Lasso & Stepwise Model

Untitled

09/02/2021

Contents

Theoretical Bases

Stepwise Regression

Stepwise regression is an iterative construction of a regression model that involving the selection of relevant independent variables to be utilized in a final model. This method examines the statistical significance of each independent variable in a linear regression model. This form of regression is mostly used when numerous independent variables produce insignificant outcomes because of the small sample size or high correlation between independent variables.

Stepwise regression can be undertaken either by regressing each independent variable at a time and including it in the final regression model if it is statistically significant or selecting all the relevant independent variables and including them in the model, only to eliminate those that are not statistically significant. Alternatively, an amalgamation of both can be applied. This creates three methods for the stepwise regression:

1.Forward selection: Each independent variable is tested and included in the model if it is statistically significant. This process is repeated till all significant variables are captured

2.Backward elimination: Here we start with a group of independent variables, remove one at a time then test to see if the removed variable was statistically significant

3 Bidirectional elimination: This combines both forward selection and backward elimination to determine which independent variable is included or excluded from the final model.

Lasso Regression

"LASSO" denotes Least Absolute Shrinkage and Selection Operator. Lasso regression is a regularization technique that is used over linear regression for better predictions. It does this through shrinkage. This is where data values are shrunk towards a central point like the mean. With lasso regression, a penalty term is introduced which penalizes the absolute size of the regression coefficients. By curtailing the sum of the absolute value of the estimates, some coefficients can become zero, thus removing them from the model. A bigger penalty will further cause more shrinkage towards zero. This is useful when you wish to select variables for a model and helps in situations where there are high levels of multicollinearity. The lasso regression can be represented by:

Residual Sum of Squares + Lambda * (Sum of the absolute value of the magnitude of coefficients)

. Lambda demotes the amount of shrinkage

. When lambda is zero, all variables are maintained and the estimate is equivalent to that of linear regression.

. As lambda increases more coefficients will be set to zero and in effect eliminate the feature or variable. Theoretically, when lambda is equal to infinity all the coefficients will be removed.

. As lambda increases bias increases

Data Presentation

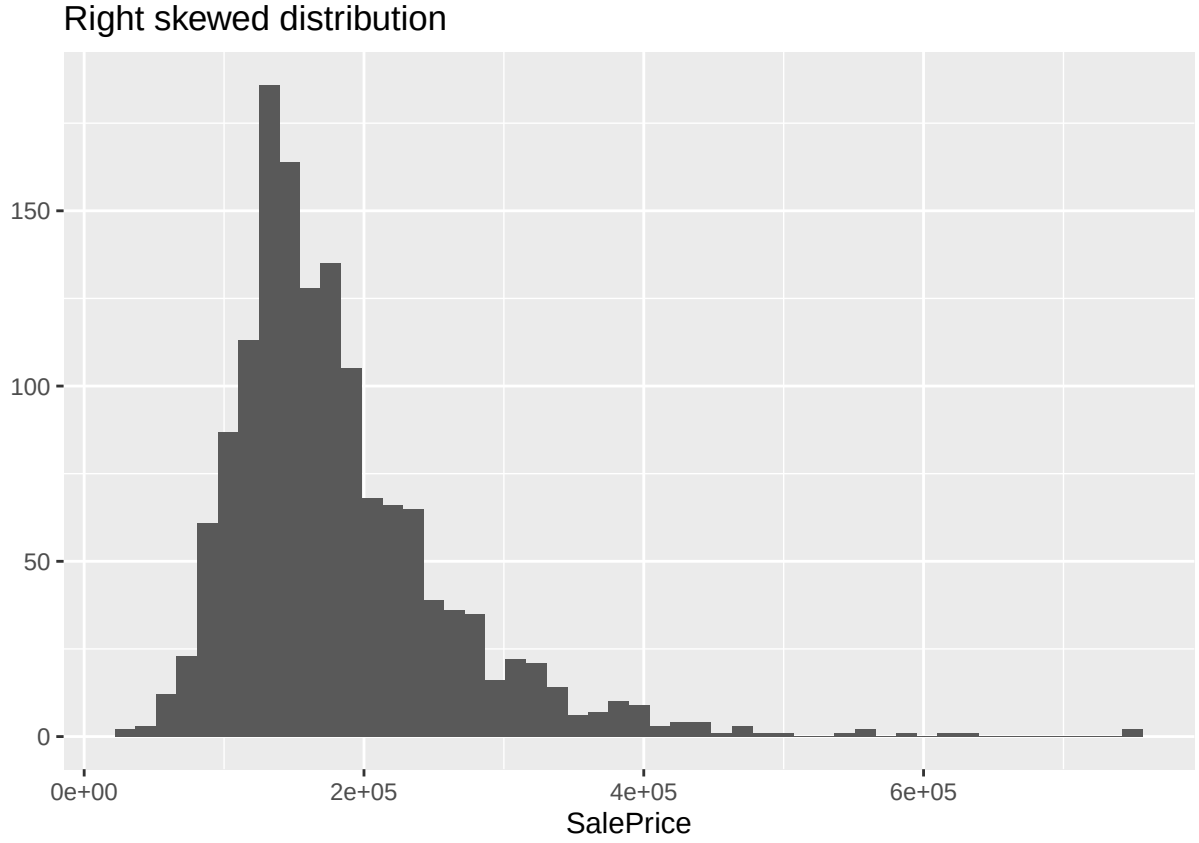
The combination of the test and train set reveals the dataset has **2917** observations and **81** variables with a mixture of integer, character and numerical values.

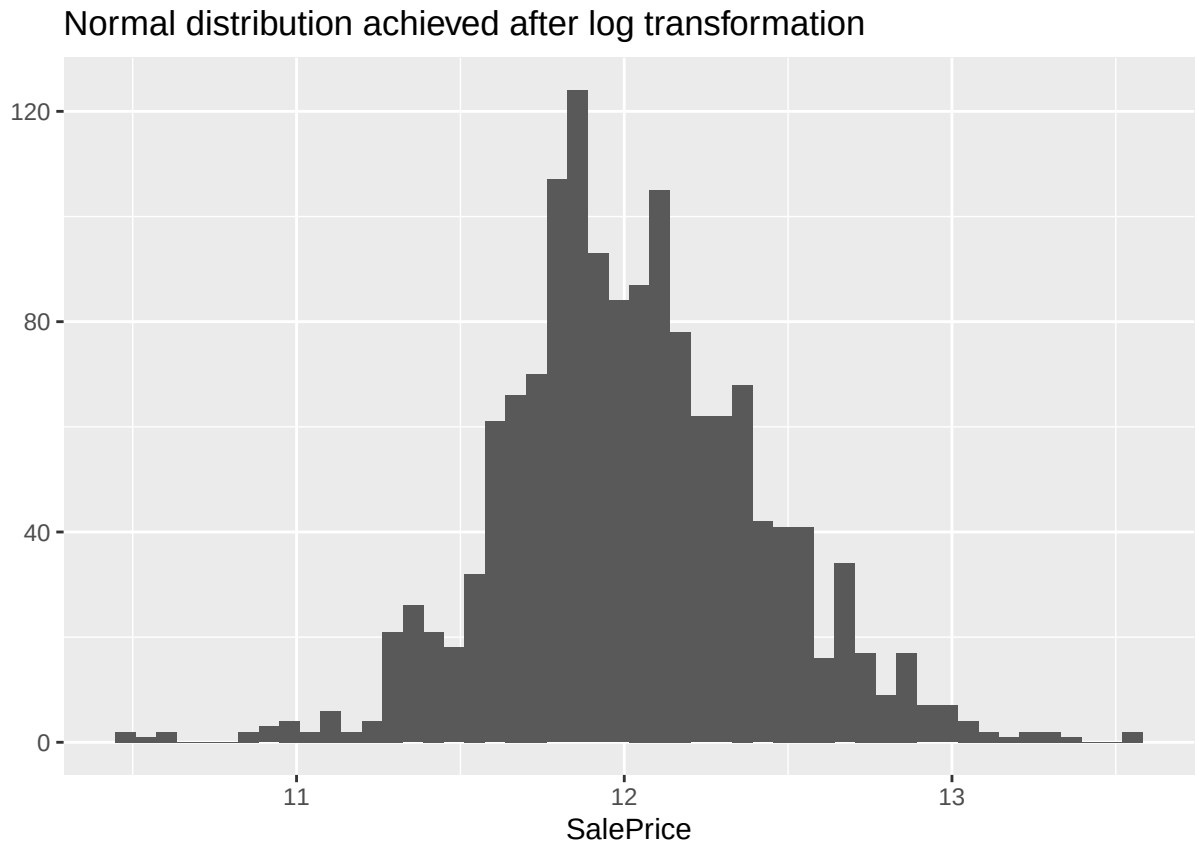
```
## 'data.frame':    2919 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr  NA NA NA NA ...
##  $ LotShape     : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr  "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr  "2Story" "1Story" "2Story" "2Story" ...
##  $ 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 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr  "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr  "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr  "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr  "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr  "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : chr  "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
```

```
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr  "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr  NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : chr  "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr  NA NA NA NA ...
##  $ Fence        : chr  NA NA NA NA ...
##  $ MiscFeature  : chr  NA NA NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900
```

Data Pre-processing

The GrLivArea variable was the most correlated numeric independent variable to Salesprice, therefore outliers were removed to ensure they don't adversely impact correlation. Additionally, the distribution of the dependent variable needed to be normal to allow for inferences to be made, thus the SalesPrice variable was logged to ensure its normality.





The dataset has a total of 1459 missing figures which will be filled through imputation. For a number of the categorical variables, NA values will be replaced with "None" since these variables represent the absence of Basements, Garages and others, however for some categorical variables the N/A values will be replaced with the most occurring value. For most of the numerical variables, N/A will be replaced with zero. We will also specify the order of the levels while converting categories to integer ranks. Dummy categorical variables will also be encoded, with some skewed features being transformed.

Objective

The goal of this study is the utilization of the Lasso and Step-wise model to make predictions. However before predictions are made with the test set, a validation set will be obtained from the train set. With this we can perform cross-validation and evaluate the model performance with the root mean squared error(RMSE), The RMSE is the square root of the mean of the square of all the errors. It can be calculated by squaring the residuals, finding the average of the residuals and taking the square root of the result. It measures the difference between the predicted and actual values, with a lower RMSE value being better than a higher value.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Once the root mean squared error is computed, the model with the lowest one will be selected, predictions will be made from the test set and the results will be saved and submitted.

Modelling

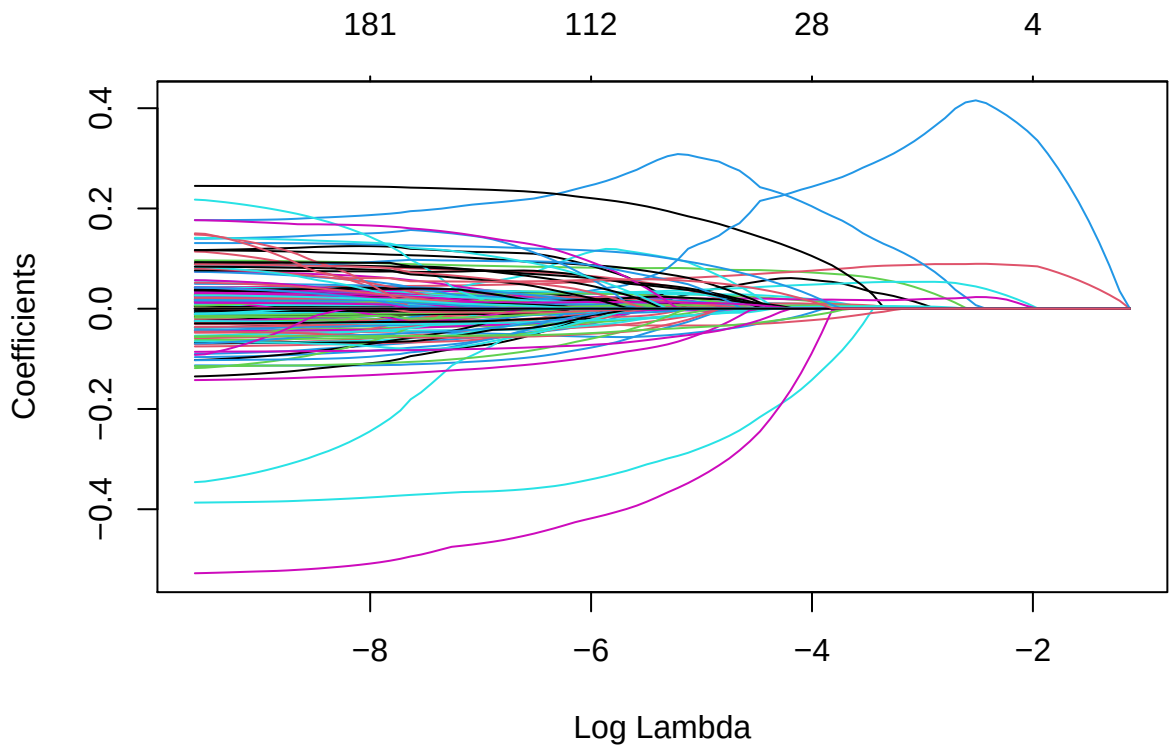
Lasso Model

The first model we will utilize is the lasso model. For now, we will test our model against a validation set. This will enable the evaluation of model performance through the root mean squared error for later comparison with the step-wise model.

```
set.seed(76)
model_lasso = glmnet(as.matrix(training[, -59]),
                             training[, 59])
```

A visual inspection of the model demonstrates how the lasso regression works. As we can see certain coefficients reduce rapidly as lambda is increased and is ultimately zeroed out. This clearly illustrates that certain parameters perform better than others.

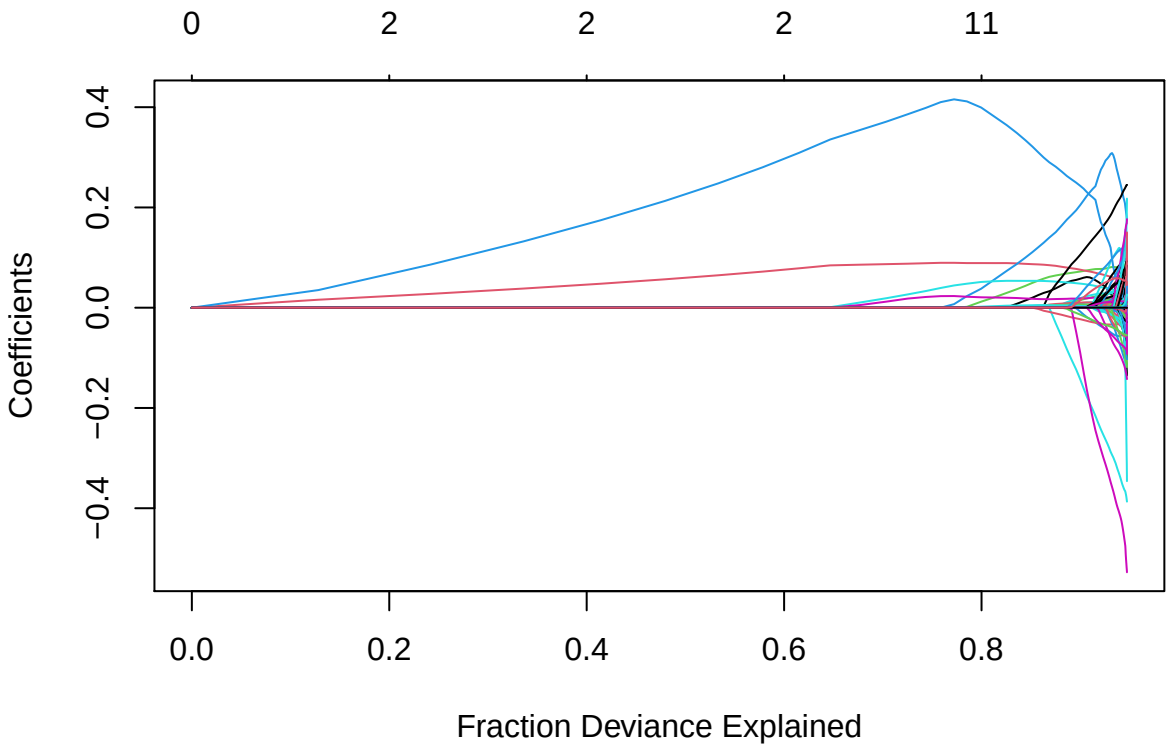
```
plot(model_lasso, xvar = 'lambda')
```



Again we see that about 60% of the variability is being explained by just two variables, with 80% percent being explained by 10 variables. The gain in fraction deviance explained by the remaining variables isn't significant as the rapid growth of such variables at the tail end

of the plot illustrates overfitting, thus more emphasis must be placed on the few variables that had steady growth. As stated earlier, this shrinkage and zeroing out make the lasso regression amazing at feature selection.

```
plot(model_lasso, xvar = 'dev')
```



Finally, we will cross validate the lasso model with the cv.glmnet function then make predictions with the validation set.

```
# cross validate lasso for better model performance
set.seed(76)
model_lasso_cv = cv.glmnet(as.matrix(training[, -59]),
                           training[, 59])

# Prediction with the validation set
pred_las <- predict(model_lasso_cv,
                    newx = as.matrix(validation[, -59]), # Column 59 is the dependent varia
                    s = "lambda.min")
```

Stepwise Model

Next is the stepwise model. Just like before we will train and test our model on a validation set. We will then store the predictions for model evaluation through the root mean squared error.

For the method, we will employ the forward approach where each independent variable will be tested and included in the model if it is statistically significant. This will be done till all significant variables are captured.

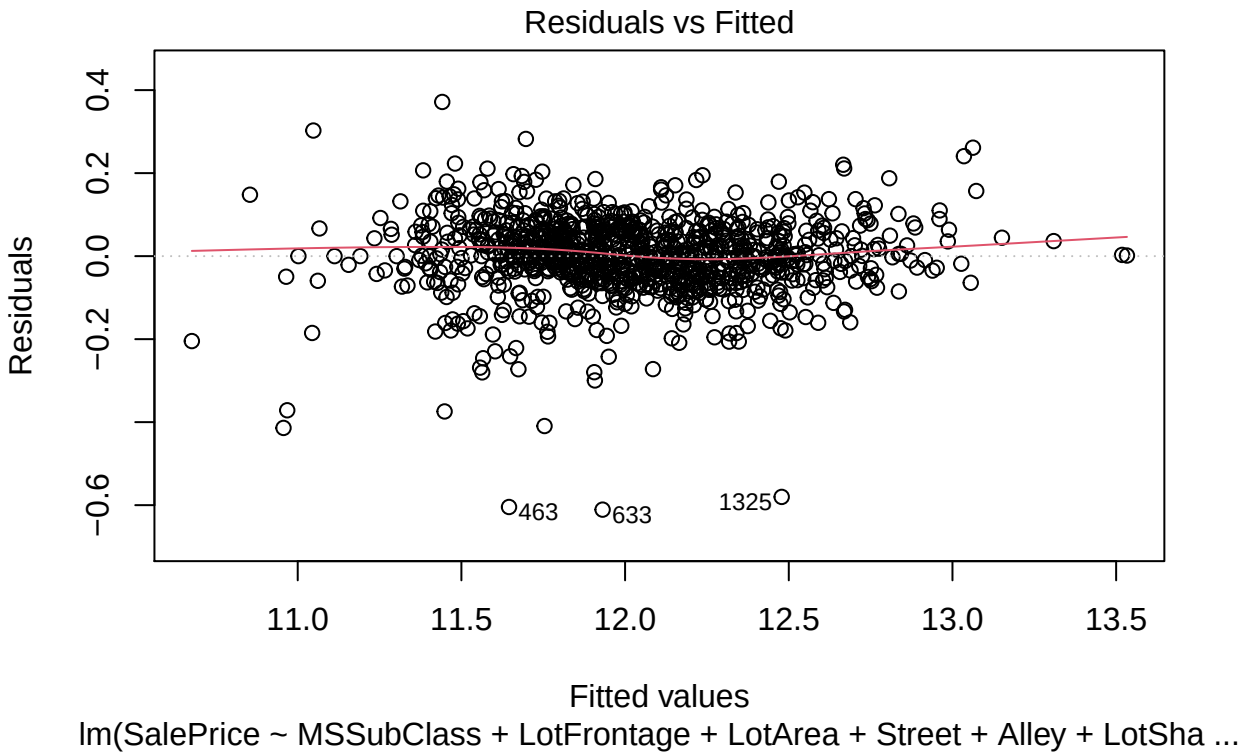
```
set.seed(76)
model_step <- step(lm(formula = SalePrice ~ .,
                      data = training),
                   direction = "forward")
```

Before we make predictions with the stepwise model. Some assumptions of the stepwise regression should be assessed.

The regression model is linear

The horizontal line shows an almost linear relationship between the residuals and the fitted values. Therefore the regression model is linear

```
plot(model_step, 1)
```



The mean of residuals is zero

the mean of residuals is approximately zero, this assumption holds true.

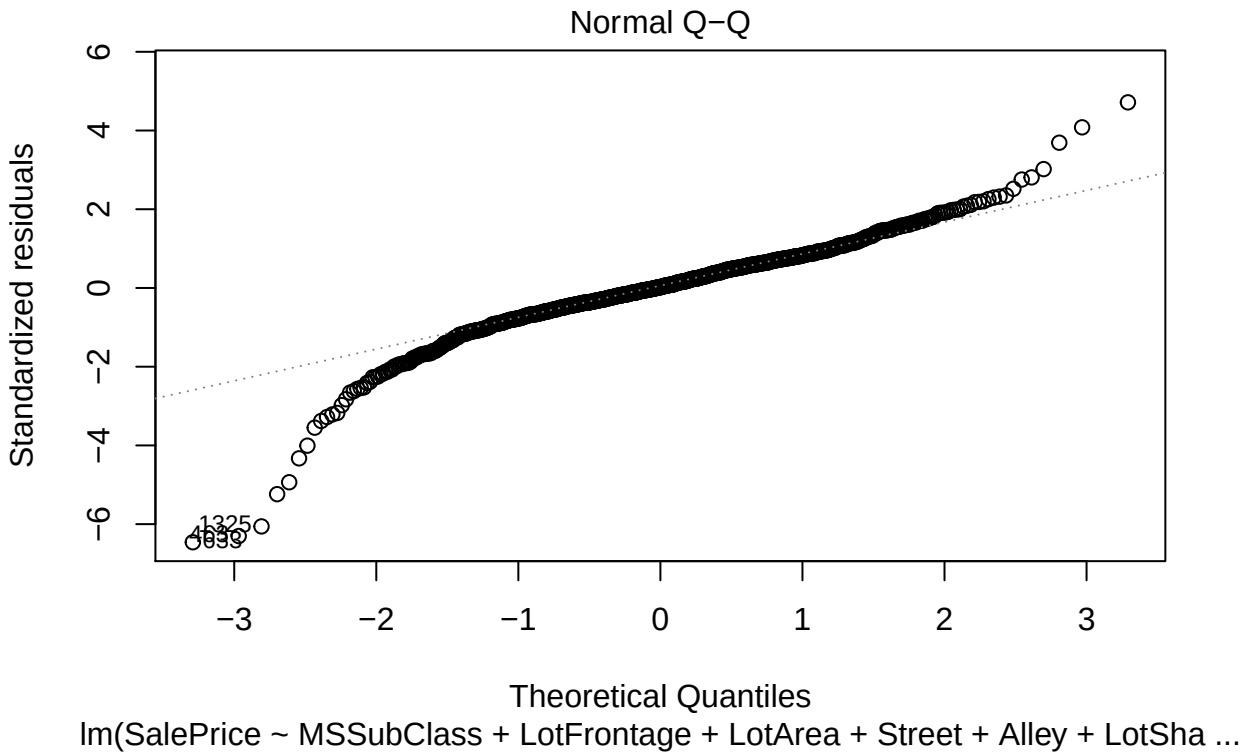
```
format(mean(model_step$residuals),
       scientific=F)
```

```
## [1] "0.00000000000000000004490681"
```

Residuals are normally distributed

The standardized residuals of the plot don't follow the normal QQ line. Therefore the residuals are not distributed.

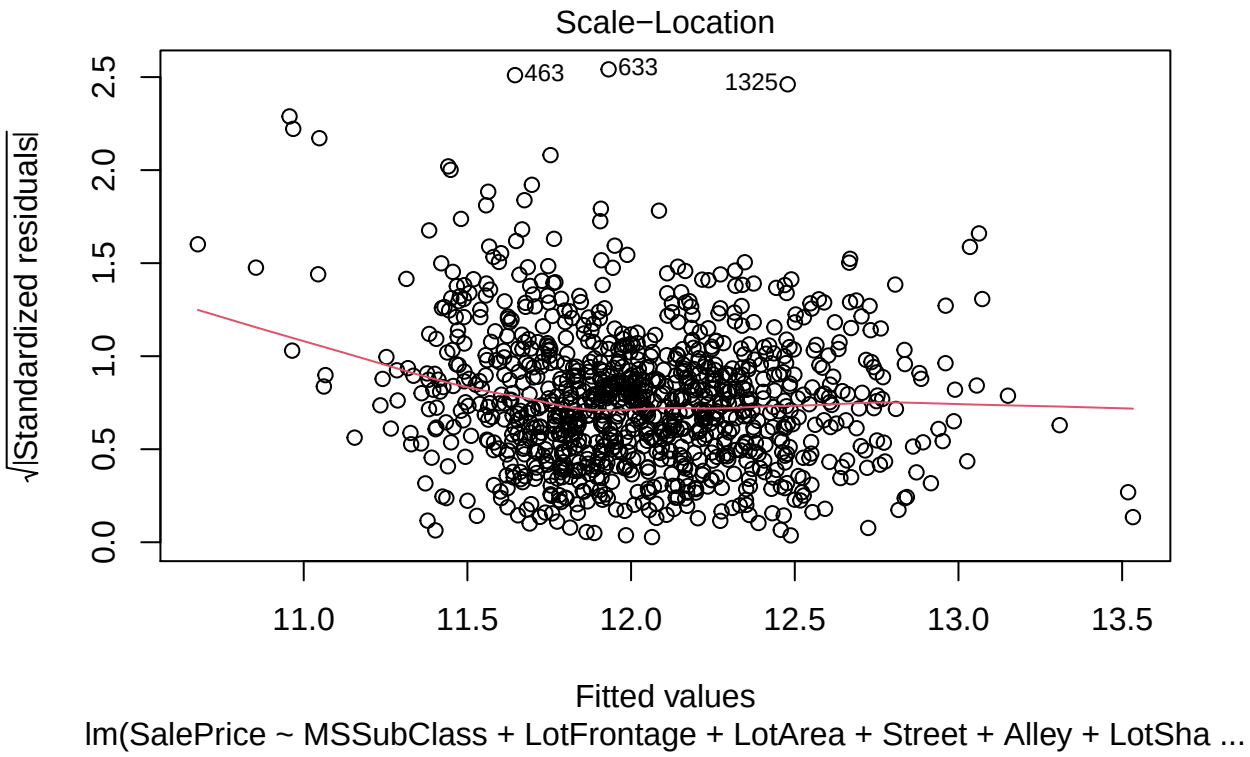
```
plot(model_step, 2)
```



Homoscedasticity of residuals

From the plot, the residuals are not homoscedastic. It is therefore heteroscedasticity.

```
plot(model_step, 3)
```



Autocorrelation of residuals

The Durbin-Watson test can be used to check for the autocorrelation of the residuals.

The null hypothesis H0: Residuals are autocorrelation

and the alternate hypothesis H1: Residuals are not autocorrelation

The Durbin-Waston test presents a D-W Statistic of 1.925 (close to 2.0) and a p-value of $>$ 0.05, thus, we do not reject Ho: Residuals are not autocorrelated.

```
durbinWatsonTest(model_step)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.01371545      1.970556     0.6
##  Alternative hypothesis: rho != 0
```

we will now save the predictions for model comparison later on.

```
pred_step <- predict(model_step, newdata = validation)
```

Comparison of results and Conclusion

We have trained and tested both models on a validation set and now we will determine the best performing model with the lowest root mean squared error.

```
lasso_rmse<-rmse(validation$SalePrice, pred_las)

step_rmse<-rmse(validation$SalePrice, pred_step)

rmse_results <- data_frame(method = "Lasso Model",
                           RMSE = lasso_rmse)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Stepwise Model",
                                     RMSE = step_rmse ))

rmse_results %>% knitr::kable()
```

method	RMSE
Lasso Model	0.1042815
Stepwise Model	0.1153647

From our cross-validation, the worse performing model was the Stepwise model with a root mean squared error of approximately **0.1154**. The Lasso model on the other hand achieved a root mean squared error of approximately **0.1043**, making it our most powerful model.

We will now retrain the lasso model on the whole training set, make final predictions and write them for submission.

```
# Retraining on training set
set.seed(76)
final_lasso = cv.glmnet(as.matrix(train[, -59]),
                        train[, 59])

## Predictions
pred_lasso = data.frame(exp(predict(final_lasso,
                                    newx = as.matrix(test[, -59]),
                                    s = "lambda.min")) - 1)

# Saving lasso prediction results
df<-data.frame(Id=test_id, SalePrice=pred_lasso$X1)
head(df)
```

```
##     Id SalePrice
```

```
## 1 1461  119720.5
## 2 1462  163305.6
## 3 1463  183102.3
## 4 1464  201560.3
## 5 1465  200790.4
## 6 1466  170478.0
```

```r
write.csv(df, "results_lasso.csv", row.names = FALSE)
```