House Prices Prediction

# Step 1: Competition Description



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 Armes, Iowa, this competition challenges you to predict the final price of each home.

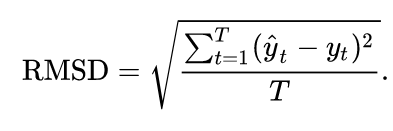
## practice Skills

Creative feature engineering

Advanced regression techniques like random forest and gradient boosting

## evaluation method

Based on the housing information data provided by the contestant in Ames, Iowa. I picked the Root Mean Square Error(RMSE):



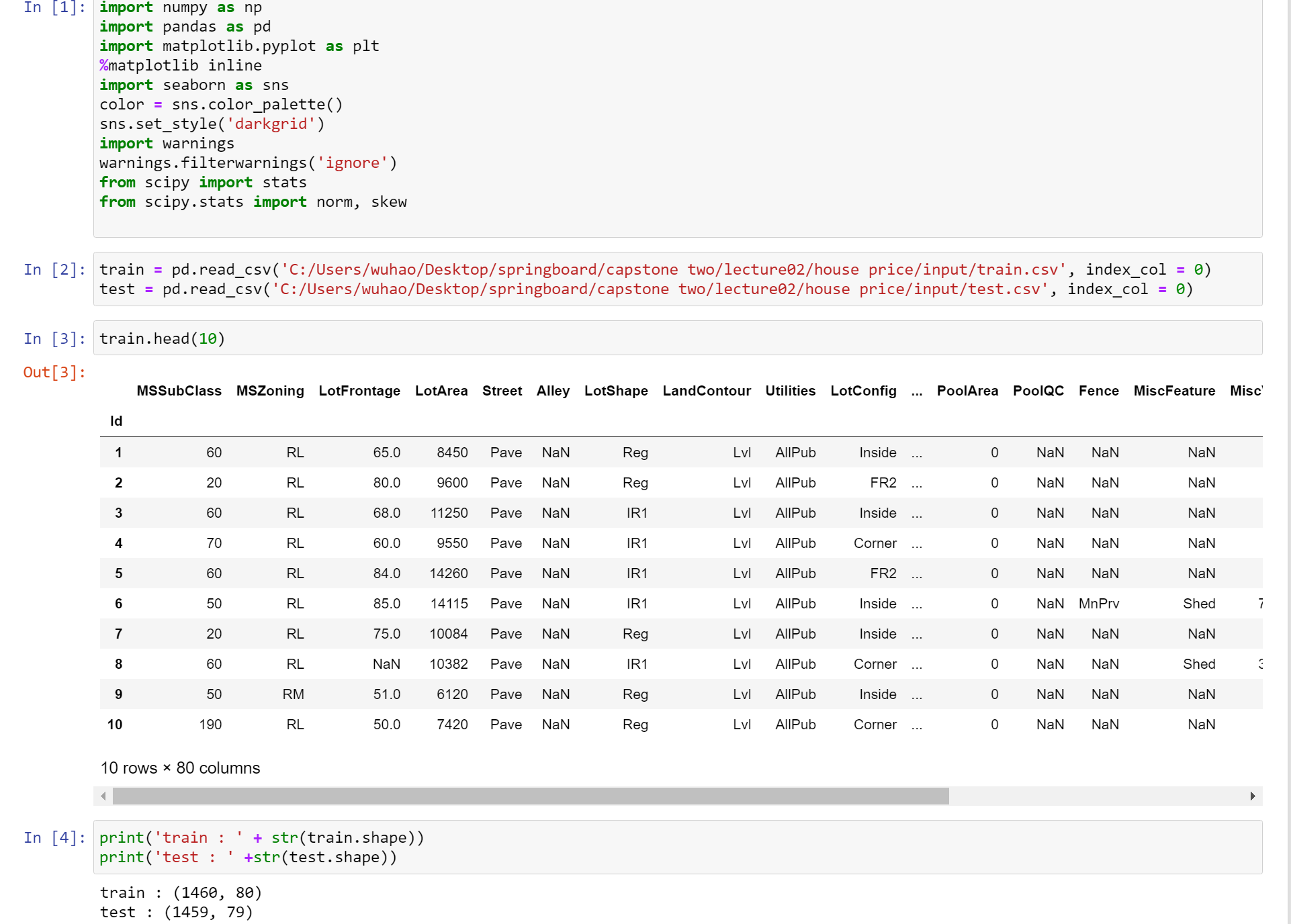
# Step 2: Data Collection

The Kaggle provided us with the data set needed for this competition, including the training data set(train.csv), test data set(test.csv), and the sample\_submission. I assume that the sample\_submssion\_data are the final accurate SalePrice.

Each data record represents the relevant information of each house. There are 1460 rows for train and test respectively. There are 79 feature columns of the data. 35 are numerical and 44 are categorical features. The detailed description of features is in the txt file.

# step 3: Data cleaning

## import data and some toolkit



## Exploratory Data Analysis

Many people may think that a data set with more than 80 columns will be a little difficult to start, but at first, we should understand the data at the beginning. We followed those two steps:

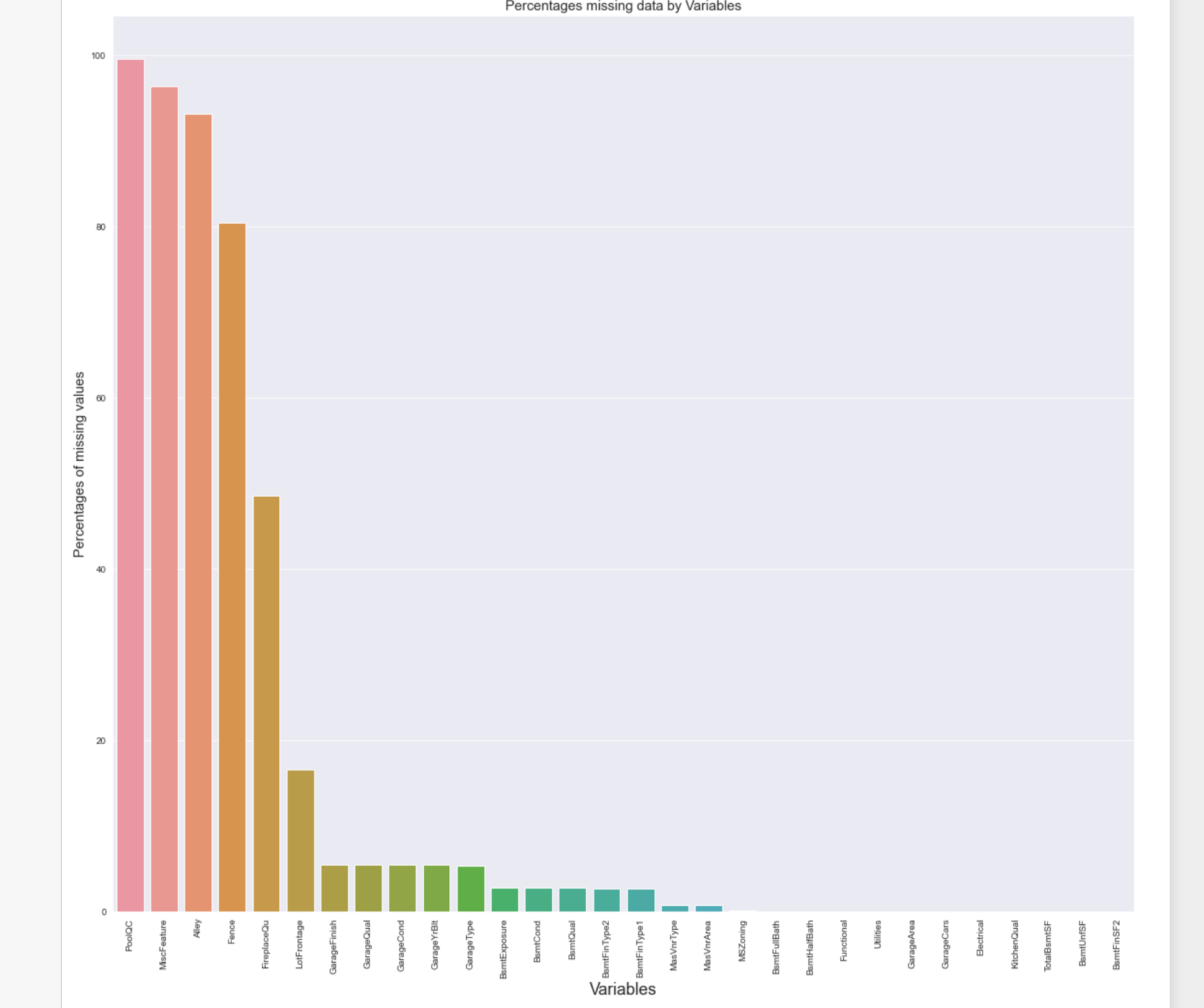
1. Observe the missing values and outliers of the data. We need to find out which features have missing values and which features have outliers.
2. Find the correlation map between the features and predict values. There are 80 variables, but not everyone should be used in the models. We will use a heatmap to help us to filter those features.

## 

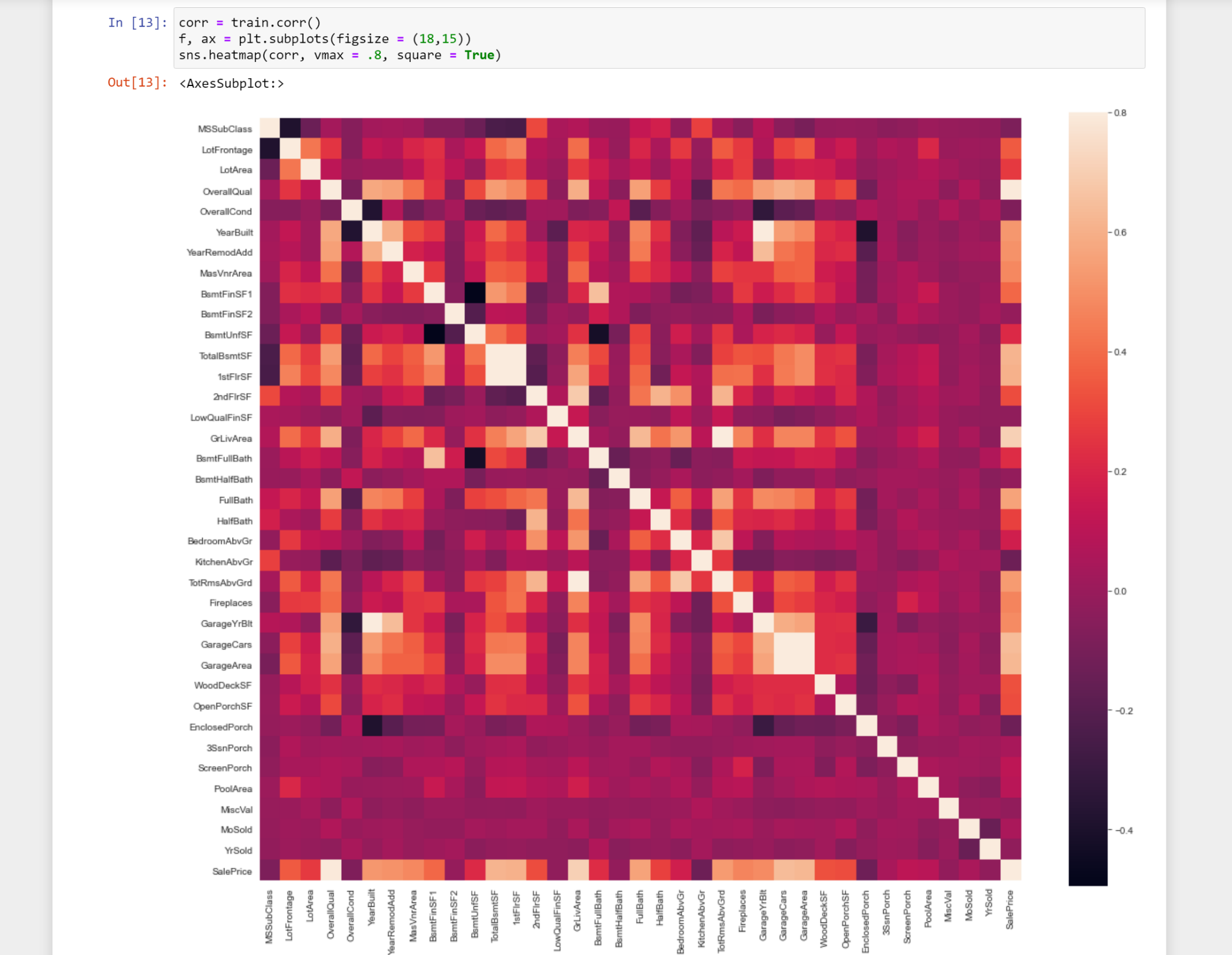
## Missing data



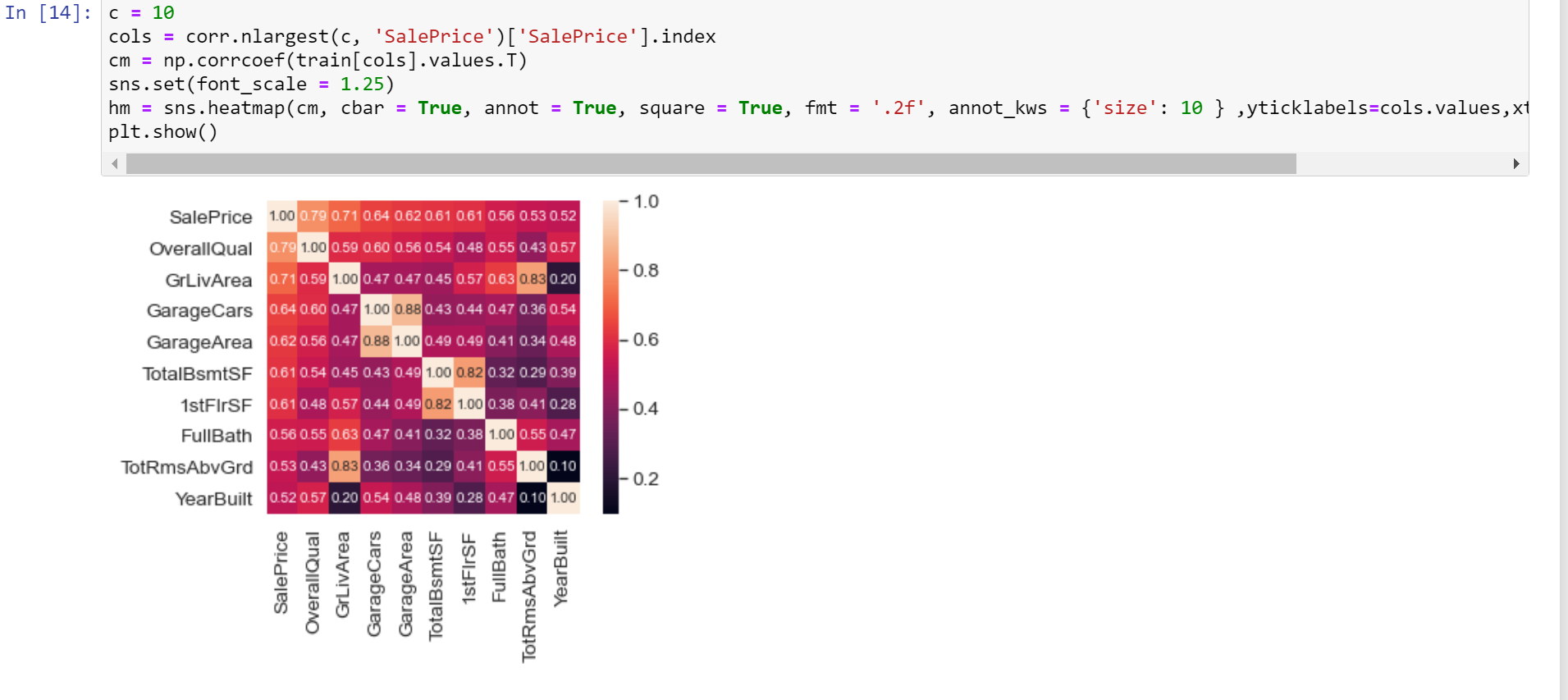
the distribution of missing data



## Data correlation exploration



We pick those features which correlation values are over 50%. We choose 10 features.

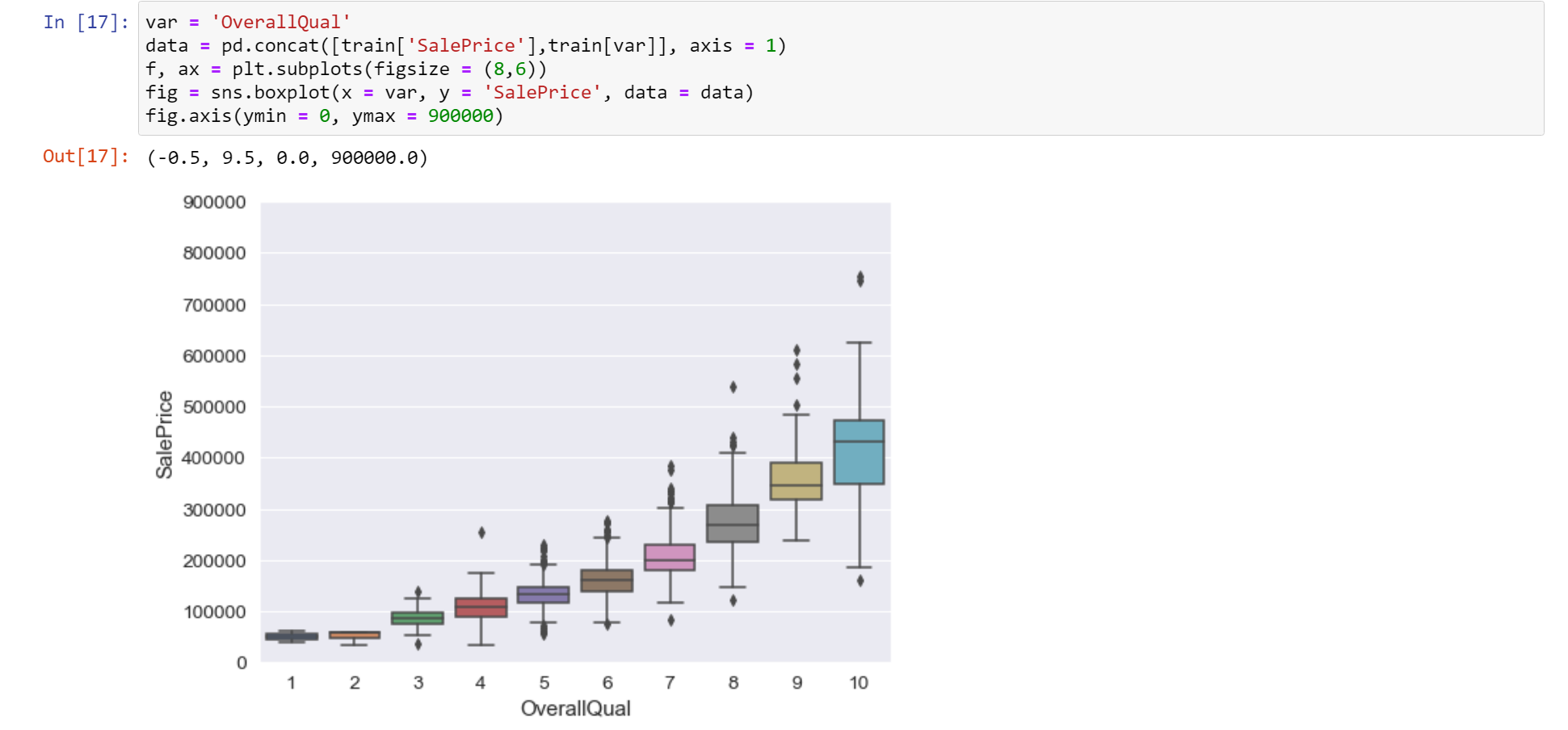


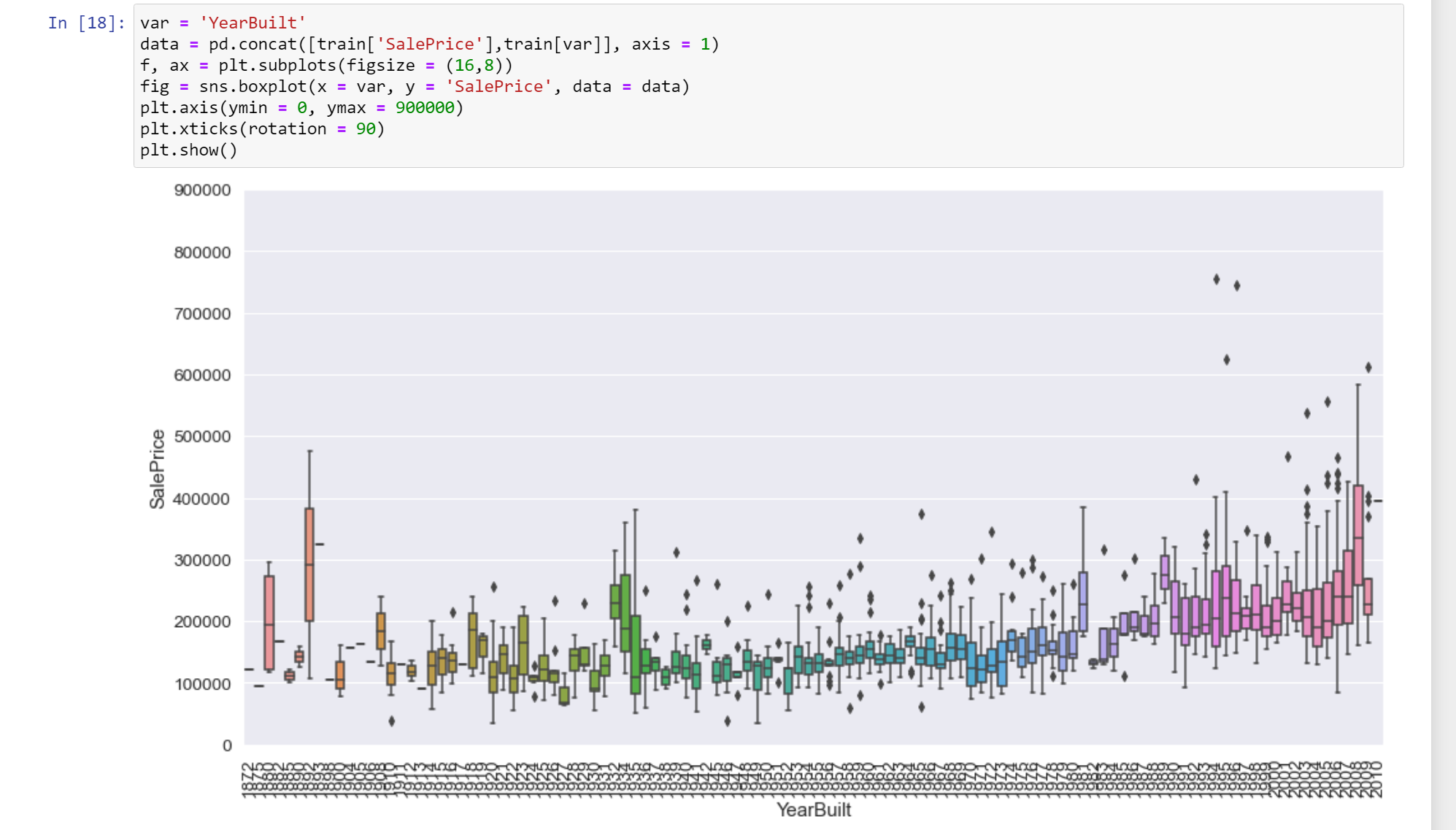
It can be found that'OverallQual', 'GrLivArea' and'TotalBsmtSF' have a strong correlation with housing prices.' GarageCars' (how many cars can be placed in the garage) and'GarageArea' (garage area) also has a strong correlation with house prices, but the correlation between these two variables is also very strong. So we only need to keep on the two features to avoid the problem of multicollinearity. The same situation applied to ‘TotalBsmtSF’ and ‘1stFloor’.

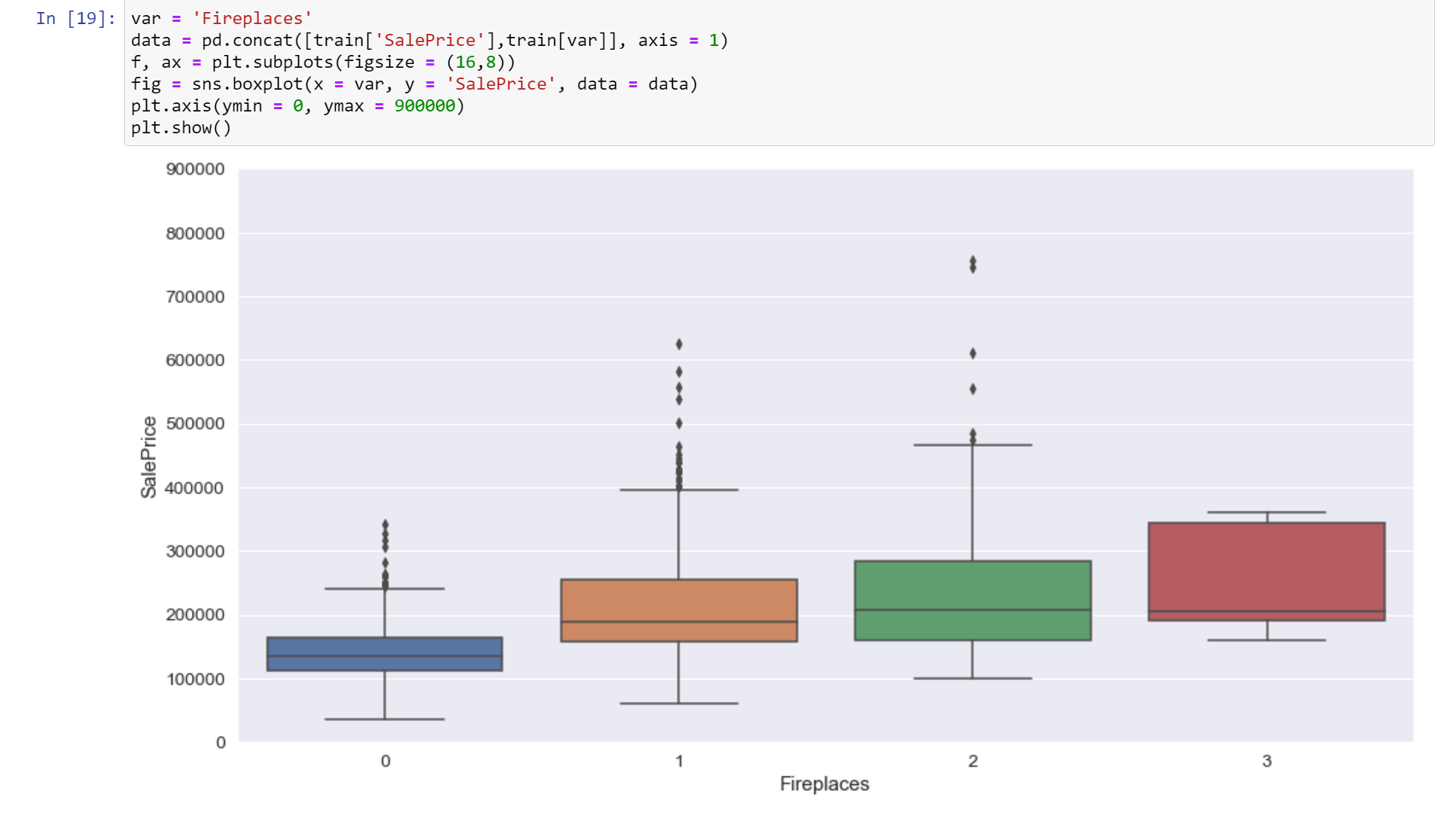
## Visual Exploration

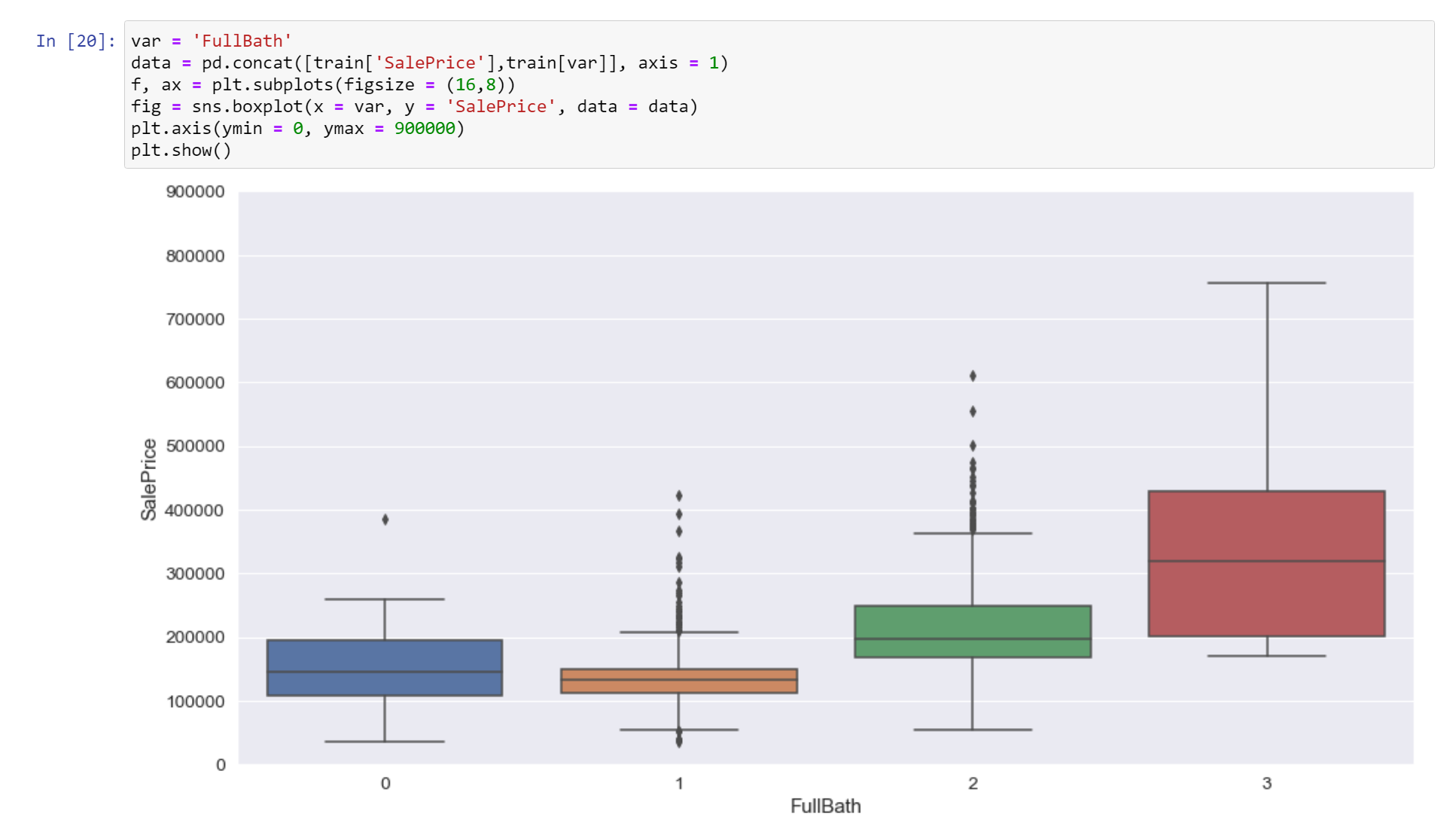


for discrete numerical variables, we use boxplot to describe the relationships between SalesPrice and those variables.

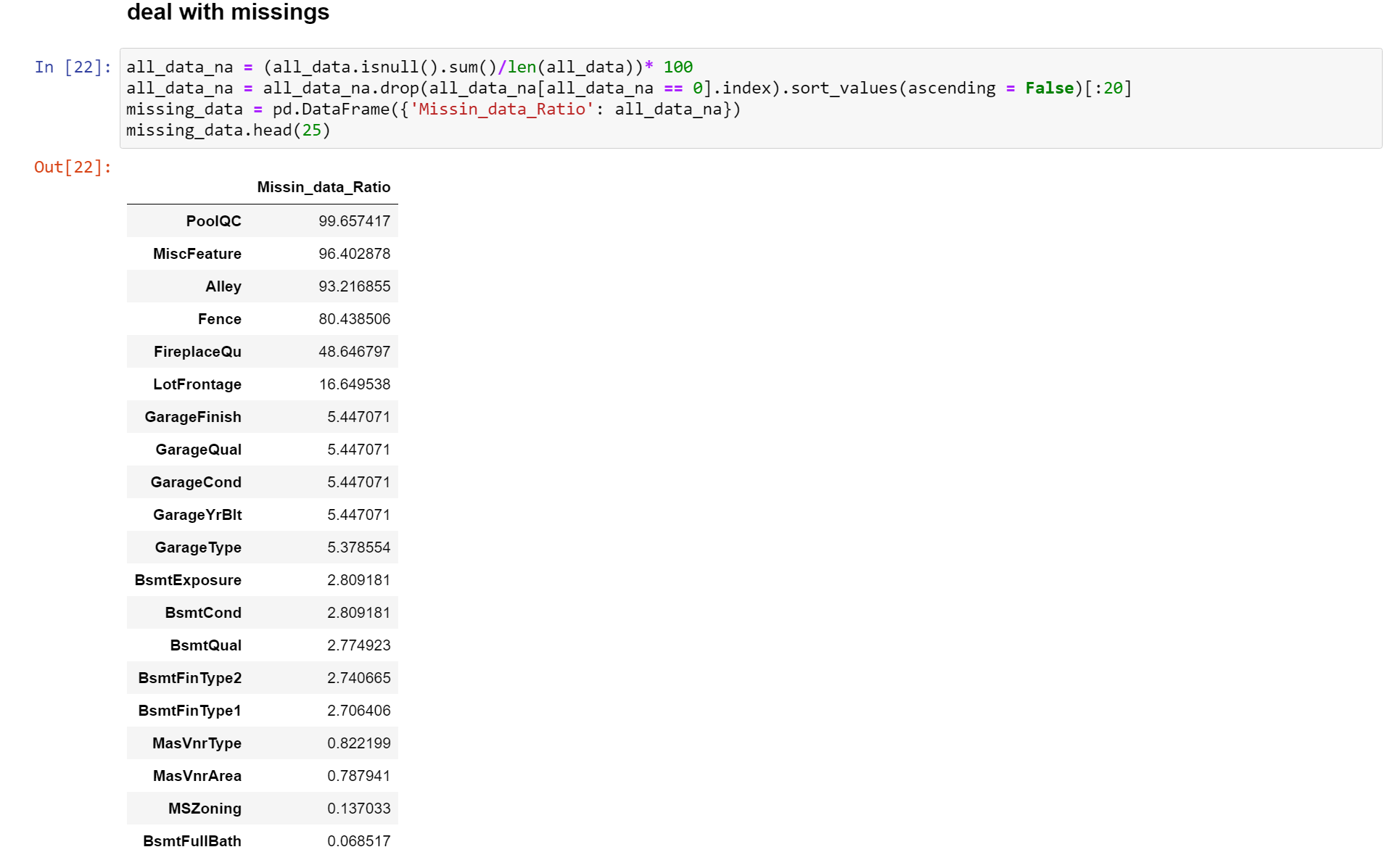








deal with the missing features

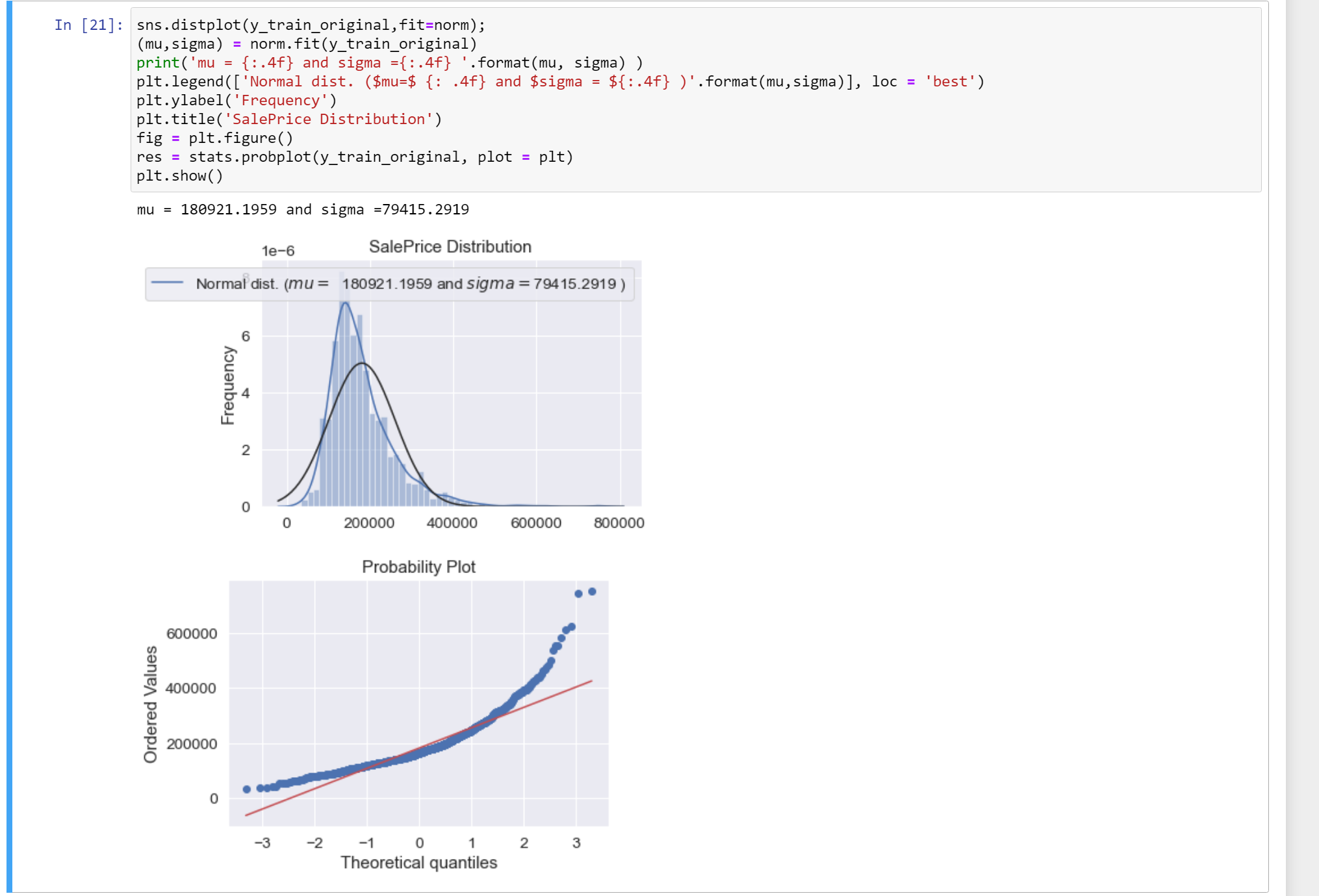


delete those features whose missing values were over 50%

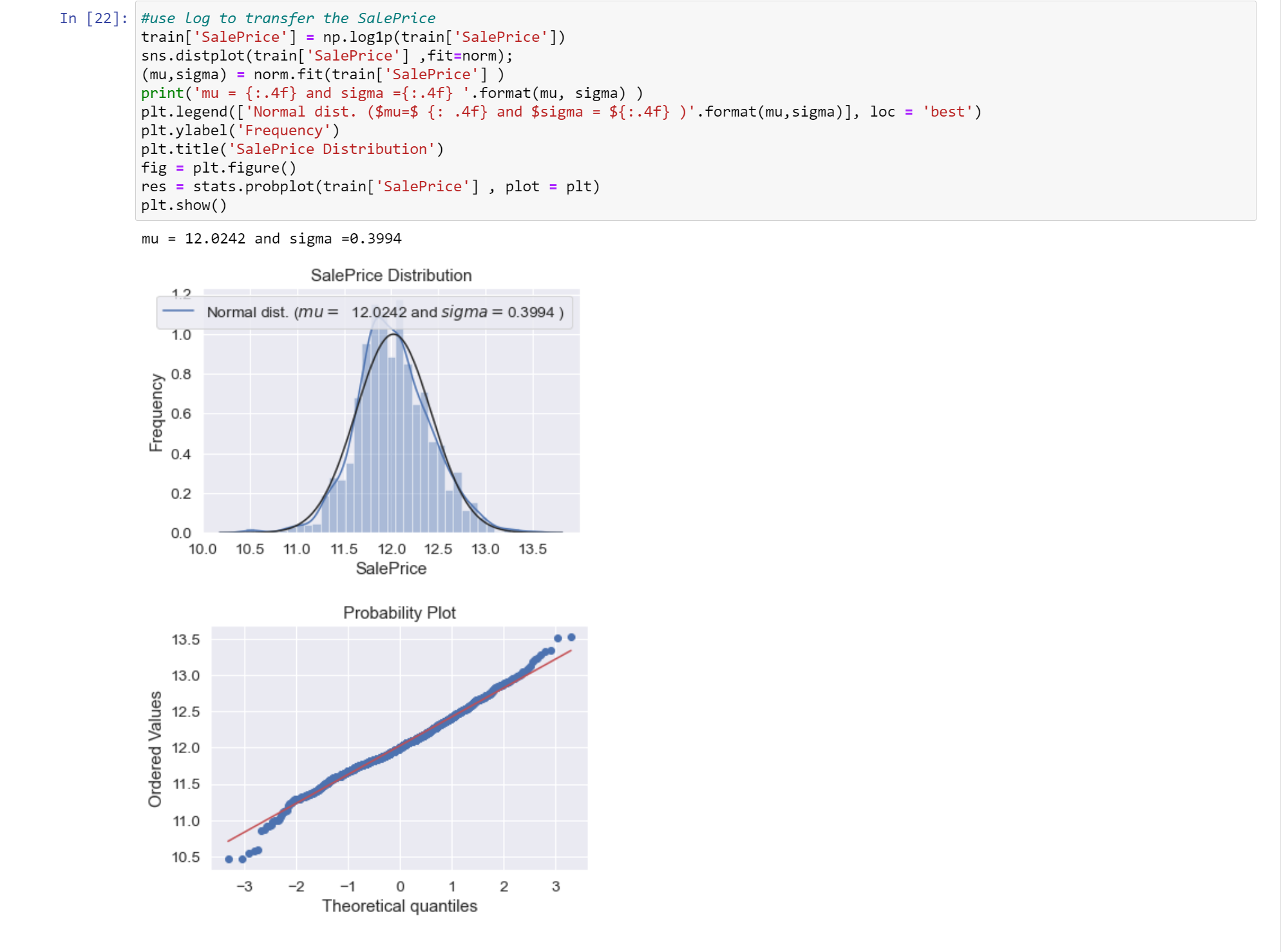


Then we find that there is no missing value right now.

find the distribution of SalePrice:



From the histogram distribution and qq-plot of the sale price, we can find that the distribution of house price is skewed to the right. We need some transformations to make it to normal distribution. The logarithmic is used here.



# Step 4: Feature Engineering

Feature Engineering is mainly divided into three parts: feature preprocessing, feature extraction, and feature screening.

Feature Conversion: The most common feature processing is the conversion of discrete features. There are generally two general methods that we can use: Label Encoding and One\_Hot Encoding.

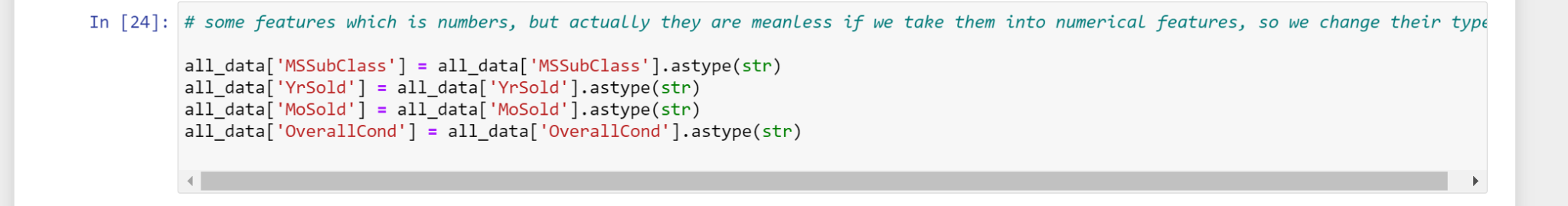
Feature Extraction: This is the most creative and important part of feature engineering.

There are many methods for this. We can use addition, subtraction, multiplication, and division transformations on features. Furthermore, we can use square root or logarithmic function to do those tasks.

Feature Selection: Lots of features do not mean accuracy to predict. Reversely, multicollinearity will impact your prediction badly. Therefore, we need to filter those features to increase the accuracy.

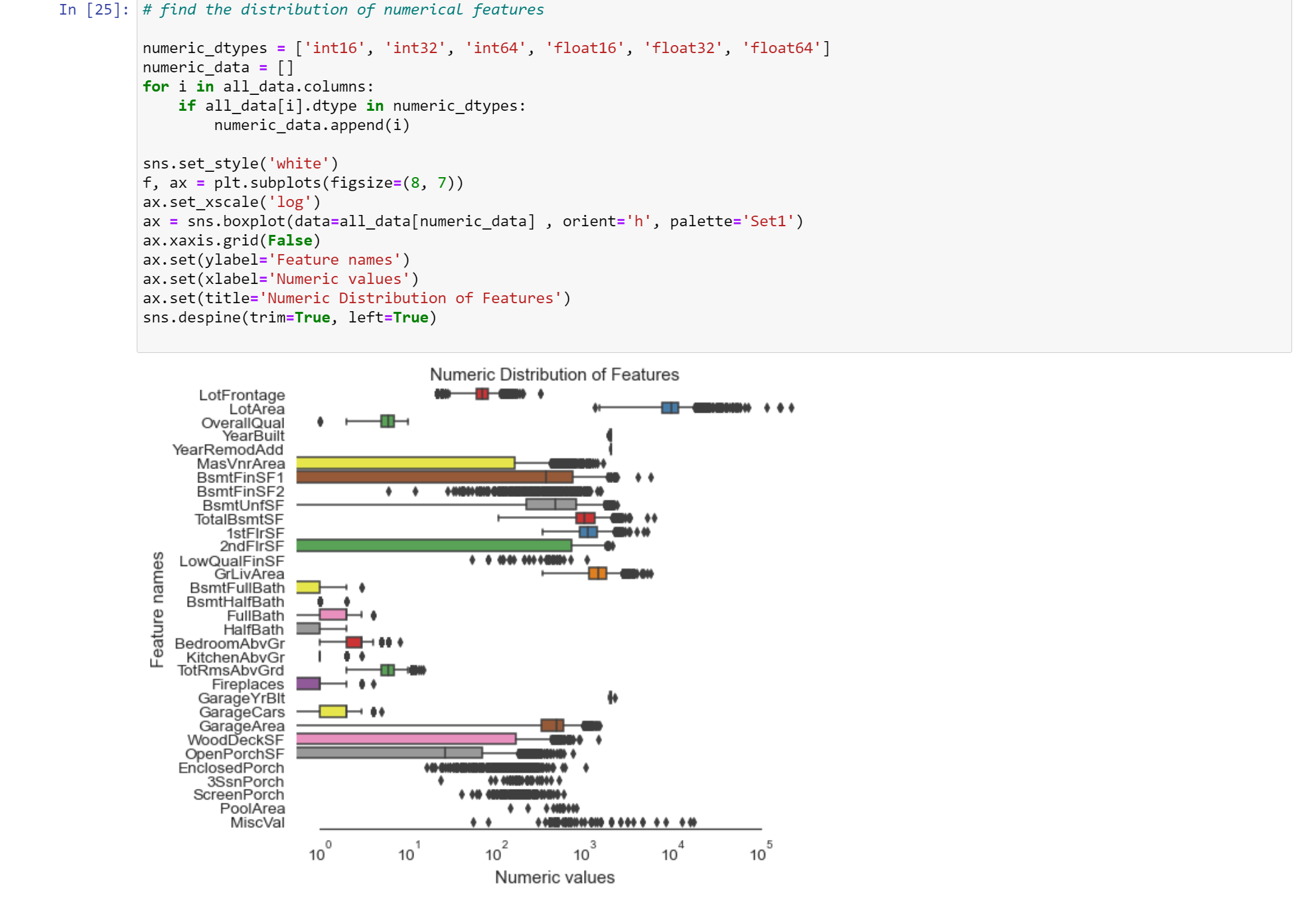
We will do the above procedure steply.

Lastly, some features are numerical but lack meanings, such as years and categories, etc. Here we will convert them into strings.



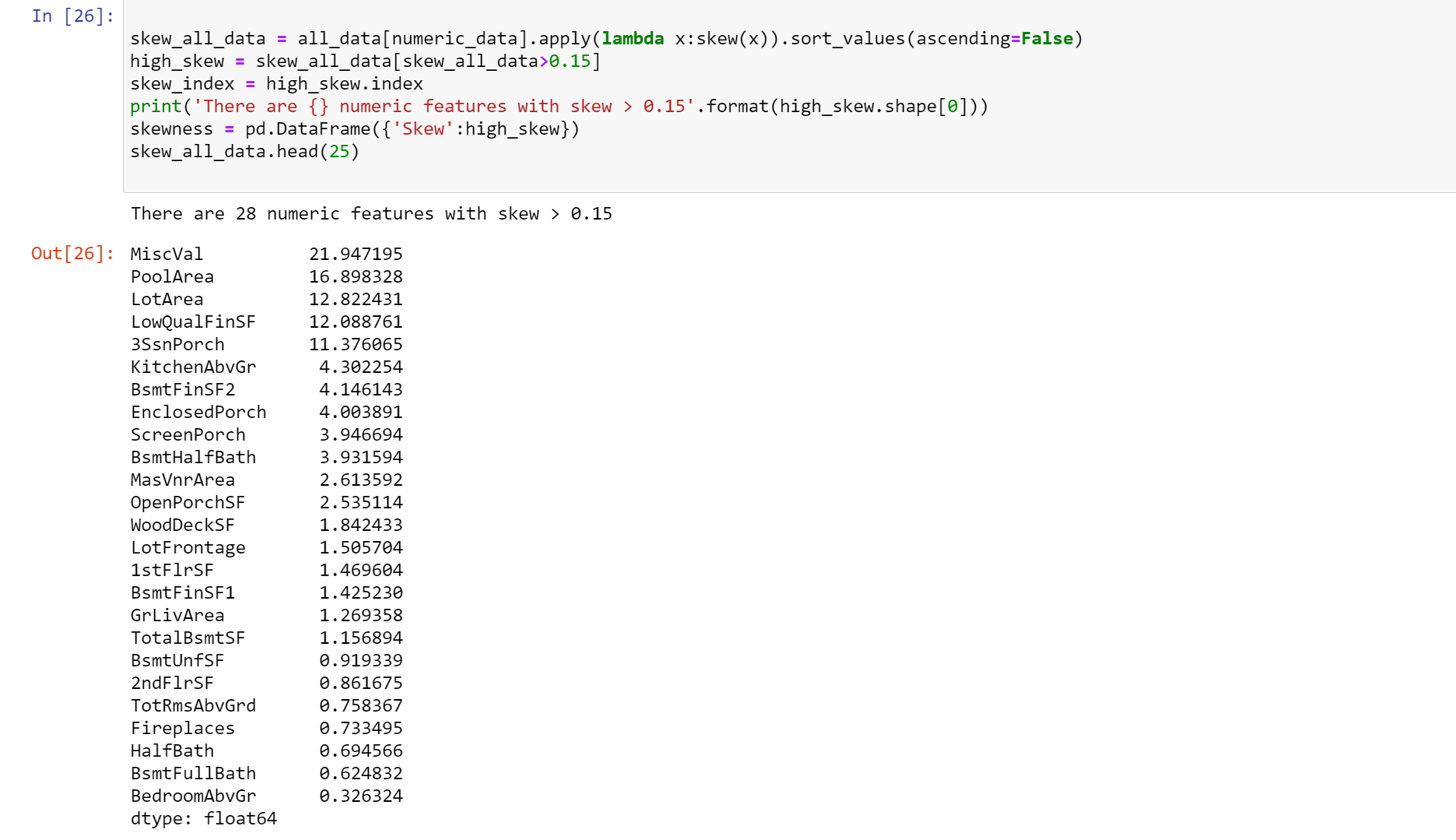
## Feature Conversion:

Some data need to be normalized, so thereby increasing the model efficiency. There are many ways of data conversion. We tried the logarithmic conversion for SalePrice in the data cleaning step. Here we will use the box-cox transformation.

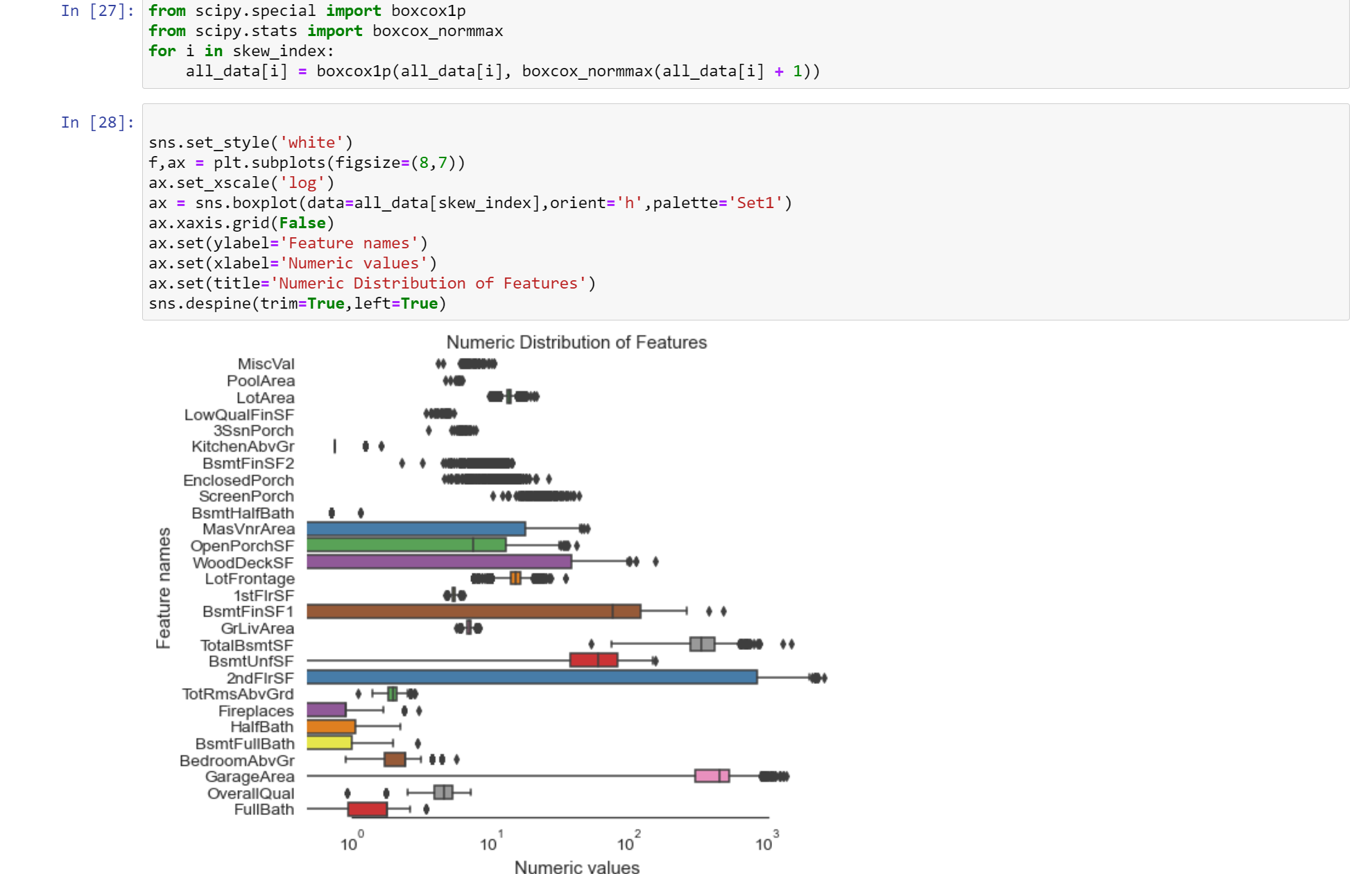


Unfortunately, data are skew seriously.

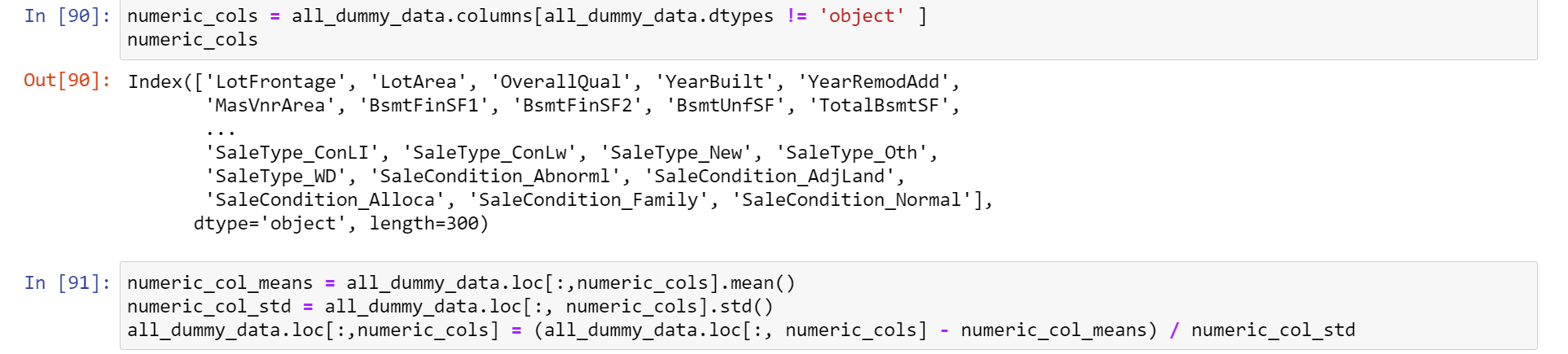
We calculate the skewness first.



Use the scipy function boxcox1p to do the Box-Cox conversion.

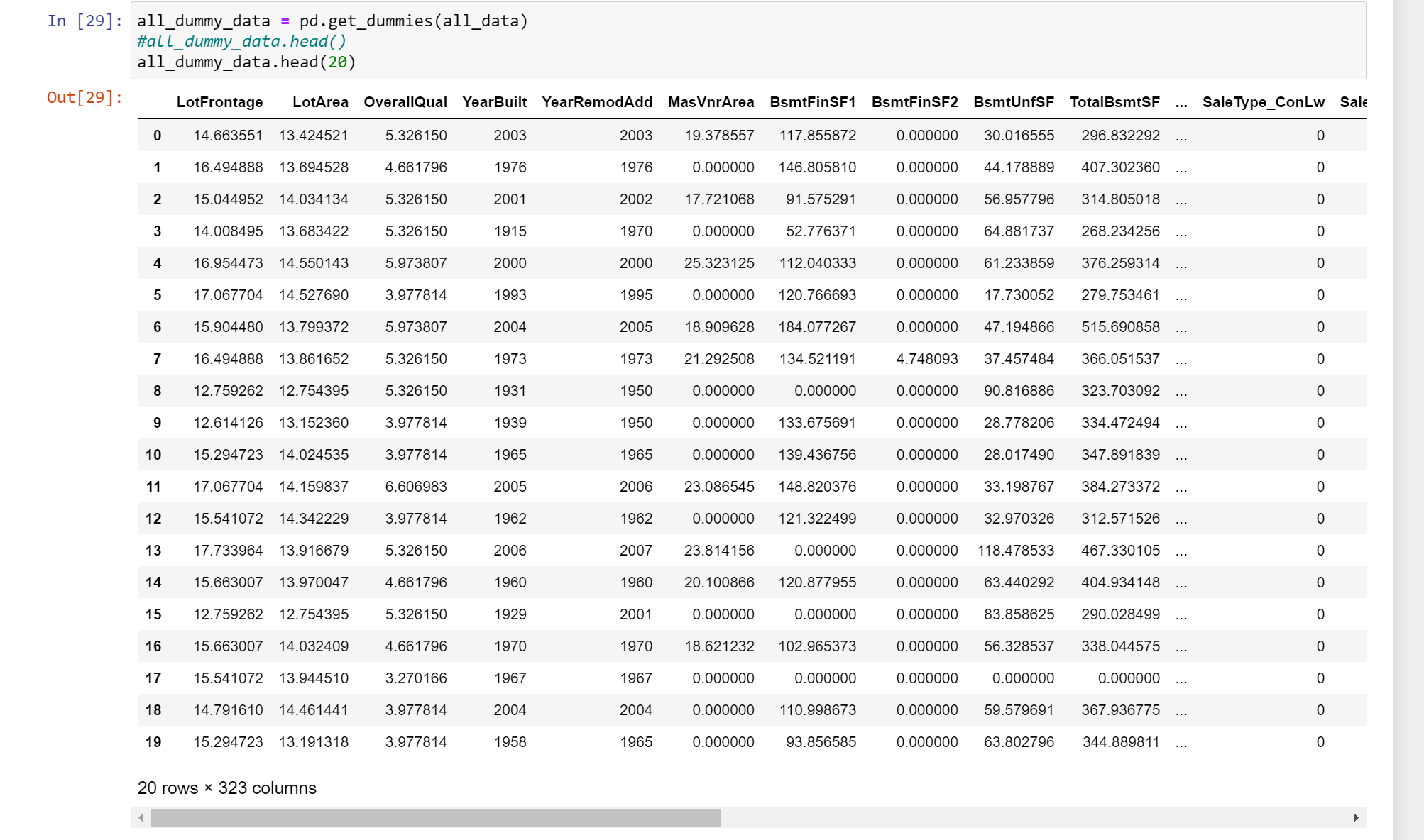


the final step is to normalize those features:

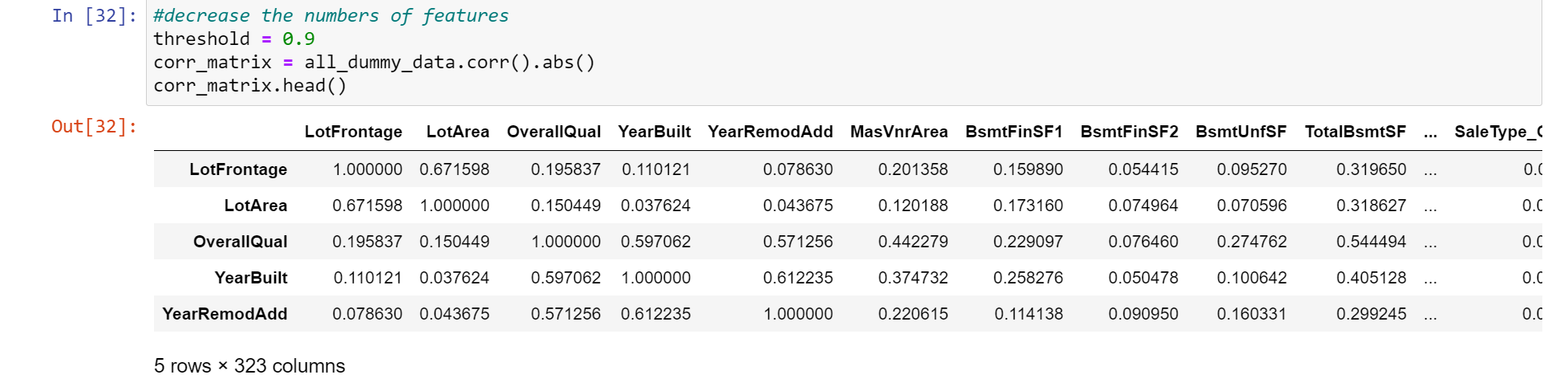


## Feature Extraction:

Here we use One-Hot Enconding for the categorical features.

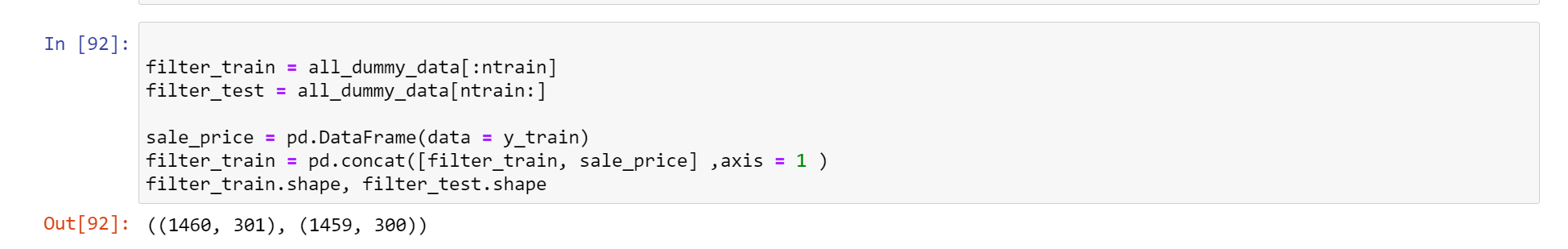


check if all features become numerical.

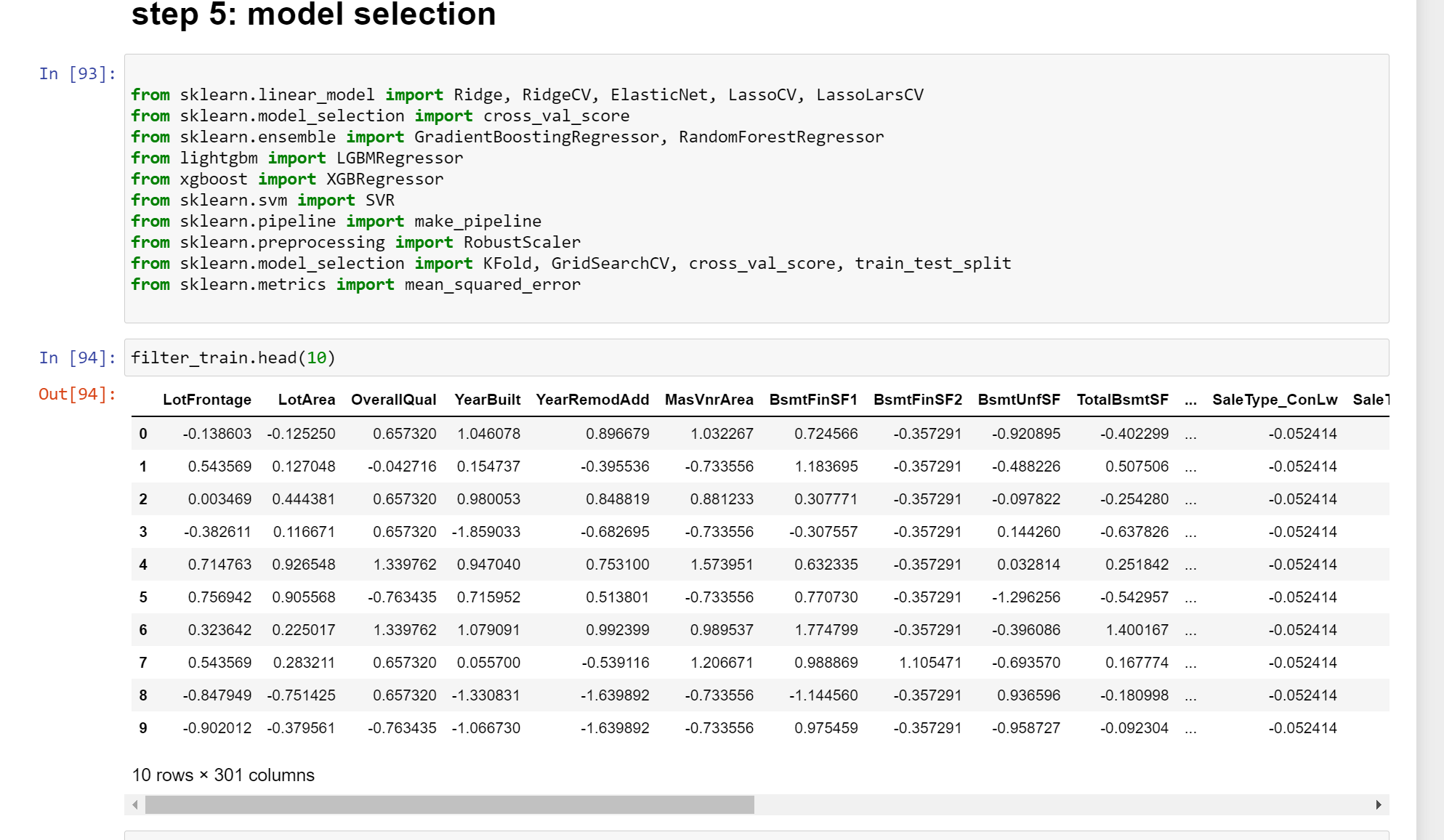


## Feature Selection:

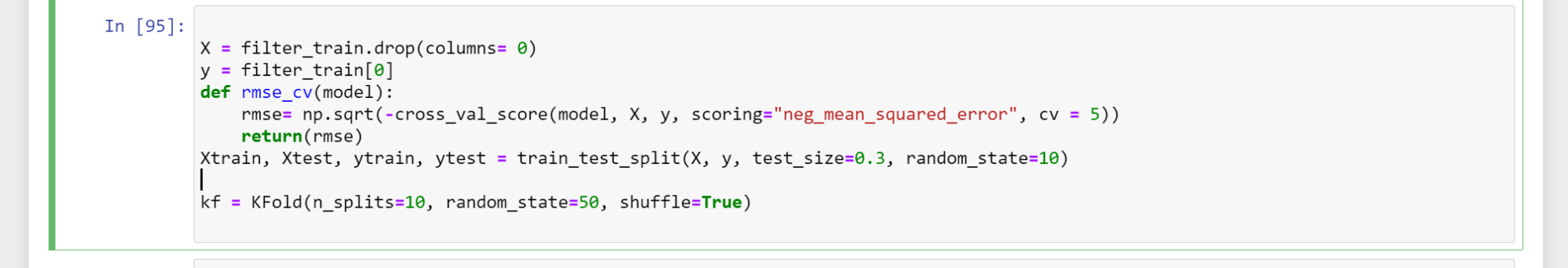
Conduct a correlation matrix, and keep only the features which the coefficient of correlations are over 0.9. This step is for avoiding collinearity.

Split the data back to training data and test data

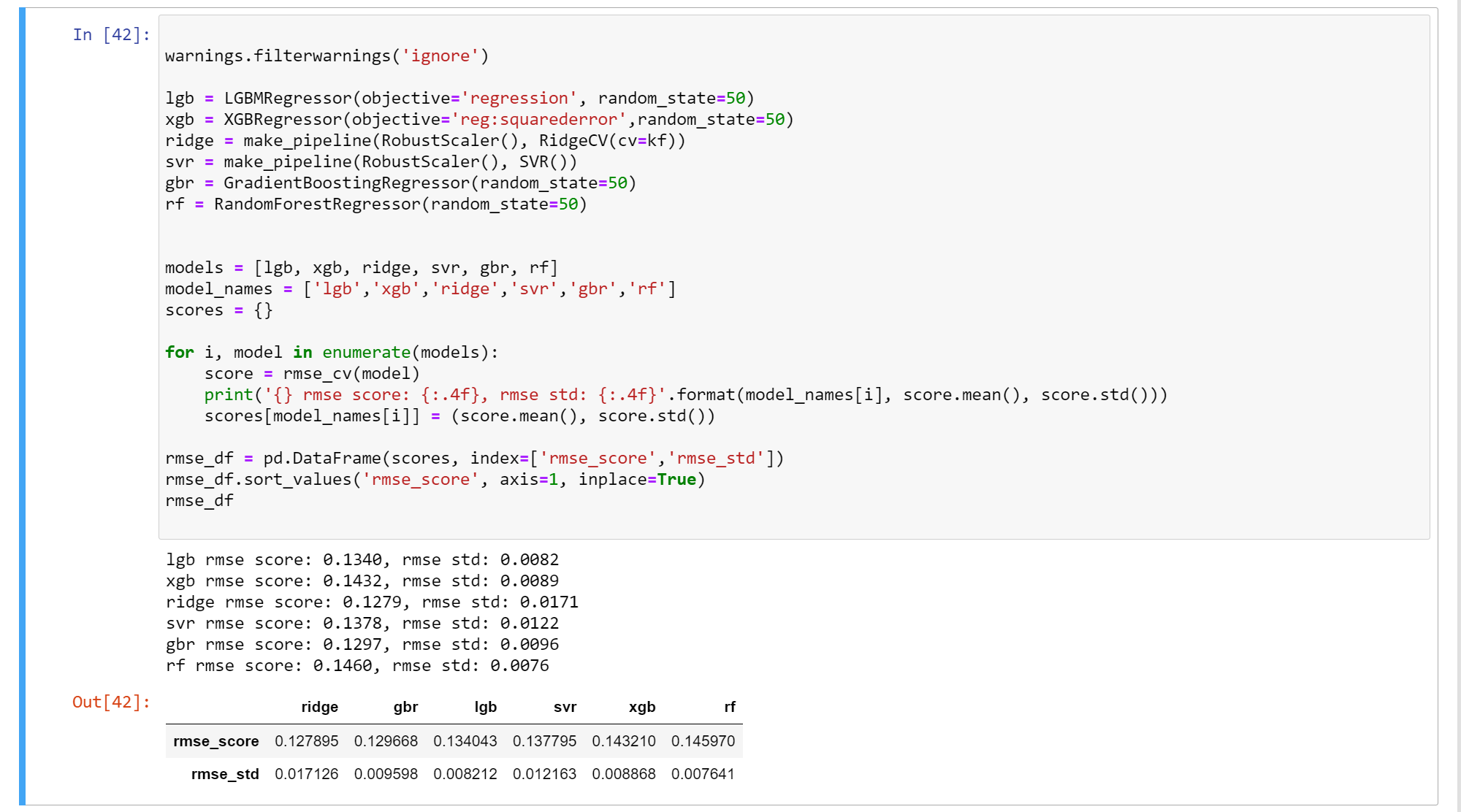
# Step 5: Model Selection



Before modeling, divide the dat set, define the cross-validation mode and measurement indicators.

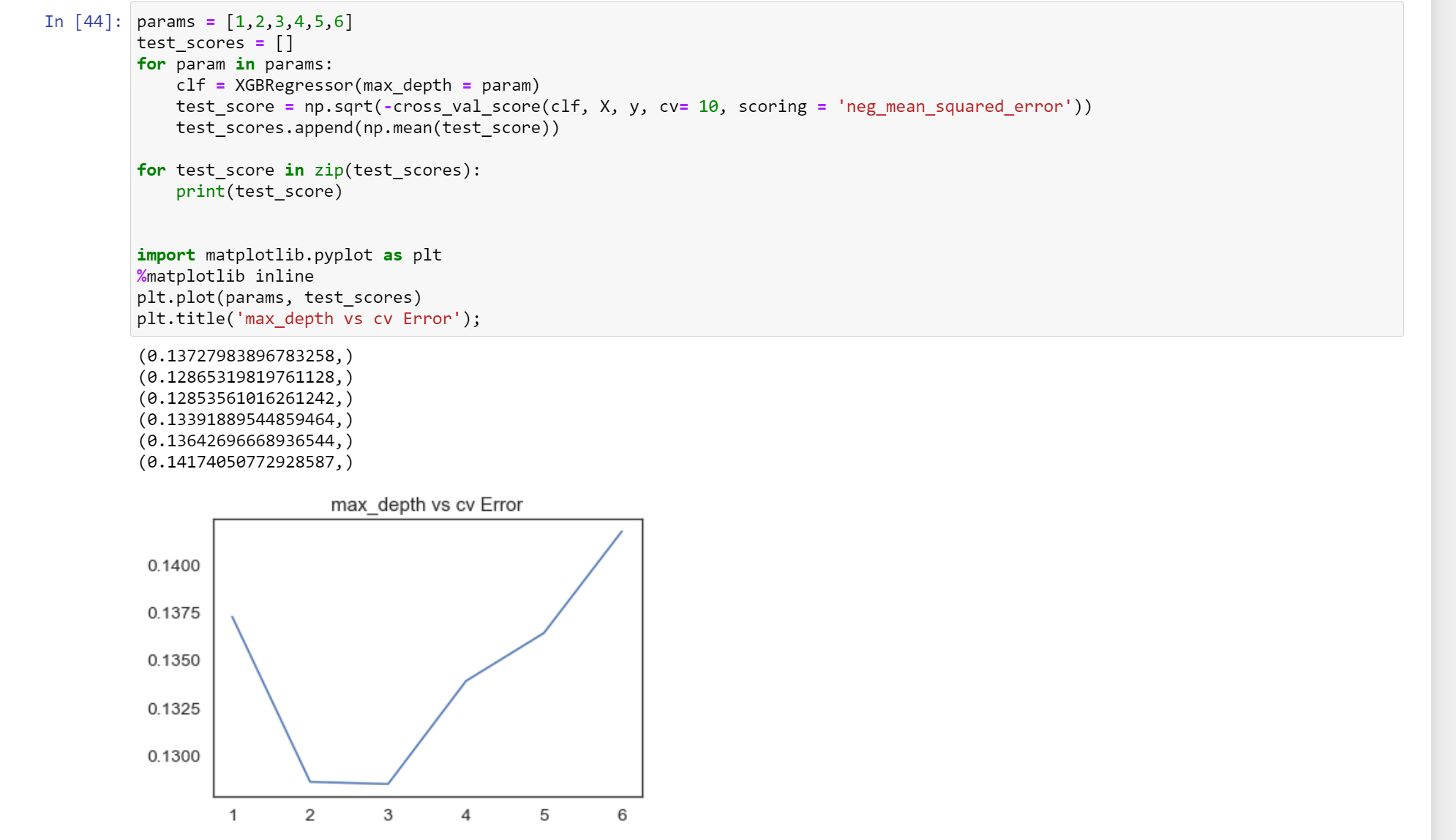


Now we tried to set some bench models.



From above, we can concluded that the XGBoost will be the best mode as it has reletively low RMSE\_score and RMSE \_ std.

Furthermore, we want tried different maximun depth of a tree to explore the lowest RMSE.



Therefore, we can see that when max\_depth = 2, 3, we can have the lowest RMSE for prediction.

# Conclusion and Summary :

We pick XGBoost as the best model. Obviously, we have lots of ways can imporve our results. Here we just pick three of them:

1.Feature Conversion: we can add more procesure in this step. For example, we can add some more addition features or log form features to decrease the number of features.

2.Model Parameter Tuning: Every model can became better when we tune the parameters.Parameter optimization is a very time-consuming process. In short, there are some space for the inprovement of every model.

3.Model Stacking: We create multiple layers to tune the XGBoost model and give differernt layers a special weights to optimaize the model.