

Assignment 2 by Jason Alexander Chan a1867657

Concepts in Artificial Intelligence and Machine Learning, Trimester 1 2022

Contents

[1 Background 4](#_Toc98635060)

[2 Report deliverables 4](#_Toc98635061)

[3 Methodology 4](#_Toc98635062)

[3.1 Inspecting the Raw Data 4](#_Toc98635063)

[3.2 Processing ‘Missing’ values 8](#_Toc98635064)

[3.3 Implementing the Python Regression model (6 marks) 9](#_Toc98635065)

[3.3.1 Further pre-processing 10](#_Toc98635066)

[3.3.2 Regression 10](#_Toc98635067)

[3.3.3 Results 10](#_Toc98635068)

[3.4 Dropping features 13](#_Toc98635069)

[3.5 Conservative 14](#_Toc98635070)

[3.6 Aggressive 14](#_Toc98635071)

[3.7 Results 14](#_Toc98635072)

[3.8 Hyper Aggressive 16](#_Toc98635073)

[3.8.1 Results 17](#_Toc98635074)

[4 Conclusion 19](#_Toc98635075)

[5 Appendix 20](#_Toc98635076)

[5.1 Feature names in the training and test data set. 20](#_Toc98635077)

[5.2 Feature weights vs. Feature 23](#_Toc98635078)

**List of Figures**

[Figure 1: Snippet from Train.csv 4](#_Toc98635079)

[Figure 2: Data inspection steps taken 4](#_Toc98635080)

[Figure 3: Scatter plots of training features vs. SalePrice 5](#_Toc98635081)

[Figure 4: Histogram plots of training features 6](#_Toc98635082)

[Figure 5: Histogram plot of SalePrice 7](#_Toc98635083)

[Figure 6: Correlation plot of training data 7](#_Toc98635084)

[Figure 7: Fixing the ‘Missing’ Data 8](#_Toc98635085)

[Figure 8: Overall workflow for the regression model 9](#_Toc98635086)

[Figure 9: Regression Result on Dropped data set 11](#_Toc98635087)

[Figure 10: Regression Result on Dropped and Unskewed data set 11](#_Toc98635088)

[Figure 11: Regression Result on Imputed data set 11](#_Toc98635089)

[Figure 12: Regression Result on Imputed and Unskewed data set 12](#_Toc98635090)

[Figure 13: Imputed, unskewed scatter plots 13](#_Toc98635091)

[Figure 14: Revisiting the best performing model 15](#_Toc98635092)

[Figure 15: Regression Result after dropping features from Conservative List 15](#_Toc98635093)

[Figure 16: Regression Result after dropping feature from Aggressive List 15](#_Toc98635094)

[Figure 17: Feature weights in the model after aggressive dropping 16](#_Toc98635095)

[Figure 18: Regression Result after dropping feature from the Hyper Aggressive List 17](#_Toc98635096)

[Figure 19: Figure 17: Feature weights in the model after hyper aggressive dropping 17](#_Toc98635097)

# Background

This assignment aims to teach the basic flow of machine learning. The objective is to predict house sales price using a regression model. There are two deliverables: runnable code and a PDF report which includes the methodology, analysis and results.

The training data set is a csv file that contains 38 features on 1168 houses, while the test data set is another csv file with with 38 features on 292 houses.

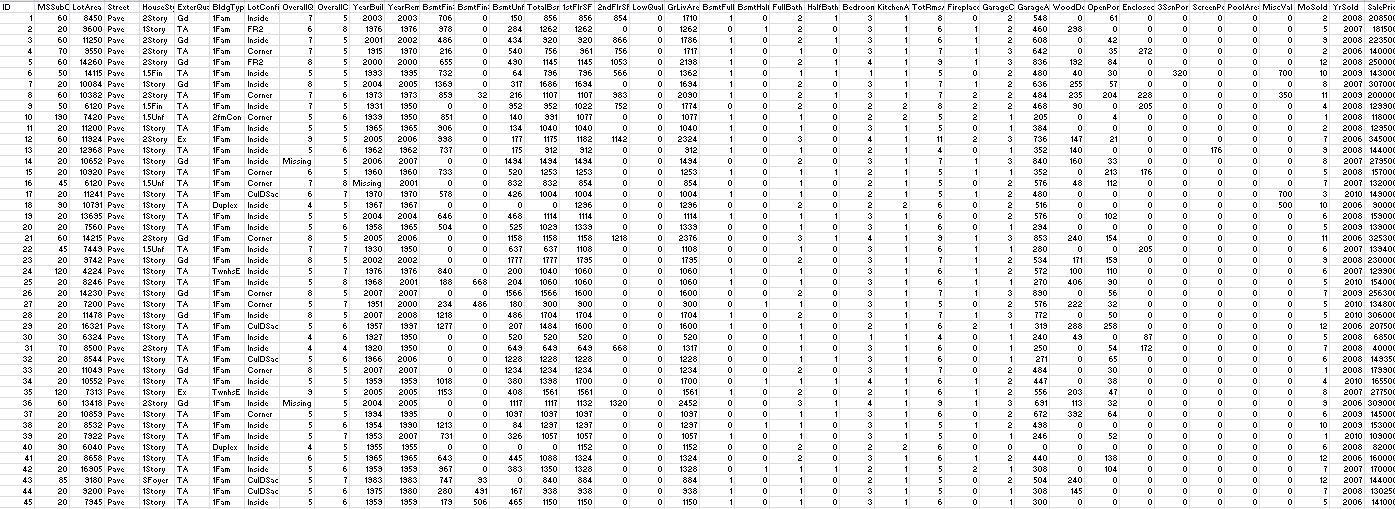


Figure : Snippet from Train.csv

# Report deliverables

1. A description of how to handle the missing values in the code. Report the results. (2.5 marks).
2. A description of the regression technique you used and report the results (3 marks)
3. A description of the feature selection you applied and report the results (2.5 marks)

# Methodology

## Inspecting the Raw Data

Before performing any data processing, it’s important to first inspect the raw data to get an initial impression about the training data set. I use the below steps to do this. I implemented this assignment in a python notebook in which I also created the plots for the report.

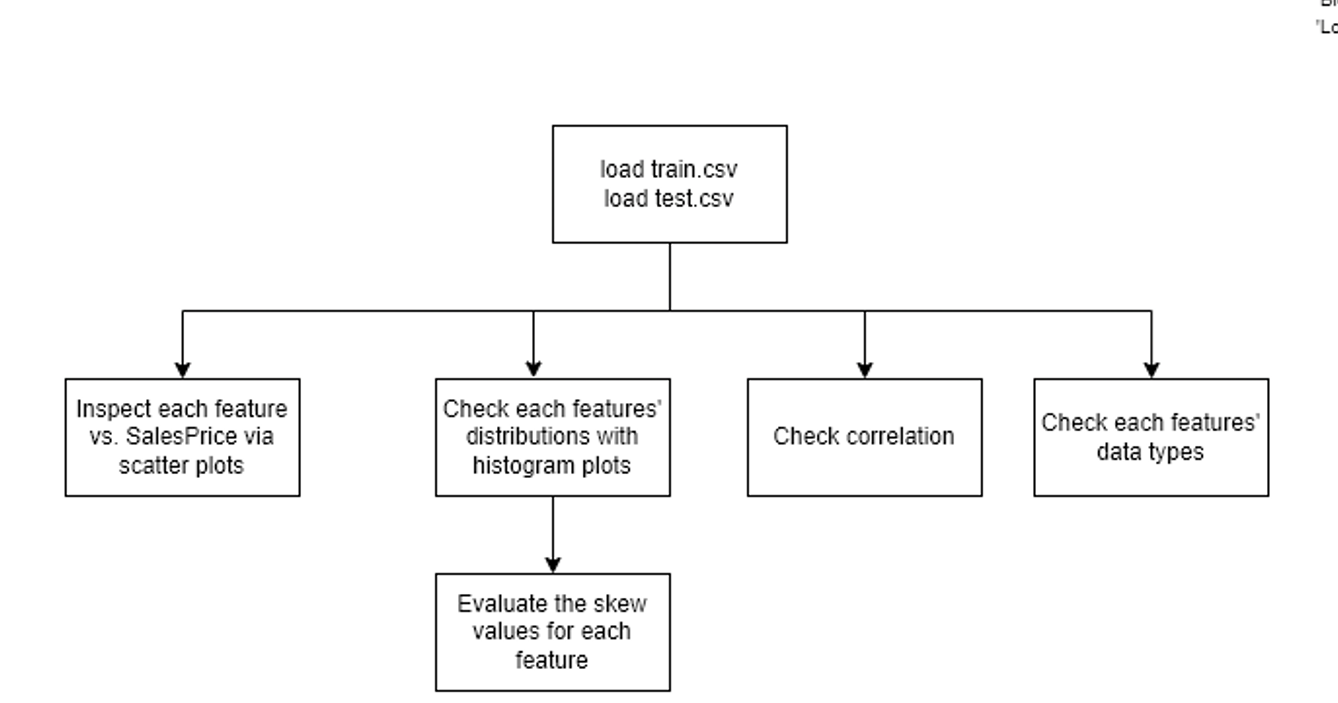
****

Figure 2: Data inspection steps taken

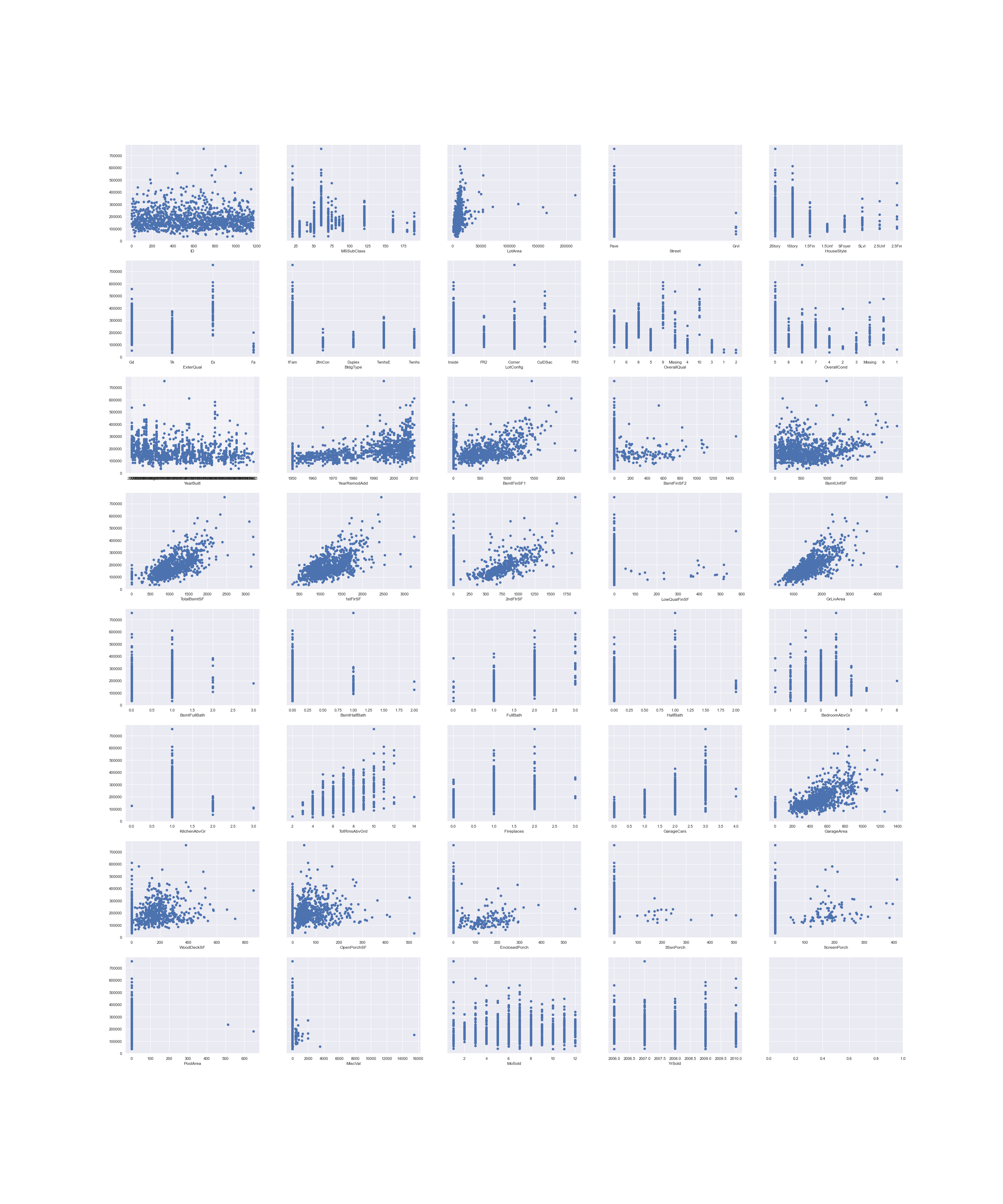


Figure 3: Scatter plots of training features vs. SalePrice

The scatter plot gives initial hints about the features that affect SalePrice – these are the features[[1]](#footnote-1) that have a rough trend from the bottom left of the plot to the top right such as GrLivArea, LotArea, TotalBsmtArea. Curiously, there are many features about basements suggesting that this dataset was taken from a cold region. OverallQual and OverallCond should be sorted numerically but isn’t because their initial data types are strings. OverallQual and OverallCond have an unexpected value called ‘Missing’. This is the first hint that about the ‘Missing’ data.



Figure 4: Histogram plots of training features

The histogram plots reveal data distributions for each feature. We can immediately see features that are highly skewed (such as LotArea, BsmtFinSF1), somewhat normally distributed (TotRmsAbvGrd), sparse (MiscVal). The skewed data will need to be transformed for the regression for the models to better discern relationships between those features to SalePrice. The sparse data are probable candidates to be dropped. ID offers no valuable information because it’s evenly distributed across the range and should be dropped.

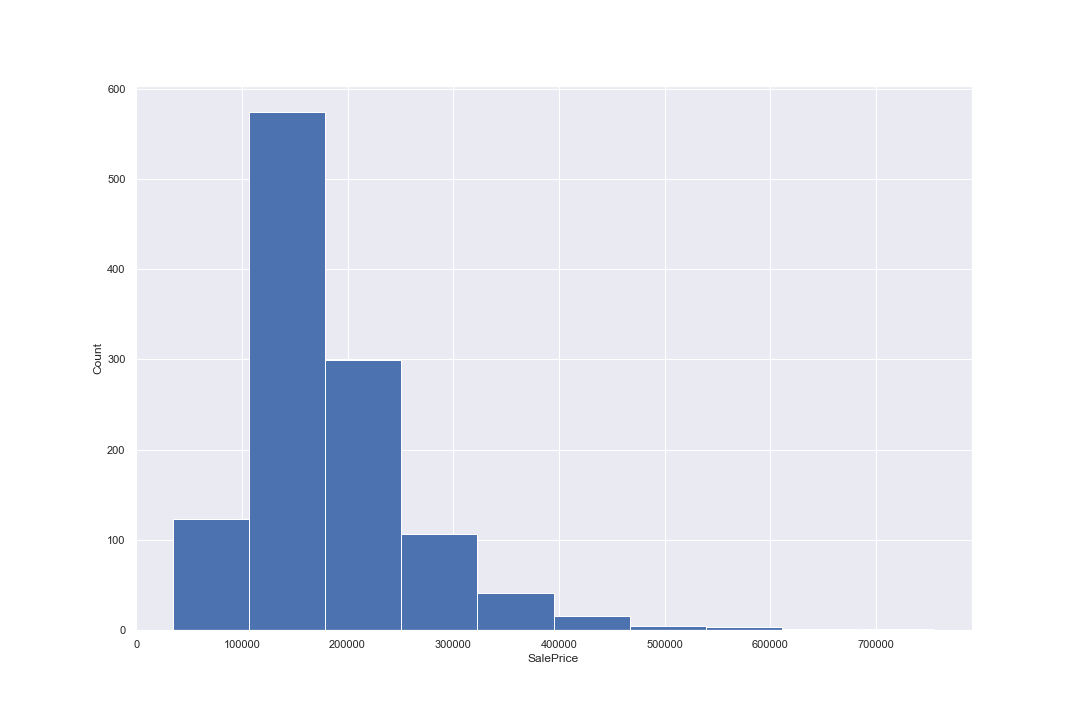


Figure 5: Histogram plot of SalePrice

The above plot is a histogram of SalePrice in the training data. It is highly skewed and will need to be transformed for the regression.

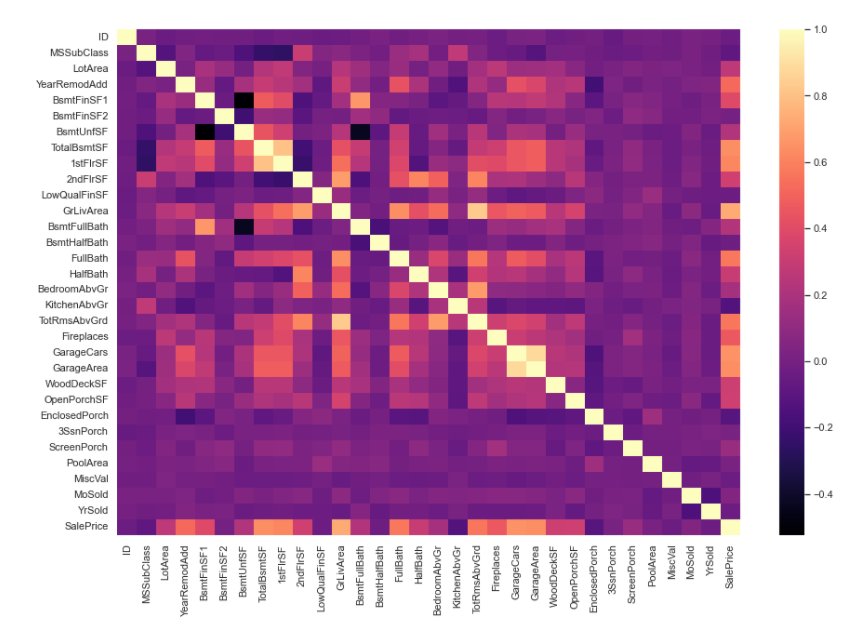


Figure 6: Correlation plot of training data

The above plot reveals the correlation of each feature with the other features[[2]](#footnote-2) in the training data. SalePrice has a strong correlation with YearRemodAdd, features related to surface area like TotalBsmtSF, 1stFlrSf, 2ndFlrSF, GrLivArea, features related to bathrooms such as FullBath, HalfBath, Garages, WoodDeckSF and OpenPorchSF. This intuitively makes sense because they are desirable in a property. Fireplace has a strong correlation to SalePrice too suggesting that these properties could be located in a cold region.

There is very weak correlation between ID, EnclosedPorch, 3SsnPorch, ScreenPorch, MiscVal, MoSold and YrSold to the other features.

## Processing ‘Missing’ values

Missing data can affect machine learning models. In general, machine learning models require data sets to be adequately prepared and transformed if it’s to have any hope of offering potentially meaningful predictive models. Missing values can cause a run time error during the pre-processing or training phases of the machine learning workflow.

To start the process, I imported test.csv and train.csv as a panda data frame and did the raw data inspection in the previous section to get an initial impression. I could clearly see the string ‘Missing’ in columns. I confirmed this by evaluating the datatype of an entry with ‘Missing’ which returned a dtype(‘O’), which is a panda string type. After that I did some sample inspections for: NaNs, Infs, empty cells. In practical machine learning problems the data set will have any combination of these. There were none of these in the training and test data.

Since the ‘Missing’ data seemed only to be the ‘Missing’ string, I used the below steps to process the training and test data to fix them.

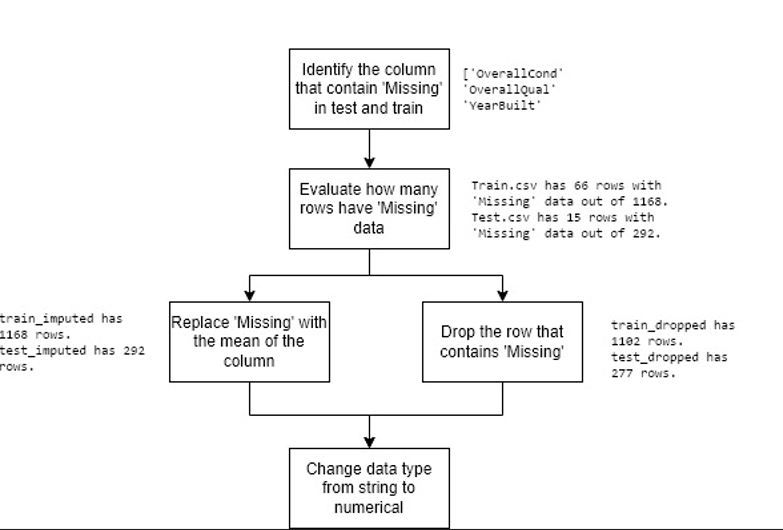
****

Figure 7: Fixing the ‘Missing’ Data

Then I identified the columns that contain the ‘Missing’ string: these columns are OverallCond, OverallQual, YearBuilt. I created a function to do this and confirmed that training and test only had ‘Missing’ data in the same columns. I also counted how many rows were affected: training data had 66 rows out of 1168 are affected (5.6%), test data had 15 out of 292 rows are affected (5.1%).

I applied two techniques to deal with the ‘Missing’ data:

1. Imputing. This involved replacing the ‘Missing’ fields with the numerical mean of the column.. I quickly discovered this didn’t work because the columns had mismatched data types: ‘Missing’ is a string while the numeric values are floats. I then replaced all instances of ‘Missing’ with NaNs, the I cast the entire column as floats, and replaced NaNs with the mean.
2. Dropping the rows that contain ‘Missing’ data. This is acceptable because only approximately 5% of the rows are affected. This wouldn’t be acceptable if the number of rows were 50% for example because too much of the training data is discarded.

Finally, I change the data type of the entire column to numerical so that they can be used in the regression, which only accepts numerical data types.

## Implementing the Python Regression model (6 marks)

The overall workflow I used is shown below.

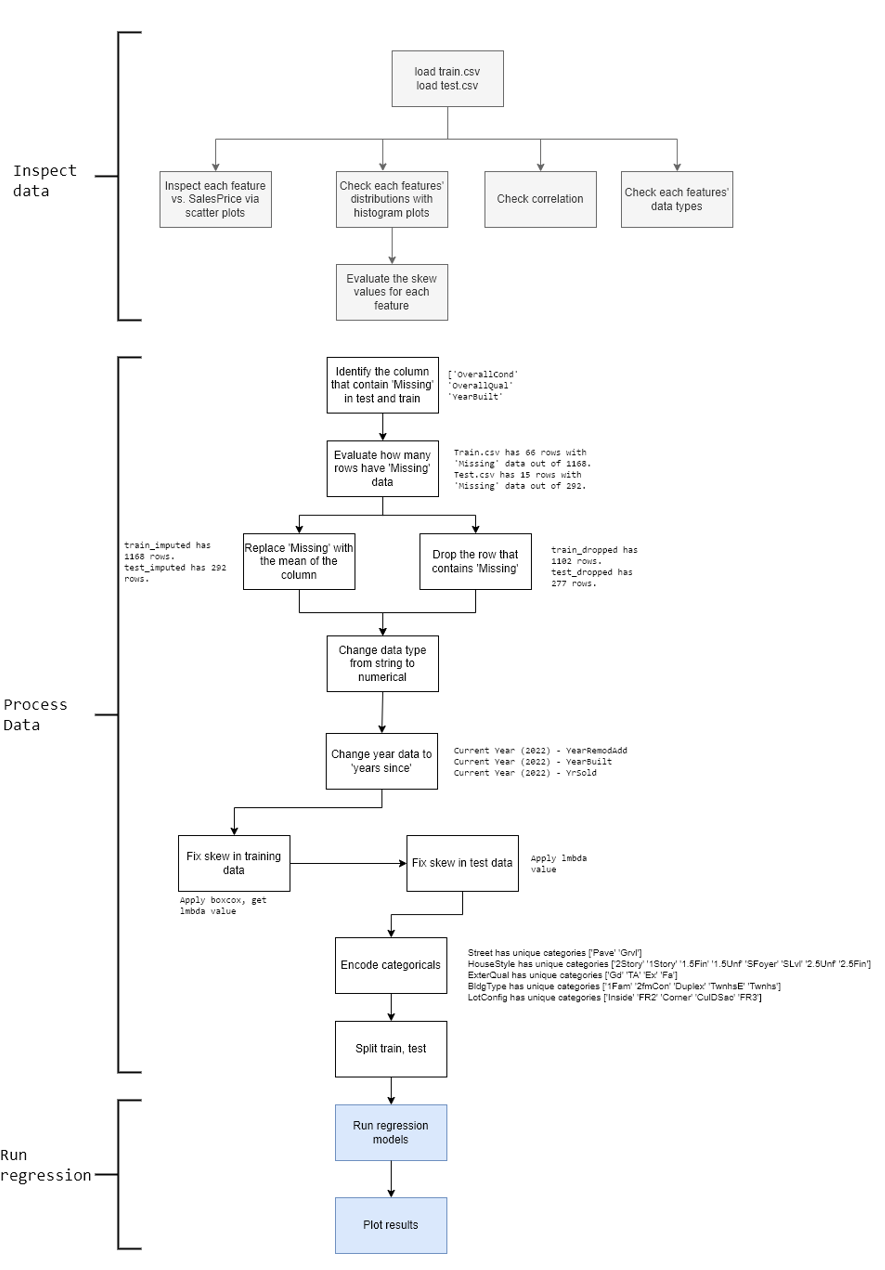


Figure 8: Overall workflow for the regression model

### **Further** pre-processing

There were extra pre-processing steps I needed to employ.

* Changing year data to ‘years since’ simply involves deducting the year data from the current year (2022). The result is a year difference rather than absolute year. This is more meaningful data into the regression model – it matters to a customer how many years ago that a property was remodelled/renovated rather than the year in which that happened. My initial attempts to use absolute year resulted in high error and poor prediction results.
* Fixing skew in the training data. Fixing skewed data is important for linear regression because the technique itself based on the assumption that the features are linearly independent and normally distributed. Since the training data has skew, we can use boxcox transformation function to transform each feature to more closely resemble a normal distribution. Note: a normal distribution isn’t guaranteed but the transformation is effect is generally better than its untransformed distribution for linear regression. Applying boxcox also evaluates a corresponding lmbda value (ranges from -5 to 5), which is the parameter used to unskew that particular feature.
* Fixing the skew in the test data using the lmbda value. The training data needs to be similarly unskewed using the same lmbda value for the particular feature in the training data.
* Encode categoricals. Regression modelling can only take numerical data but we have string categorical values. These are processed into numericals via the one-hot-encoding technique whereby each unique category is given a new column, the value for each row is a Boolean: 1 for true, 0 for false.
* Split train, test.

### Regression

I used five regression models to evaluate the prediction results: linear regression, ridge regression, Huber regression, ransac regression and theil sen regression. I employed five because I was curious about each of their performance. All five were part of SciKitLearn’s linear\_model library, which was exceptionally convenient and easy to use. The Root Mean Square Error (RSME) and the correlation coefficient (R2) of the predicted SalePrice vs. the actual SalePrice in the test data were evaluated.

I created four combinations of train, test data sets because I was curious about what effect (if any) these specific pre-processing steps would have. All other pre-processing steps such as one-hot-encoding, change year data to ‘years since’ etc. were applied to these four combinations.

1. Dropped rows for ‘Missing’ data
2. Dropped rows for ‘Missing’ data, and unskewed
3. Imputed values for ‘Missing’ data
4. Imputed values for ‘Missing’ data and unskewed

### Results

The R2 plots are shown on the next page. For each of these combinations of train, test data sets I plotted the predicted SalePrice vs. the actual SalePrice in the test data. The subplot title annotates the linear regression model used, the R2 value and the RSME value.

The best model is linear regression on the imputed and unskewed training data, where R2 is 0.818 and RSME is 0.09. It is highlighted in green on the next page. *Imputing* the ‘Missing’ data is marginally better than dropping the rows with ‘Missing’ Data.

**Results Dropped**

Unsatisfyingly, the R2 value is roughly in the order of 0.6. The best model is are the linear regression and ridge models with R2 scores of 0.656 and the worst is Huber at 0.496. We can see a slight upward bend in all the plots highlighting the poor prediction result. One outlier can be seen in the bottom right corner in all the plots. The data points seem to gravitate to the bottom left corner, this is the result of leaving the skew in the data features – recall that the raw data plots exhibited a similar trait. The RSME value is in the order of 40-50,000 because the data has not been transformed and left with their original units, skew and outliers.

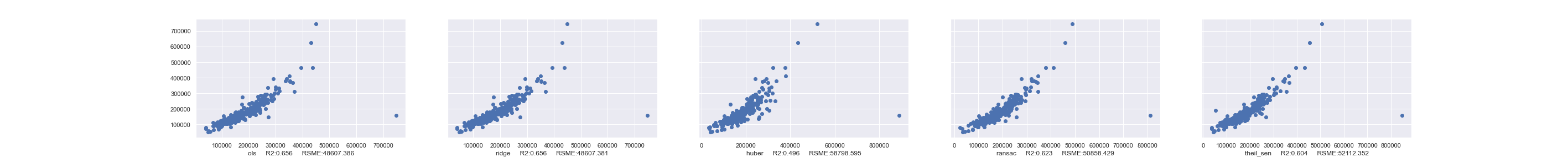


Figure 9: Regression Result on Dropped data set

**Results: Imputed**

Imputing the ‘Missing’ value had a small but immaterial effect. The R2 score for linear regression and ride improved to 0.660 each, while theil sen improved from 0.604 to 0.611. Huber significantly improved from 0.496 to 0.653. However, imputing seems to have a strongly negative effect on the ransac model reducing its R2 value from 0.623 to 0.225 – looking at its plot, the data has been cleaved into three chunks, which is a poor result.

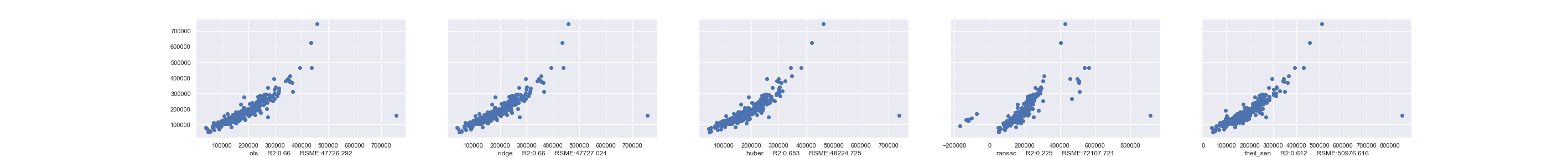


Figure 10: Regression Result on Dropped and Unskewed data set

**Results: Dropped and Unskewed**

Unskewing the data has significantly improved the prediction result. The R2 value is roughly in the order of 0.7-0.8 when their skew has been transformed with boxcox vs. 0.6 when skew was left in the feature data. The best result is the linear regression model at 0.813. The worst result is theil sen 0.659 because the predictions exhibit a wider variance from the trend line than the other four models. The plot exhibits a straighter correlation trend line unlike the previous two results. The transformation has noticeably spread the data points across the range of values. The RSME value is < 1 due to the transformation. All models have one far right outlier. Notice that the SalePrice values plotted in the x and y axes are their transformed values.

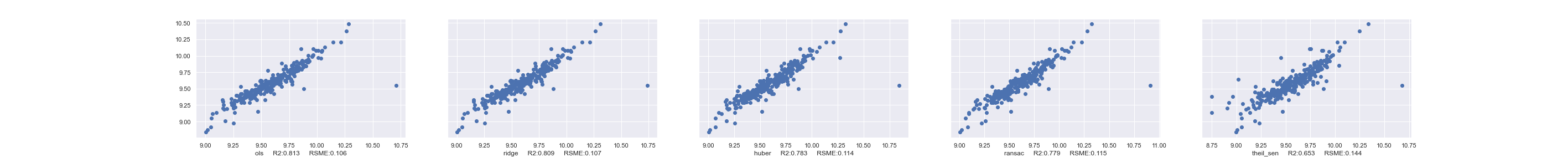


Figure 11: Regression Result on Imputed data set

**Results: Imputed and Unskewed**

Unskewing the imputed data has a marginally better prediction result than unskewing the dropped data. The best R2 value is 0.818 by the linear regression model (highlighted in green). However, it seems that unskewing the imputed data makes theil sen and ransac terrible predictors with R2 values of 0.276 and 0.336 respectively - their trend lines are roughly straight but they have significantly more outliers, which drag their R2 values down.

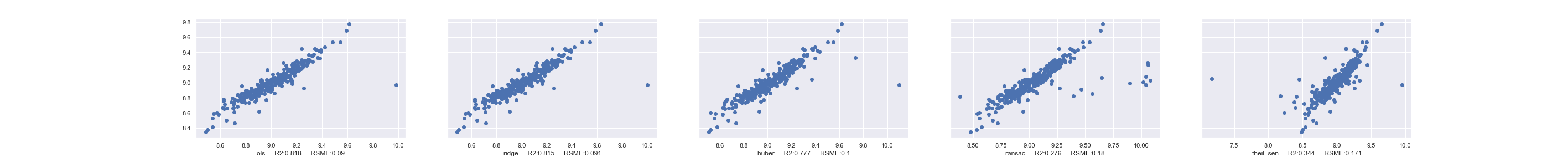


Figure 12: Regression Result on Imputed and Unskewed data set

## Dropping features

To better determine which features to drop I plotted the imputed, unskewed scatter plots of the training data (the better solution). From this scatter plot I could form an opinion about which features to drop.

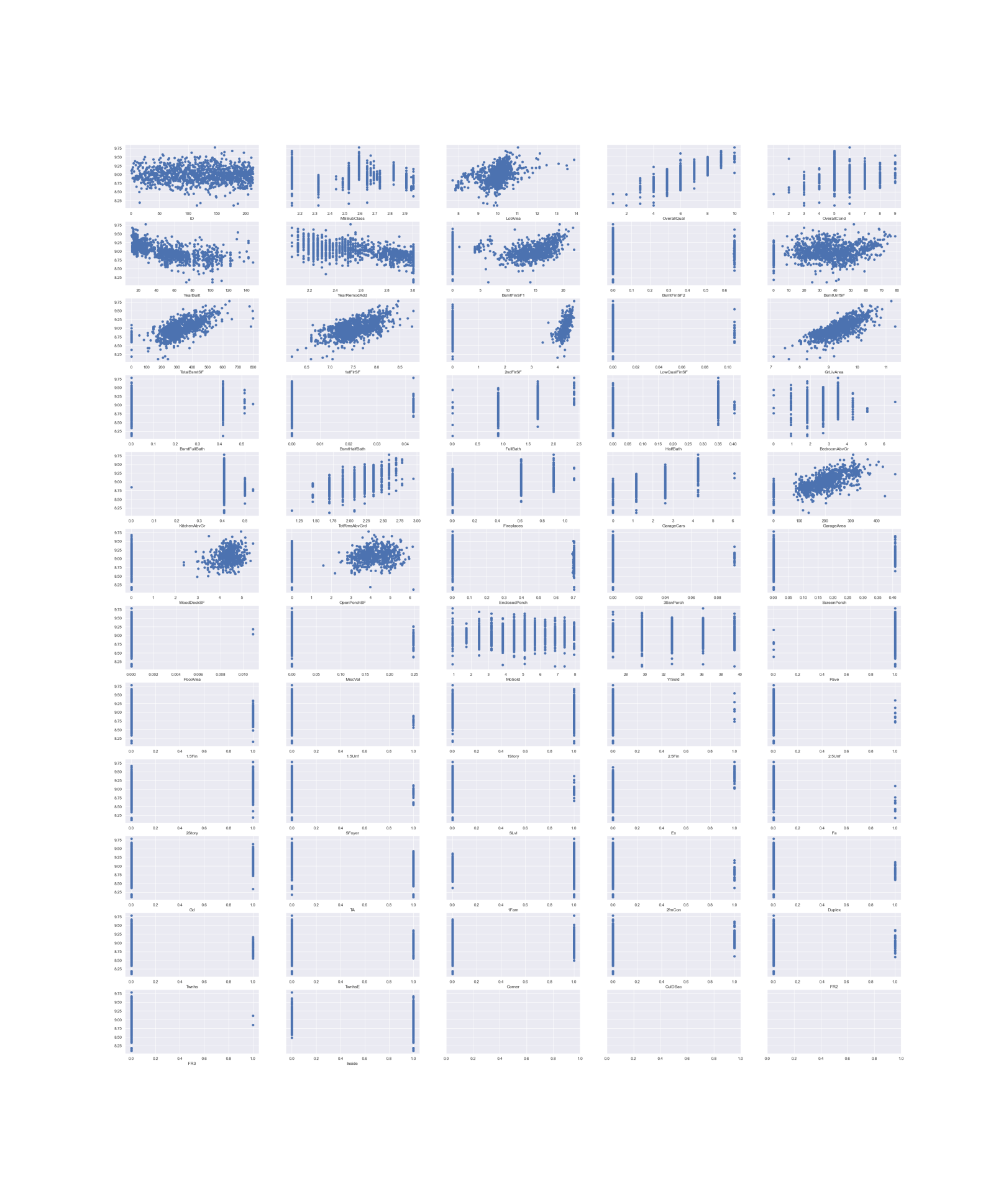


Figure 13: Imputed, unskewed scatter plots

Two lists of columns were identified that contain the features to drop. The first is called *Conservative* and second is called *Aggressive*. *Conservative* drops the features that I’m highly confident will have negligible impact to the prediction performance. *Aggressive* extends the list of features to drop in *Conservative* by dropping extra features that I’m reasonably confident won’t affect the prediction performance.

## Conservative

* Drop ID because it contains no information about the house. Confirmed by inspecting the histogram plot in the first section.
* Drop MSsubclass because that information is captured in the categorical features anyway. It’s therefore redundant. Confirmed by reading the *data\_description.txt*.
* Drop MoSold – assume monthly price trends immaterial.
* Drop MiscVal – sparse feature set. Where data exists its value is single outlier at 16,000 which is a small fraction of house price value. Confirmed by inspecting the histogram plots in first section.

## Aggressive

Aggressive drops the list in *Conservative* plus:

* Drop Street because number of gravel is << paved. One hot encoded so drop Pave.
* Drop 3SsnPorch because sparse and data with values have narrow band horizontal trend vs. SalePrice suggesting no correlation. Per inspection of the scatter plot.
* Drop LowQualFinSF because sparse and data with values have narrow band horizontal trend vs. SalePrice suggesting no correlation. Per inspection of scatter plot.
* Drop YrSold because year range is only 4 years between 2006-2010 and don't expect materially price fluctuations over that time. However, this is during GFC so year should it might actually affect prices – unsure.
* Drop FR3 because sparse. Per inspection of the scatter plot of the imputed and unskewed data set.

## Results

The results are shown in the next page. In summary, the conservative drop only resulted in a marginal decrease in R2 correlation of the predicted SalePrice vs. the actual SalePrice in the test data vs. baseline in the linear regression model. The aggressive drop resulted in no decrease in R2 correlation vs. the conservative drop.

**Baseline: imputed, unskewed**

Recall that the baseline best result is the linear regression model with R2 = 0.818 and RSME = 0.09.

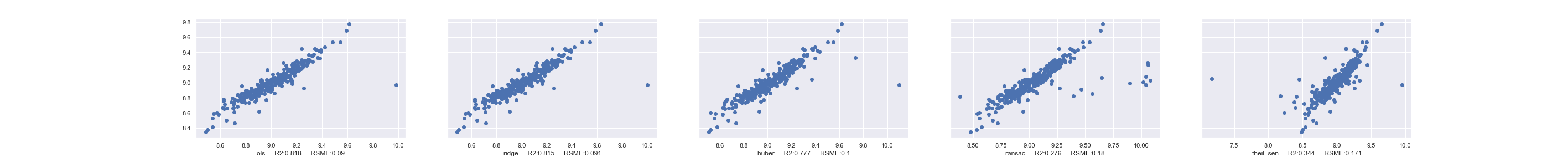


Figure 14: Revisiting the best performing model

**Results: Conservative drop**

The conservative drop had negligible impact on the linear model: its R2 value dropped from 0.818 to 0.816, while its RSME increased from 0.09 to 0.091. Surprisingly, the ransac model improved *significantly*: its R2 jumped from 0.276 to 0.78, while its RSME dropped from 0.18 to 0.099 – the outliers have predominantly disappeared from the plot. No change to the ridge model. The huber and theil sen models decreased slightly in prediction performance.

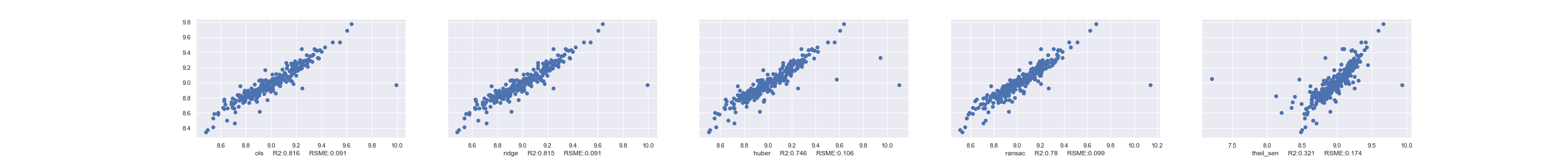


Figure 15: Regression Result after dropping features from Conservative List

**Results: Aggressive drop**

The aggressive drop had no impact on the linear regression model vs. the conservative drop. In total vs. baseline, 9 features were dropped out of 57 in the imputed, unskewed data set (16%) with negligible effect on the linear regression’s R2 prediction correlation and RSME value. The ridge model improved fractionally from R2: 0815 to R2: 0816. The Huber and ransac models dropped slightly in prediction performance, while the theil sen model significantly deteriorated from R2: 0321 to 0.227.

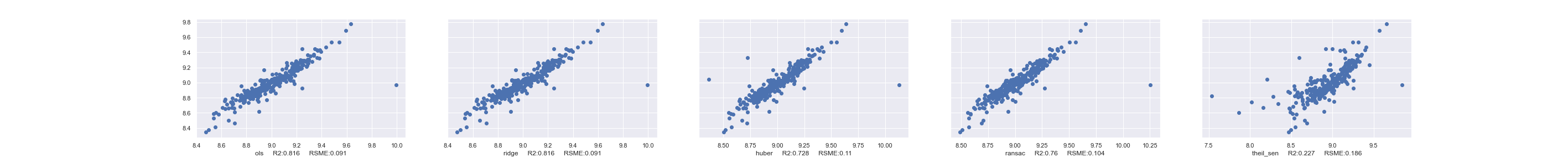


Figure 16: Regression Result after dropping feature from Aggressive List

## Hyper Aggressive

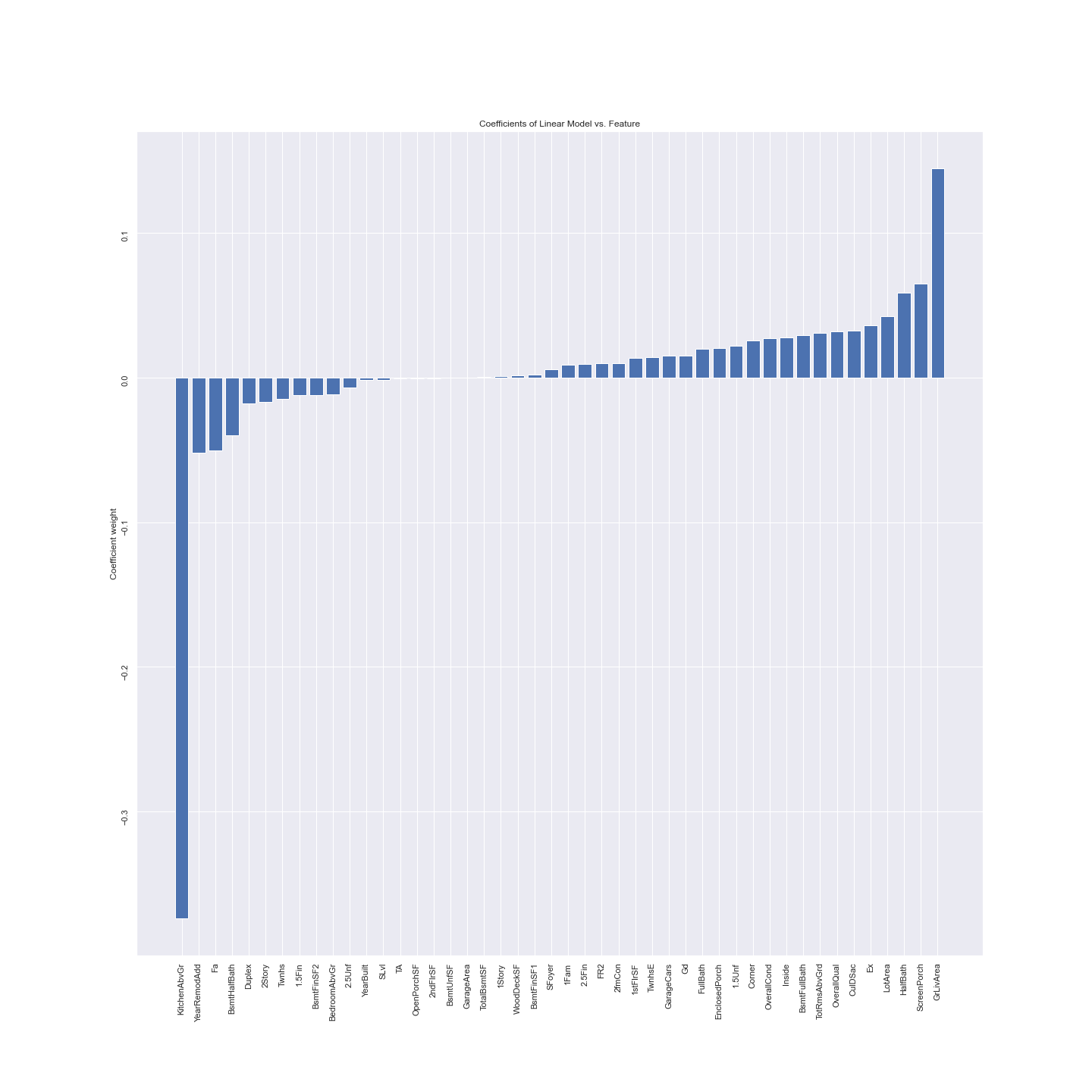
Examining the features weights in this model we see that after aggressive dropping we see a number of features that have no impact on SalePrice per the below plot of the model coefficients. This section explores ‘what if’ these features are blindly dropped without applying domain knowledge or judgement to see how non-sensible the results become.

Figure 17: Feature weights in the model after aggressive dropping

There are six features that have almost zero impact on the SalePrice even after the aggressive feature drop are listed below. This will form the hyper aggressive drop list. *Hyper Aggressive* drops the list in *Aggressive*  plus:

* TA: Average/Typical exterior material quality
* OpenPorchSF: Open porch area in square feet
* 2ndFlrSF: Second floor square feet
* BsmtUNfSF: Unfinished square feet of basement area
* GarageArea: Size of garage in square feet
* TotalBsmtSF: Total square feet of basement area

### Results

After dropping the *Hyper Aggressive List*, the SciKitLearn linear regression model and the ridge regression model are the best prediction models with R2: 0.833 and RSME: 0.086. Huber and ransac both improved significantly and now have R2 values uplifted from 0.7 to 0.8. Theil sen still performed the worst.

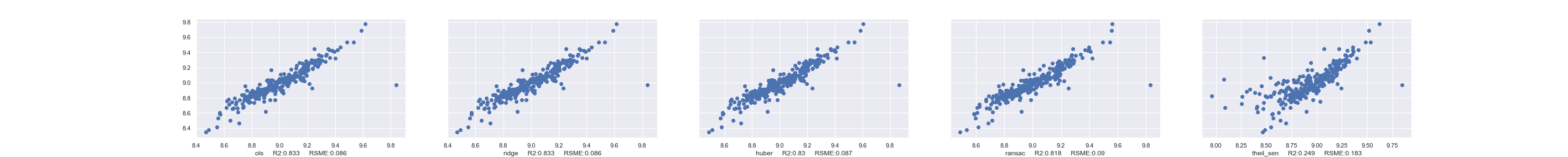


Figure : Regression Result after dropping feature from the Hyper Aggressive List

The feature weights plot reveals that there is still one feature with no effect on the prediction. However, notice that it is BsmtFinSF2, which in the previous aggressive drop has a weight approx. -0.5. The feature with the strongest negative weight is now FullBath and not KitchenAboveGr. KitchenAboveGr has a weight that is almost zero.

This underscores *how sensitive machine learning models are to the number of features in the training data* – the weights are relative to the features in the training set, dropping features changes that relative relationship. There is no objective ‘truth’ about what the worst three features are to SalePrice. Only the worst three features vs. the other features being evaluated against.

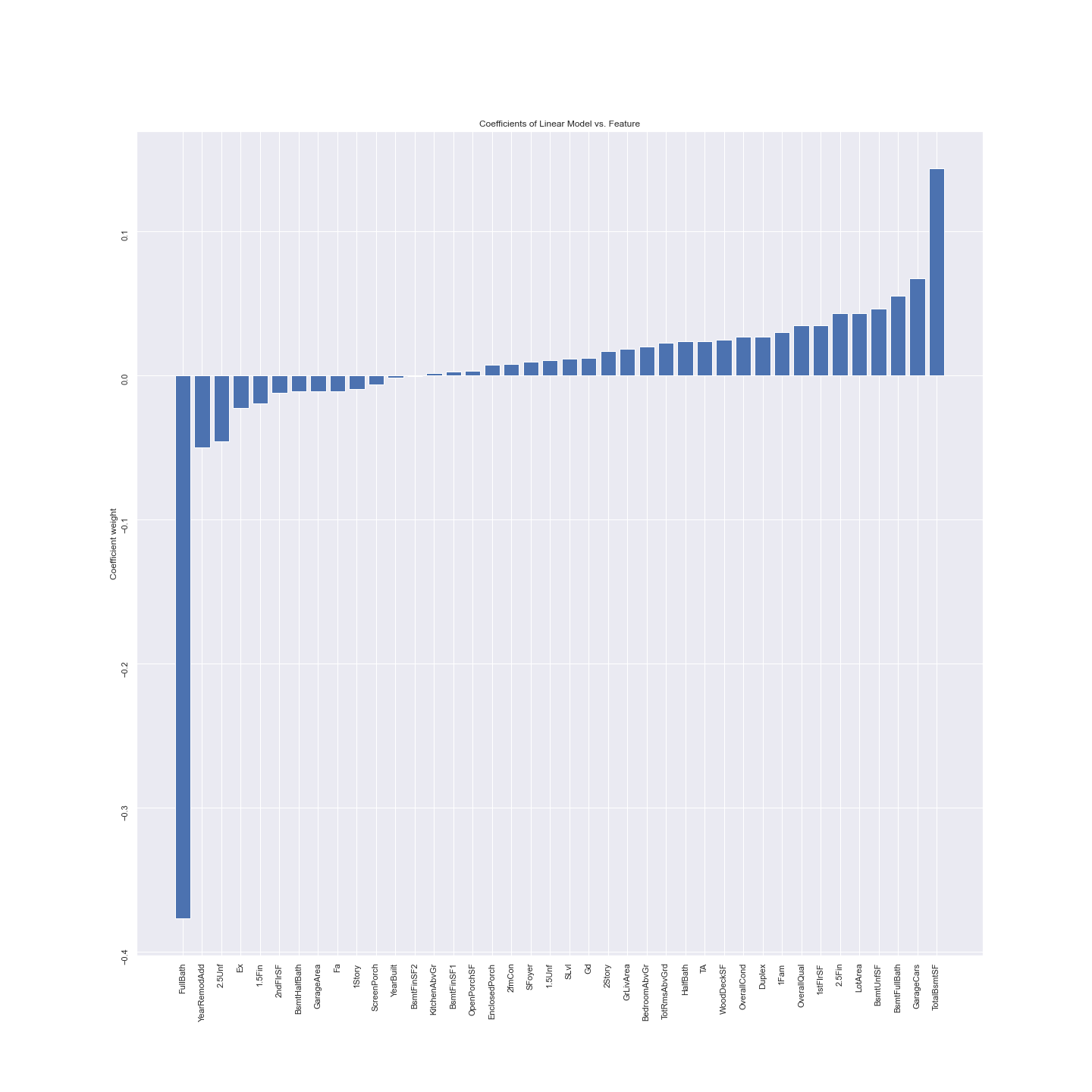


Figure : Figure 17: Feature weights in the model after hyper aggressive dropping

It doesn’t make intuitive sense that TotalBsmtSF, GarageCars and BsmtFullBath are the top three predictors of SalePrice. Again, this highlights that extreme care and judgement must be applied to feature selection and dropping. Blindly dropping features as was the case in Hyper Aggressive, produced a model that deceptively has a higher R2 score but with suspicious feature weightings.

The results from hyper aggressive feature dropping are rejected.

# Conclusion

This assignment created a SalePrice prediction model based on the training data set. The best prediction model is the default SciKitLearn Linear regression model because it exhibited the highest R2 correlation score of 0.816 (after aggressively dropping features) and the lowest RSME (after normalising the skewed data) of 0.091. Aggressive feature dropping applied judgement to the selection of features dropped.

The top 3 features that positively impacts SalePrice *in this model* are:

* GrLivArea: Above grade (ground) living area square feet
* ScreenPorch: Screen porch area in square feet
* HalfBath: Half baths above grade

The bottom 3 features that negatively impacts SalePrice *in this model* are:

* KitchenAbvGr: Kitchen above grade
* YearRemodAdd: The number of years since remodelled
* Fa: Fair exterior material quality

To achieve this model the training and test data set underwent these pre-processing steps: ‘Missing’ value cleaning, data-type recasting, normalisation via boxcox, one-shot-encoding, and re-baselining of ‘Year’ data. Five linear regression models were evaluated: SciKitLearn’s default linear regression, ridge, huber, ransac, thiel sen.

Imputing the ‘Missing’ data had marginal improvements in the prediction result vs. dropping the rows with the ‘Missing’ data. Unskewing or normalising the data via boxcox had a significant improvement on the prediction result. Conservatively and Aggressively dropping features had immaterial impact on the prediction performance of the model: likely because I correctly estimated the features with the least impact on the models by inspected the scatter plots of the unskewed, imputed training data set.

If this model were to be used in production the lmbda values from the boxcox transformation needs to be applied to the incoming features used in the prediction. The predicted sale price then needs to be inversely transformed with its lmdba value to set the units back to dollars. Finally, to use the model in production the model’s coefficients need to be outputted.

# Appendix

## Feature names in the training and test data set.

MSSubClass: Identifies the type of dwelling involved in the sale.

20 1-STORY 1946 & NEWER ALL STYLES

30 1-STORY 1945 & OLDER

40 1-STORY W/FINISHED ATTIC ALL AGES

45 1-1/2 STORY - UNFINISHED ALL AGES

50 1-1/2 STORY FINISHED ALL AGES

60 2-STORY 1946 & NEWER

70 2-STORY 1945 & OLDER

75 2-1/2 STORY ALL AGES

80 SPLIT OR MULTI-LEVEL

85 SPLIT FOYER

90 DUPLEX - ALL STYLES AND AGES

120 1-STORY PUD (Planned Unit Development) - 1946 & NEWER

150 1-1/2 STORY PUD - ALL AGES

160 2-STORY PUD - 1946 & NEWER

180 PUD - MULTILEVEL - INCL SPLIT LEV/FOYER

190 2 FAMILY CONVERSION - ALL STYLES AND AGES

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl Gravel

Pave Paved

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

ExterQual: Evaluates the quality of the material on the exterior

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

Po Poor

BldgType: Type of dwelling

1Fam Single-family Detached

2FmCon Two-family Conversion; originally built as one-family dwelling

Duplx Duplex

TwnhsE Townhouse End Unit

TwnhsI Townhouse Inside Unit

LotConfig: Lot configuration

Inside Inside lot

Corner Corner lot

CulDSac Cul-de-sac

FR2 Frontage on 2 sides of property

FR3 Frontage on 3 sides of property

OverallQual: Rates the overall material and finish of the house

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

OverallCond: Rates the overall condition of the house

10 Very Excellent

9 Excellent

8 Very Good

7 Good

6 Above Average

5 Average

4 Below Average

3 Fair

2 Poor

1 Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

BsmtFinSF1: Type 1 finished square feet

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

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

BedroomAbvGr: Bedrooms above grade (does NOT include basement bedrooms)

KitchenAbvGr: Kitchens above grade

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Fireplaces: Number of fireplaces

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

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

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

## Feature weights vs. Feature

This is an export of the model coefficients for the default SciKitLearn linear model which has R2: 0.816 after aggressive feature dropping.

[(-0.3734872798555275, 'KitchenAbvGr'), (-0.051771295253718916, 'YearRemodAdd'), (-0.05026859494818263, 'Fa'), (-0.03982719654142506, 'BsmtHalfBath'), (-0.018081318009989522, 'Duplex'), (-0.01689225239913952, '2Story'), (-0.014867339528942099, 'Twnhs'), (-0.012196090616225291, '1.5Fin'), (-0.01213043848650553, 'BsmtFinSF2'), (-0.011722463334269491, 'BedroomAbvGr'), (-0.007188628269582042, '2.5Unf'), (-0.0016479946247560653, 'YearBuilt'), (-0.0014747299421606092, 'SLvl'), (-0.0007485274183368379, 'TA'), (-0.0005806192700200294, 'OpenPorchSF'), (-0.0005037269277154845, '2ndFlrSF'), (-0.0003764092849373459, 'BsmtUnfSF'), (8.463047158928254e-05, 'GarageArea'), (0.0002213818698828733, 'TotalBsmtSF'), (0.0009099171758397131, '1Story'), (0.0015492388259483309, 'WoodDeckSF'), (0.0019496941079335136, 'BsmtFinSF1'), (0.005508336218518583, 'SFoyer'), (0.008721618373679131, '1Fam'), (0.009246357468438809, '2.5Fin'), (0.009823109935782995, 'FR2'), (0.01001182530754149, '2fmCon'), (0.013671113790862782, '1stFlrSF'), (0.01421521385773139, 'TwnhsE'), (0.015008724901592715, 'GarageCars'), (0.015239336279579904, 'Gd'), (0.01984853278156071, 'FullBath'), (0.02052551449378043, 'EnclosedPorch'), (0.022087090364317597, '1.5Unf'), (0.02537586540210916, 'Corner'), (0.027328256765655385, 'OverallCond'), (0.02772181371300706, 'Inside'), (0.029403867815887566, 'BsmtFullBath'), (0.03086802089831474, 'TotRmsAbvGrd'), (0.031836831921297576, 'OverallQual'), (0.032106694387762424, 'CulDSac'), (0.03577778608693005, 'Ex'), (0.04210612719469776, 'LotArea'), (0.05826818635369496, 'HalfBath'), (0.06482861468878101, 'ScreenPorch'), (0.14433500777418887, 'GrLivArea')]

1. Please refer to the Appendix for the feature descriptions [↑](#footnote-ref-1)
2. Please refer to the appendix for the feature descriptions. [↑](#footnote-ref-2)