University of Missouri - Kansas City

Big Data Programming

# House Prices

****

**Team members:**

Han Zhou

Gulnoza Khakimova

Yujing Wu

**Project proposal**

**Goals and Objectives**

**Motivation**

Big Data analysis is becoming more popular year by year. Trained models are helping to make predictions at areas like medicine, auto industry, education, media etc. For our project we decided to implement Prediction of House Prices. Create a model which will be used in Real Estate Investment.

**Significance**

Created model can be used to predict house prices, it might be a good tool for Real Estate Investors who might look into house price predictions before buying a house and making an investment, which might help them to save or earn money. Single house buyers also can used model before purchasing a house in order to know if they are buying not overpriced house for their area. As we know real estate market is blooming, which means that property prices are very high.

**Objectives**

With 79 variables which describe the features about house, we try to predict the final price of each home. It means we create a prediction machine, give a data set which contains the price of the home as the input to train the machine. And next, when there is a new buyer, we can input the features and predict the price for the house from the machine we have trained before.

**System Features**

System features of this project will include prediction of house’s price. The prediction will base on the given dataset which is the training set. After training the machine, it can predict the final price of each home.

## **Increment 1**

### **Dataset**

File descriptions

We have two data set files. One is train.cvs which store data for training our model. And also we have test.cvs which collect data for validating the accuracy of our data.

* train.csv - the training set
* test.csv - the test set

In training data set file , we have 80 columns and 1460 rows. Each column is a unique feature. And in testing data file, we have 81 columns and 1459 rows. Each column is a unique feature. But the problem here is that, in our testing data file, we do not have the prediction feature called “SalesPrice”. It lead to a situation that we can not evaluate our results. So we give up our original testing data set. And try to divide the original training data set into two parts: training set , validation set.

The amount of new training data is 70 percent of the original one. That is to say we have 1022 rows and 81 columns in training set.

The amount of new testing data is 30 percent of the original one. In other words we have 438 rows and 81 columns in validation set.

### **Detail design of Features**

Here's a brief version of what you'll find in the data description file.

1. Id.
2. SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
3. MSSubClass: The building class
4. MSZoning: The general zoning classification
5. LotFrontage: Linear feet of street connected to property
6. LotArea: Lot size in square feet
7. Street: Type of road access
8. Alley: Type of alley access
9. LotShape: General shape of property
10. LandContour: Flatness of the property
11. Utilities: Type of utilities available
12. LotConfig: Lot configuration
13. LandSlope: Slope of property
14. Neighborhood: Physical locations within Ames city limits
15. Condition1: Proximity to main road or railroad
16. Condition2: Proximity to main road or railroad (if a second is present)
17. BldgType: Type of dwelling
18. HouseStyle: Style of dwelling
19. OverallQual: Overall material and finish quality
20. OverallCond: Overall condition rating
21. YearBuilt: Original construction date
22. YearRemodAdd: Remodel date
23. RoofStyle: Type of roof
24. RoofMatl: Roof material
25. Exterior1st: Exterior covering on house
26. Exterior2nd: Exterior covering on house (if more than one material)
27. MasVnrType: Masonry veneer type
28. MasVnrArea: Masonry veneer area in square feet
29. ExterQual: Exterior material quality
30. ExterCond: Present condition of the material on the exterior
31. Foundation: Type of foundation
32. BsmtQual: Height of the basement
33. BsmtCond: General condition of the basement
34. BsmtExposure: Walkout or garden level basement walls
35. BsmtFinType1: Quality of basement finished area
36. BsmtFinSF1: Type 1 finished square feet
37. BsmtFinType2: Quality of second finished area (if present)
38. BsmtFinSF2: Type 2 finished square feet
39. BsmtUnfSF: Unfinished square feet of basement area
40. TotalBsmtSF: Total square feet of basement area
41. HeatingQC: Heating quality and condition
42. CentralAir: Central air conditioning
43. Electrical: Electrical system
44. 1stFlrSF: First Floor square feet
45. 2ndFlrSF: Second floor square feet
46. LowQualFinSF: Low quality finished square feet (all floors)
47. GrLivArea: Above grade (ground) living area square feet
48. BsmtFullBath: Basement full bathrooms
49. BsmtHalfBath: Basement half bathrooms
50. FullBath: Full bathrooms above grade
51. HalfBath: Half baths above grade
52. Bedroom: Number of bedrooms above basement level  
    53 Kitchen: Number of kitchens
53. KitchenQual: Kitchen quality
54. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
55. Functional: Home functionality rating
56. Fireplaces: Number of fireplaces
57. FireplaceQu: Fireplace quality
58. GarageType: Garage location
59. GarageYrBlt: Year garage was built
60. GarageFinish: Interior finish of the garage
61. GarageCars: Size of garage in car capacity
62. GarageArea: Size of garage in square feet
63. GarageQual: Garage quality
64. GarageCond: Garage condition
65. PavedDrive: Paved driveway
66. WoodDeckSF: Wood deck area in square feet
67. OpenPorchSF: Open porch area in square feet
68. EnclosedPorch: Enclosed porch area in square feet
69. 3SsnPorch: Three season porch area in square feet
70. ScreenPorch: Screen porch area in square feet
71. PoolArea: Pool area in square feet
72. PoolQC: Pool quality
73. Fence: Fence quality
74. MiscFeature: Miscellaneous feature not covered in other categories
75. MiscVal: $Value of miscellaneous feature
76. MoSold: Month Sold
77. YrSold: Year Sold
78. SaleType: Type of sale
79. SaleCondition: Condition of sale
80. Heating: Type of heating

### **Analysis**

We set up the whole work into four parts. At each one we do some related work to deal with the data so that we can generate better results from the data.

In the first place, we have to analyze our original data. Since there might be some features which do not have some relationships with the dependent variable. We have to plot or calculate to find out the most mattered features to apply in our further research.

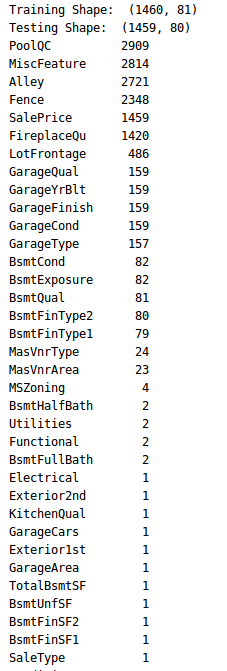
Second, we need to do some pretreatment to our original data. For there is a huge amount of data, null values or noise values are unavoidable. To solve this, we can take some measures to remove or replace the null values or noise values. After cleaning the data, model will be easier to fit and we will get a better result.

Next, we can apply the data to train a model. We can use 79 features of them to be independent variable. And 1 feature is dependent variable—SalesPrice. We use training set to train the model so that when we give the test set as input the model can return a prediction of dependent variable.

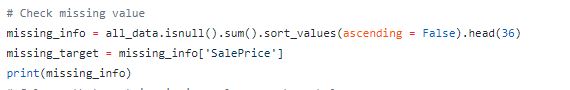
Finally, we can evaluate our results. The last step, we can compare the training results and real value. Thus we can get a result about the accuracy of the model. And the higher the accuracy could be, the better our model would be.

### **Implementation**

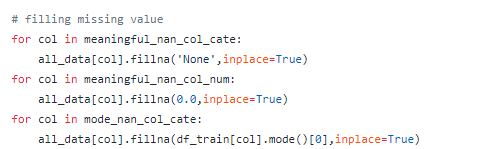
We load the data. And do some preparation for the pretreatment of data. First of all, we combine test set and train set to do EDA of features and Check missing value. Columns that contain missing values are shown below.There are 35 columns contain missing value (includes salePrice). Below is list of rows which contain missing data.



Code below shows how to check missing data:



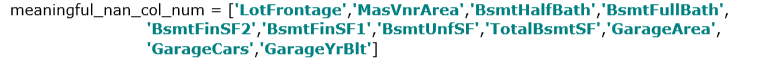
After identifying missing data we need to fill them out.



Based on data description, we found that the missing value in the following columns simply means 'None'.



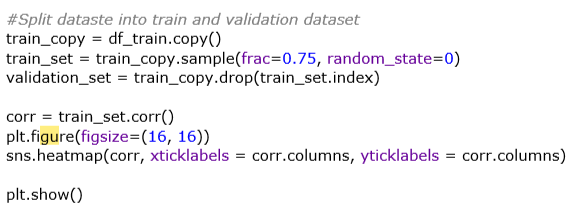
Also, some of the 'None' value need to be expressed as 0.0



Then we fill the list with most common term since there are only a few missing value and replace null value with 0.

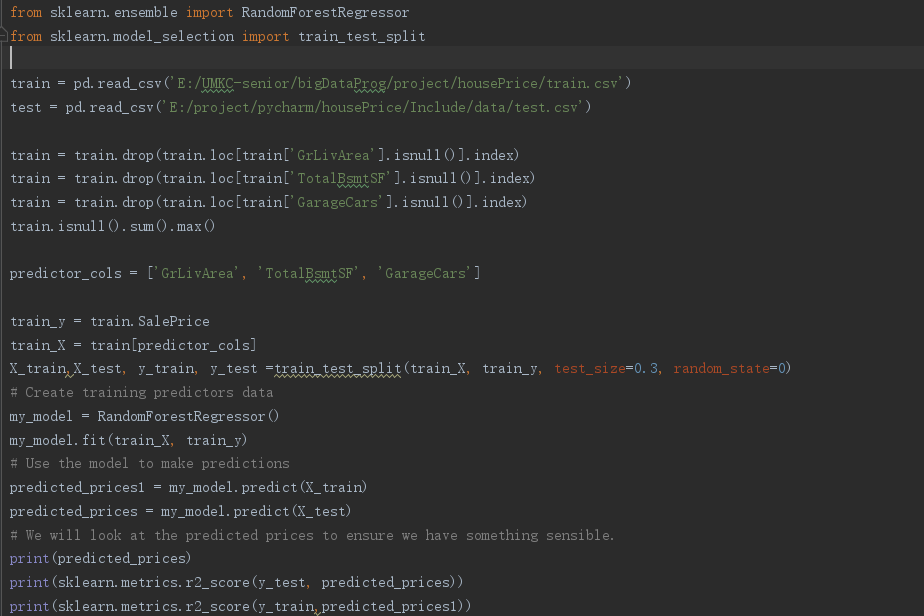


Since we have told before, original test set do not have target column. We use two ways to split the data. One is



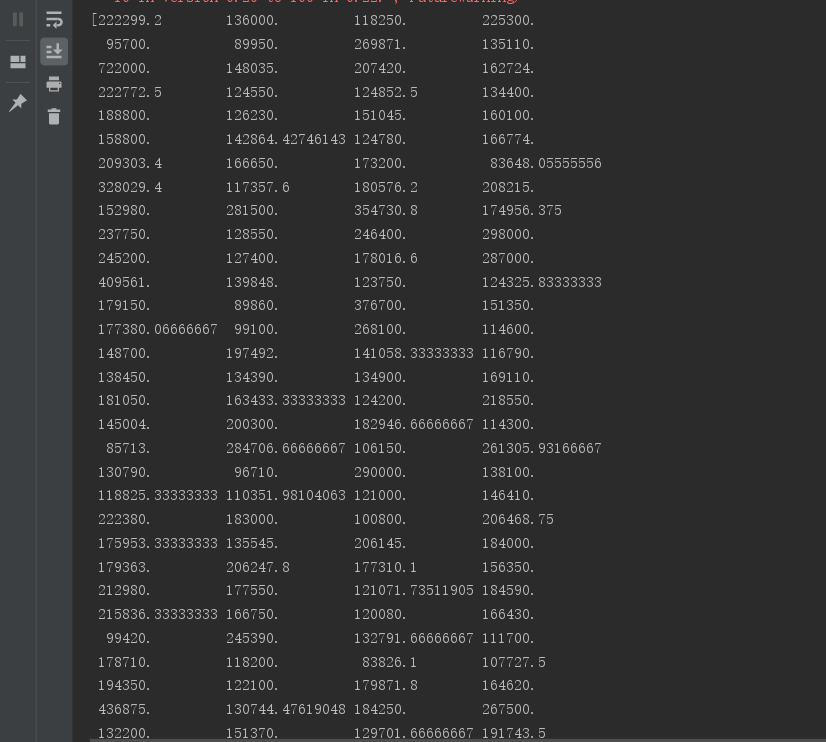
### **Code**

In this function, we use drop to clean the data. We choose three top mattered features to analyze and remove the rows with null values. Then we train\_test\_split function to split our data to get train set and validation set. Next, we fit the model with train data. Finally, we print out the score of the model using score function of metrics library.

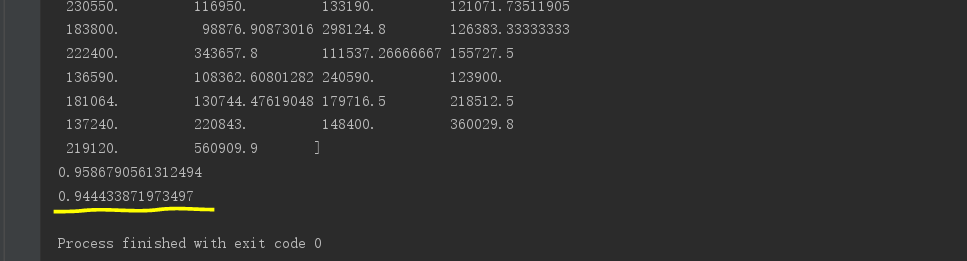


Result

The first part of the results is the prediction of the salesPrice.



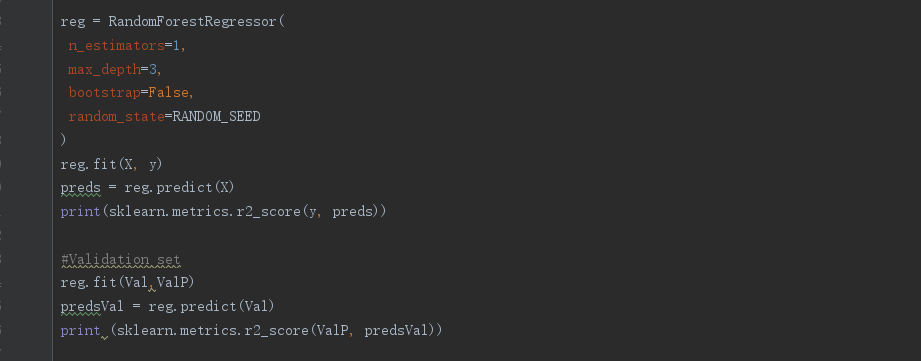
The second part is the score of the model.



* Define RandomForestRegressor() implementation

In this implementation, we define the function by ourselves. As the same as the above one, we print out the score of the model using score function of metrics library.

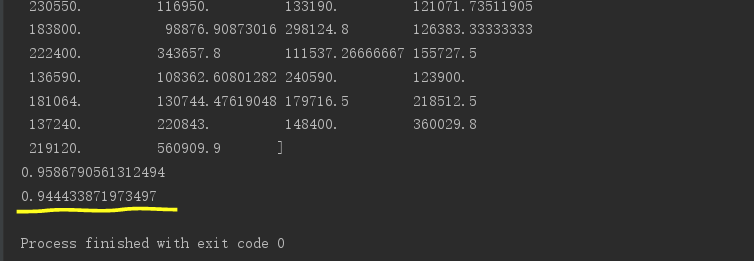
Code:



Result:

### **Preliminary Results**

The score of the model from the skl-learn library reach up to 0.94, which is high enough to prove the model is excellent and can get a great prediction.



### **Project Management**

### **Implementation status report**

In the first place, we can not apply the function. Because there are so many null values. That makes our input set invalidate. To fix this

|  |  |  |
| --- | --- | --- |
|  | **Activity** | **Result** |
| 1 | Run model with input data set | Error: the input is invalid |
| 2 | Run model with input data set | Error: the input is invalid |
| 3 | Run model with input data set | Error: No module named sklearn |
| 4 | Run model with input data set | Error: No module named pd |
| 5 | Run model with input data set | Successful |

We pretreatment the data bases on some fundamental analysis. After that we met some problems for importing packages in the IDE. The IDE provides an easy way to solve this problem. And then we can run the class successfully. Work completed: Description

In the first place, we finish analyzing our original data. We display some plots and calculation. And we find out the most mattered features to apply in our further research.

Second, we have done some pretreatment to our original data. For there is a huge amount of data, null values or noise values are unavoidable. To solve this, we take some measures to remove or replace the null values or noise values.

Next, we apply the data to train a model in two approaches. We use training set to train the model and give the test set as input. For each method, we print out the prediction of dependent variable and the sores of the model.

Finally, we evaluate our results. The last step, we can compare the training results and real value. Thus we print out a result about the accuracy of the model. And the higher the accuracy could be, the better our model would be.

### **Responsibility (Task, Person)**

**Gulnoza(Class ID: 13 ):** Analyzed original data, cleaned dataset, implemented random forest algorithm to train our model, tested the model.

**Han Zhou(Class ID: 30):** Done pretreatment of the original data. Use RandomForestRegressor() function to implement the idea. Give evaluate scores on the results. Help finishing report.

**Yujing Wu(Class ID: 28 )**

**Work to be completed**

### **Description**

We have to keep on fixing the model. Through adjust the parameters to improve our own model. Also, we can figure out some deep relationships in the data so that we can imply more functions to obtain more significant information.

### **Issues/Concerns**

Some problems related with big data will be hard to make sense and even harder to imply them. Thus it is quite important for us to define good questions and use what we have learned from class flexibility.

### **References/Bibliography**

1. h[ttps://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/](https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/)
2. <https://www.kaggle.com/rdenadai/house-prices-advanced-regression-techniques>

## 

## **Increment 2**

### **Introduction**

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 project challenges us to predict the final price of each home.

### **Background**

It is our job to predict the sales price for each house. For each Id in the test set, must predict the value of the SalePrice variable.

### **Dataset**

**File descriptions:** We have two data set files. One is train.cvs which store data for training our model. And also we have test.cvs which collect data for validating the accuracy of our data.

* train.csv - the training set
* test.csv - the test set

In training data set file , we have 80 columns and 1460 rows. Each column is a unique feature. And in testing data file, we have 81 columns and 1459 rows. Each column is a unique feature. But the problem here is that, in our testing data file, we do not have the prediction feature called “SalesPrice”. It lead to a situation that we can not evaluate our results. So we give up our original testing data set. And try to divide the original training data set into two parts: training set , validation set.

The amount of new training data is 70 percent of the original one. That is to say we have 1022 rows and 81 columns in training set.

The amount of new testing data is 30 percent of the original one. In other words we have 438 rows and 81 columns in validation set.

**Data fields:**

SalePrice - the property's sale price in dollars. This is the target variable that we'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

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

### **Analysis of data**

We set up the whole work into four parts. At each one we do some related work to deal with the data so that we can generate better results from the data.

In the first place, we have to analyze our original data. Since there might be some features which do not have some relationships with the dependent variable. We have to plot or calculate to find out the most mattered features to apply in our further research.

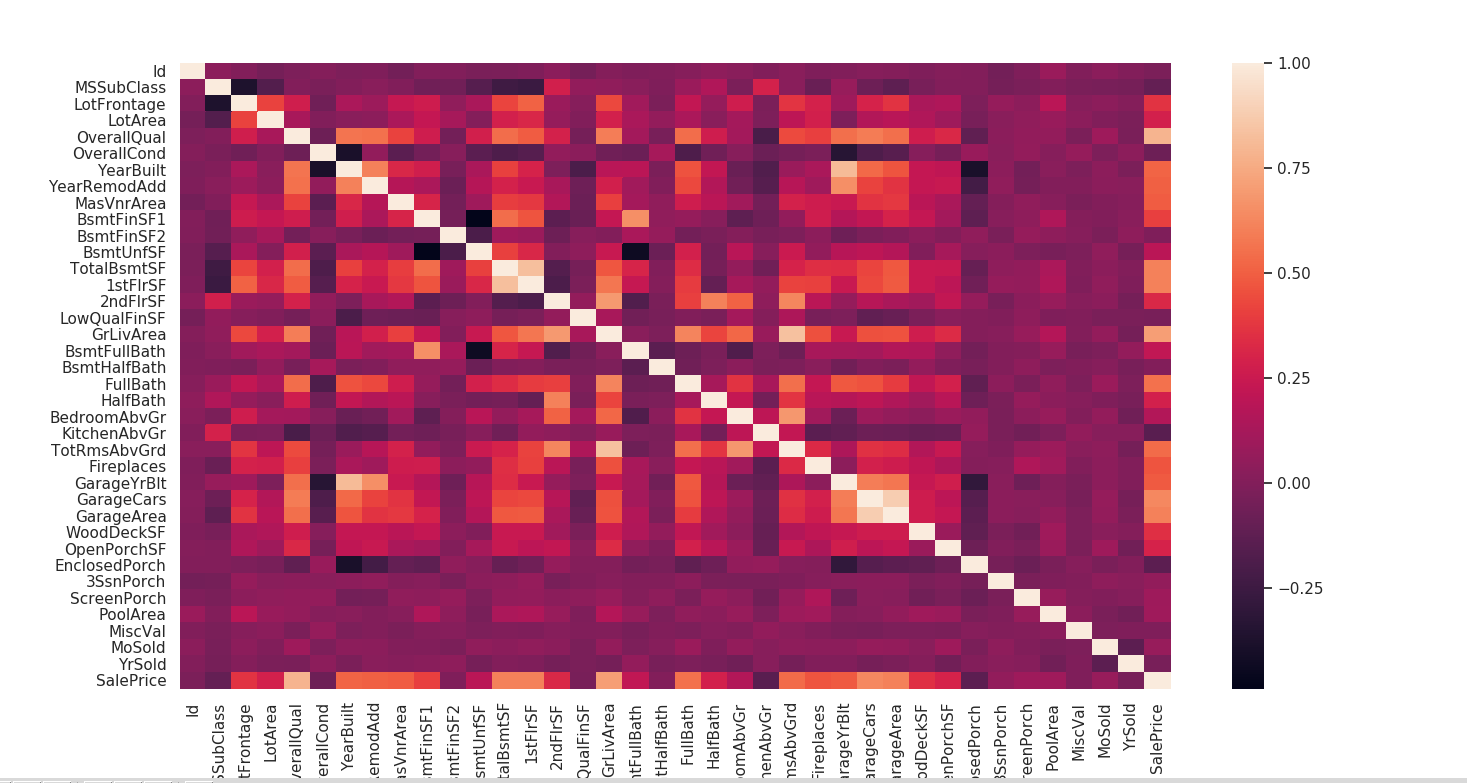
Second, we need to do some pretreatment to our original data. For there is a huge amount of data, null values or noise values are unavoidable. To solve this, we can take some measures to remove or replace the null values or noise values. After cleaning the data, model will be easier to fit and we will get a better result.

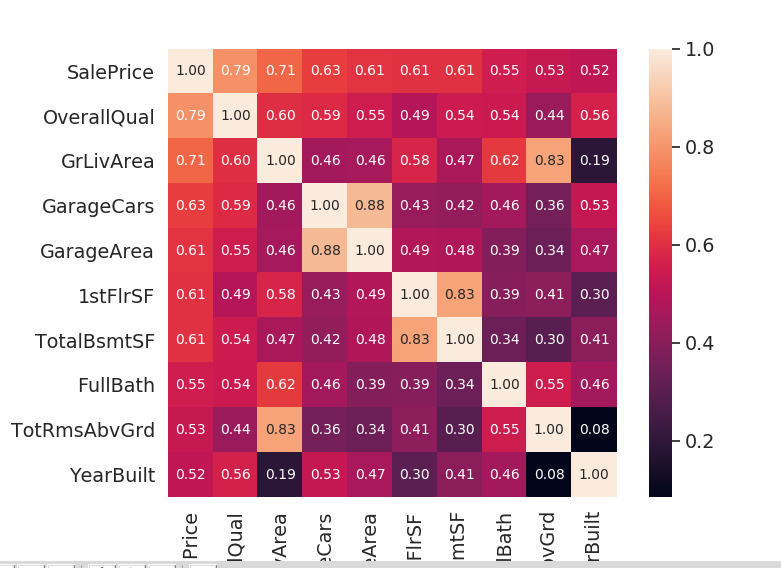
Next, we can apply the data to train a model. We can use 79 features of them to be independent variable. And 1 feature is dependent variable—SalesPrice. We use training set to train the model so that when we give the test set as input the model can return a prediction of dependent variable.

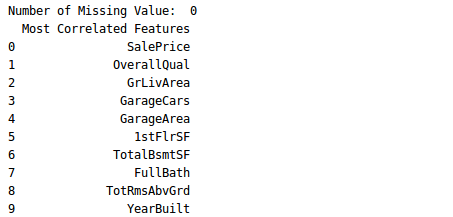
Finally, we can evaluate our results. The last step, we can compare the training results and real value. Thus we can get a result about the accuracy of the model. And the higher the accuracy could be, the better our model would be.

### **Implementation and Result**

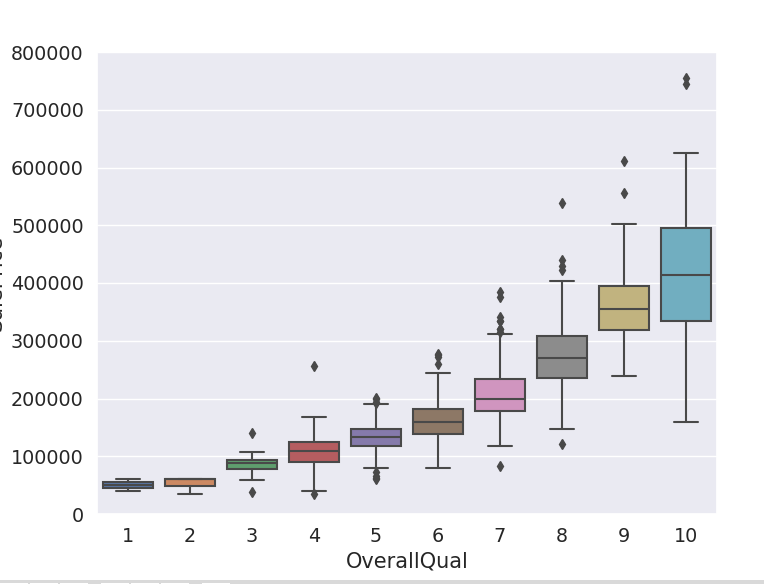
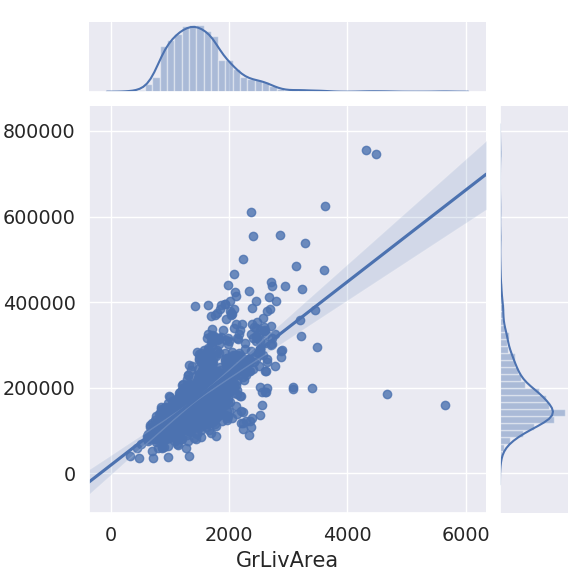
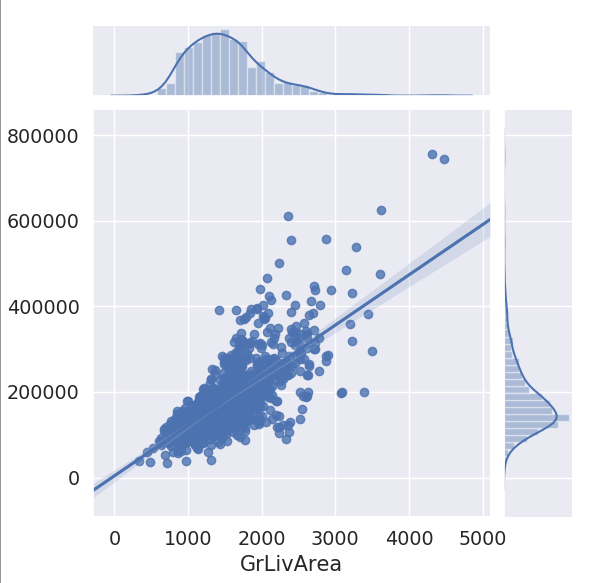
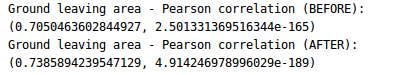
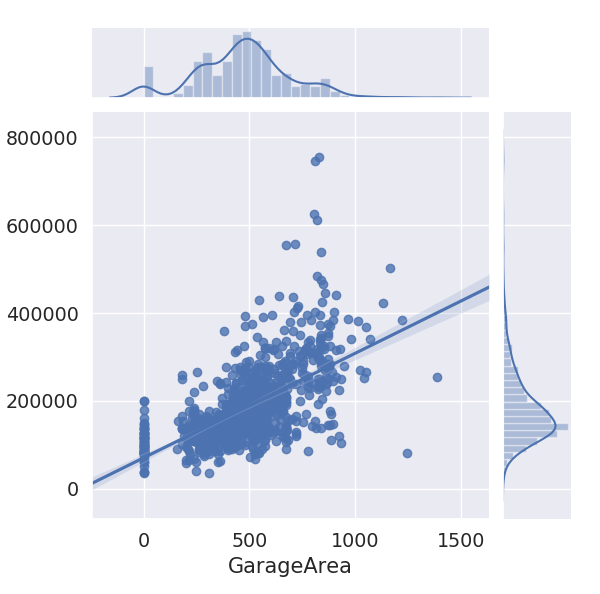
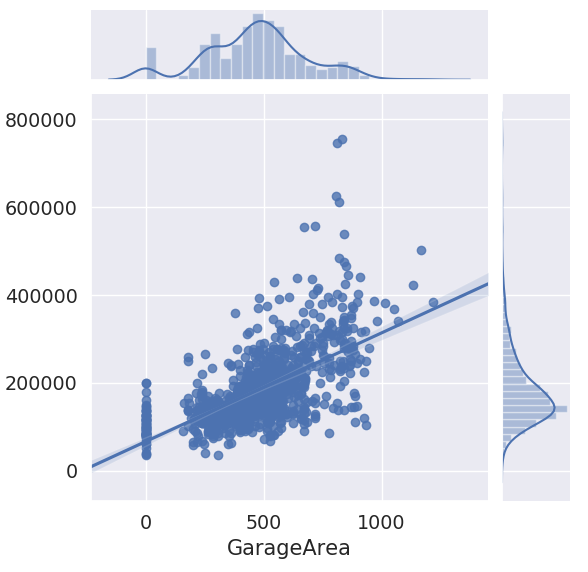
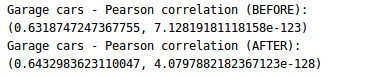
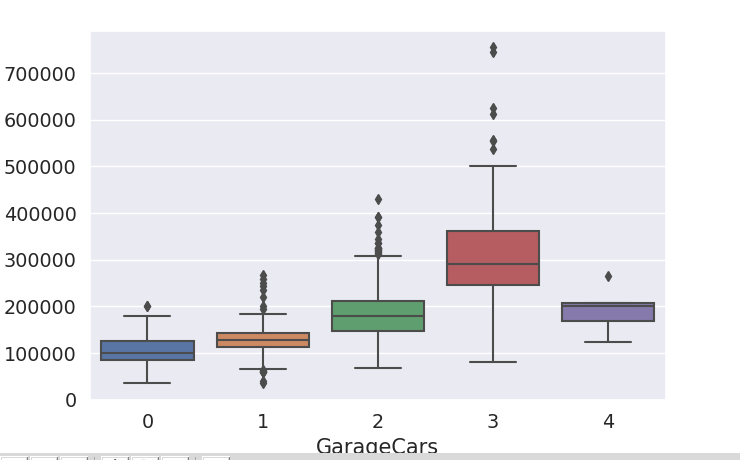
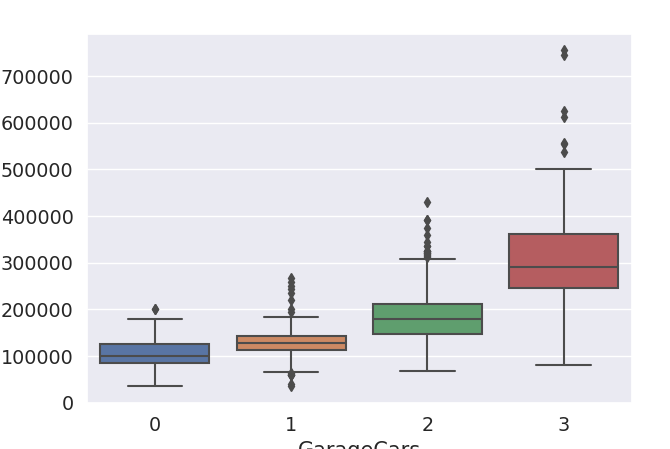
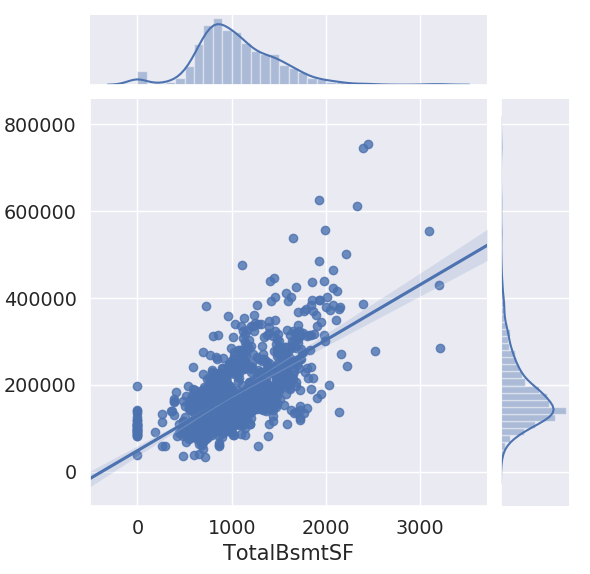
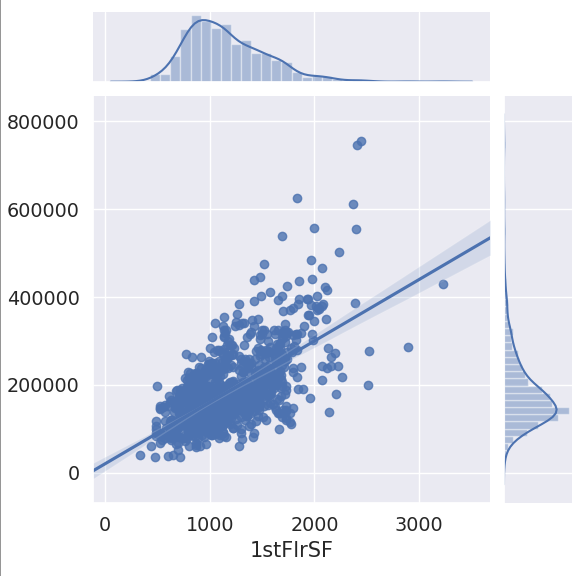
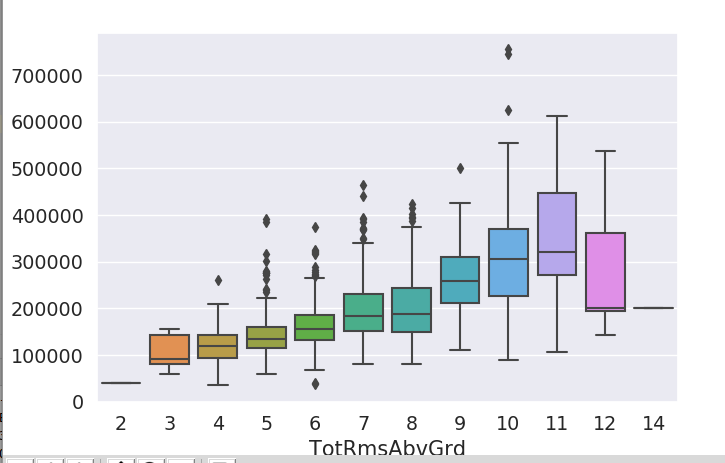
Lets display all data features that we have in matrix form

 Below is a list of top 10 heatmap, features which actually will be needed in predicting house prices.





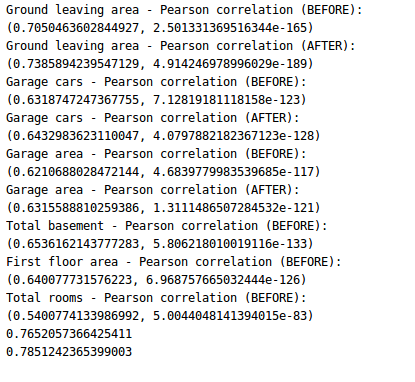
Now we will visualize each feature in order to clean and organize data before training.

* Overall Quality vs Sale Price  
    
  Everything looks good, better quality of the house result in higher price.
* Living Area vs Sale Price   
  Looks good except those two dots which does not make a sense, we can remove them and then we can check if it increased Pearson's correlation.  
    
    
  Yes, Pearson's correlation increased after cleaning the data.
* Garage Area vs Sale Price  
  
* Removing outliers manually (More than 1000 sqft, less than $300k) and check Pearson's correlation  
    
  
* Garage Cars vs Sale Price  
  
* Removing outliers manually (More than 4 cars, less than $300k)  
  
* Basement Area vs Sale Price  
    
  Looks good. no need to clean it.
* First Floor Area vs Sale Price  
  
* Total Rooms vs Sale Price  
  

After analysis and pretreatment on original data, we finally get our valid input set for fitting the model. We use two approaches to do apply our ideas. The first one is to call function RandomForestRegressor() from skit-learn library. The second one is to define the function by our selves. For each function, we fit the model with training set and evaluate it with validation set.

* Call RandomForestRegressor() implementation

Below is calculation of Pearson correlation after cleaning the data:



### **Responsibility (Task, Person)**

**Gulnoza(Class ID: 13 )**

**Han Zhou(Class ID: 30)**

**Yujing Wu(Class ID: 28 ):** Done with the increment 2.

### **References/Bibliography**

1. <https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/>
2. <https://www.kaggle.com/rdenadai/house-prices-advanced-regression-techniques>