The aim of this Assignment is to perform basic exploratory data analysis (EDA) for predicting the house prices, from the housing data set taken from kaggle.

The tasks performed in this notebook include:

A. Examination:

1.Deal with nan(Not a Number).

2.Feature selection.

B. Transformation

1.Normality check.

2.Convert strings to categorical values via Label Encoder.

3.Correlation matrix.

4.Dimensionality reduction.

C. Exploration :

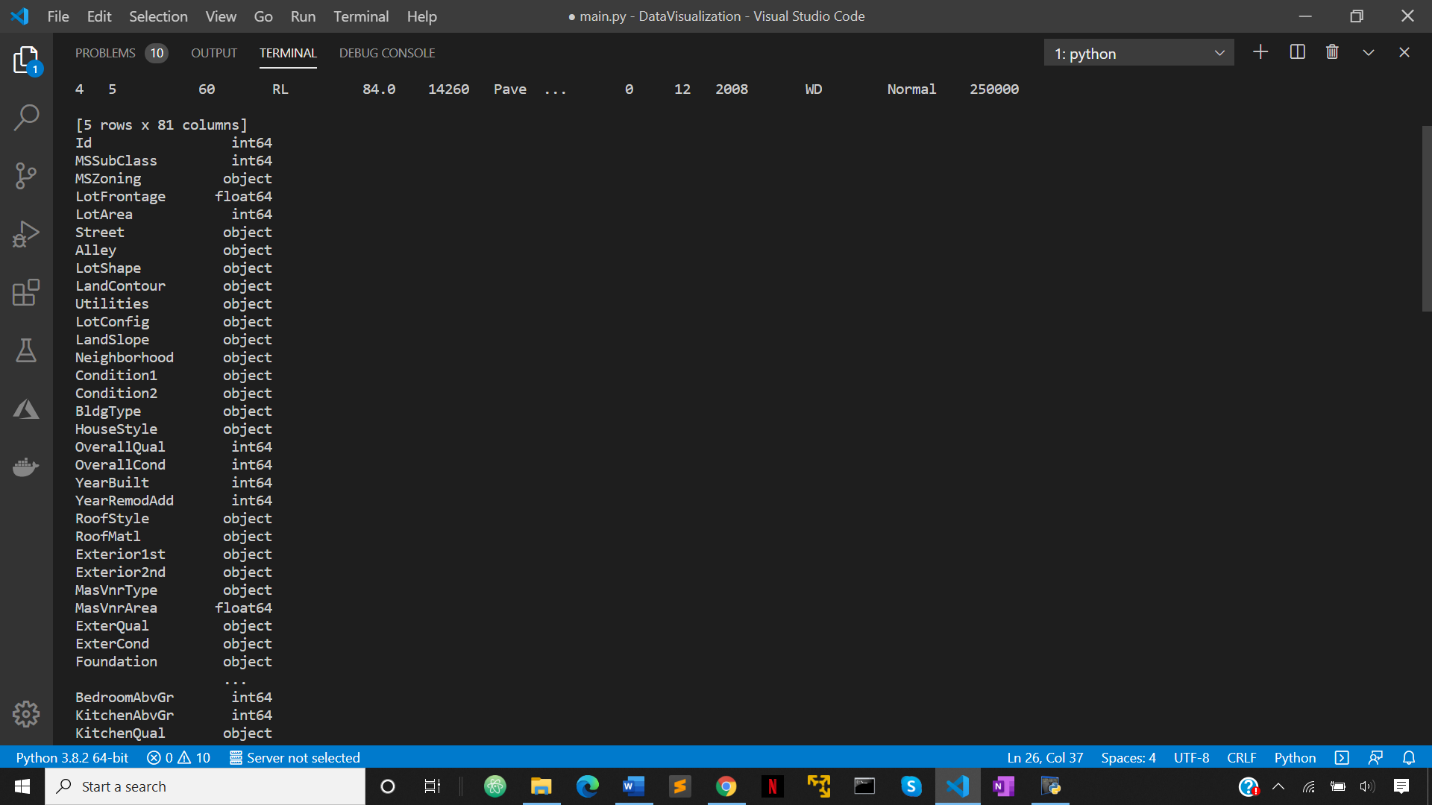
1.Feature importance.

2.Outlier deletion & normalization.

I also show the basic of the so-called 'stacked-ensemble model' to boost our prediction accuracy!

First, let's start with looking at what's inside of the given csv file.

Our Data consists of 81 columns and the details about the housing data can be found in data\_description.txt and there seems to be several data types.



1. Convert strings to categorical values via Label Encoder:

We need to convert our 'objects' which are essentially strings to categorical variables. To encode categorical variables we use Label Encoder.

Label encoding is simply converting each value in a column to a number.

2. Dealing with nan :

After converting strings to categorical values we need to eliminate nan which is essentially not a number and is not useful for the analysis.

training data+++++++++++++++++++++

LotFrontage: 259 nans

MasVnrArea: 8 nans

GarageYrBlt: 81 nans

testing data++++++++++++++++++++++

LotFrontage: 227 nans

MasVnrArea: 15 nans

BsmtFinSF1: 1 nans

BsmtFinSF2: 1 nans

BsmtUnfSF: 1 nans

TotalBsmtSF: 1 nans

BsmtFullBath: 2 nans

BsmtHalfBath: 2 nans

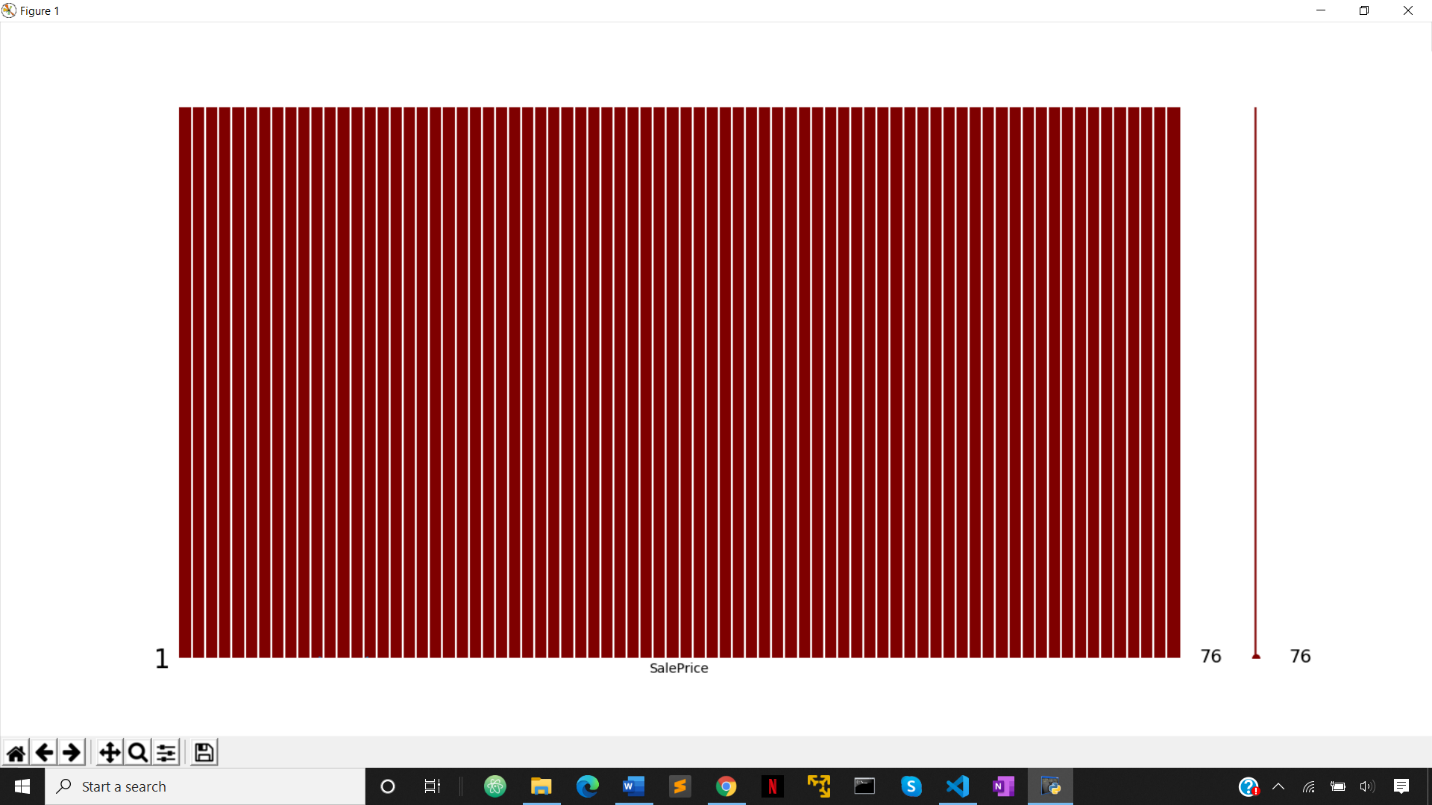
GarageYrBlt: 78 nans

GarageCars: 1 nans

GarageArea: 1 nans

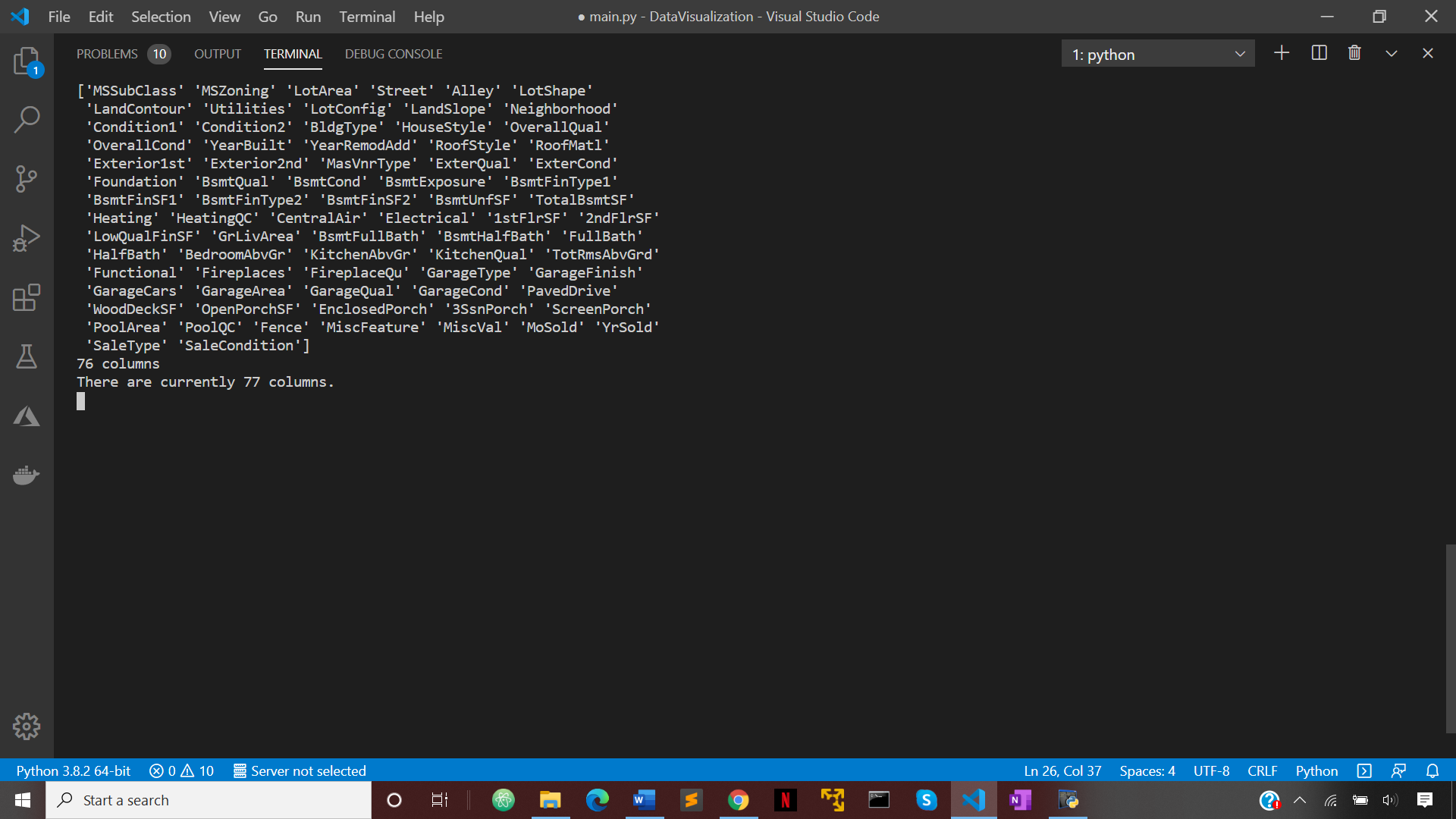
Actually, there are many columns in the test set to have nan. 'LotFrontage', 'MasVnrArea' and 'GarageYrBlt' have many nans in the both sets, so we just drop them.

For others, there are a few nans in each column, so we just replace them with the median of the corresponding column.



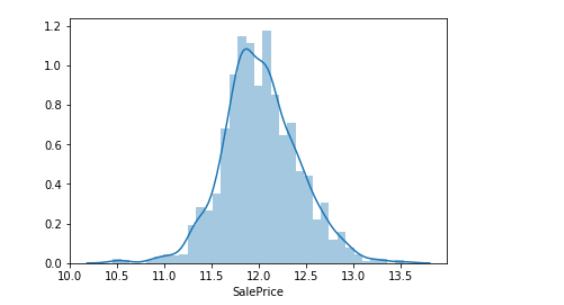
3. Normality check :

OK, there are no nan any more! Let's check our columns again. There are many '...SF'. Maybe we make a new feature which takes the sum of the all.



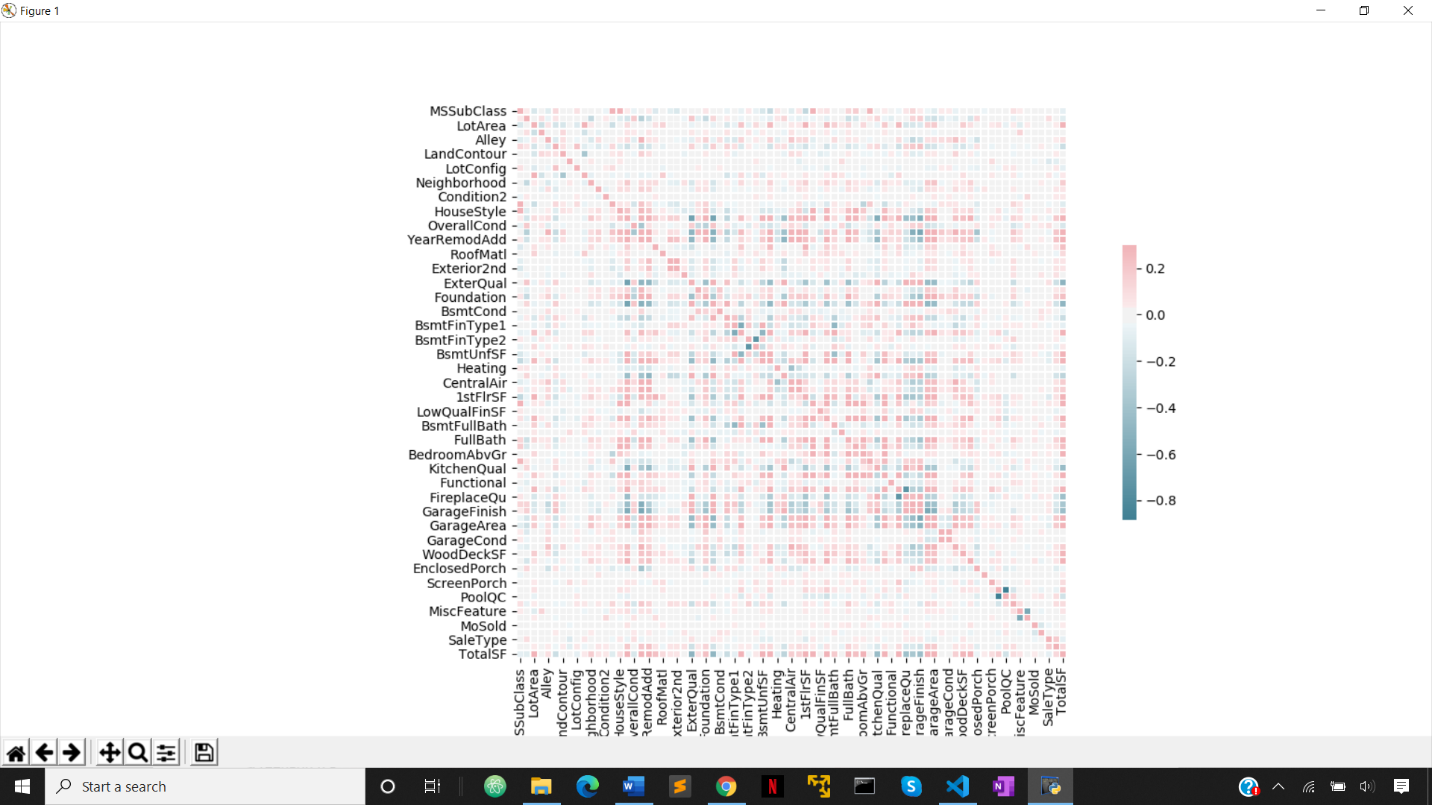
Now let's have a look at the target distribution. As this is a regression task, we want the target to be normally distributed.

We use log-transform to make them normally distributed.



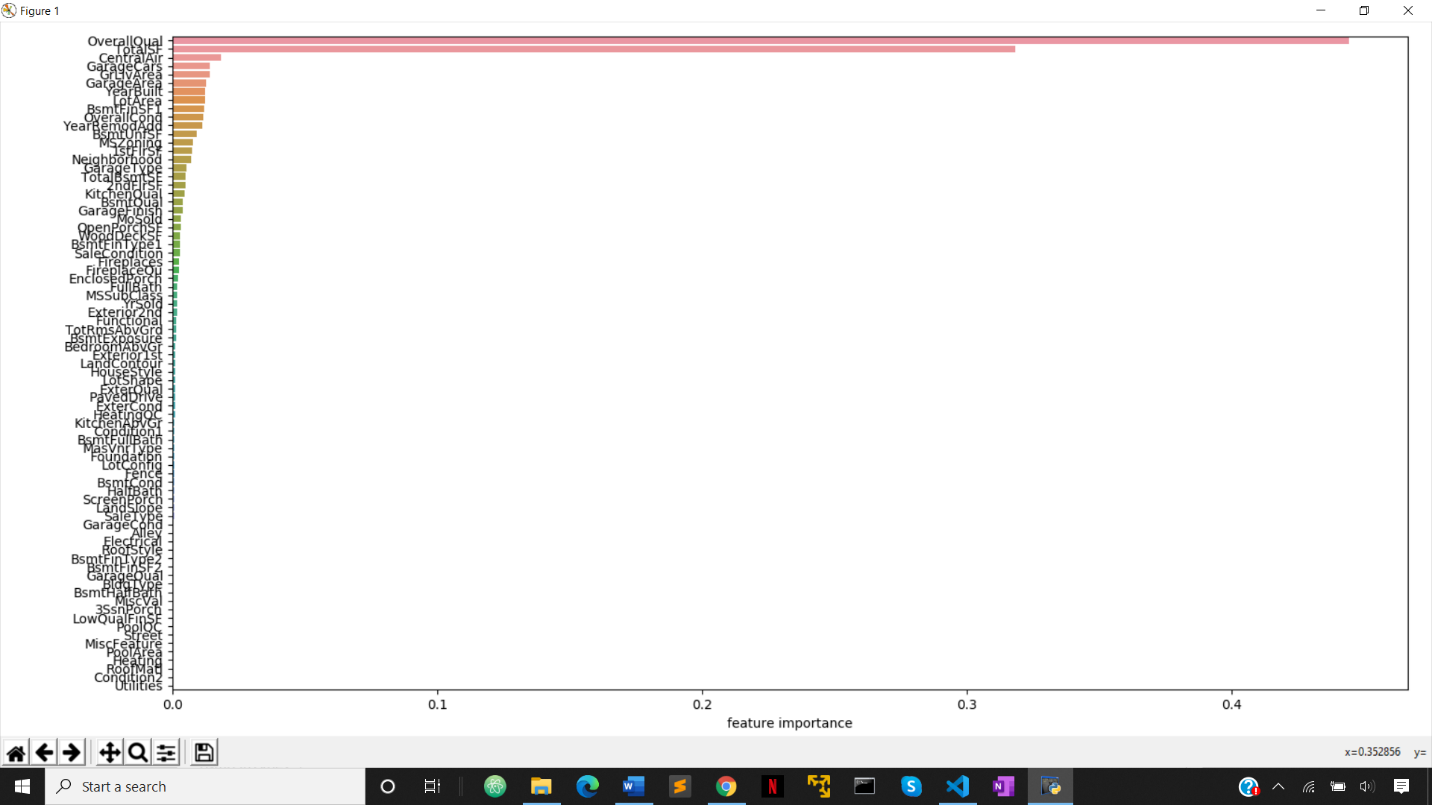
4. Correlation matrix :

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data.



5. Feature Importance :

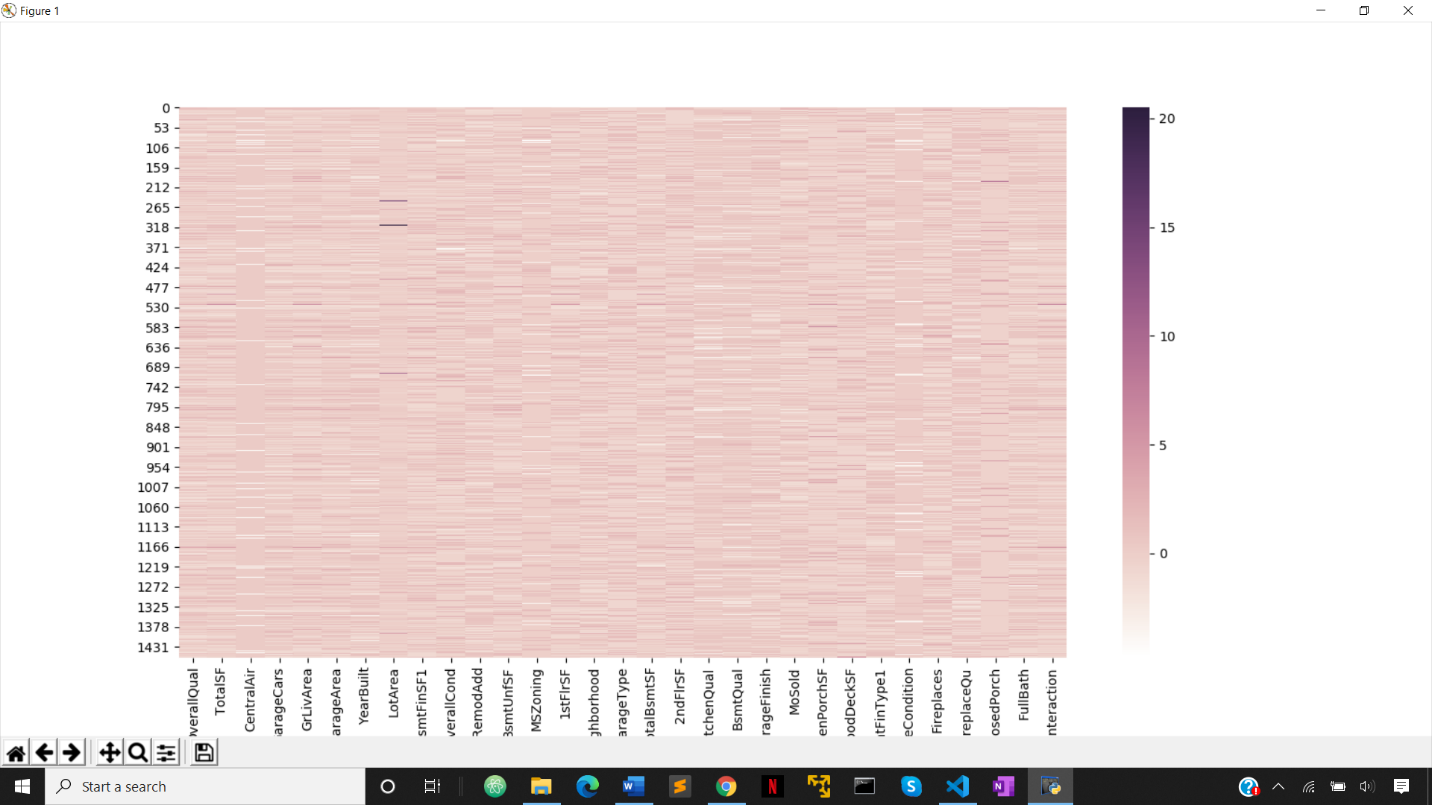
Which features are important? Let a random forest regressor tell us about it.



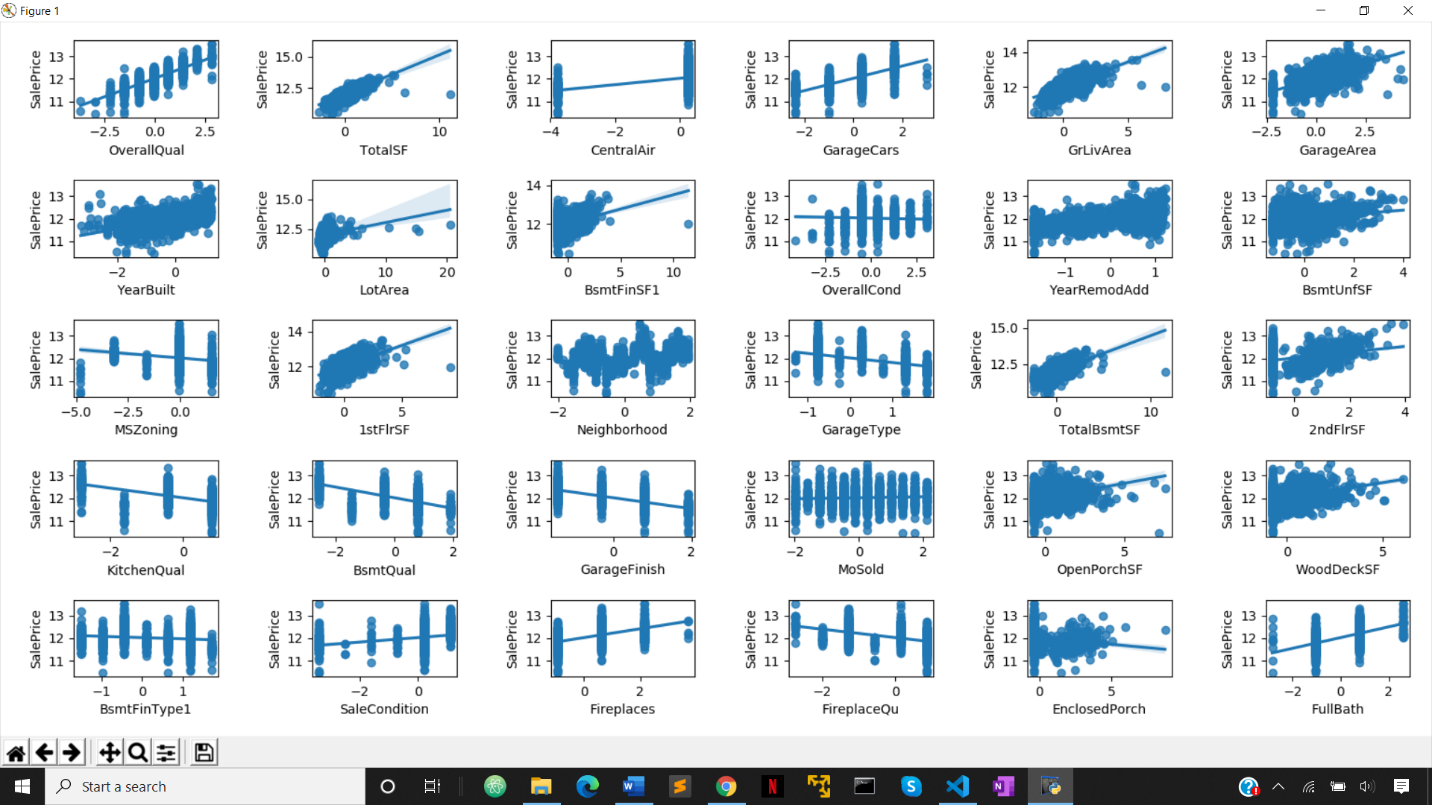
Surprisingly, only two features are dominant: 'OverallQual' and 'TotalSF'.

So instead of using all the 77 features, maybe just using the top 30 features is good enough (dimensionality reduction, in a way).

Here, we make a new feature called 'Interaction': simply the multiplication between the top 2 features. Also, we normalize the data via z-scoring.



Let' see how the important features are related to our target "SalePrice" using sns plot.



Basically we can see clear linear relationships in many panels. We do not see a lot of outliers, which is good.

Maybe let's just remove some outlier data points found in the 'totalSF' and 'GrLivArea', which are apparently out of the linear regression line.

Now it is time to have fun with the stacked-emsembling model!

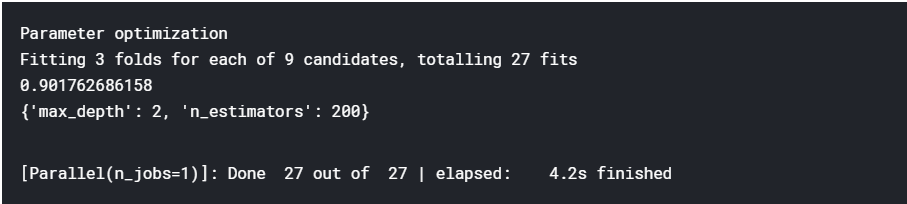
The intuitive idea of 'emsembling' is that we may be able to get better prediction performance if we average predictions by multiple models, rather than relying on a single model.

There are always pros and cons in machine learning algorithms, so averaging multiple model predictions may compensate one another.

Here we use three models: XGBoost, NN and SVR.

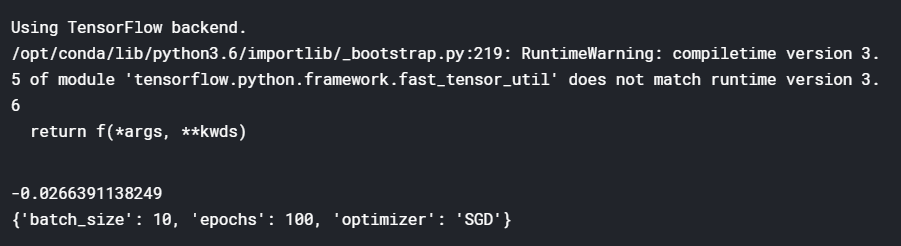
XGBoost:

There are some parameters in XGBoost that can be optimized. We use 'GridSearch' to explore which combination of parameters yield the best prediction in this dataset.



Moving on to neural network: we simply use Keras for easy implementation of multi-layer perceptron.

We can again take advantage of grid search, but that requires Keras' wrapper 'KerasRegressor' (or 'KerasClassifier') to utilize scikit\_learn's gird search. Here we try to select the best ones with respect to 'optimizer', 'batch\_size', and 'epochs'.



SVR (support vector regressor) is done easily using sklearn. Again, we use grid search to optimize some of the SVR's hyperparameters.

Q.What are Ensemble methods?

Ans. Emsemble methods are simply averaging predicted results across models to improve overall accuracy. This method is based on the idea that prediction by each model may be independent of one another and thus averaging predictions across models may compensate one another and yield better solution.

Stacking methods are part of emsemble methods, but they use weighted averaging of predictions by models.

The idea is, of course, you want to assign more weights on better models and less weights on worse ones, based on the dataset.

To determine the weights, we treat predictions by models as new features (predictors) and train a linear model to predict our targets.

Here is the very nice documentation about the stacked-emsemble model. 'https://blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/'

So let's first make a new matrix where each column represents prediction by each model.

We train a linear regressor for this new training matrix and predict our target! We use Lasso GLM to avoid overfitting.

We predict our target 'SalePrice', but do not forget to convert it back to ordinal scale! It has been in log-scale for normality.

Note that, here we used a grid-search to optimize hyperparameters, but it is still not good enough because explored parameter spaces are still narrow. We could still optimize hyperparameters and go further up.

Finally we output our results to output.csv predicting the house price.