Statistical analysis on Ames House Prices

Supervised Learning Algorithms

Professor:

Silvia Salini

Presenter:

Shojaat Joodi Bigdilo

Student number: 14088A

June 26, 2023

List of contents:

1. Introduction

The following analysis takes place on the **Ames House Price Estimation dataset** was compiled by Dean De Cock for use in data science education The goal of this analysis is to explore the possibility of exploiting the data by employing **supervised statistical learning** techniques with the **goal** of ask a house buyer to describe their dream house and give them the approximate price of it.

2. Data Collection

The dataset analyzed in this study has been uploaded under the name ”House Prices - Advanced Regression Techniques” on the Kaggle[1] platform.

In this dataset, there are 1460 observations with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. Among explanatory variables, there are 37 integer variables, such as Id, MSSubClass, LotFrontage, and 43 factor variables, such as MSZoning, Street, LotShape. Descriptive analysis and quantitative analysis will be used. Target variable (SalePrice) has a continuous value.

The dataset is composed by features the following variables:

- **SalePrice** - the property's sale price in dollars. This is the **target variable** that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation

[1] https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement
- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area
- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **1stFlrSF**: First Floor square feet
- **2ndFlrSF**: Second floor square feet
- **LowQualFinSF**: Low quality finished square feet (all floors)
- **GrLivArea**: Above grade (ground) living area square feet
- **BsmtFullBath**: Basement full bathrooms
- **BsmtHalfBath**: Basement half bathrooms
- **FullBath**: Full bathrooms above grade
- **HalfBath**: Half baths above grade
- **Bedroom**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **WoodDeckSF**: Wood deck area in square feet
- **OpenPorchSF**: Open porch area in square feet
- **EnclosedPorch**: Enclosed porch area in square feet
- **3SsnPorch**: Three season porch area in square feet
- **ScreenPorch**: Screen porch area in square feet
- **PoolArea**: Pool area in square feet
- **PoolQC**: Pool quality
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

3. Data Preprocessing

This dataset has not been cleaned and needs to be preprocessed to handle missing or incoherent values. In this analysis, dataset separated in two sub-dataset (categorical_data sub-dataset and numeric_data sub-dataset) in order to implement different methods on each type of data, then after some operation on them they are concatenated together for model implementation.

All steps of preprocessing:

3.1. Discovering original type of each feature

For discovering original type of each feature in the dataset in order to manipulate them in the next steps, we applied three following steps:

Step1: Discovering features with numerical value in dataset while their original type is categorical

Step2: Discovering features with categorical value in dataset while their original type is numerical

Step3: Separating rest Numerical features and Categorical features

Description of each Step:

Step1: Discovering features with numerical value in dataset while their original type is categorical:

In this dataset exist some features that have numerical value, but actually their type are categorical data because already they are encoded in original dataset. Therefore first, before any operation on numerical data these features are separated from numerical data and named “original_cat_dataset” sub-dataset, then it will be added to categorical sub-dataset; when already categorical sub-dataset encoded.

These variables are:

TotalBath: New created features which displays total number of Bath
OverallQual: Overall material and finish quality
OverallCond: Overall condition rating
BedroomAbvGr: Number of bedrooms above basement level
TotRmsAbvGrd: Total rooms above grade
Fireplaces: Number of fireplaces
GarageCars: Size of garage in car capacity. number of cars that can park
MoSold: month sold
KitchenAbvGr: Number of Kitchens above grade

Step2: Discovering features with continues value in dataset while their original type is categorical:

In dataset the value of 'MSSubClass' (type of dwelling involved in the sale) is numeric and continuous, but originally its type is categorical data, therefore, first it is converted to categorical for some operation. Finally, in encoding step it will be encoded to numerical value.

Step3: Separating Numerical features and Categorical features:
In this step, two different sub-dataset for all **numerical** features and all **categorical** features are created in order to handle missing values and other operations on them with their specific methods.

3.2. Missing Data

For handling missing data, exists two methods: **imputing** or **removing** missing values. In this analysis three following steps implemented:

3.2.1. Step1: handling Not exist data

Some missing data actually are not missing data. For example, some features like “FireplaceQu” (Fireplace quality) have value of “NA” in dataset which means 'No Fireplace'. Therefore, these kinds of features are imputed manually and their “NA” value is converted to 0 and for rest category 1, 2, 3 respectively.

These variables are:
"GarageFinish": “NA” in dataset means 'No Garage'
"GarageQual": “NA” in dataset means 'No Garage'
"GarageCond": “NA” in dataset means 'No Garage'
"BsmtCond": “NA” in dataset means 'No Basement’
"BsmtQual": “NA” in dataset means 'No Basement’
“FireplaceQu” : “NA” in dataset means 'No Fireplace'

3.2.2. Step2: Removing features with more than 50 percent missing values

Finding missing data and removing features that have more than 50 percent missing values, by setting the threshold percentage. The result shows that features named "Alley", "PoolQC", "Fence", and "MiscFeature", which all of them are categorical features, and have more than 50 percent missing values, therefore these are removed from the dataset.

By eliminating these features with a high proportion of missing data, the dataset was streamlined and prepared for further analysis. This process ensured that only features with a reasonable amount of complete information were retained, thus minimizing the potential biases associated with missing data.

3.2.3. Step3: Imputing features with less than 50 percent missing values

For remaining features which have features with less than 50 percent missing values, implemented **Mean** imputation for **'Numerical'** data and **Mode** imputation for **'Categorical'** data.

3.3. Feature Combination

In this step some features that represent **same meaning** are combined together, then previous features are deleted from dataset in order to avoid **dimensionality** issue.

- "**TotalBath**": Creating Total number of bathrooms by Combination of four following features.
  "BsmtFullBath": basement full bathrooms
  "BsmtHalfBath": basement half bathrooms
  "FullBath": Full bathrooms above grade
  "HalfBath": Half baths above grade

- "**AllPorchSF**": Creating total Square Feet of Porch area by Combinations of four following features
  OpenPorchSF: Open porch area in square feet
  EnclosedPorch: Enclosed porch area in square feet
  X3SsnPorch: Three season porch area in square feet
  ScreenPorch: Screen porch area in square feet

4. Operation on Numerical data

4.1. Outliers' Detection

Outliers[2] are extreme values that deviate significantly from the majority of the data points. They can have a substantial impact on the statistical properties of the data, such as the mean and standard deviation.

In this analysis, outliers are detected in two ways for some important features, then they are treated in two different scenarios in order to analyze algorithms in two different scenarios and compare their results.

Hint: First of all, the 'categorical_data', 'numeric_data', 'original_cat_dataset' sub-datasets, and 'SalePrice' column concatenated, because when we delete one value from **numerical** dataset during outliers elimination, it deletes just it's row in that dataset, therefore we need to merge all subsets to delete this row from all of them.

4.1.1. Two Steps for detecting outliers:

Step1: Using a **function** which uses **Z-score** method to detect outliers by setting upper and lower limit to 3 standard deviations distance from Mean of data.

Step2: Detecting outliers of some important features Based on scatter plot of every 'feature value' and 'target value'.

4.1.2. Two Steps for handling outliers:

[2] Link: https://towardsdatascience.com/outlier-why-is-it-important-af58adbefecc

This section is implemented in two different scenarios:

Scenario 1: In first scenario, outliers are detected, but none of them removed from dataset, and all algorithms are implemented without removing outliers.

Scenario 2: In second scenario, all algorithms are implemented with removing some outliers of some features.

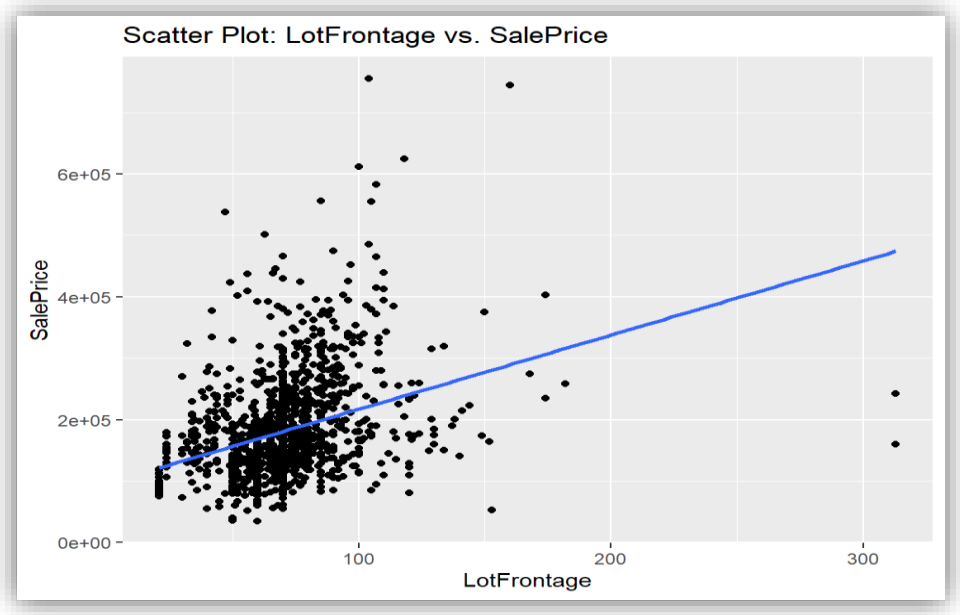
In this scenario, in order to ensure the integrity and accuracy of our analysis, we implemented a rigorous outlier removal process on the dataset. Outliers, which are extreme values that deviate significantly from the majority of observations, can distort the relationship between variables and lead to biased or unreliable results. We employed Z-score technique to identify and remove outliers from the dataset, specifically focusing on variables such as 'GarageArea', 'LotArea', 'GrLivArea', and 'LotFrontage'. By removing these outliers, we aimed to create a more reliable and representative dataset for analysis, minimizing the potential influence of extreme values on our results.

Result of outlier:

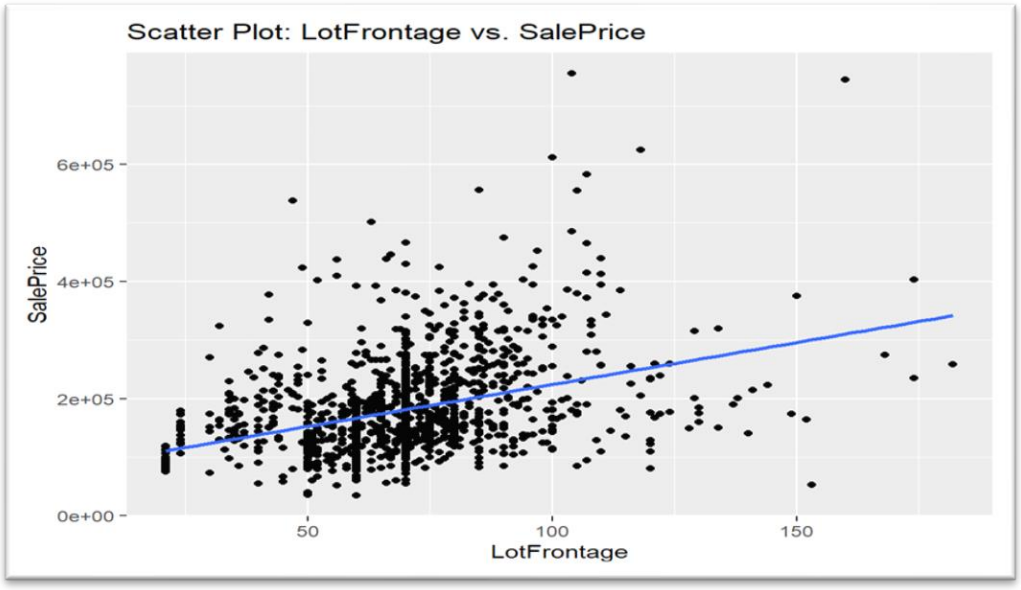
Based on the results of the outlier detection analysis, it was found that many of the numerical features in the dataset contain outliers. Among these variables, the features "LotFrontage," "LotArea," "GarageArea," and "GrLivArea" were identified as important variables with outliers that require further attention.

LotFrontage: Linear feet of street connected to property

- Ploting the "Scatter Plot" for **"LotFrontage"** **<u>before</u>** deleting outliers



- Ploting the "Scatter Plot" for "LotFrontage" **<u>After</u>** deleting outliers



To investigate the outliers in more detail, a scatter plot was created specifically for the "LotFrontage" feature. The scatter plot revealed interesting findings. Despite the initial indication from the outlier detection **function** that there were **15 outliers**, the **scatter plot** showed **two** data points that had the exact same value of **313**. These two data points were considered as outliers and were chosen to be removed from the dataset.

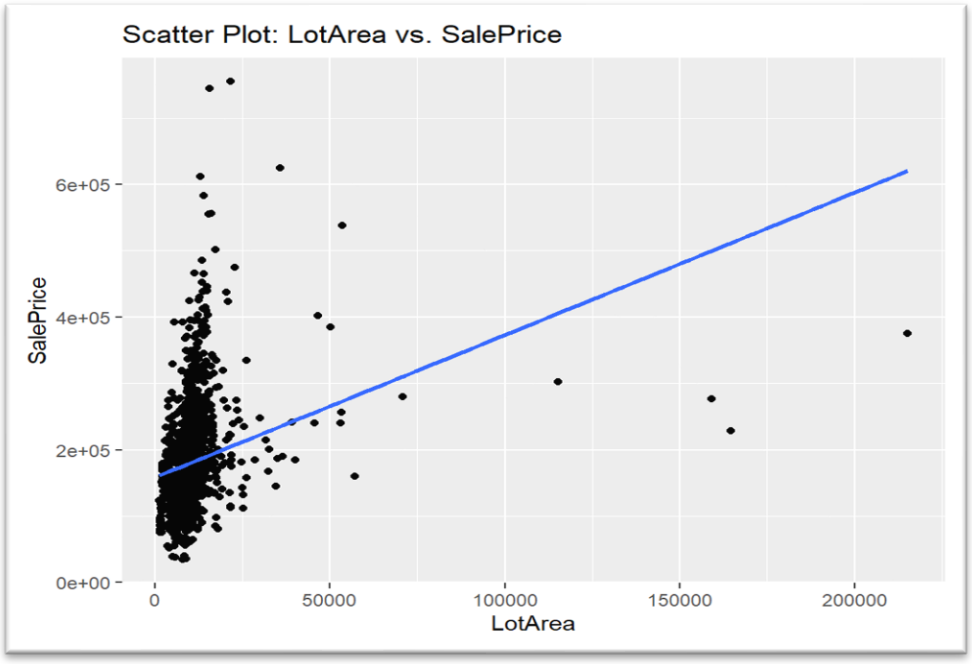
This discovery emphasizes the importance of visualizing the data and manually inspecting outliers, as automated detection methods may not capture all anomalies accurately. By identifying and removing these specific outliers in the "LotFrontage" feature, we ensure that our subsequent analysis and modeling are based on a cleaner and more reliable dataset.

By removing these two outliers while keeping the other 13 outliers, we strike a balance between ensuring the dataset's integrity and retaining potentially valuable information. Deleting all outliers could lead to a loss of important insights or patterns that exist in the data.

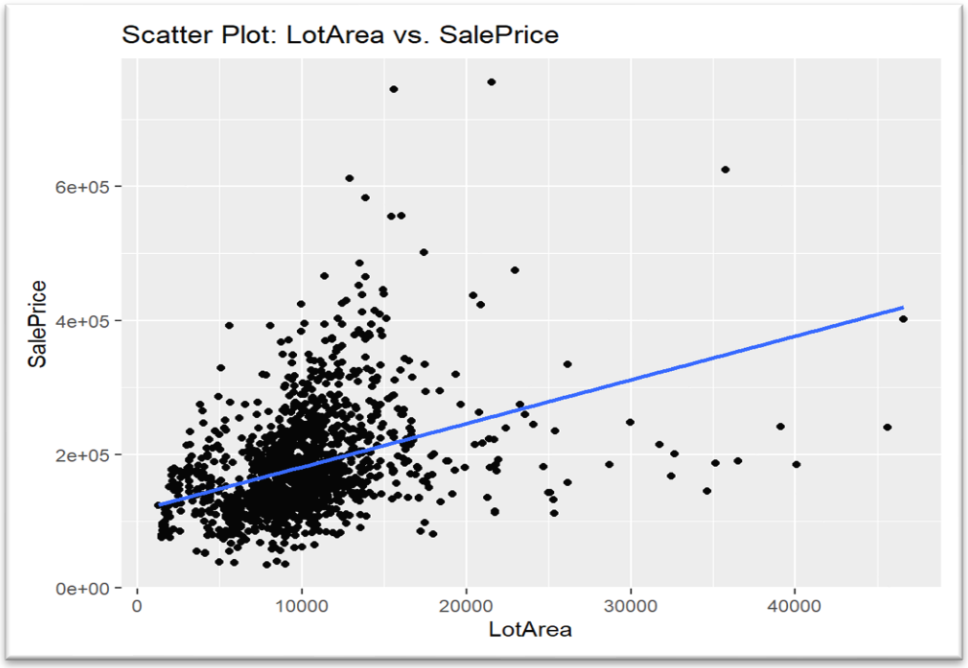
After deleting these two outliers from the "LotFrontage" feature and plotting the scatter plot again, it is observed that the linear line representing the relationship between the "LotFrontage" and the target variable (SalePrice) has changed. This change in the linear line indicates that the outliers had a significant impact on the estimated regression model.

LotArea: Lot size in square feet:

- Plotting the "Scatter Plot" for "**LotArea**" **before** deleting outliers



- Plotting the "Scatter Plot" for "**LotArea**" **After** deleting outliers (10 outliers deleted)



Description:

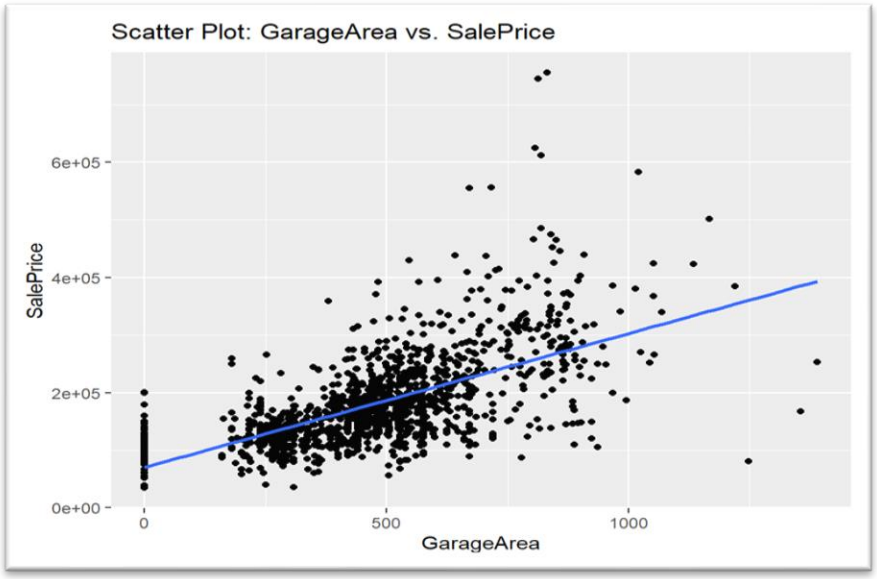
After analyzing the "LotArea" feature and plotting its scatter plot before and after deleting outliers, it was observed that the outliers were causing the linear regression line to deviate from the majority of the data points. This deviation indicated that the outliers were exerting a strong

influence on the estimated regression model between "LotArea" and the target variable, potentially biasing the results and affecting the accuracy of predictions.

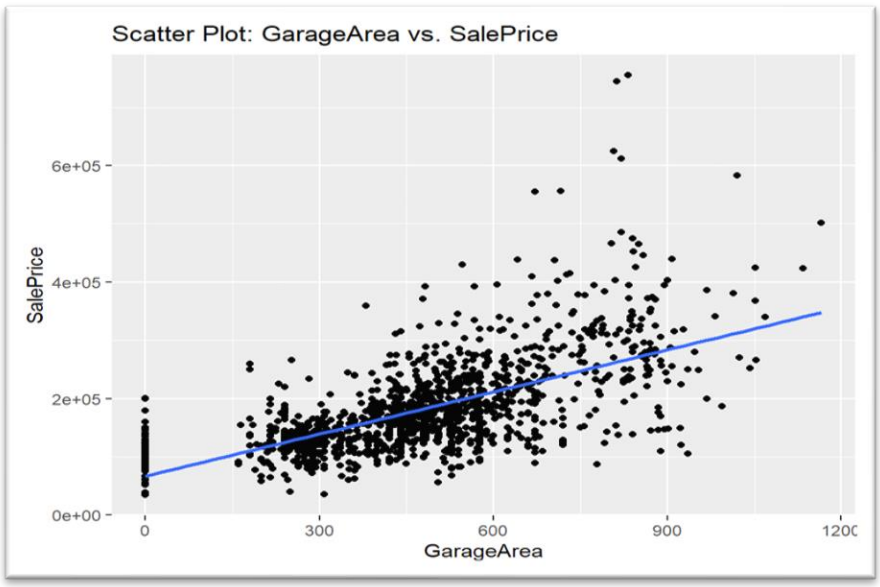
To address this issue, a decision was made to remove outliers with values exceeding 50,000 from the dataset. After removing these outliers, the scatter plot for "LotArea" was plotted again. The result showed that the linear regression line between "LotArea" and "SalePrice" improved significantly. The line became more aligned with the majority of the data points, indicating a stronger relationship and a better fit of the regression model.

GarageArea: Size of garage in square feet:

- Plotting the "Scatter Plot" for " **GarageArea**" **Before** deleting outliers



- Plotting the "Scatter Plot" for " **GarageArea**" **After** deleting outliers (**4 outliers deleted**)



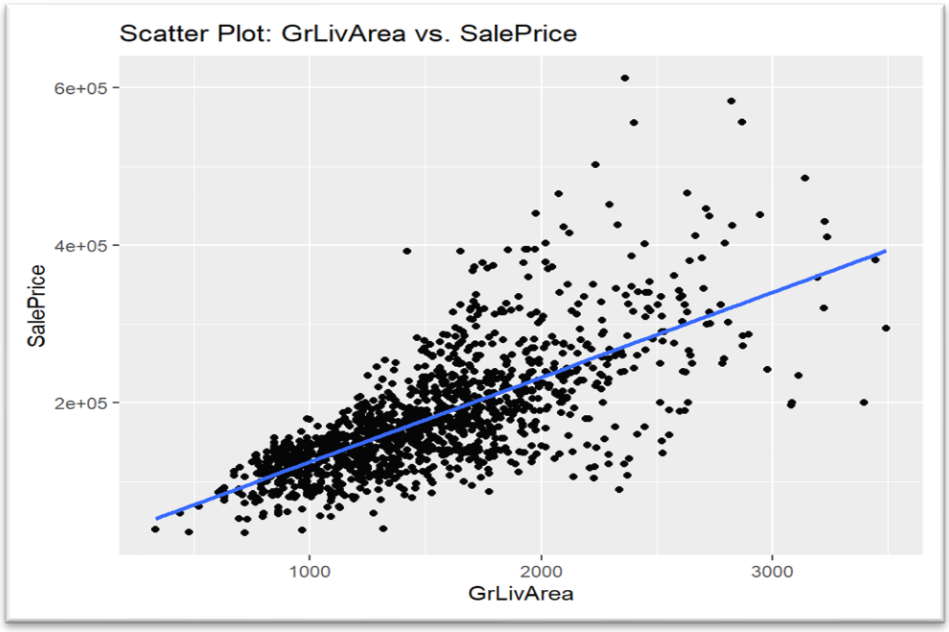
Description:

The outliers were identified as data points with extremely high values of "GarageArea," which were affecting the linear relationship between the feature and the target variable. Recognizing the need to address this issue, a decision was made to remove outliers exceeding 1,200 from the dataset, and 4 outliers were deleted. After removing these outliers and plotting the scatter plot again, a moderate improvement in the linear regression line was observed.

GrLivArea: Above grade (ground) living area square feet:

- Plotting the "Scatter Plot" for " **GrLivArea**" **Before** deleting outliers

- Plotting the "Scatter Plot" for " **GrLivArea**" **After** deleting outliers (5 outliers deleted)



Description:
Plotting the "Scatter Plot" for "GrLivArea" before deleting outliers, shows some outliers that affect the linear regression line between this feature and the target variable (SalePrice), but after deleting 5 outliers with values more than 3500 from the dataset, then plotting again, the line got better.

4.2. Log transformation and Skewness

Log transformation[3] is frequently employed as a method to decrease the skewness in a distribution and achieve a **symmetric** shape. Skewness measures the lack of symmetry in a probability distribution, where a distribution that is right-skewed exhibits a longer tail on its right side. Log transformation can help to "stretch out" the tail of a right-skewed distribution, making it more symmetric and easier to analyze.

If the **skewness**[4] is between -0.5 to +0.5 then we can say the data is fairly symmetrical. If the skewness is between (-1 to -0.5) or (0.5 to 1) then the data is moderately skewed. And if the skewness is less than -1 and greater than +1 then our data is heavily skewed.

Since when the **skewness** is between -0.5 to +0.5 the data is **fairly** symmetrical. Therefore, this analysis considers threshold equal to 0.50. Therefore, in first step, by putting **threshold (0.5)**, numerical features that their skewness were more than **0.5** detected, then log transform method implemented on them to reduce their skewness.

Features with skewness more than 0.5:

BsmtFinSF1	X1stFlrSF	WoodDeckSF	LotFrontage
BsmtFinSF2	X2ndFlrSF	PoolArea	LotArea
BsmtUnfSF	LowQualFinSF	MiscVal	MasVnrArea
TotalBsmtSF	GrLivArea	AllPorchSF	

4.3. Data Distribution and Scaling on numerical sub-dataset

In this part following four steps applied on numerical data:

Step One: Checking Normality of **some** features by QQ Plot and Histogram.

Step Two: Checking Normality of all features by a function which used Shapiro-Wilk Test.

Step Three: Scaling all numeric data to same range, to reduce non-normality of data.

Step Four: Checking again and deleting most skewed data after scaling.

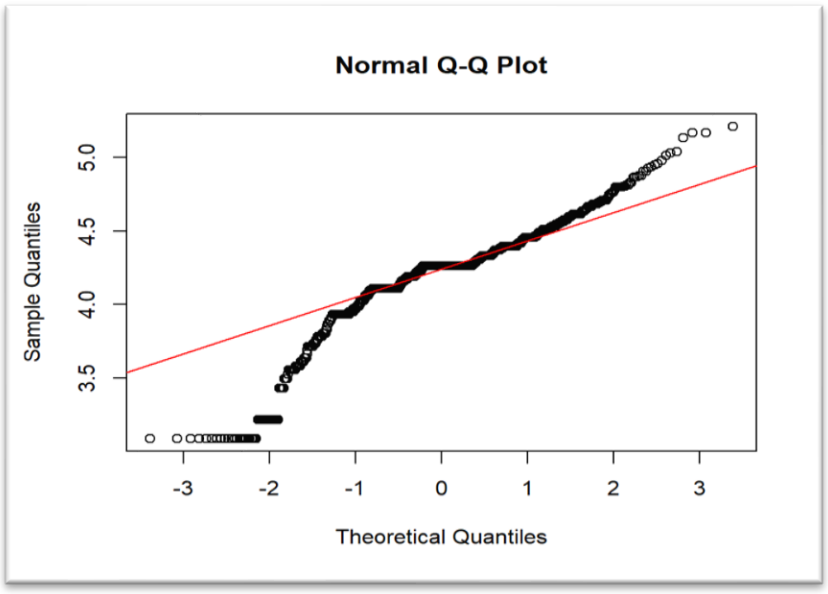
[3] Link: https://tariqueakhtar-39220.medium.com/log-transformation-and-visualizing-it-using-python-392cb4bcfc74
[4] Link:https://vivekrai1011.medium.com/skewness-and-kurtosis-in-machine-learningc19f79e2d7a5

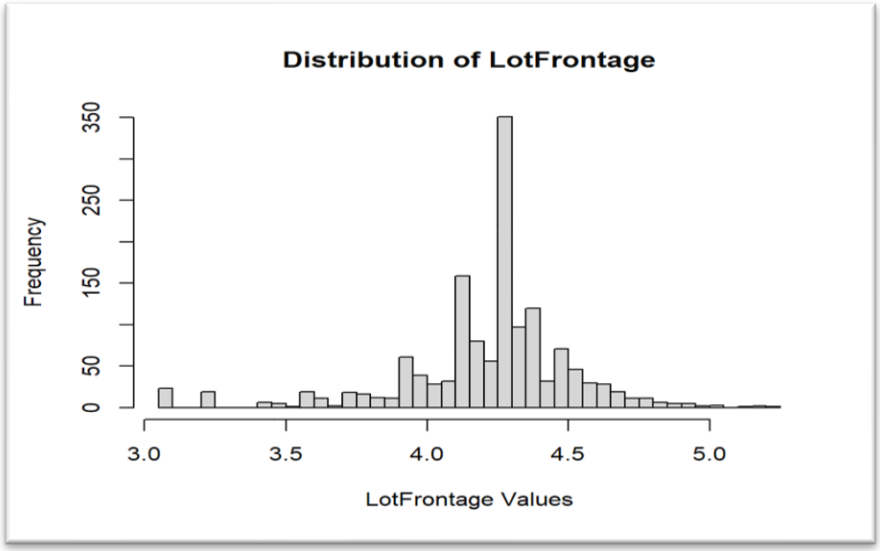
4.3.1. Step One: QQ Plot

A QQ plot (quantile-quantile plot) is a graphical tool that compares the quantiles of a dataset against the quantiles of a theoretical distribution, such as the normal distribution. If the **data points fall** approximately along a **straight line**, it suggests that the data follows a **normal** distribution.

Plotting some important features which play critical role in house price in real world.

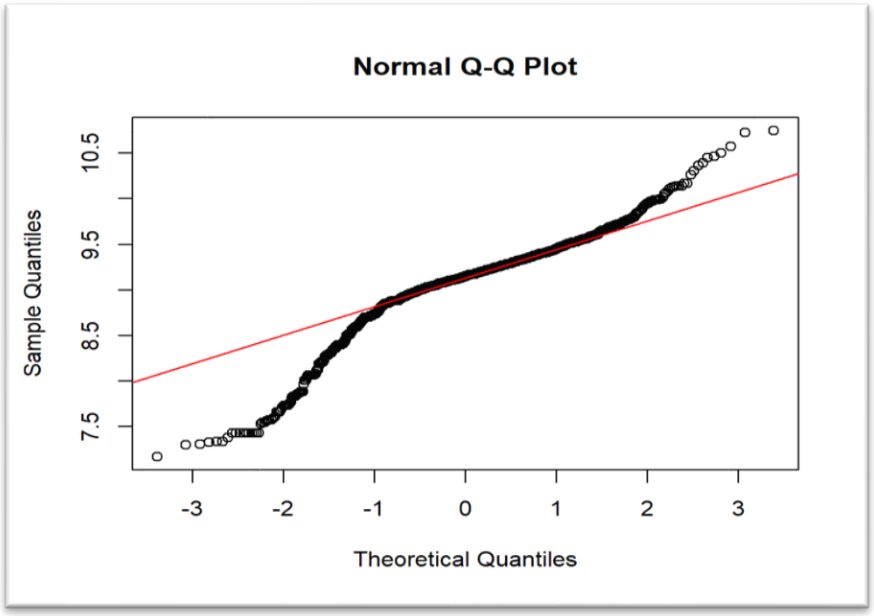
QQ Plot and Histogram for **LotFrontage** feature:

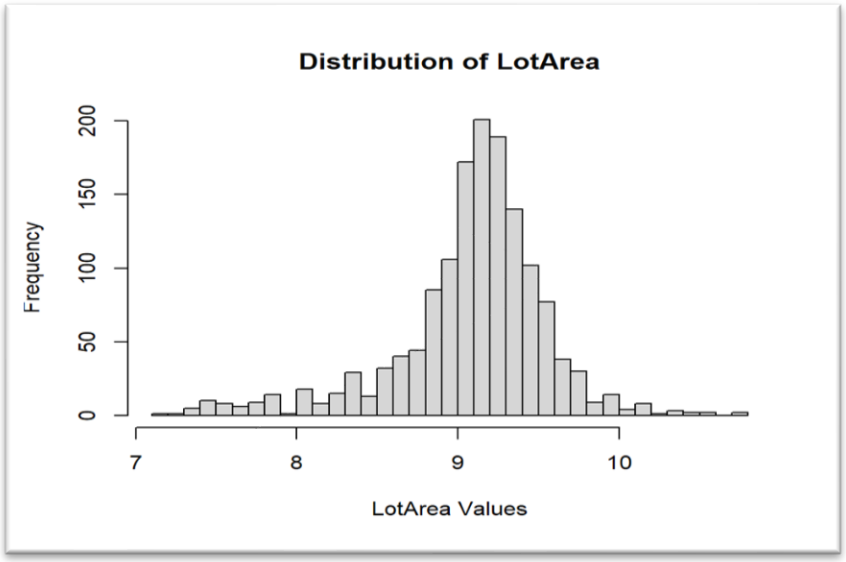




Description: The Plots display that the distribution of 'LotFrontage' feature is not normal.

QQ Plot and Histogram for '**LotArea'** feature:





Description: The Plots display that the distribution of '**LotArea'** feature is not normal.

4.3.2. Step Two: Shapiro-Wilk Test

The Shapiro-Wilk test is a statistical test that assesses whether a sample comes from a normally distributed population.

If the p-value is greater than the significance level (e.g., 0.05), we accept the null hypothesis, indicating that the data follows a normal distribution. But in the following result all P-Values are so small than 0.05, which shows **distribution of all numerical data are not normal.**

Result of **Shapiro-Wilk Test** for numerical features:

	Feature	Statistic	P_value
W	LotFrontage	0.90084939	1.550487e-29
W1	LotArea	0.91757626	2.658395e-27
W2	YearBuilt	0.92549214	4.039110e-26
W3	YearRemodAdd	0.86266334	1.058573e-33
W4	MasVnrArea	0.70432084	1.197613e-44
W5	BsmtFinSF1	0.73204813	3.746496e-43
W6	BsmtFinSF2	0.38253489	1.141789e-56
W7	BsmtUnfSF	0.69333765	3.294143e-45
W8	TotalBsmtSF	0.96538207	4.725792e-18
W9	X1stFlrSF	0.99655345	2.683206e-03
W10	X2ndFlrSF	0.66465450	1.343104e-46
W11	LowQualFinSF	0.10899182	3.300060e-63
W12	GrLivArea	0.99576752	4.829529e-04
W13	GarageYrBlt	0.92822536	1.088052e-25
W14	GarageArea	0.97916949	1.394419e-13
W15	WoodDeckSF	0.71400634	3.864062e-44
W16	PoolArea	0.03184558	9.702591e-65
W17	MiscVal	0.17732704	9.388331e-62
W18	YrSold	0.89677487	4.911084e-30
W19	AllPorchSF	0.80604465	1.974978e-38

4.3.3. Step Three: Scaling data

To scale[5] the data to a common range in R, there are various methods such as Min-Max Scaling, Z-Score Standardization, and Quantile Transformer Scaler. Here is used most common used methods, Z-Score Standardization.

In this analysis is used the "**scale()"** function, which **standardizes** the values by subtracting the mean and dividing by the standard deviation, which called Z-score normalization, transforms the values of a variable so that they have a mean of 0 and a standard deviation of 1.

Also In this part, the number of outliers for example for "LotFrontage","GarageArea", and "LotArea" before scaling checked, then all numerical features are scaled. Finally, to assess the impact of scaling on the presence of outlier, once again number of outliers are checked after scaling. But it was observed that the number of outliers remained unchanged. This indicates that scaling did not effectively address the issue of outliers in this dataset.

4.3.4. Step Four: Deleting most skewed data after scaling

In this step some features that even after scaling have more skewness are deleted. Since when the skewness is more than 1, we can say data is heavily skewed. So here threshold is considered equal to 1.

Based of result, most skewed features are: "BsmtFinSF2", "LowQualFinSF", "PoolArea", "MiscVal". These high skewed features are deleted from numerical sub_dataset.

[5] https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

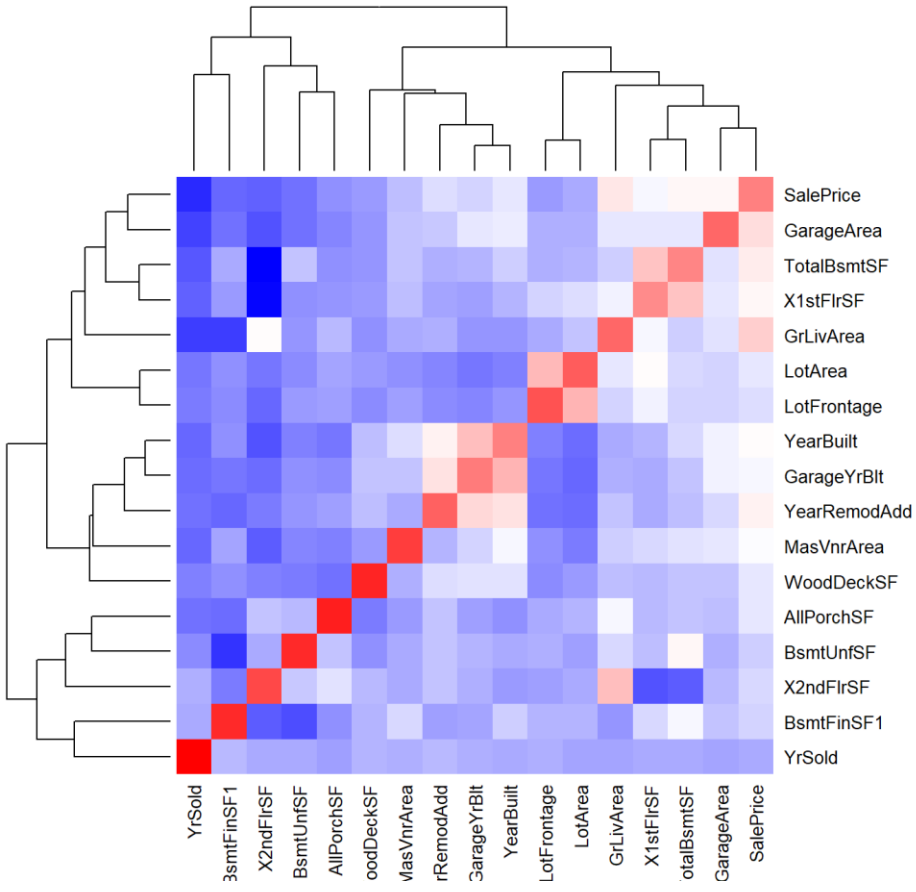
4.4. Multicollinearity between numerical features

Multicollinearity can adversely affect model performance and stability. Highly correlated features can lead to unstable estimates, inflated standard errors, and inaccurate coefficient interpretations. By dropping highly correlated features, you can improve the stability and performance of the model.

In this part we take **two step** to analyze correlation:

First step: Finding correlation between numerical features and plotting it.

Ggplot library are used in order to plot correlation heatmap, to find most important variable on target variable.



Second step: Dropping correlated features with Correlation bigger than 80%

Dropping correlated features with a correlation coefficient greater than 80% is done to address multicollinearity in this dataset. Multicollinearity occurs when two or more independent variables in a regression model are highly correlated with each other.

Result of correlation matrix and function with threshold shows that none of the features have high correlation with together.

5. Converting categorical data to numerical

Converting categorical data to numerical in order to implement some operation on these data.

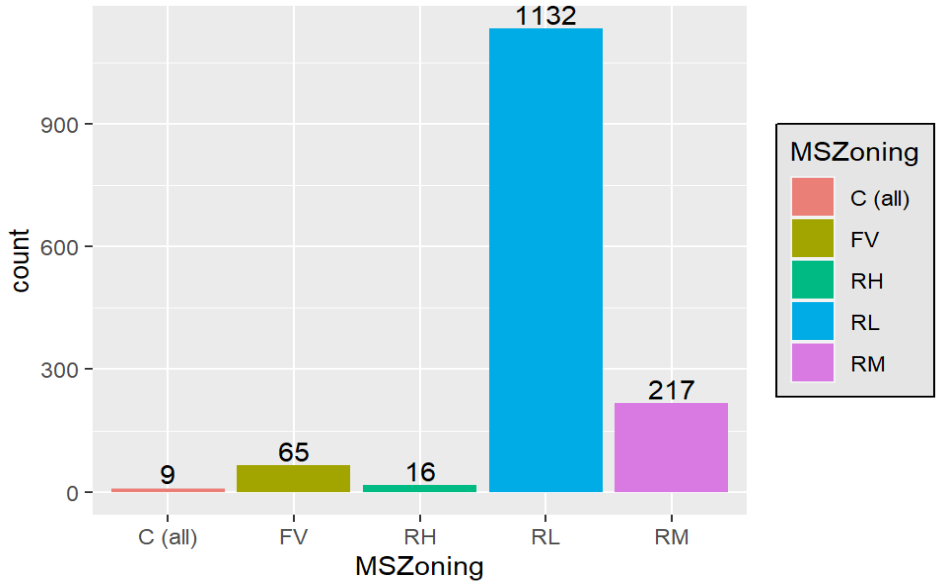
5.1. First Step: plotting some important features

- Counting number of houses sold in different categories of "**MSZoning**" by barplot:

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park
RM	Residential Medium Density

Figure: Distribution of MSZoning



Description: majority of houses sold is belong to Residential Low Density and Residential Medium Density.

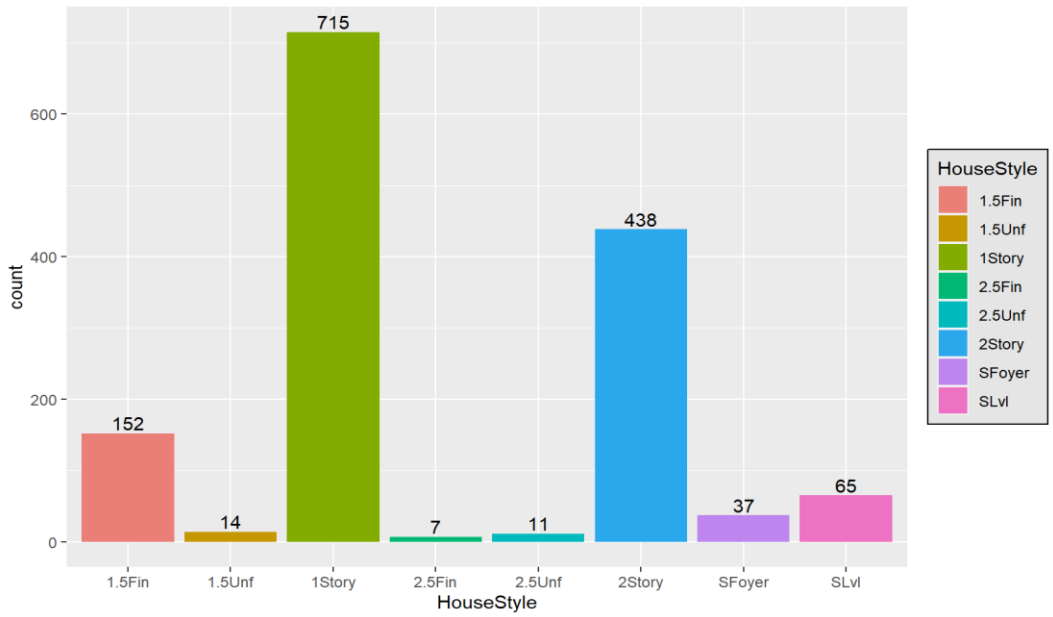
- Counting number of houses sold in different categories of "**HouseStyle**" by barplot:

HouseStyle: Style of dwelling

1Story	One story
1.5Fin	One and one-half story: 2nd level finished
1.5Unf	One and one-half story: 2nd level unfinished
2Story	Two story
2.5Fin	Two and one-half story: 2nd level finished
2.5Unf	Two and one-half story: 2nd level unfinished
SFoyer	Split Foyer

SLvl Split Level

Figure : Distribution of HouseStyle



5.2. Second Step: Separating categorical features in two group

In this analysis categorical features are grouped in two groups, first group contains features with more than four categories (factors) plus some categorical feature that are ordinal, but they have less that four factor. Second group contains features with less than equal four categories (factors). Because the goal is applying **label** encoding on first group with more than 4 categories to avoid very large feature set as we can end up with **dimensional** issue. Therefore, **factor** function from **superml** library used to apply **label encoding** on them. Then for features with less than equal 4 categories one-hot encoding method is applied to create dummy variable (creating one column for each category of feature).

After encoding all categorical features, now data is ready for any operation. Therefore, we merge numerical and categorical sub-datasets into one dataset named 'all_data'.

6. Model Fitting

After Descriptive Analysis we are moving into Predictive Analysis section. In this data analysis, two scenarios were considered to evaluate the impact of outliers on the results. The first scenario involved performing the analysis with all data points, including the outliers. The second scenario involved removing some outliers and conducting the analysis with the remaining data points. Ultimately, a comparison is made between two different scenarios.

The following algorithms are implemented in both scenarios:

1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Regression Trees
5. Random Forest Model
6. SVR (SVM Regression)
7. Gradient Boosting Regressor

6.1 Linear Regression Model

In Linear Regression Model, the relationships between Dependent and Independent Variables are expressed by equation with coefficients. The aim of this model is to minimize the sum of the squared residuals.

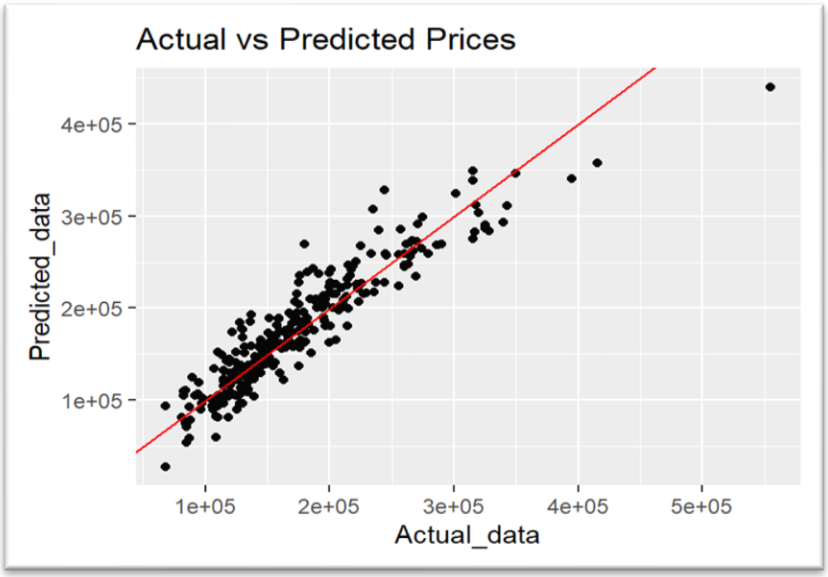
We will apply the Linear Regression algorithm to try to predict the dependent variable "SalePrice" by using the other variables in the dataset as predictors.

Summary of training data : here just result of variables that have strong impact are shown.

```
Call:
lm(formula = train_label ~ ., data = train_data)
Residuals:
    Min      1Q  Median      3Q     Max
-147464  -15908   -1063   13359  175396
Coefficients: (8 not defined because of singularities)
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        30368.87   39790.74   0.763 0.445502
TotalBath           6017.99    1820.35   3.306 0.000978 ***
OverallQual        11325.63    1126.66  10.052  < 2e-16 ***
OverallCond         6930.80     975.25   7.107 2.16e-12 ***
BedroomAbvGr      -10997.83    1578.67  -6.967 5.64e-12 ***
TotRmsAbvGrd        7644.24    1133.37   6.745 2.49e-11 ***
Fireplaces          8432.09    1698.69   4.964 8.02e-07 ***
KitchenAbvGr      -27614.86    4921.18  -5.611 2.55e-08 ***
BsmtCond           -9875.55    2428.24  -4.067 5.11e-05 ***
BsmtQual            5714.72    1992.71   2.868 0.004214 **
Functional          3170.02     976.55   3.246 0.001206 **
FireplaceQu        -4859.30    1060.08  -4.584 5.10e-06 ***
GarageType          2268.52     622.88   3.642 0.000283 ***
SaleCondition       2897.67     789.49   3.670 0.000254 ***
MasVnrTypeBrkFace -10885.50    3424.67  -3.179 0.001522 **
MasVnrTypeNone     21290.51    7237.25   2.942 0.003333 **
BsmtExposureGd     19141.90    3555.72   5.383 8.96e-08 ***
LotArea             7372.27    1408.84   5.233 2.00e-07 ***
YearBuilt          10334.73    2191.90   4.715 2.73e-06 ***
MasVnrArea         16243.46    3392.13   4.789 1.91e-06 ***
TotalBsmtSF        15257.23    2066.43   7.383 3.08e-13 ***
X1stFlrSF          10337.86    3234.55   3.196 0.001433 **
X2ndFlrSF          14373.50    3217.72   4.467 8.77e-06 ***
GarageArea          8120.22    2020.18   4.020 6.24e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 27180 on 1080 degrees of freedom
Multiple R-squared:  0.881,       Adjusted R-squared:  0.8733
F-statistic: 114.2 on 70 and 1080 DF,  p-value: < 2.2e-16
```

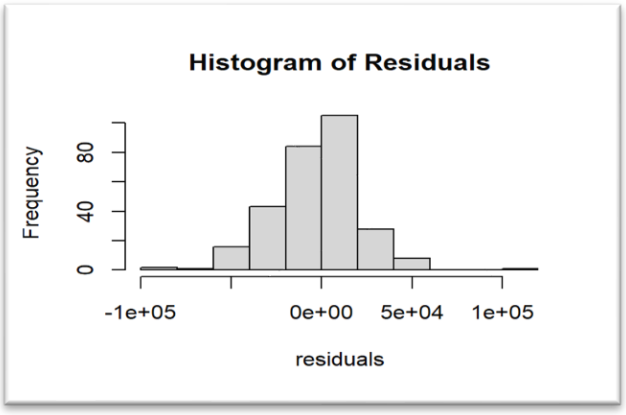
1. **Coefficients**: This table displays the estimated coefficients for each predictor variable in the model. The coefficients represent the expected change in the target variable associated with a one-unit increase in the corresponding predictor variable, while holding all other variables constant. For example, the coefficient of TotalBath is 6017.99, indicating that for every additional unit of TotalBath, the predicted value of the target variable (train_label) is expected to increase by approximately 6017.99.
2. **Residual standard error**: This value represents the standard deviation of the residuals. It provides an estimate of the average distance between the observed values of the target variable and the predicted values from the model. A lower residual standard error indicates a better fit of the model to the data.
3. **Multiple R-squared**: The multiple R-squared value measures the proportion of variance in the target variable that can be explained by the predictor variables included in the model. It ranges from 0 to 1, with a higher value indicating a better fit. In this case, the multiple R-squared is 0.881, suggesting that the predictor variables explain approximately 88.1% of the variance in the target variable.

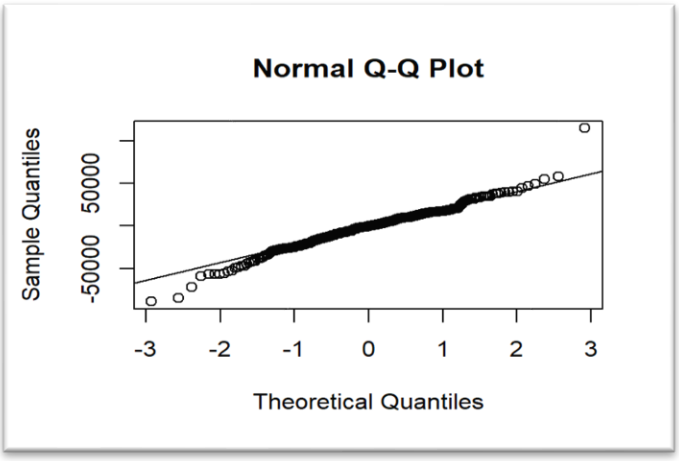
Plotting the scatter plot of actual vs predicted values



By visualizing the actual vs predicted values in a scatter plot, we can assess the model's accuracy and see how closely the predicted values align with the true values. If the points cluster tightly around the reference line, it indicates a good fit between the model's predictions and the actual data. Deviations from the reference line suggest areas where the model may be underestimating or overestimating house prices.

Plot a histogram and a Q-Q plot of the residuals






```
        Shapiro-Wilk normality test
data:  residuals
W = 0.9743, p-value = 4.839e-05
```

The **Shapiro-Wilk normality test** was conducted on the residuals of the model. The p-value is 4.839e-05. This result indicates that the **residuals** do **not** follow a **normal** distribution.

```
      Durbin-Watson test
data:  lm_model
DW = 1.9745, p-value = 0.8046
```

The **Durbin-Watson test** was conducted on the model residuals to assess the presence of **autocorrelation**. The test statistic (DW) is 1.9745, therefore, it indicates that there is no substantial autocorrelation present in the model residuals.

Result of Linear Regression Prediction With keeping and Removing Outlier

Model name	Mean Squared Error (**MSE**)	Root Mean Squared Error (**RMSE**)	R-squared Value
Linear Regression, **keeping** Outlier	1070460328	32717	**0.8319571**
Linear Regression, **Removing** Outlier	598023443	24454	**0.8656613**

Description of result:

A comparison between linear regression models, one with outliers and the other with outliers removed, reveals significant differences in their performance metrics. The model that includes outliers exhibits higher mean squared error (MSE) of 1,070,460,328 and root mean squared error (RMSE) of 32,717, indicating larger prediction errors. The R-squared value of 0.832 implies that approximately 83% of the target variable's variance is explained by this model. In contrast, the model with outlier removal demonstrates improved performance, with a lower MSE of 598,023,443 and reduced RMSE of 24,454. The R-squared value also increases to 0.866, indicating a better fit to the data and explaining around 87% of the variance. These findings suggest that eliminating outliers positively impacts the accuracy and predictive capability of the linear regression model.

6.2. Lasso Regression[6]

It is used over regression methods for a more accurate prediction. Lasso regression is a regularization technique. In regularization, what we do is normally we keep the same number of features but reduce the magnitude of the coefficients.

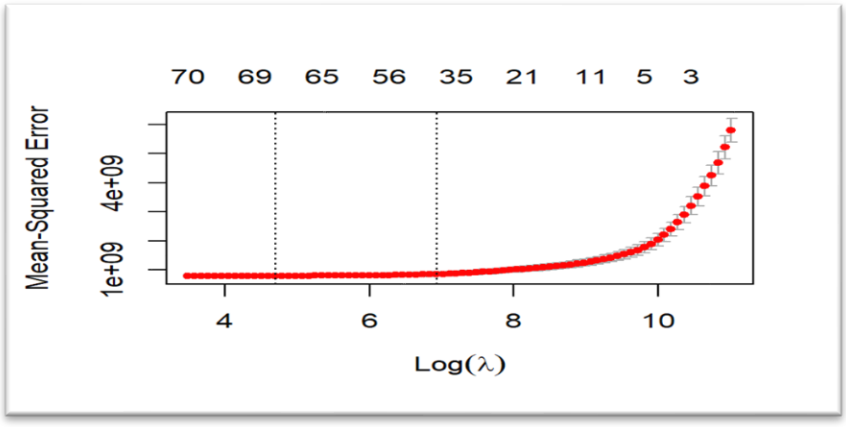
The primary goal of LASSO regression is to find a balance between model simplicity and accuracy. It achieves this by adding a penalty term to the traditional linear regression model, which encourages sparse solutions where some coefficients are forced to be exactly zero. This feature makes LASSO particularly useful for feature selection, as it can automatically identify and discard irrelevant or redundant variables.

The penalty term is the sum of the absolute values of the coefficients (L1 regularization).

Result of training data:

```
Call:  cv.glmnet(x = x, y = train_label, alpha = 1)

Measure: Mean-Squared Error
    Lambda Index   Measure       SE Nonzero
min  109.4    69 796342500 62602484      68
1se 1020.0    45 853834283 76041160      45
```



[6] Link: https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/#:~:text=Lasso%20regression%20is%20a%20regularization,i.e.%20models%20with%20fewer%20parameters

The lasso regression model was performed using the cv.glmnet function. The selected measure for evaluation was the mean squared error. Two values of lambda, namely min and 1se, were reported along with their respective indices, measures, standard errors, and the number of non-zero coefficients. The minimum lambda value was 109.4, and its corresponding index was 69, and it resulted in **68 non-zero coefficients**. The 1se lambda value was 1,020.0, with an index of 45, and it resulted in **45 non-zero coefficients**.

Result of Prediction With keeping and Removing Outlier

Model name	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared Value
Linear Regression, **keeping** Outlier	1070460328	32717	0.8319571
Linear Regression, **Removing** Outlier	598023443	24454	0.8656613
Lasso Regression, **keeping** Outlier	1056470035	**32503**	0.**8347327**
Lasso Regression, **Removing** Outlier	575769012	**23995**	0.**8698855**

Description of result: Lasso has low MSE and RMSE than linear regression, which shows its performance is better than linear regression.

6.3. Ridge Regression

Ridge tries to balance the bias-variance trade-off by shrinking the coefficients to reduce overfitting, but it does not select any feature and keeps all of them. While Lasso tries to balance the bias-variance trade-off by shrinking some coefficients to zero.

The penalty term is the sum of the squares of the coefficients (L2 regularization).

Result of Prediction With keeping and Removing Outlier

Model name	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared Value
Linear Regression, **keeping** Outlier	1070460328	32717	0.8319571
Linear Regression, **Removing** Outlier	598023443	24454	0.8656613
Lasso Regression, **keeping** Outlier	1056470035	32503	0.8347327
Lasso Regression, **Removing** Outlier	575769012	23995	0.8698855
Ridge Regression, keeping Outlier	**1063788979**	**32615**	**0.8342116**
Ridge Regression, **Removing** Outlier	**561731890**	**23700**	**0.8717279**

Description of result:

First: The comparison between Lasso and Ridge regression models reveals a minimal difference in their RMSE, indicating similar predictive accuracy. However, the specific objectives and requirements of the analysis should be taken into account. Lasso regression, known for its feature selection capability, assigns some coefficients to zero. While the RMSE remains relatively

unchanged, eliminating the features ignored by Lasso can simplify the model and improve interpretability. Nevertheless, a thorough evaluation of the context, significance of the ignored features, and overall model performance impact is necessary before making any decisions. But all in all, I decide to keep these features.

Second: result shows that when we remove outliers, the performance of the Ridge regression Increase significantly in comparison with keeping outliers. The best performance (lowest MSE and RMSE) belongs to Lasso regression with removing outliers.

6.4. Decision tree Regression[7]

A regression tree is basically a decision tree that is used for the task of regression which can be used to predict continuous valued outputs instead of discrete outputs.

In the Regression Tree algorithm, we do the same thing as the Classification trees. But we try to reduce the Mean Square Error at each child rather than the entropy.

Result of Prediction With keeping and Removing Outlier

Model name	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared Value
Ridge Regression, **Removing** Outlier	**561731890**	**23700**	**0.8717279**
Decision Tree Regression, **keeping** Outlier	1535356777	39183	0.7638792
Decision Tree Regression, **Removing** Outlier	1070643240	**32720**	0.7657119

Description of result:

Result shows that for this dataset, Decision Tree Regression does not perform well in comparison with previous Regression models, in both scenarios; keeping or removing outliers.

6.5. Random forest regression

Random forest regression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning. The trees in random forests run in parallel, meaning there is no interaction between these trees while building the trees.

Random forest operates by constructing a multitude of decision trees at training time and outputting the class that's the **mode** of the classes (classification) or **mean** prediction (regression) of the individual trees.

[7] Link: https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047

Result: Random Forest regression with Removing and keeping outlier for different tree

Model name	Number of tree*	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared Value
Ridge Regression, Removing Outlier	-	**561731890**	**23700**	**0.8717279**
Random forest R, keeping outlier	**500 ***	581439044	24113	0.9172295
Random forest R, Removing outlier	**500 ***	380863994	**19515**	**0.9127023**
Random forest R, keeping outlier	800	576978835	**24020**	**0.9180853**
Random forest R, Removing outlier		382806328	19565	0.9122580
Random forest R, keeping outlier	1000	578224157	24046	0.9179589
Random forest R, Removing outlier		382514369	19557	0.9123435

* Default value: **ntree=500**

Description of result:

First: Result shows that when we remove outliers, the performance of the Random Forest regression Increase in comparison with keeping outliers. The best performance (lowest MSE and RMSE) belongs to Random Forest regression with removing outlier with 500 tree number.

Second: result shows that for this dataset, Random Forest regression perform better than previous Regression models, with any number of trees in both scenarios; keeping or removing outliers.

6.6. Support Vector Regression (SVR) [8]

Support Vector Regression (SVR) gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data.

SVR is a powerful algorithm that allows us to choose how tolerant we are of errors, both through an acceptable error margin(ϵ) and through tuning our tolerance of falling outside that acceptable error rate.

[8] Link: https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

Result: SVM Regression with Removing and keeping outlier for different Parameters

SVM-Kernel,	outlier	Degree*	Gamma**	Num of Support Vectors	Mean Squared Error (MSE)	(RMSE)	R-squared Value
radial	removed	-	**default**: 0.0128	750	579614300	24075	0.8679594
	keep			723	976780652	31253	0.8638785
radial	**removed**	-	**0.002**	732	458625168	**21415**	**0.8989202**
	keep			716	**837503633**	**28939**	**0.8926195**
radial	removed	-	0.001	747	523686947	22884	0.8883535
	keep			743	982615457	31346	0.8749117
linear	removed	-	default: 0.0128	818	561762947	23701	0.8709347
	keep			791	978936598	31287	0.8581726
polynomial	removed	3 *	default: 0.0128	827	586601826	24219	0.8648458
	keep			811	1073364108	32762	0.8401107
polynomial	removed	5	default: 0.0128	941	1361251255	36895	0.68596
	keep			951	2279539225	47744	0.66418
sigmoid	removed	-	default: 0.0128		9131002618	95556	0.2366
	keep			1045	13482369268	116113	0.1383

* Parameter needed for kernel of type polynomial (default: 3)

** parameter needed for all kernels except linear (default: 1/(data dimension)) . here default : 0.0128

Visualizing for best model: method is kernel = "radial", gamma = 0.002

Description of result:

First: Result shows that when we remove outliers, the performance of the Random Forest regression Increase in comparison with keeping outliers, because the amount of Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) decrease significantly.

Second: The best performance (lowest MSE and RMSE) belongs to Support Vector Regression with **radial** kernel with removing outlier and Gamma 0.002 for both scenarios, while lowest performance belongs to SVM with **sigmoid** kernel in both scenarios, since it has lowest MSE and RMSE.

Third: Random Forest regression perform better than this Regression models, with any number of parameters in both scenarios; keeping or removing outliers.

6.7. Gradient Boosting Regression[9]

Gradient boost is a machine learning algorithm which works on the ensemble technique called 'Boosting'. Like other boosting models, Gradient boost sequentially combines many weak learners to form a strong learner. Typically, Gradient boost uses decision trees as weak learners.

[9] https://www.numpyninja.com/post/gradient-boost-for-regression-explained

Result: Gradient Boosting Regression with keeping and Removing outlier for different Parameters

N.trees	Outlier	Interaction. depth	shrinkage	CV.folds	Mean Squared Error (MSE)	(RMSE)	R-squared Value
100 *	removed	1	0.01	0	**1988340658**	**44590**	**0.7342502**
	keep				3311512444	57545	0.7299473
300	removed	1	0.01	0	835410799	28903	0.8450883
	keep				1571362866	39640	0.8262502
1000	removed	1	0.01	0	500320375	22367	0.8866444
	keep				780066017	27929	0.8822713
1000	removed	2	0.01	0	**417348540**	**20429**	0.9062905
	keep				630897850	25117	0.9033936
1000	removed	4	0.01	0	**359426064**	**18958**	**0.**9195737
	keep				495760801	22265	**0.9236055**
1000	removed	4	0.01	5	**365436080**	**19116**	0.9173652
	keep				500560735	22373	0.9238221
1000	removed	4	0.01	10	**350358444**	**18717**	0.9212387
	keep				477181082	21844	0.9273591
1000	**removed**	**4**	**0.1**	**10**	**321213263**	**17922**	**0.9277352**
	keep				618396387	24867	0.9090994
1000	removed	4	0.005	10	393971041	19848	0.9094066
	keep				592476249	24340	0.9139783

* Default value of each parameter:
n.trees = 100, number of trees
interaction.depth = 1,
shrinkage = 0.1,
cv.folds = 0

Description of result:

First: Result shows that when we remove outliers, the performance of the Gradient Boosting Regression Increase in comparison with keeping outliers, because the amount of Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) decrease significantly.

Second: The best performance (lowest MSE and RMSE) belongs to Gradient Boosting Regression with 1000 trees number, Interaction.depth 4, shrinkage 0.1, and CV.folds 10. While lowest performance belongs to Gradient Boosting with **100 tree** in both scenarios, since it has lowest MSE and RMSE. As a result, Gradient Boosting with low number of trees has low performance.

Third: Gradient Boosting Regression performance becomes better when the number of **Interaction.depth** is increased from 2 to 4, and also when the number **CV.folds** (cross-validation folds) increase from 5 to 10.

Description of important parameters in Gradient Boosting Regression:

1. **interaction.depth**: Integer specifying the maximum depth of each tree (i.e., the highest level of variable interactions allowed). A value of 1 implies an additive model, a value of 2 implies a model with up to 2-way interactions, etc. Default is 1.

2. **shrinkage**: a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually work, but a smaller learning rate typically requires more trees. Default is 0.1.

3. **cv.folds:** Number of cross-validation folds to perform. If cv.folds>1 then gbm, in addition to the usual fit, will perform a cross-validation, calculate an estimate of generalization error returned in cv.error.

Summary of result of all Algorithms:

Comparison of Models when Outlier Removed: Descending Order By MSE and RMSE

Model name	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	R-squared Value
Decision Tree Regression, **Removing** Outlier	1070643240	**32720**	0.7657119
Linear Regression, **Removing** Outlier	598023443	**24454**	0.8656613
Lasso Regression, **Removing** Outlier	575769012	**23995**	0.8698855
Ridge Regression, **Removing** Outlier	561731890	**23700**	0.8717279
SVM regression, Kernel: radial, Gamma: 0.002 Removing outlier	458625168	**21415**	0.8989202
Random forest Regression, Removing outlier N-tree: 500	380863994	**19515**	0.9127023
Gradient Boosting Regression, with **Removing** outlier, Parameter: 1000 trees number, Interaction.depth 4, shrinkage 0.1, and CV.folds 10	**321213263**	**17922**	**0.9277352**

Summary:

Result shows that for this dataset, Decision Tree Regression does not perform well in comparison with other Regression models. While Gradient Boosting Regression (with Removing outlier, 1000 trees number, Interaction.depth 4, shrinkage 0.1, and CV.folds 10) and Random forest Regression with 500 tree number are best model to predict House price. This means that based on requirement (values of features) that a buyer mentions, we can predict price of house he or she want buy.

By comparing the two scenarios, it was observed that the presence or removal of outliers had a notable effect on the results. In the case where outliers were retained, the outcome of all regression models showed moderate result in terms of Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-squared values. However, interestingly, the results for all these Regression models showed improvement when the outliers were removed. The MSE and RMSE decreased and the R-squared values increased, indicating that the removal of outliers had a positive impact on the predictive performance and accuracy of these models.

Furthermore, the analysis revealed the importance of understanding the nature and context of the outliers. Outliers can arise due to various reasons such as measurement errors, data entry mistakes, or other reasons. It is crucial to investigate the outliers and determine whether they are valid data points or erroneous values. In the case of this analysis, removing outliers resulted in improved results for regression models.

These findings highlight the importance of addressing outliers in the data analysis process. Outliers have the potential to significantly impact the results and distort the model's ability to accurately capture the underlying patterns and relationships in the data. By carefully considering the presence of outliers and their potential influence, more accurate models can be developed.

Reference:

https://posit.co/download/rstudio-desktop/

https://www.r-project.org/other-docs.html

https://ggplot2.tidyverse.org/

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

https://towardsdatascience.com/outlier-why-is-it-important-af58adbefecc

https://tariqueakhtar-39220.medium.com/log-transformation-and-visualizing-it-using-python 392cb4bcfc74

https://vivekrai1011.medium.com/skewness-and-kurtosis-in-machine-learningc19f79e2d7a5

https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/#:~:text=Lasso%20regression%20is%20a%20regularization,i.e.%20models%20with%20fewer%20parameters

https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047

https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2

https://www.numpyninja.com/post/gradient-boost-for-regression-explained