

A

PROJECT REPORT

ON

“KAGGLE COMPETITION:

HOUSE PRICES: ADVANCED REGRESSION

TECHNIQUES”

VIRAJ BHAKTA

TABLE OF CONTENTS

	PAGE
1. PROBLEM STATEMENT	3
2. INTRODUCTION	3
3. GOAL	3
4. DATASET DESCRIPTION	3
5. METRICS	4
6. EXPOLATARY DATA ANALYSIS	5
7. WORKFLOW	10
8. RESULT	14
9. CONCLUSION	16

Kaggle Competition: House Prices: Advanced Regression Techniques

PROBLEM STATEMENT

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

INTRODUCTION

- Predicting the residential house price can guide real estate agents during the decision making and pricing process, therefore assistant their customers to find the best selling price for their houses.
- Industry Leaders also use the prediction of the house price for various purposes.
- Zillow uses the prediction of the house price to provide guidance to its online selle and buyer.
- The objective of this project is to predict the final price of a house given 79 explanatory variables that describe almost all aspects of a house in Ames, Iowa.
- This project is from a Kaggle playground competition.

GOAL

The Ultimate goal of this project is to train a model based on the given data to predict the house price as accurate as possible evaluated by the Kaggle Leaderboard.

DATASET DESCRIPTION

There are four files, which will be used during the competition.

File Descriptions:

- train.csv - the training set
 - Training Dataset Consists of 1460 observations, each with 81 features including the sale price of the house.
 - The training dataset is mixed with categorical and numerical features as well as some missing values.
- test.csv - the test set
 - The test dataset has 1459 observations, each with 80 features.

- Our task is to fill the sale price of the test dataset using model trained from training dataset.
- data_description.txt - full description of each column, originally prepared by Dean De Cock but lightly edited to match the column names used here
- sample_submission.csv - a benchmark submission from a linear regression on year and month of sale, lot square footage, and number of bedrooms

Data fields

- There are 79 explanatory variables in dataset. Here's a brief version of what you'll find in the data description file.
- 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)

METRICS

Root Mean Squared Logarithmic Error (RMSLE) is root mean square error of log transformed predicted and log transformed actual values.

The Root Mean Squared Logarithmic Error (RMSLE) will be used to as the evaluation metrics for this project.

$$\text{RMSLE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

n is the total number of observations in the (public/private) data set,

\hat{y}_i is your prediction of target, and

y_i is the actual target for i .

$\log(x)$ is the natural logarithm of x .

The value of RMLSE is higher when the difference between predicted and actual house prices is larger.

EXPOLATORY DATA ANALYSIS

1) MISSING VALUES:

Here we have drawn a heat map of missing data by each feature to check how many missing values we have for every column. But from these figure we cannot get clear idea about every columns. So we have to plot another graph to check how many missing values do we have?

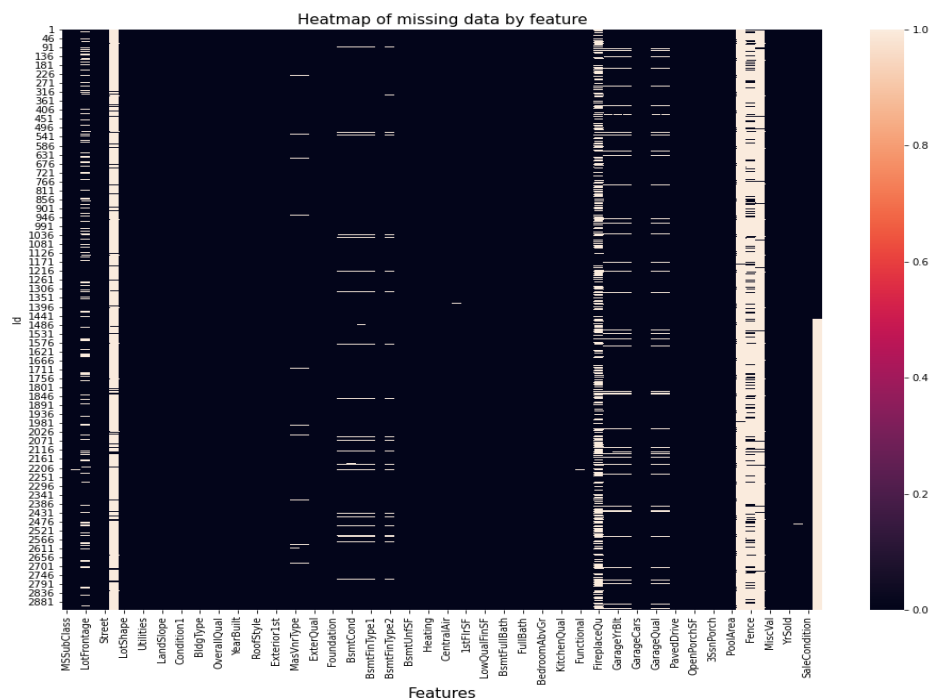
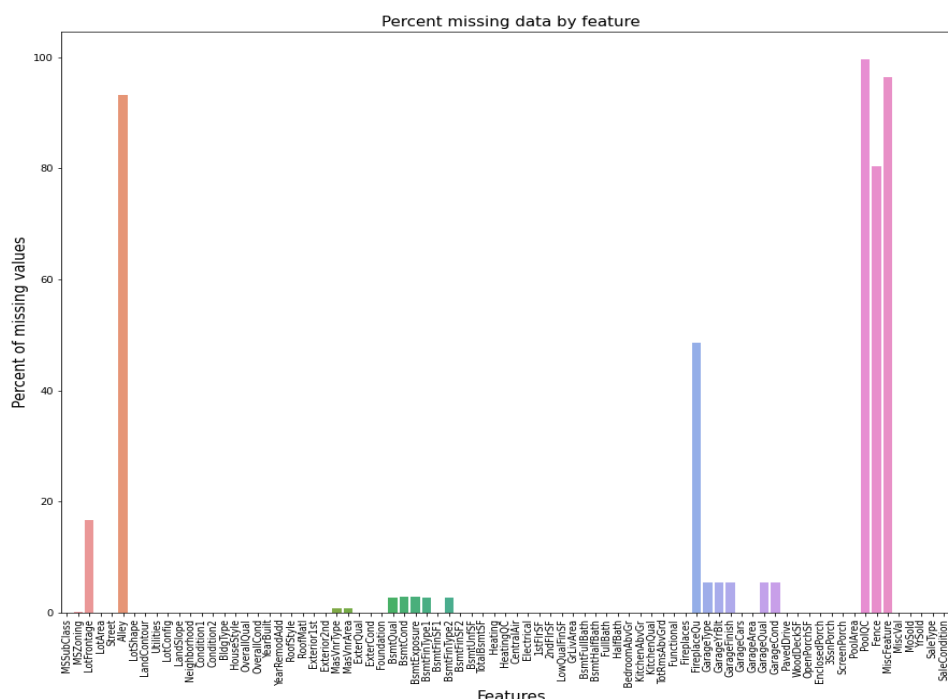


Fig 1. Heat map of Missing values



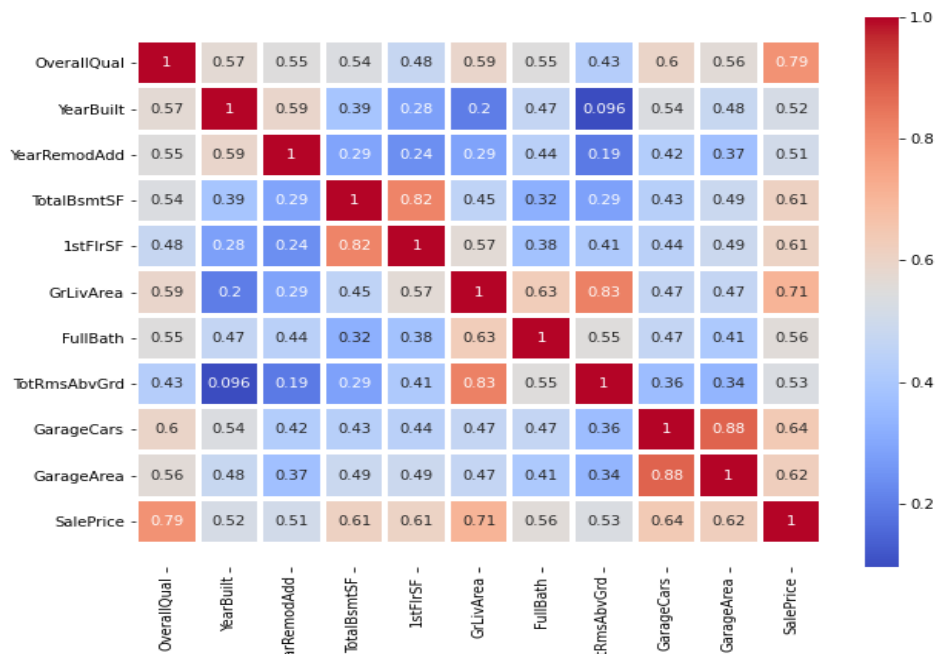


Fig 3. Correlation

Then I have used `seaborn.regplot()` to check for linear relationship with our target variable which will be determined through regression line.

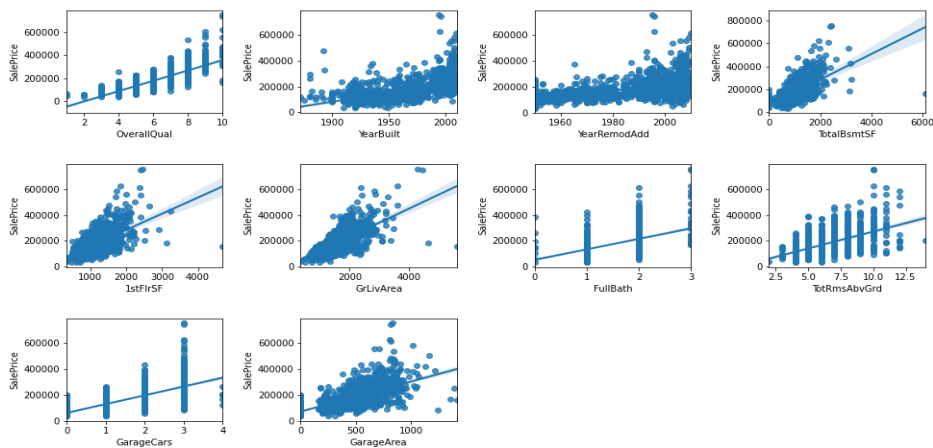


Fig 4. Reg plot of Highly Correlated Feature with Sale Price

3) Skewness:

It is a measure of how much the probability distribution of random variable deviates from normal distribution.

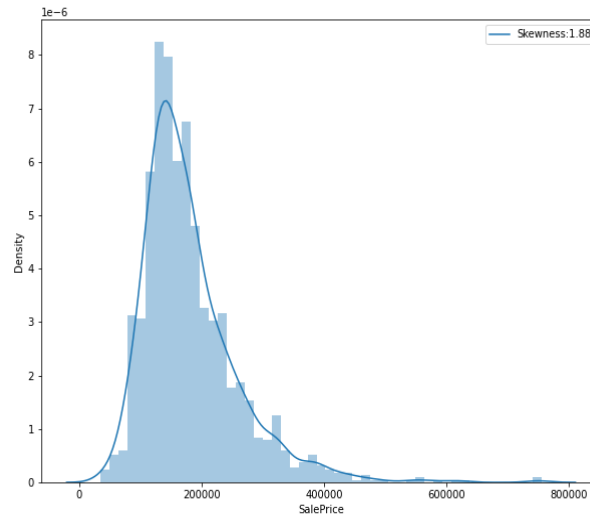


Fig 5. Original Sale Price

From the above figure 5 we can see that the histogram of our target variable i.e. SalePrice is skewed towards left. From figure, we can see that skewness = 1.88. So with these condition, we have applied log transformation to normalize the data of our target variable. Below figure 6, we can see that our Skewness is decreases to 0.12.

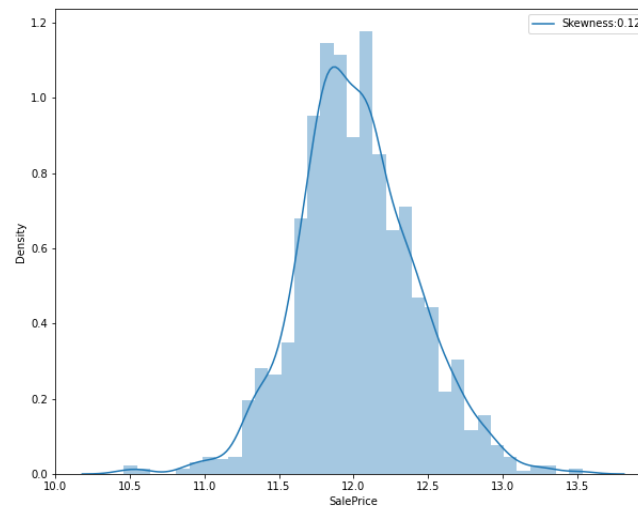


Fig 6. Log Transformation of Sale Price

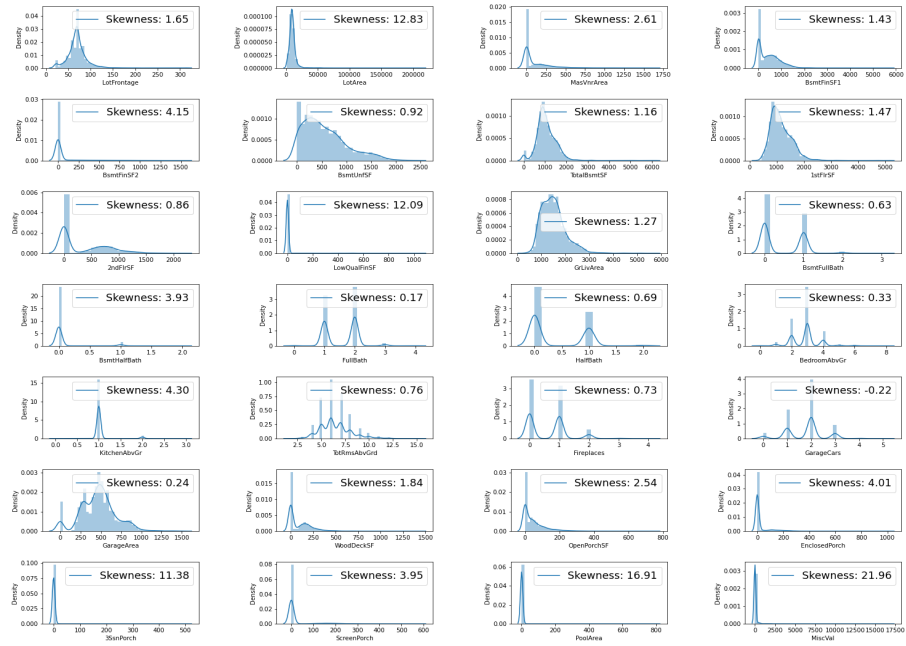


Fig 7. Skewness before Log Transformation

Above figure 7 show us about the skewness of numerical data. So we have to use log transformation to decrease their skewness. Below figure 8 clearly illustrate that we were able to normalize the data of numerical variables.

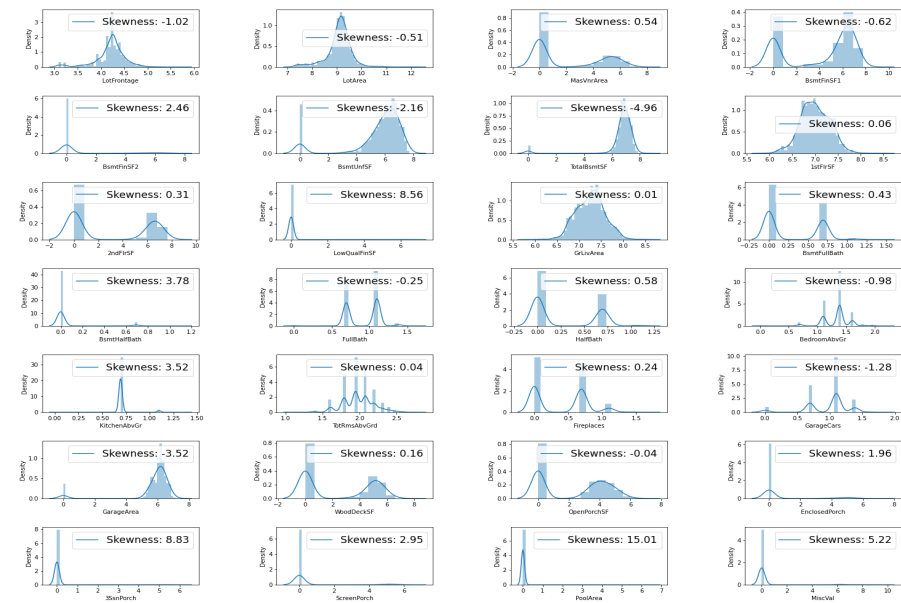
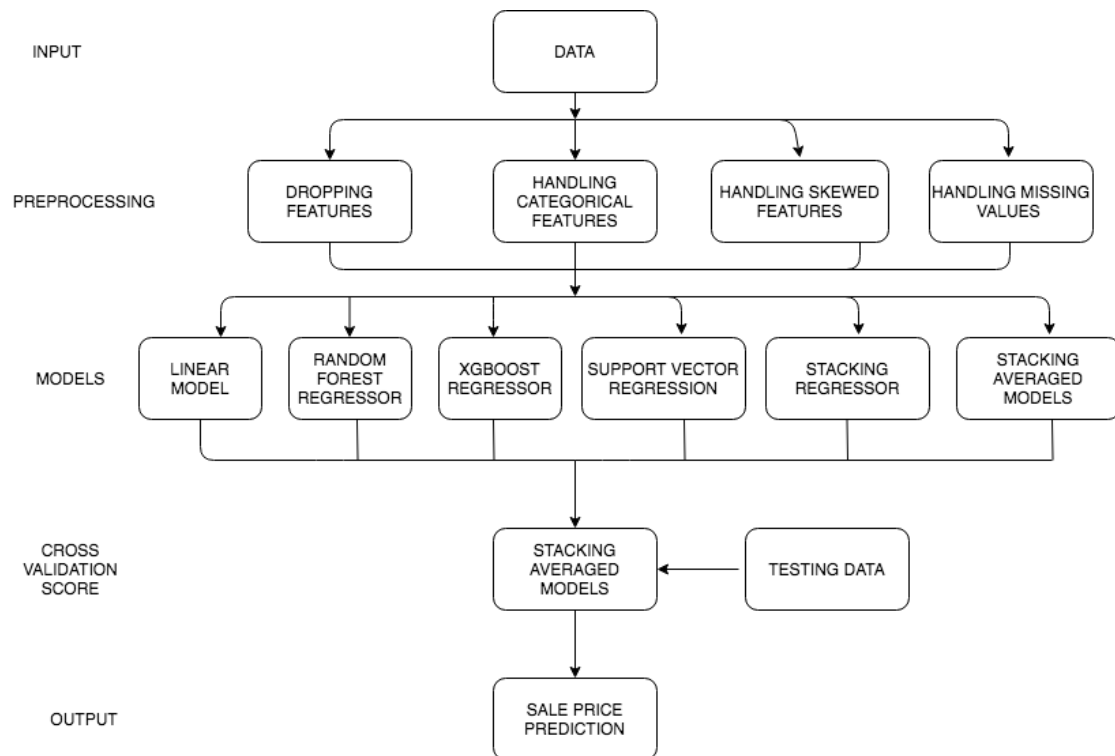


Fig 8. Skewness after Log Transformation

WORKFLOW



INPUT

- Merged training dataset and testing dataset.
- Dataset consist of 2919 observations with 81 Features.

DATA PREPROCESSING

1) DROPPING FEATURES

- a. Dropping the features which have missing values greater than 20 percent.
- b. Following are the feature which will be drop:

['Alley', 'FireplaceQu', 'PoolQC', 'Fence', 'MiscFeature', 'SalePrice']

2) HANDLING MISSING VALUES

Following are the features which are having missing values:

['MSZoning', 'LotFrontage', 'Utilities', 'Exterior1st',
'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'BsmtQual', 'BsmtCond',
'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Electrical', 'BsmtFullBath',

'BsmtHalfBath', 'KitchenQual', 'Functional', 'GarageType',
'GarageYrBlt','GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
'GarageCond','SaleType']

3) STEPS

- a. First I have deal with all Bsmt Features that are having missing values. So, I have replaced NAN value by NA if it is dtype = object otherwise I replaced it with 0.
- b. For BsmtFinType2 I have created a bucket and then replace the NAN by mode.
- c. Again if there are any missing values available then I have fill data by using Mode for dtype = object and Mean for dtype = int64,float64.
- d. Then I have repeated same steps(a and c) for handling other missing values of garage features and LotFrontage.

4) HANDLING CATEGORICAL FEATURES

- Convert the categorical code into the order.
 - a. When categorical data is converted to one hot encoding we will have effect on accuracy, so I decided to convert into the order format.
 - b. For these I have used pandas library of **pandas.api.types.CategoricalDtype(categories = None, ordered = None)** : This class is useful for specifying the type of Categorical data independent of the values, with categories and orderness.
- After these I have used **pandas.get_dummies()** which is used for data manipulation. It converts categorical data into dummy or indicator variables.

5) HANDLING SKEWED FEATURES

- First, I have drawn a distplot of skewed features for checking their skewness or where our data is lying towards left or right.
- After checking skewness of features, I have used log transformation for decreasing the skewness on skewed features.
- Then again plotted distplot to check how many skewness we have decrease.

TRANSFORMATION

1) ROBUST SCALER:

- It use percentiles can be used to scale input variables that contain outliers.
- They are robust to outliers.
- It removes the median and scales the data according to the Interquartile Range (IQR).

MODELS

There are six models that I have trained on training data and test it on testing data. Following are the models that I have trained:

- Linear Model: Lasso Regression
- Random Forest Regressor
- XGBoost Regressor
- Support Vector Regressor
- Simple Stacking Model: Averaged Models
- Stacking Regressor

1) Cross Validation Score on training data (Scoring = r2)

Before Feature Engineering:

MODELS	Cross Validation Score
Linear Model: Lasso Regression	0.8882910850879332
Random Forest Regressor	0.8565763892635457
XGBoost Regressor	0.8862827850745246
Support Vector Regressor	0.8935570735290478
Simple Stacking Model: Averaged Models	0.9008365247911075
Stacking Regressor	0.8965350449952836

From the above calculation, when analyses I find out that Simple Stacking Model: Averaged Model was performing better than every single model.

2) Simple Stacking Model: Averaged Model

Averaging is a basic technique which is commonly used for problems with regression. In order to get a more consistent result, the predictions are simply averaged here, and while this approach is simple, it almost always provides better results than a single model.

We will construct a class that inherits from the classes of Scikit-Learn's BaseEstimator, RegressorMixin, and TransformerMixin for ease of use, since we just need to overwrite the fit and predict techniques in order to get a working model that can be used like any other model of Scikit-Learn.

The only difference is that we need to move the base models while constructing an entity that will then be trained in the fit method and used in the prediction method for forecasts. Following is the code:

```
averaged_models = AveragingModels(models = (lasso,svr_reg,
xgb2_reg,gbr_reg))
```

For Simple Stacking Model: Averaged Models I have used combination of Lasso regressor, Support Vector Regressor, XGBoost Regressor and Gradient Boosting Regressor.

FEATURE ENGINEERING

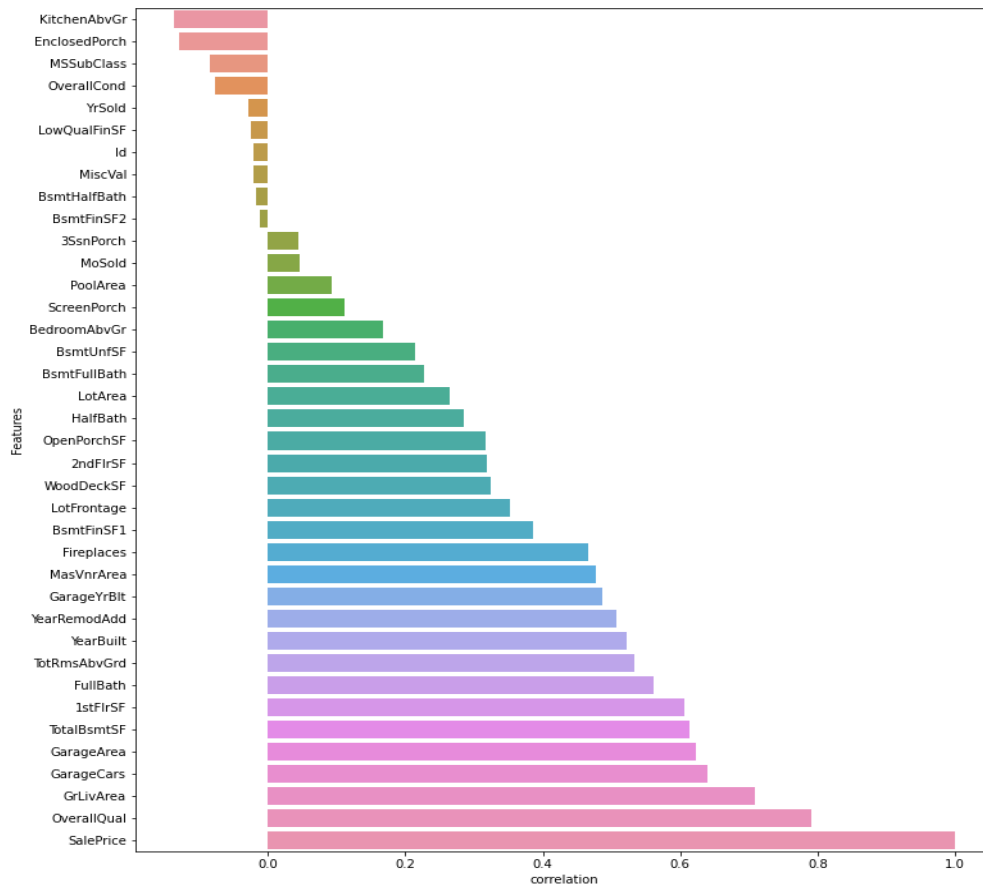
Below graph we will be using in Feature Engineering to improve our model. We will be dropping some columns, which are less correlated with target variable.

- In these Process, I have drop following feature:
['YrSold','LowQualFinSF','MiscVal','BsmtHalfBath','BsmtFinSF2','3SsnPorch','MoSold']
- Then I have followed same workflow, which I did during the creation of base models.
- After dropping some columns, I was able to improve my models as follows:

After Feature Engineering:

MODELS	Cross Validation Score
Linear Model: Lasso Regression	0.8927806501857067
Random Forest Regressor	0.8589694101475068
XGBoost Regressor	0.8870542936849423
Support Vector Regressor	0.8999152655571999
Simple Stacking Model: Averaged Models	0.9045028299349802
Stacking Regressor	0.8959452376914213

From the above calculation, when analyzed I find out that Simple Stacking Model: Averaged Model was performing better than every single model. So I decide to use this model for my final submission to Kaggle Competition.



RESULT

BEFORE FEATURE ENGINEERING






1) XGBOOST:

Rank: 1719

1717	Ronald Martis		0.13366	4	1mo
1718	KooKaik		0.13367	8	2mo
1719	Viraj Bhakta		0.13370	6	2d
1720	Tina #3		0.13371	10	1mo





2) SUPPORT VECTOR REGRESSOR:

Rank : 1459

1458	Tung Vo		0.12902	25	13d
1459	Viraj Bhakta		0.12903	7	~10s
Your Best Entry ↑ Your submission scored 0.12903, which is an improvement of your previous score of 0.13370. Great job!  Tweet this!					
1460	KAstev		0.12906	3	17d
1461	Sean Duan		0.12907	3	1mo






3) STACKING REGRESSOR

Rank: 970

970	Viraj Bhakta		0.12391	10	~10s
Your Best Entry ↑ Your submission scored 0.12391, which is an improvement of your previous score of 0.12903. Great job!  Tweet this!					
971	Lev Novitskiy		0.12392	50	17d
972	Bryan Cover		0.12396	6	2mo

4) SIMPLE STACKING MODEL: AVERAGED MODELS





Rank: 618

618	Viraj Bhakta		0.12126	11	~10s
Your Best Entry ↑ Your submission scored 0.12126, which is an improvement of your previous score of 0.12391. Great job!  Tweet this!					
619	rnakao		0.12127	6	2mo
620	oscar matias torres		0.12129	6	1mo
621	ura yukitaka		<> Ensemble predicti...	17	2mo

AFTER FEATURE ENGINEERING

1) SIMPLE STACKING MODEL: AVERAGED MODELS

Rank: 311

310	aoleo		0.11933	16	2mo
311	Viraj Bhakta		0.11934	13	~10s
Your Best Entry ↑ Your submission scored 0.11934, which is an improvement of your previous score of 0.12126. Great job!  Tweet this!					
312	HelenaHlz		0.11935	22	9d

CONCLUSION:

From these kaggle competition, I want to conclude that by doing these project I came across many new concepts like how can we deal with skewness, categorical data and numerical data. To deal with categorical data, we have used `pandas.get_dummies()` which converts categorical variables into dummy variables. Furthermore, we have used `RobustScaler()` transformation technique to transform the data. Then we started to create some base models like linear model, Random Forest Regressor, XGBoost Regressor, Support Vector Regressor, Stacking Regressor and Simple Stacking Model: Averaged Model. From these we were able to get best result with Simple Stacking Model: Averaged Model. Another thing that I came across was Feature Engineering, So after dropping some columns which are not correlated with our target variable, we were able to achieve better score. At last, by doing these project I was able to handle 79 Explanatory variables and learn new concepts that are related to data science field. And I was able get my rank in top 6% i.e. 311 out of 5196 Candidates.