUVs28pbxKrKWP8ADuy8ahgCVmYj7tXNUsdZ1Hxj4r0vQlsUjvdNtLaWa5kZTbKwmXeiKpEhAJwpKDj71VJJfDr/AF+v4aii3pzad/6+86zVPF2i6PeC2vribzBGJpDBayzLBGScPK8alYlOG+ZyowrHOAcU7zx9pFj4kXRZLfVpbhoTMJbfSbmaIrlRwyIdw+cfMMqOhIPFUbvwxrVjc6hFoAsLi11SyitZZL6d45LYohj3qqowlBUg7Mpyp5+bInm8ParpGo6Pd+HYrO/+waa2nPHf3b2+VzGQ4ZY5Mn93yCB160L4tdv+H/4H3+Qa8vn/AMN/wfuF1DxxbaR40vNI1Ld5MdhBdQpa2k1xO5Z5VclIwzbFCJztwC3J5FX77xroNhZ2tzLevLFdwfaYja20twTDgHzSI1YqnI+ZsDnrSxaLcp4y1LV2eLyLvTre1RQx3B0eZiTxjH7xcc9jxXGf8K11K1s9IePy724t9Hh0y7t11u706M+WSQ6yQKS4O5htZfQgjkFdP68/+B95o7XbXl+n/BO2v/Fmk6feRWkkl1czyxLMqWNjPd7UY4Vm8pG2g4OC2M4OOhqHxB4iOmahpthb/JcXd1EheeyuHhMbNgqJUQoshxwGYc4z1rD1rwpq9xa2dtoOm6dpslvaRQQ6lDq9xFPabRgJhYv9IjU/MEkYK56qOtdLrel3Gp2thHFJHvtr6C4kZ8jcqOGbGAeTjgU7K69fwuZtvlfe342/zE1PxRpmlailhcm8lumQSGKzsJ7oopJAZ/KRtgJBwWxnBx0NP1DUNI0XULSa/McF3qcyWMMohJeZ/mZIywBOB85GeBk+tYXi7QNZ1e+36LaWNrN5SpHrC6lNbXMBBJwY0iImRSd3lu4RjwQOtdDqA1gGwGlNZMouF+3NdBgTDg7jHt/jzjGeMZpLp6lPrbt/X4mhXNar470rR/EkWiXdrq8lzJC8oe20m4njwuzOCiHd98crkDoxBwD0tc5r+l6s3iLTdb0KCyuprS3ntnt7y6e3UrKY23B1jkOQYsY287uvHJ11GrE2r+NNC0O8e01G6lW6jthdtBDaTTSCEkjzNqKTtBU5P8PGcZGbr67p0d3cWz3GJre0W9lXY3ywksA2cc/cbgc8dKqW+j3SeOLrW5WhEE+mwWojViWDpJKzdgMYkGD14PArL8QeHtcufEF9e6IdPKajpI0+RruR1NuVaRg4RVPmA+ZgjcmMZyc4qJOSjpvr+tv0KtG/3fpf7tTU1PxfoukQW8t5cTN9phM8UdtaS3EhiABMhSNWZUGRliAASATk1sQzR3EEc0EiyRSKHR0OQykZBB7iuD1vwLf3N7p1/Z7bqWHTU0+4tv7Zu9OUhTkOJIAS3JYFWXnIORjB7HRbBNL0GwsIraO0S1t44Vt4pmlWIKoG0OwDMBjGSAT1IrWy19f8/wDgGSvp/Xb/AIJBrHiXTNDnggvpLh7i4DNHb2lpLdSsq4y3lxKzBRkAtjGSBnkVFqHi3SdMNut014ZLiLz1ih0+4mkSP+9IiIWjHOMuF5B9DVTV9L1q38Ux674ejsLt5LT7HcWt9cPANocurpIqPg5ZgVK88cjHMF3pXiGHXn1rS7XTJri/sIrW8t7m+kRLdo2dg0biFjIP3rAgqmdoPfiNbf152/QvS/8AXl+WvqXNR8deHdLl8q5v2kk+yLehLW3luC1uxIEoEatlODlhwOM4yMu1XxroOi3TWt/dyi4S2F4YYLWWZxCSR5m2NWO0FTk/w8ZxkZo+F/B03hrVIGW5Se1t9FtdNVjkSM8TyEsR0AIcY5PetKHR7iPx1d60Xj+zTadDaKgJ3h0klYkjGMYkHf1p9V8/1t+g9NfRfpf7tSTV/E2l6JHAb+WYtcAtFDbWstxK6gZZvLjVm2jIy2MDIyeRWV4o1/UrLRbHXfDt9ps2myy2ysktq8pnSaZEDJIsqhflfPKtmoPGHhG81fXLHV9NzNLb28ltJbHV7nTgysysGEsGWyCuNpUg5zkbRmxe+FpZ/h9ZaDp8FtYNbvaMIftMk0caxTRyMokZdzcIQCQCTjOKS2v5/r/kCtdJ7f19xra34isPD0SSakt6UcM2610+e5ChcZLeUjbRz1OP0rN1Lx/oumahp9q8ep3X9ox+bBNZaZPcRsm3cCGjQ7sjsuSOpAHNQeMvD2s67eWqWNxnTvJkjngGpz2LI7EYl3QqWlwu4eWWVTnk9CK0Ph3xBZ6J4WeC302bU9CjMDwPeyJDKhiMe4SCEsDwpxs7kZ4yTWzf9df+B9/3TK+lv6/rX+rX1PEnicaTZ2X2XdHcXs8CxPdWFy0Kq8qKQ7Ih8tiGwocr8xGeM1d1jxJpuhyxRXzXLzTKXSG0s5rqTaMZYpErMFBIG4jGSBmotf0u81vw9DagQQ3X2m1nkXzCyL5c8cjgNtBPCEA4GeM4qrq2maxB4mGt+H4bG8llsxZzW99dPbqoVy6urLHJz8zAjAzxzxim9Pvf5afjp+I9OW5NeeNvD9lJbRvfNPLeW/2m2is7eS5eeLIG5FiViw5BOAcDk8c1q6dqNrq2m29/p8vnW1wgkjfaVyD6ggEH1BAI6Guc8N+D7jw/qWnyG5jnhtdKNm7chnkMokLBecL1xzkcCtTwppM+heG7fT7t43liaQs0RJU7pGYdQOzCkttfP83b8LEXfNbp/wABfrcg8WeIm0KzhW3Oy7uZo0ikmsria3UGRFbe8SEISGIXcQM47Zq1q/iXTdDnhgvTdSTzqzpDZ2U11IVUgFisSMQuSBkgDJpfEmmTaxoj2ds0ayNNDIDISBhJVc9AeymsnxjoupauUGl6bYzSCFkS+fVZ7C5tiTyEeGJmKnCkruAO0ZB7K7L0OojcSRq6hgGAIDKVP4g8j6GsnV/Fej6HfQ2Wo3EovLiNpYbaC2lnllVSAxVI1ZmxuBIAJAyegJrRsIbi3062hvbn7VcxxKss+wJ5rgYLbRwMnnFZ0ukTv44ttZDx/Z4tPltWUk7y7SIwOMYxhD39Kv7Vumv5P9Sbvlv10/Mhm8a6HFY2V0txc3C3yNJBFa2M88zKpAZjFGhdQpIBLKMEgHBOKq6l8QtD02702ErqN4uprvt57DTZ7mJl2ls7o0IY4U8LkjqQBzWXpvhXxB4dmtr7S002+uVjuYJre5upIE2SXLTI6yCJzkbsFdvORz8vNgeEtT0rSPDw0prS+vtHuJZnS6me3imMqSB8MEkK4MmQMHgY96XT+u//AAwanYwTLcW8c0YcLIoZRIjIwBGeVYAg+xAIrP1nxFpugiEajJN5lwWEUNtbS3Er7RliI4lZiBxk4wMjJ5FX7Zp2tYmvI44rgoDLHFIXVWxyAxClhnuQM+grkvGXhG71rW9N1jTSZZrOCa2e2Oq3OnB0kKNuE0ALZBjHylSDnsVFD3Kj5mveeLtFsbGxunupJ49QTfarZ20ty8yYDFgkSs20AjLYwMjPUVJp/ifR9VayXT71bj7fDLPbFEbEqRsFdgcYwCy/XORkVgQ+GNU0i40O90Gx01HsbGWxmsri/mdIkd0fckxjLuQU5DKN2eoxzkeE9I1qTRvDWr6Z9gnmtYb61uFuJHhUCW4BMigK5ODH9wkZz98UNpP7/wDgfpr5k31t5f5f8E7S48U6Xb6TbamDeXFrdDMTWenz3DEYzkpGjMo9yAKyvE/iLUIvBY8TeE7/AE+SzjtvtWLqzkl+0IQCu0iRNnGeob6DFUJvCXiJPDPh/TYbqGZbGEx30EGpXGnrM+BtkWaJTJhfm/d8A7gSflFT2/hHU0+ELeEpXtBdx2Zs4plldo3UcK7ErlSRyRhsep60R8xvyO1ByoNc/wCIfGmm+GdRsLLULfU5Zb59kZs9NnuFB2s3JRTk/IflGW74xkjoAMKBWD4o0m/1BtKvNIS2lu9LvftSQXUzQxygxSRkF1RyvEmfunpjvkHUFtr2Jr7xTpenWtnPdG8zerugt4rCeW4YAAkmFUMigZGcqMEgHBIrRsb621KxivLGZZoJRlHXv7Y6gg8EHkEYNc9qOma8+saX4gsLXTpNRgs5bS4sZ710hxI0bFkmELNkGMDmMbg3bAzoeFdEfw/oK2U0olmeea5lZc7Q8srSsq5/hBcgeoGaelhPdf1/X9MbrWuXNlqFppekWcN7qd3HJMkVxcmCNYoyodmcI5HLqAApyT2GTVrQtXTXNHivkhe3ZmeOWGQgtFIjlHQkcHDKRkdcVQ1zSdQbXLHXdDjtZ760gmtTb3c7QxyRylGPzqjlSDGp+6c8jjrVnw1o8mh6HHaXEqTXDyy3FxJGpVWllkaR9oJJA3MQM9gKFa3n/X6fiNmtRRRSAKKKKACiiigArh/E3jHU9IOt39sbCPT/AA8I2u7e5jYzXasquTG4cBOGKrlW3MCPl613Fcn4n8Ff8JRqsUl4+lraLH5TsdMD3uw53olyX+RWB2kBM4LYIJyDqh6dRviLxFqNhr1hBb3trp2nXESs13d6TPcxs7PtCGVJESEnIxvPJYY9K66sDxPoOoeIbRtMi1SGy0q4j8q8jW0L3EiZ5CS+YFTI4yUYjJIIOMb4GAAKF8PzZOtzmNX8c2mkX2o27aXqd0ulxRz309vEhjgiZS28lnBYAKSVUFvRTT9a8bWWjTzItjfahHa2wu72azRClpCc4kfc6lshWO1AzYU8cjOLqPhnV9c8VeJ7WK9+waVqNvbQXJksi7Tx7GDiGTcoRsEgkhwMjgd5vEHw1s9W1z+1LWDQ5JXto7eRNa0YagoWPO0p+8RkOGIPJBwvAxzKv1/rf/gfiNW/r+vX8DVuvGcMWr3OnWOkapqctqIXmeziQoiSglXyzruxjkLlu4BHNTal4o+x6w2l6do+o6xdxxLNOll5KiBWJC7mlkRcttbABJ4JIHGZ9N0T+z9b1HUfOVvt0cCeUkWwR+WpXjk8HPTt71S1Dw/qy+IptX8Oava2Ml3CkV1Be2JuY5ChOx12yRsrAMwPJBGOBjNPqKN7ahrnjS10O4nifTtQvBZ24ur+S1SMrZQnOHfc6luFc7UDthTxyMxal46t7G+v7W00bVdUfT4EuLhrOOLYsTKWDB5JFU8D7oO70BHNUPEfw5t9c8QHWVi0Ke6mt0hnXWNFF8nyZ2tGPMRkPzEHlgQF6YydlfDOLjW5ftSj+1raO32pDgQ7I2TI55HzZxxjFTJyUbx3/wCH/wCAP7S7df6/r/POm8dXP/CTW+l2HhfU9QgubP7XFeQT2yrImUG5VeVTt+fnOGz0UjmuvrlpPCmo2smkXGh6ra293p2n/wBnu95YtOksfyHIVZUKnMfqep+tdSOnPWtXa2nn+en4ER5t5eX5a/jcwdU8Uix1n+ytP0jUNZvkgFxNFY+SogjYlULNLIi5Yq2ACT8pOAKj1TxhFpt9NbR6TqV8bSBbi+e1WIrZo2SC+6RSxwrHbGHOB05XKaj4e1UeJJNa8Oavb2E1zbJb3UN7ZG5ilCMxR1CyRsrDe4PzEEEccZqHUvCup3F9ez6ZrNvZpqtukGopJYmUttUrvhPmDy2Ktj5hIPlXjg5h35fd3NNL67Be+PLWC9uLXTtI1TV3t7WK8kayjj2eTIGIYPJIin7v3c7jkYBAJGDrnirW9V8UadYeHrXXodPm09r1bnTBp5e6BMe1h9pchUAcggqrEkY4FdNYeEodNuNQNrPtgurCCxjiKf6lYldQc5+bIf26e9Jo/hT+yr3SLj7Z5v8AZuk/2bt8rb5nMZ39Tj/V9OevXir05/L/AID/AOB9/wB2d5cvn/wV/wAH+t2a5q1/Z67odhDFfwQT3aLLfLFbyRTfJITCwMgdCdoO5UI4A7nEXi3xa+kWmp2+m2N9dXVpYPcy3FtHGyWeVbYzhmBbJUnaiueORyM7Wq6V/ac2nSed5X2G7FzjZnfhGXb14+/nPPSsTXPCF9qN9qkmmaymnwaxZi2vEazEzgqrqrxsXAU4fBDKwIAxtPNZS5nBpb6/kv1NVZSXbT83/X9WJr3xemmQ28I0++1W8+xLeXEdiseYYsY8xt7oOSGwq5Y4OBxTLnx1arqH2LStK1PWZjZRX6mxjj2GGQsA2+R0UH5PukgnIwDhsUfEPw4tda1G01FY9Hnu4bNbN/7Y0kX0TIpJUqu9CjZJ5DYIOCDgEbOneG/sGsz6h58I87Tbex8i3t/Kjj8oyHco3HCnzMBewHU1b1u/X9bfp+PymOi+S/S/6/h86p8c2lwlh/YmmajrM19ZrfJBZrEjRQN913Mroq5OQFzuOG4wCRPq3i6PSpVt10nUb67Fr9ruLa0ERe2i5G5y0iqeQQApYnacA4zWXYeB9R0OHTH8Pa5BBd2mmxadcNd2BmiuUjyUbYsqMjAs3RyMMQQeCE8SfD2LxDqltq0yaJc6ilqtrOdU0cXkDgEsGSMyK0Zyzfxng4OcAglv7vd/rb9L9fTo9P6+V/1sdhaXUN9Zw3dpIJYJ41kjdejKwyD+RrB8Taxe2OoaXa20F9BDcXsCSX8UcEkWGfBhZXkDjd/eVTjNbljb/ZNPt7bbCvkxKmLeLy4xgY+VMnavoMnA71W1jSv7WitE87yfs15DdZ2bt3lsG29RjOOtN25lba6+65GvI+9n99jN1Txgmn67No9ro2qapew2yXbpZxxhRGzOud8jooIKH5ScnPAOGxmN8RJZtW0u20nwvqmpWup2LX1vcwzWyb0AjPCSSqePNAO7aQegYZI349D2eKL/AFj7Rn7ZZQ2nk7PueW0jbs55z5vTHbrzWLB4Kv8ATNP8PLour20V9oti1j593YtNHNGwj3Hy1lQq2YlI+Y9xg9QK3X+t/wDgGjtfT+tv+Caet+KY9HuUtIdL1DVLwwNcvbWSx7oogcF2MjovXgAEsecA4OKdx4+shcLBpOmanrUkmnR6nH/Z8SEPA5YAhndRn5c7c5OflBwcV/FXw/t/Emr22rNHo817Fb/ZnGraSt9CybtwKoXVlYHOCGxhiCDwRp6T4a/szWhqPnW4/wCJXb6f9ntbbyYk8pnbci7jtU78BecAdTSXn/W//A/H5LTX+u3/AAfwLia7anWJ9NlSWCaG0S73SqArxkkEg56qV+YHGMj1rP1Xxa+maHHq0fh/Vr60a2a5lNuIFaBAN3zLJKpJxzhdx47HipfEXhldfmtJBcC3MW+G4/db/PtpMeZD1GA21OecY6c1leMvAC+Lr6OS4uLJrX7M1s9vfWAuhFk8ywZdRFLg43lX6LxwQU720/r+tPxCNr6/1t/wRfGF7f21rpOtaNrd3bwTX1jA1osUJhmjmuEVi2+MyAlXxwwxgcV0Wqar/ZZs/wDQb28+13SW3+iQ+Z5O7P7yTkbUGOW7ZFZd14YuL/wrpmk3uoRGaxuLSdriK2KrJ5EqSABC5K5CAZ3HGc+1amqWmoXZs/7M1P8As/ybpJbj/R1l+0RDO6Ln7ucj5hyMVo+W1l3/AA0/4JKu1d9vx1/4BfoooqBhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAcPrWrazcXniO40rVGsY/DkalbYQRsl3J5ImYSlgW2bWVRsKEHcSTxjs7WcXVnDcKColjVwCORkZrE1jwfa6xeTTPf31rFeRrDf21tIgjvY1zhX3KWXgkEoUJBwScDGi2lA67BqS3t4ghtmtxZpLi3bLA7ymOXG3AOeATQr9f63/PQHbp/W3/BOe8W3mqWGs21y17q9joccQNxc6dHaOkb78ZmEqtJsxjmMcDJOAMixqF5rNp4+0aA6jCdLvjOotY7Xa3yxbsvIWbd8wyNoTAJB3cEXvEHhpPEarb32p30enMoW40+AxrFdLnOHYoZMHoQrqCOD1NVNX8Hzarrlvqa+JtYsXtSxtobZLQxw7l2tjfAzHI/vE9eMUlpG3m/6/wAhSu3p5f1/mVPFt5qlhrNtcte6vY6HHEDcXOnR2jpG+/GZhKrSbMY5jHAyTgDIr61q2sXF54juNK1RrGPw5EpS2EEbpdyeSJmEpYFtm1lUbChHzEk8Y3de8Mr4iRbe+1S/TT2UJc2EJiWO6Gc4dtnmYPQhWUEcHqai1jwfaaxeTStfX1pDdxrDfWttIgjvY1zhX3KWXgkExlCQcEnAwWfLZPuUmr3a7GVqnifULrW9Gi0R1hsVvIE1GRkDGQyoSIFz0IBDseoygHU4y/H3inVNH8Rtb2uo3tlAlpFJDLa2sc1vHK0pUm9d0JiixtIYMmVWXnIUV0mqfDzwnq99He3nh/TTdJOs7TiyiLylegdipLD/AAFO1jwXa6vf3Vx/aOoWcd/AtvqFtbNH5d5GuQFbejMvysy5jKHB68Ahx0fz/Cy/USvfXsvzM3WdQ1m91DXP7I1ZtOXQbeORIlgjkS7lKGQiXcC3l4CjCFG5bnpjrdPuxf6ba3iqUFxCkoUjkbgDj9axtW8G2mqzOyX1/YRXEK215BZyKq3cK5wjllLLwzDdGUbDdeBjoERY41RFCqowoHYULr/Xf89Ba6f12/4JxmtahrN5qWu/2RqzacugwJIkQgjkS6lKGQiXcC2zAUYQo3Lc9MdZp92L/TbW8VSguIUlCkcjcAcfrWRrXhC21q7lmOoX9kt1EsF9DaSIq3kQJwjllLLwzDdGUbDdeBi+2kKdatdQS8vIltrd4Fs45cW7hip3MmOWG3AOeAT60K/X+t/z0G+n9dv+Cch4q8S6ivii702xl1e3tNLsoru7l0e3t5JgJDJ8zfaAV2KsZO1AzsTwMDBta14ovJtU0qLw5cxtYpe2qX91sD+aJiNsK9gSrB2PUApj72Rq6z4QttY1CS8TUL/T5Li3FreCzdFF3CCSEfcjFcb3wyFW+Y/N0xBqnw38IaxNHNeeHtMM6SxyGYWUReTZjCsxUkqQACPTiiOjV9rr89flb+raCmrp27P8v8/0G+KrPVZriBdG8Tajpt1dERQ20ENq8S45aVhJCzkBeSNwBO0DBNQeOb290uC3udP1vULK4AAjiFnG9nIQwybiVoj5S4OCfMTjO3J4rpRpsQ1k6kXkaUW4t0Q42Rru3ErxkFvlzzj5F44rO8Q+Gf8AhJFNvdaxqNvp0sflXNhbeSsdyufmDMYzIMj5TsdeOmDzQuhWhS1++1mx8V+Hlh1CBdNvr77PJbJa/Ow+zzOd0hY8bkUgKqkY5JziofG91q9jcWd1a3OsWmkQwyvf3GkxWkjRYKbXdZ1ZigXzCRGpbjoat654PfW9SsrtfEWq6eLGQS20FolqUjfYybv3kLsSVdhgkjnpVrWvDj65ZRWdxrepQWvlmK6itzCn2xTgEO3lllyAR+7KfePtg9A0ur9jm/FXiPUj4jGk6RLqwhtdPW/uJdHgtpJ3V2ZVObgeWFAQkqoLsWXbwGrtdNu4NQ0q0vLO4+1W9xCksU+B+9VlBDcAdQc9BWTqnhG21C4juLO/vdJmW2FpI9gyAywA5Ebb0YAAk4ZcMMnBGa2rS1hsbOG0tIxFBBGscUa9FVRgD8hTWz/rv+lvx8iXe/8AXZfrf8PMlooopDCiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACuQ1bX9Z07xakF3PaaXozyxRwTTabLci6LY3AzJKqW53EIokXBOCC2do6+sW78JaTfar9vvFvJpPMWUwPqFwbcsuNpMG/yjggH7vUZ680dUHQ57x540vfDU7tpl7aSm2gSaewOk3Fy5BY8tPE+yAMAQpkUjKk8gHDNV8WeJI/EV/Z6amlLawXRtI2uI5GcH7GLjecMAQCCNvG7cOV2/N0Wq+DdE1u8mudRt53e4h8mdYryaKOZcEDeiOFcjccFgSOMEYGJYvCukQ5K20jsXEjPLcSSMzCHyNxZmJJ8v5cnr1680teV9yla+pyNz428RSafq2pWKaXDa6TplvqTxzxSO84eFpHiDB1CfdwHw2M8qaueKdO0ybxJ4O1yPTrVdQn1VEN35C+cYzbTnYXxnHTjOOKnj+Hemy+IL25v4S9i0VrFbW8d3MissKkbZkBCyrnGA+4deOTnqLzTLTUJbOS7h8x7Kf7RbncRsk2sueDz8rsMHjmnpzXXe/yv/l/WpFO6haW9mvvX9f0jifF3jvVtL8QXum6HZvM+n2kdy0Q0i7vGvGfcRErw/LDwmN77uW+7hTnvYpPNhSQoyF1DbWGCuex96ytV8LaXrN6t5dLdw3Kp5Znsb+e0d0zkKzQupYAkkBs4ycYya11UIgUZwowMkk/metC21G99DgPF3jvVtL8QXum6HZvM+n2kdy0Q0i7vGvGfcRErw/LDwmN77uW+7hTm/rPiLWdP16AzyW+kaK6wbbi50yW6812J3q0kcqi3x8ozIu3LdTyBs6r4W0vWb1by6W7huVTyzPY389o7pnIVmhdSwBJIDZxk4xk0y98I6TqN8Lq9F9KcqWhOpXHkPtxjdCJPLboM7lOcc5oXmEvIx9RufFh+JEOm6drOmQafLYSXKwz6W8pG141OWE6kt8xwRgAcFSean8Uw3beI/DkkzWM2njUUCQPbyCaObypf3iyrKFxjjaUPU89MbGr+HNO1uaCa9F0k1urLHNZ3s1q4VsEqWidSQSoOCSOBVy6sLa9a2a5j3tayiaE7iNrgFc8Hnhj19aOkfJ/rf8tA6y81+lvzON1jxT4ig8Tz6fpiaWtst0LRHuI5GcM1o0+84YAhdhG3jduxlMZOd4k+I1/pnhe01ixubPzzpUeo3GmjSrm6bDLu+aaJwsCnkBpFI4J5wa7mXw/pk1815Jbbp2mE5fzG++IjFnGcfcYjHTnPXms+58BeHbqOOKWylEKWi2bQx3kyRywqpVUkRXAkwCcFwSM042S1/rf/AIA5WurdtfW/+RmX934sf4iwafp2s6Zb6fNp73SwT6W8pXa8anLCZSW+Y4IwAOqk81V8W+OtU0vxJc6Tols0ktjYx3br/ZF3em6Zy4WJTBxD/qz877hluF+U10t74T0q/is1n+3q1lGYoZodSuIZthxlWlSQO4O1SdxOSM9adqvhbS9Zu47u7W6iuY4/KE9lfT2sjJnO1midSwByQGJAJOOppdEl5/rYUd23/W3/AAfvMTWvEus2GqW0spg0fRpIoWNzdaZLd7ncnejtHIv2faNvzyLt+brwRXS3/wDa32zT/wCy/sX2Xzj9u+0b9/lbTjytvG7dt+9xjPeqN94P0jU7pZr4X0oAQNCdSuRDKFxjfEJNj9Bncp3d81ev9F0/U7zT7u+t/Nn02Yz2rb2Xy3KlCcAgH5WIwcjmj/P8P6/rqLX8C9RRRQMKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooqlc6zpdlqVtp95qVnb3t3n7PbSzqsk2OuxSct+FAF2iis3U/Emh6Lcw2+s6zp+nz3H+piurpImk5x8oYgnnjigDSorP1DxBo2k3dta6rq1jY3F0dtvDc3KRvMcgYUMQW5IHHrTdT8SaHotzDb6zrOn6fPcf6mK6ukiaTnHyhiCeeOKAMO7n1LWfHV7o1vr9xosVjZwzpHZxQNLc+YzAuTNG42LtCjaByTk9Kpa5qmrm88STW2tTaZH4etUmigWKFku/3Rk3yl0LbCQU+RkPytz0I2/Edt4Pv72ws/F0Oh3N1MzLYwamkLu5OARGsnJP3c7fao9fbwRbapp7eKD4fhv4sGwOomFZUwRjyt/Iwcfd74pWbSS/r+uw7pO7/AK/ruZdlc+IPEXiHUIotam0eC2gsp0t4rWJ2DyIWdHLqcqcYwMMCOGHSu4rMu9T0LR9UiS+vdOsL/UmVIlmlSKW6I4AAJBcjcAAM9fetOn5kRTSs3qFFFFBQUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAV5d4kngt4vG+l6lJF/a+rSodHgYZluQIIli8perbJg5JH3DljjrXqNFCdncdzP8A7ZsYtbg0Sa4/4mU1s1ysIRvmjUhWbOMDlhxnPNcv461KDQr6PUrDXfI13yFjttFHlSHUx5nEfllfMOSSAyMAvU5AIruKKOqb6f1+QuljyP4i3Ri8T6qiala2Jl0yCKbSrxQW15N7t5EDE5V8F4/kDkmYZUEKa6HxrqkGi3EGqWOtG1102yJbaH+6dtRG/iPyyPMzkkbkYBTycgEV3dFC6X6O4PX+vT/I808S3VjZXXjOz1xoheaxaRx6XA6/vLtRDtWKLu7LMXOFyV3gnGRV3xdqC6AbTUIde8rxGtpHFFo6+VIdUO77gjIMnJLAMjALnJyAa76ikkk0/wCuv+YNtq39dP8AI8v+IcosrjXLO3utHmu9f01YEsp7jF8rqHWPyYQCZgWbIGUCsGbJycek2McsWn28dy++ZIlWRj/EwAyfzqeimtP6/ruD3/r+ugUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAf/9k=)

# Plot the relationship using facetwrap() between the Living Area and SalePrice by Neighborhhod to see the linear relationship between each of neighborhood

analysis\_filter%>%ggplot(aes(x=GrLivArea,y=SalePrice,col=Neighborhood))+geom\_point()+

geom\_smooth(method="lm")+facet\_wrap(~Neighborhood)+ylab("Sale Price of House")+xlab("Living Area(in Sq.Ft) of House")+

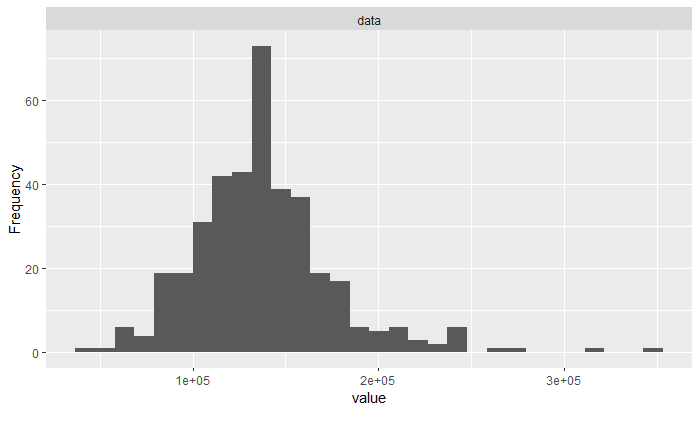
ggtitle("Plot for Relationship between Living Area and Sale Price of House for Edwards NAmes and BrkSide Neighborhood")



# Plot the histogram of SalePrice and Living Area to check Normality

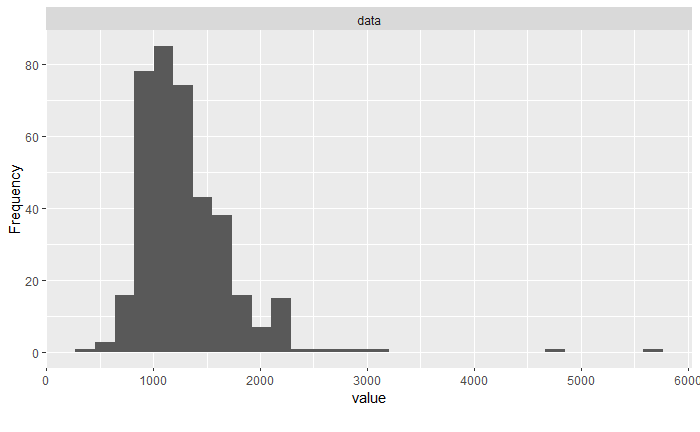
## The Sale Price seems to have some skeweness so we will use log(SalePrice) in creating model

plot\_histogram(analysis\_filter$SalePrice)



## The Living Area seems to have some skeweness so we will use log(GrLivArea) in creating model

plot\_histogram(analysis\_filter$GrLivArea)



#Linear Regression Model Between Sale Price and Living Area

## The Linear regression model has the Multiple-Rsqaure=0.4204 and adj-rsquare=0.4188 and rmse=0.2085

analysis.model=lm(log(SalePrice)~log(GrLivArea),data=analysis\_filter)

summary(analysis.model)

#Putting all plots in 3 rows and 3 columns

par(mfrow=c(2,3))

##Plot includes residuals and Standardized residuals vs fitted values, QQ plot

plot(analysis.model, bg = 'blue', pch=23)

#$Plot cook's distance to detect outliers

plot(cooks.distance(analysis.model), pch=23, bg='maroon', ylab="Cook's distance",

main = "Cook's Distance")

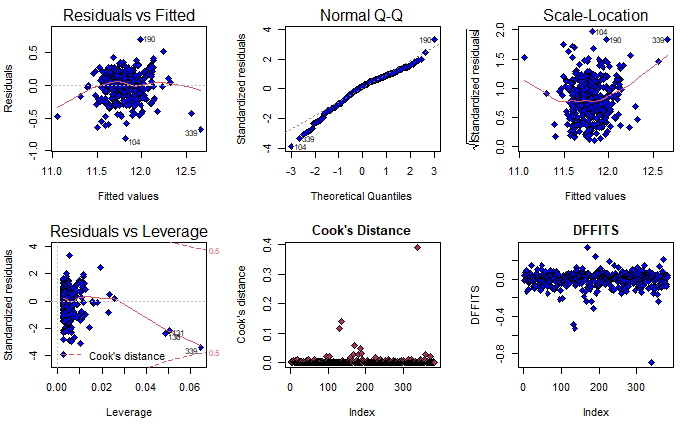
##Plot DFFITS to detect outliers

plot(dffits(analysis.model), pch=23, bg='blue', ylab = 'DFFITS', main = 'DFFITS')

# Determine which row has outlier values

analysis1.Outliers <- analysis\_filter[which(cooks.distance(analysis.model) > .05),] #View values for rows with a high cook's distance. This shows rows that could be outliers.

analysis1.Outliers



#Removing outlier with id=131

remove.outlier1=analysis\_filter[-131,]

dim(remove.outlier1)

remove\_model1=lm(SalePrice~GrLivArea,data=remove.outlier1)

summary(remove\_model1)#Multiple R-squared=0.3641 and Adj R-Squared=0.3624

#Removing outlier with id=136

remove.outlier2=analysis\_filter[-136,]

dim(remove.outlier2)

remove\_model2=lm(SalePrice~GrLivArea,data=remove.outlier2)

summary(remove\_model2)#Multiple R-squared=0.336 and Adj R-Squared=0.3342

#Removing outlier with id=169

remove.outlier3=analysis\_filter[-169,]

dim(remove.outlier3)

remove\_model3=lm(SalePrice~GrLivArea,data=remove.outlier3)

summary(remove\_model3)#Multiple R-squared=0.3278 and Adj R-Squared=0.326

#Removing outlier with id=339

remove.outlier4=analysis\_filter[-339,]

dim(remove.outlier4)

remove\_model4=lm(SalePrice~GrLivArea,data=remove.outlier4)

summary(remove\_model4)#Multiple R-squared=0.4138 and Adj R-Squared=0.4122

# Removing outliers all together

remove.outlier5=analysis\_filter[-c(131,136,169,339),]

dim(remove.outlier5)

remove\_model5=lm(SalePrice~GrLivArea,data=remove.outlier5)

summary(remove\_model5)#Multiple R-squared=0.4219 and Adj R-Squared=0.4204

# Keeping id=169 becuase it explains the model and removing the id=131,136,339

remove.outlier6=analysis\_filter[-c(131,136,339),]

dim(remove.outlier6)

remove\_model6=lm(SalePrice~GrLivArea,data=remove.outlier6)

summary(remove\_model6)#Multiple R-squared=0.4501 and Adj R-Squared=0.4487

# Keeping id=169,136 becuase it explains the model and removing the id=131,339.

remove.outlier7=analysis\_filter[-c(131,339),]

dim(remove.outlier7)

remove\_model7=lm(SalePrice~GrLivArea,data=remove.outlier7)

summary(remove\_model7)#Multiple R-squared=0.4573 and Adj R-Squared=0.4559

# So Looking at all the models after removing the outliers one by one and results of it, we decided to keep it the model 7

# which has removed the outlier of id=131 and id=339 yielding highest Adj R-Squared=0.4559 among all the models.

# Fitting the model after removing the influential outliers.

# Full Model (Additive Model)

fit.analysis1.fullmodel=lm(log(SalePrice)~log(GrLivArea)+relevel(as.factor(Neighborhood),ref="Edwards"),data=remove.outlier7)

summary(fit.analysis1.fullmodel)#Multiple R-squared=0.5041 and Adj R-Squared=0.5002

summary(fit.analysis1.fullmodel)$coefficients

confint(fit.analysis1.fullmodel)

press(fit.analysis1.fullmodel)

# Reduced Model (Interactive or Multiplicative Model)

fit.analysis1.reducedmodel=lm(log(SalePrice)~log(GrLivArea)\*relevel(as.factor(Neighborhood),ref="Edwards"),data=remove.outlier7)

summary(fit.analysis1.reducedmodel)#Multiple R-squared=0.5279 and Adj R-Squared=0.5216

summary(fit.analysis1.reducedmodel)$coefficients

confint(fit.analysis1.reducedmodel)

press(fit.analysis1.reducedmodel)

# It seems that the reduced model(interactive or multiplicative model) is working the best with Adj-Rsquared=0.5216)

|  |
| --- |
| Full Model |
| A picture containing text, receipt, screenshot  Description automatically generated |
| Reduced Model |
| Text  Description automatically generated |

par(mfrow=c(3,3))

##Plot includes residuals and Standardized residuals vs fitted values, QQ plot

plot(fit.analysis1.reducedmodel, bg = 'blue', pch=23)

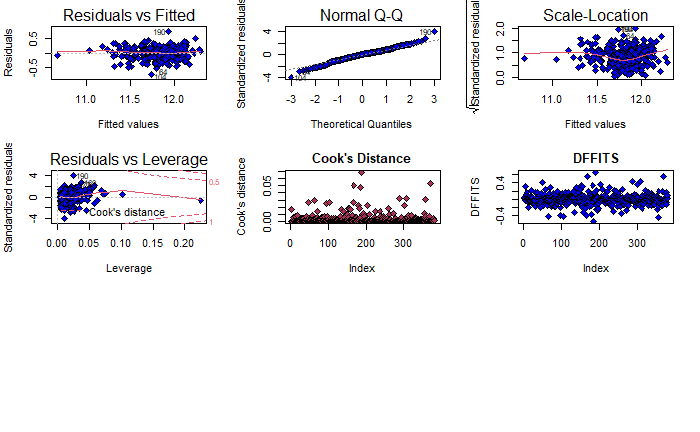
#$Plot cook's distance to detect outliers

plot(cooks.distance(fit.analysis1.reducedmodel), pch=23, bg='maroon', ylab="Cook's distance",

main = "Cook's Distance")

##Plot DFFITS to detect outliers

plot(dffits(fit.analysis1.reducedmodel), pch=23, bg='blue', ylab = 'DFFITS', main = 'DFFITS')



```

**# Analysis Question 2: Build the most predictive model for sales prices of homes in all of Ames Iowa. This includes all neighborhoods. Your group is limited to only the techniques we have learned in 6371 (no random forests or other methods we have not yet covered). Specifically, you should produce 4 models: one from forward selection, one from backwards elimination, one from stepwise selection, and one that you build custom. The custom model could be one of the three preceding models or one that you build by adding or subtracting variables at your will. Generate an adjusted R2, CV Press and Kaggle Score for each of these models and clearly describe which model you feel is the best in terms of being able to predict future sale prices of homes in Ames, Iowa.**

**## Cleaning the Train Dataset**

```{r}

dim(house\_train)

summary(house\_train$SalePrice)

# Check for the total number of columns that are character and numeric in type

numeric\_var\_train=sum(sapply(house\_train[,1:81],is.numeric))

numeric\_var\_train

char\_var\_train=sum(sapply(house\_train[,1:81],is.character))

char\_var\_train

# Check for the names of columns that are character and numeric in type

numeric\_varname\_train=which(sapply(house\_train[,1:81],is.numeric))

numeric\_varname\_train

char\_varname\_train=which(sapply(house\_train[,1:81],is.character))

char\_varname\_train

# Represent the total NA values as TRUE and FALSE

table(is.na(house\_train))

# Represent the columns having total NA values in table form

missing\_values\_train <- colSums(sapply(house\_train, is.na))

missing\_values\_train<- data.frame(house\_Variables\_train = names(missing\_values\_train), house\_NA\_Count\_train = missing\_values\_train); rownames(missing\_values\_train) <- c()

missing\_values\_train<- missing\_values\_train %>% filter(house\_NA\_Count\_train > 0)

kable(missing\_values\_train, "html") %>%

kable\_styling(full\_width = F)

length(missing\_values\_train$house\_Variables\_train)

sum(missing\_values\_train$house\_NA\_Count\_train)

# Substitute the NA with None in character

character\_impute\_train <- c("Alley", "MasVnrType","BsmtQual", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "FireplaceQu", "GarageType", "GarageFinish",

"GarageQual", "GarageCond", "BsmtCond","PoolQC","Fence","MiscFeature","Electrical")

house\_train[,character\_impute\_train] <- apply(house\_train[,character\_impute\_train], 2,

function(x) {

replace(x, is.na(x), "None")

}

)

# Substitute the NA with 0 in numeric

numeric\_impute\_train <- c("LotFrontage", "MasVnrArea","GarageYrBlt")

house\_train[,numeric\_impute\_train] <- apply(house\_train[,numeric\_impute\_train], 2,

function(x) {

replace(x, is.na(x),0)

}

)

summary(house\_train$LotFrontage)[4]

colnames(house\_train)

dim(house\_train)

summary(house\_train[,])[4]

# Unique values in each columns

for(i in colnames(house\_train)){

print(unique(house\_train[,i]))

}

# Converting character to integer

var\_facs\_train <- c("SaleCondition","SaleType","MiscFeature","Fence","PoolQC","PavedDrive","GarageCond","GarageQual","GarageFinish",

"GarageType","FireplaceQu","Functional","KitchenQual","Electrical","CentralAir","HeatingQC","Heating","BsmtFinType2",

"BsmtFinType1","BsmtExposure","BsmtCond","BsmtQual","Foundation","ExterCond","ExterQual","MasVnrArea","MasVnrType","Exterior2nd",

"Exterior1st","RoofMatl","RoofStyle","HouseStyle","BldgType","Condition2","Condition1","Neighborhood","LandSlope","LotConfig","Utilities",

"LandContour","LotShape","Alley","Street","MSZoning")

house\_train[,var\_facs\_train] <- lapply(house\_train[,var\_facs\_train] , factor, ordered = FALSE)

# Loop for converting character in integer

for(i in colnames(house\_train)){

house\_train[,i]=as.integer(house\_train[,i])

str(house\_train[,i])

}

# Removing the GrLivArea and TotalBsmtSF because they are totally correlated with X1stFlrSF,X2ndFlrSF, LowQualFinSF and BsmtFinSF1, BsmtFinSF2 and BsmtUnfSF respectively.

house\_train=house\_train%>%dplyr::select(-c(GrLivArea,TotalBsmtSF))

dim(house\_train)

colnames(house\_train)

# Plot the graphs and perform EDA

## we see some variables with some skewness so we will preceed with caution and try to do log transormation on those variables.

plot\_histogram(house\_train[,2:79])

# We don;t have any missing values now.

table(is.na(house\_train))

```

**## Cleaning of Test Dataset**

```{r}

# Read the test dataset from GitHub

house\_test=read.csv("https://raw.githubusercontent.com/RashmiAPatel19/6371-Statistics-Project/main/test.csv",header=TRUE)

head(house\_test)

dim(house\_test)

colnames(house\_test)

dim(house\_test)

# Check for the total number of columns that are character and numeric in type

numeric\_var\_test=sum(sapply(house\_test[,1:80],is.numeric))

numeric\_var\_test

char\_var\_test=sum(sapply(house\_test[,1:80],is.character))

char\_var\_test

# Check for the names of columns that are character and numeric in type

numeric\_varname\_test=which(sapply(house\_test[,1:80],is.numeric))

numeric\_varname\_test

char\_varname\_test=which(sapply(house\_test[,1:80],is.character))

char\_varname\_test

# Represent the total NA values as TRUE and FALSE

table(is.na(house\_test))

# Represent the columns having total NA values in table form

missing\_values\_test <- colSums(sapply(house\_test, is.na))

missing\_values\_test<- data.frame(house\_Variables\_test = names(missing\_values\_test), house\_NA\_Count\_test = missing\_values\_test); rownames(missing\_values\_test) <- c()

missing\_values\_test<- missing\_values\_test %>% filter(house\_NA\_Count\_test > 0)

kable(missing\_values\_test, "html") %>%

kable\_styling(full\_width = F)

length(missing\_values\_test$house\_Variables)

sum(missing\_values\_test$house\_NA\_Count)

# Substitute the NA with None in character

character\_impute\_test <- c("MSZoning","Utilities","Alley", "MasVnrType","BsmtQual", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "FireplaceQu",

"GarageType", "GarageFinish", "Exterior1st","Exterior2nd","KitchenQual","Functional","SaleType",

"GarageQual", "GarageCond", "BsmtCond","PoolQC","Fence","MiscFeature","Electrical")

house\_test[,character\_impute\_test] <- apply(house\_test[,character\_impute\_test], 2,

function(x) {

replace(x, is.na(x), "None")

}

)

# Substitute the NA with 0 in numeric

numeric\_impute\_test <- c("LotFrontage", "MasVnrArea","GarageYrBlt","BsmtFinSF1","BsmtFinSF2","BsmtUnfSF","TotalBsmtSF","BsmtFullBath","BsmtHalfBath",

"GarageCars","GarageArea")

house\_test[,numeric\_impute\_test] <- apply(house\_test[,numeric\_impute\_test], 2,

function(x) {

replace(x, is.na(x), 0)

}

)

colnames(house\_test)

dim(house\_test)

table(is.na(house\_test))

# Unique values in each columns

for(i in colnames(house\_test)){

print(unique(house\_test[,i]))

}

# Converting character to integer

var\_facs\_test <- c("SaleCondition","SaleType","MiscFeature","Fence","PoolQC","PavedDrive","GarageCond","GarageQual","GarageFinish",

"GarageType","FireplaceQu","Functional","KitchenQual","Electrical","CentralAir","HeatingQC","Heating","BsmtFinType2",

"BsmtFinType1","BsmtExposure","BsmtCond","BsmtQual","Foundation","ExterCond","ExterQual","MasVnrArea","MasVnrType","Exterior2nd",

"Exterior1st","RoofMatl","RoofStyle","HouseStyle","BldgType","Condition2","Condition1","Neighborhood","LandSlope","LotConfig","Utilities",

"LandContour","LotShape","Alley","Street","MSZoning")

house\_test[,var\_facs\_test] <- lapply(house\_test[,var\_facs\_test] , factor, ordered = FALSE)

# Loop for converting character in integer

for(i in colnames(house\_test)){

house\_test[,i]=as.integer(house\_test[,i])

str(house\_test[,i])

}

# Again checking for missing values for confirmation

table(is.na(house\_test))

# Removing the GrLivArea and TotalBsmtSF because they are totally correlated with X1stFlrSF,X2ndFlrSF, LowQualFinSF and BsmtFinSF1, BsmtFinSF2 and BsmtUnfSF respectively.

house\_test=house\_test%>%dplyr::select(-c(GrLivArea,TotalBsmtSF))

dim(house\_test)

# Plot the graphs and perform EDA

## we see some variables with some skewness so we will preceed with caution and try to do log transormation on those variables.

plot\_histogram(house\_test[,2:78])

```

**## Combining the train and test dataset by includeing one more column name of isTrainSet which has TrUe or False value**

```{r}

# Create the isTrainSet Column =True in train set

house\_train$isTrainSet=TRUE

# Create the isTrainSet Column =False in test set

house\_test$isTrainSet=FALSE

dim(house\_train)

dim(house\_test)

# Create a SalePrice Column with NA in test set

house\_test$SalePrice=rep(NA,1459)

# Combining the train and test set

house\_full=rbind(house\_train,house\_test)

dim(house\_full)

train=house\_full[house\_full$isTrainSet==TRUE,]

test=house\_full[house\_full$isTrainSet==FALSE,]

dim(train)

dim(test)

table(is.na(test))

```

**## Creating the Partition in Train data set**

```{r}

train.model=createDataPartition(y=train$SalePrice, p = 0.8, list = FALSE)

train.data <- train[train.model, ]

test.data <- train[-train.model, ]

dim(train.data)

dim(test.data)

str(test.data)

```

**## Checking for the Normality Collinearity and Outliers**

```{r}

# Checking for normality

## There is some evidence of skewness in some variable. We will do log transformation to handle this skewness.

plot\_histogram(train)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |

# Creating the model using all the variables

all.model=lm(log(SalePrice)~.-isTrainSet,data=train)

summary(all.model)# #Multiple R-squared=0.8874 and Adj R-Squared=0.8810

# Checking for Co-Linearity

par(mfrow=c(2,3))

#Plot includes residuals and Standardized residuals vs fitted values, QQ plot

plot(all.model, bg = 'blue', pch=23)

#Plot cook's distance to detect outliers

plot(cooks.distance(all.model), pch=23, bg='maroon', ylab="Cook's distance",

main = "Cook's Distance")

#Plot DFFITS to detect outliers

plot(dffits(all.model), pch=23, bg='blue', ylab = 'DFFITS', main = 'DFFITS')

# Determine which row has outlier values

analysis1.Outliers <- train[which(cooks.distance(all.model) > .05),] #View values for rows with a high cook's distance. This shows rows that could be outliers.

analysis1.Outliers# id=524,813,1183,1299,1424 are outliers

remove.outlier1=train[-1424,]

dim(remove.outlier1)

remove\_model1=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier1)

summary(remove\_model1)#Multiple R-squared=0.8907 and Adj R-Squared=0.8845

remove.outlier2=train[-1299,]

dim(remove.outlier2)

remove\_model2=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier2)

summary(remove\_model2)#Multiple R-squared=0.9077 and Adj R-Squared=0.9025

remove.outlier3=train[-1183,]

dim(remove.outlier3)

remove\_model3=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier3)

summary(remove\_model3)#Multiple R-squared=0.8868 and Adj R-Squared=0.8805

remove.outlier4=train[-813,]

dim(remove.outlier4)

remove\_model4=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier4)

summary(remove\_model4)#Multiple R-squared=0.8878 and Adj R-Squared=0.8815

remove.outlier5=train[-524,]

dim(remove.outlier5)

remove\_model5=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier5)

summary(remove\_model5)#Multiple R-squared=0.8958 and Adj R-Squared=0.8899

remove.outlier6=train[-c(1424,524),]

dim(remove.outlier6)

remove\_model6=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier6)

summary(remove\_model6)#Multiple R-squared=0.8993 and Adj R-Squared=0.8936

remove.outlier7=train[-c(1299,524),]

dim(remove.outlier7)

remove\_model7=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier7)

summary(remove\_model7)#Multiple R-squared=0.9201 and Adj R-Squared=0.9156

remove.outlier8=train[-c(1299,524,1424),]

dim(remove.outlier8)

remove\_model8=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier8)

summary(remove\_model8)#Multiple R-squared=0.9201 and Adj R-Squared=0.9156

dim(remove.outlier8)

remove.outlier9=train[-c(1299,524,1424,813),]

dim(remove.outlier9)

remove\_model9=lm(log(SalePrice)~.-isTrainSet,data=remove.outlier9)

summary(remove\_model9)#Multiple R-squared=0.9205 and Adj R-Squared=0.916 and RMSE=0.1155

par(mfrow=c(2,3))

#Plot includes residuals and Standardized residuals vs fitted values, QQ plot

plot(remove\_model9, bg = 'blue', pch=23)

#Plot cook's distance to detect outliers

plot(cooks.distance(remove\_model9), pch=23, bg='maroon', ylab="Cook's distance",

main = "Cook's Distance")

#Plot DFFITS to detect outliers

plot(dffits(remove\_model9), pch=23, bg='blue', ylab = 'DFFITS', main = 'DFFITS')

|  |  |
| --- | --- |
| **Before Removing Outliers** | **After Removing Outliers** |
|  |  |

```

**## Selecting the model 9 in which id=542,812,1299,1424 is removed which is performing best with lowest rmse and highest adj-Rsquared**

```{r}

fit.analysis2=remove.outlier9

fit.analysis2.model=remove\_model9

dim(fit.analysis2)

table(is.na(fit.analysis2))

summary(fit.analysis2.model)

```

**## Creating the model using forward variable selection**

**Multiple R-Squared=0.9179**

**Adjusted R-Squared=0.9155**

**Kaggle RMSE=0.13985**

**CV Press=20.53273**

```{r}

# Applying forward variable selection method

ols\_step\_forward\_aic(model = fit.analysis2.model,details = TRUE)

# Generating the model with the variables generated by forward variable selection method

forward.model=lm(log(SalePrice)~Condition2+Alley+LandContour+GarageArea+FullBath+HalfBath+FireplaceQu+

EnclosedPorch+MiscFeature+LotShape+Foundation+GarageCond+LowQualFinSF+ExterQual+YrSold

+WoodDeckSF+MasVnrType+PavedDrive+YearRemodAdd+ExterCond+MSZoning+BsmtQual+BsmtFinSF2+BsmtUnfSF+

Fireplaces+Street+BsmtFullBath+BldgType+ScreenPorch+KitchenQual+CentralAir+

Functional+KitchenAbvGr+LotArea+SaleCondition+GarageCars+BsmtFinSF1+OverallCond+OverallQual+YearBuilt+X2ndFlrSF

+X1stFlrSF,data=fit.analysis2)

summary(forward.model)

# Calculating the CV Press of the forward linear regression model

ols\_press(forward.model)

#Doing the prediction on partitioned test set

prediction=predict(forward.model,newdata=test.data)

prediction

#Performing inverse log transform

value=2.718^prediction

value

# Checking the RMSE of the model

rmse.model=rmse(test.data$SalePrice,value)

rmse.model

# Comparing the model predicted values with observed values

table=data.frame(Id=test.data$Id,ObsSalePrice=test.data$SalePrice,PredSalePrice=value)

table

# Predictions on the Original Test Set

predictiontest=predict(forward.model,newdata=test)

predictiontest

#Performing inverse log transform

pred\_value=2.718^predictiontest

pred\_value

# Putting the predicted values in a dataframe

output.df=data.frame(Id=test$Id, SalePrice=pred\_value)

head(output.df)

dim(output.df)

table(is.na(output.df))

# Putting the dataframe in a csv to submit on the kaggle to check the Score

write.csv(output.df,file="C:/Users/ARTH PATEL/Desktop/MSDS@SMU/6371-LSA/Stats Project/kaggle\_submission\_forward.csv",row.names = FALSE)

```

**## Creating the model using backward variable selection**

**Multiple R-Squared=0.9188**

**Adjusted R-Squared=0.9162**

**Kaggle RMSE=0.13700**

**CV Press=19.45764**

```{r}

# Applying backward variable selection method

backward.var=stepAIC(fit.analysis2.model,direction = "backward")

backward.var$anova

# Generating the model with the variables generated by backward variable selection method

back.model=lm(log(SalePrice) ~ MSZoning + log(LotArea) + Street + Alley + LotShape +

LandContour + Condition2 + BldgType + OverallQual + OverallCond +

YearBuilt + YearRemodAdd + Exterior1st + Exterior2nd + MasVnrType +

ExterQual + ExterCond + Foundation + BsmtQual + BsmtExposure +

log(BsmtFinSF1) + log(BsmtFinSF2) + log(BsmtUnfSF) + HeatingQC + CentralAir +

X1stFlrSF + X2ndFlrSF + LowQualFinSF + BsmtFullBath + FullBath +

HalfBath + KitchenAbvGr + KitchenQual + Functional + log(Fireplaces) +

FireplaceQu + GarageCars + GarageArea + GarageCond + PavedDrive +

log(WoodDeckSF) + log(EnclosedPorch) + ScreenPorch + MiscFeature +

YrSold + SaleCondition,data=fit.analysis2)

summary(back.model)

# Calculating the CV Press of the backward linear regression model

ols\_press(backward.model)

#Doing the prediction on partitioned test set

prediction=predict(back.model,newdata=test.data)

prediction

#Performing inverse log transform

value=2.718^prediction

value

# Checking the RMSE of the model

rmse.model=rmse(test.data$SalePrice,value)

rmse.model

# Comparing the model predicted values with observed values

table=data.frame(Id=test.data$Id,ObsSalePrice=test.data$SalePrice,PredSalePrice=value)

table

# Predictions on the Original Test Set

predictiontest=predict(back.model,newdata=test)

predictiontest

#Performing inverse log transform

pred\_value=2.718^predictiontest

pred\_value

# Putting the predicted values in a dataframe

output.df=data.frame(Id=test$Id, SalePrice=pred\_value)

head(output.df)

dim(output.df)

table(is.na(output.df))

# Putting the dataframe in a csv to submit on the kaggle to check the Score

write.csv(output.df,file="C:/Users/ARTH PATEL/Desktop/MSDS@SMU/6371-LSA/Stats Project/kaggle\_submission\_backward.csv",row.names = FALSE)

```

**## Creating the model using stepwise variable selection**

**Multiple R-Squared=0.9188**

**Adjusted R-Squared=0.9162**

**Kaggle RMSE=0.13926**

**CV Press=19.0847**

```{r}

# Applying stepwise variable selection method

stepwise.var=stepAIC(fit.analysis2.model,direction = "both")

stepwise.var$anova

# Generating the model with the variables generated by stepwise variable selection method

step.model=lm(log(SalePrice) ~ MSZoning + log(LotArea) + Street + Alley + LotShape +

LandContour + Condition2 + BldgType + OverallQual + OverallCond +

YearBuilt + YearRemodAdd + Exterior1st + Exterior2nd + MasVnrType +

ExterQual + ExterCond + Foundation + BsmtQual + BsmtExposure +

log(BsmtFinSF1) + log(BsmtFinSF2) + log(BsmtUnfSF) + HeatingQC + CentralAir +

X1stFlrSF + X2ndFlrSF + LowQualFinSF + BsmtFullBath + FullBath +

HalfBath + KitchenAbvGr + KitchenQual + Functional + log(Fireplaces) +

FireplaceQu + GarageCars + GarageArea + GarageCond + PavedDrive +

log(WoodDeckSF) + log(EnclosedPorch) + ScreenPorch + MiscFeature +

YrSold + SaleCondition,data=fit.analysis2)

summary(step.model)

# Calculating the CV Press of the backward linear regression model

ols\_press(stepwise.model)

#Doing the prediction on partitioned test set

prediction=predict(step.model,newdata=test.data)

prediction

#Performing inverse log transform

value=2.718^prediction

value

# Checking the RMSE of the model

rmse.model=rmse(test.data$SalePrice,value)

rmse.model

# Comparing the model predicted values with observed values

table=data.frame(Id=test.data$Id,ObsSalePrice=test.data$SalePrice,PredSalePrice=value)

table

# Predictions on the Original Test Set

predictiontest=predict(step.model,newdata=test)

predictiontest

#Performing inverse log transform

pred\_value=2.718^predictiontest

pred\_value

# Putting the predicted values in a dataframe

output.df=data.frame(Id=test$Id, SalePrice=pred\_value)

head(output.df)

dim(output.df)

table(is.na(output.df))

# Putting the dataframe in a csv to submit on the kaggle to check the Score

write.csv(output.df,file="C:/Users/ARTH PATEL/Desktop/MSDS@SMU/6371-LSA/Stats Project/kaggle\_submission\_stepwise.csv",row.names = FALSE)

```

**## Creating the model using custom variable selection**

**Multiple R-Squared=0.8596**

**Adjusted R-Squared=0.8568**

**Kaggle RMSE=0.13926**

**CV Press=34.379**

```{r}

# Generating the custom model

custom.model1=lm(log(SalePrice) ~ Neighborhood+GarageCars+SaleCondition+RoofStyle+CentralAir+Fireplaces+

X1stFlrSF\*X2ndFlrSF+ScreenPorch+BsmtFinSF1\*BsmtFinSF2+KitchenQual+BsmtFullBath\*BsmtHalfBath+

YearBuilt+PoolQC+HouseStyle+LotArea+BsmtFinType2\*BsmtFinType1+BsmtFinType1+Electrical+

Electrical\*CentralAir+GarageFinish+GarageYrBlt+GarageType,data=fit.analysis2)

summary(custom.model1)

# Calculating the CV Press of the custom linear regression model

ols\_press(custom.model1)

#Doing the prediction on partitioned test set

prediction=predict(custom.model,newdata=test.data)

prediction

#Performing inverse log transform

value=2.718^prediction

value

# Checking the RMSE of the model

rmse.model=rmse(test.data$SalePrice,value)

rmse.model

# Comparing the model predicted values with observed values

table=data.frame(Id=test.data$Id,ObsSalePrice=test.data$SalePrice,PredSalePrice=value)

table

# Predictions on the Original Test Set

predictiontest=predict(custom.model,newdata=test)

predictiontest

#Performing inverse log transform

pred\_value=2.718^predictiontest

pred\_value

# Putting the predicted values in a dataframe

output.df=data.frame(Id=test$Id, SalePrice=pred\_value)

head(output.df)

dim(output.df)

table(is.na(output.df))

# Putting the dataframe in a csv to submit on the kaggle to check the Score

write.csv(output.df,file="C:/Users/ARTH PATEL/Desktop/MSDS@SMU/6371-LSA/Stats Project/kaggle\_submission\_custom.csv",row.names = FALSE)

```