BOSTON UNIVERSITY

**CS 521 Information Structures with Python**

# House Price Prediction

**Wenqing Fan, Hanfang Zhang, Zengrui Luo**

**Date: 08/06/2022**

**Project Summary**

In this project, our team tried to analyze which features will influence the sale price for a house and tried to use machine learning models such as linear regression and machine learning to predict the sale price based on the features we discovered. The dataset we used is file: House Prediction Data.csv, all works we did are in a jupyter notebook.

**Part I Data analysis**

We programmatically downloaded a data-set for analysis named as House Prediction Data.csv, and utilized pandas to read the csv file and compute the correlation matrix about different features of the dataset. Finally create the heatmap to visualize the correlation matrix. The heatmap represents several features which may help to predict the sale price:

['GrLivArea','TotalBsmtSF','LotArea','GarageArea','WoodDeckSF','ScreenPorch','PoolArea','MasVnrArea','OpenPorchSF','3SsnPorch']

Because these values have high scores in the correlation matrix.

We also used a parallel plot to analyze the connection in different features which may influence the sale price. Unfortunately, because there are lots of features in the dataset and there are many records, the parallel doesn’t illustrate the relationship clearly.

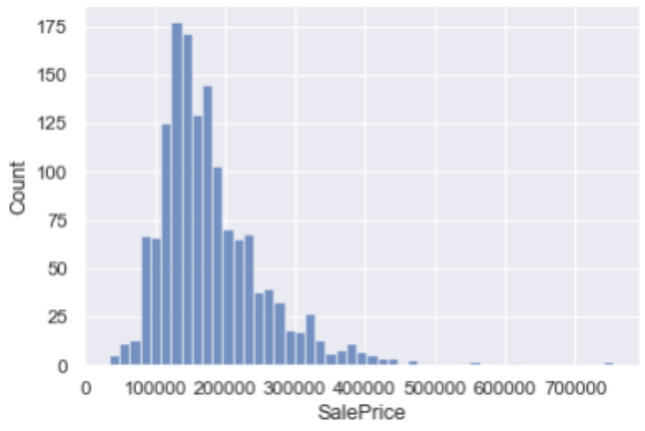
Then I analysis the missing value in the dataset:

| | |
|---|---|
| MSZoning: 0.0014% missing values | Functional: 0.0007% missing values |
| Alley: 0.9322% missing values | FireplaceQu: 0.4865% missing values |
| Utilities: 0.0007% missing values | GarageType: 0.0538% missing values |
| MasVnrType: 0.0082% missing values | GarageFinish: 0.0545% missing values |
| BsmtQual: 0.0277% missing values | GarageQual: 0.0545% missing values |

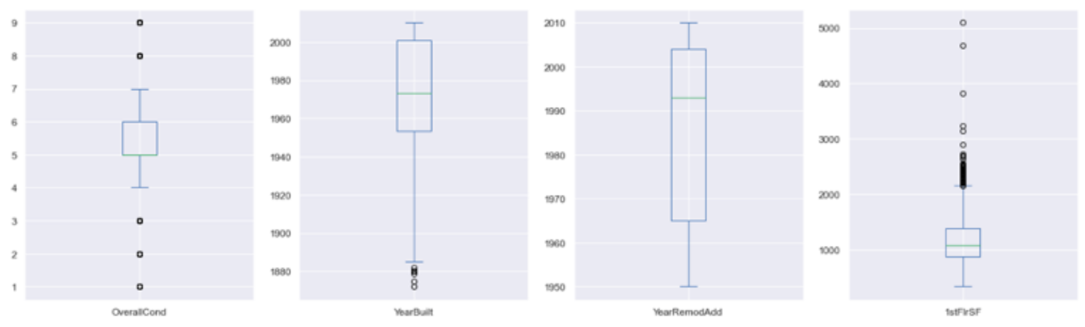
| | |
|---|---|
| BsmtCond: 0.0281% missing values | GarageCond: 0.0545% missing values |
| BsmtExposure: 0.0281% missing values | PoolQC: 0.9966% missing values |
| BsmtFinType1: 0.0271% missing values | Fence: 0.8044% missing values |
| BsmtFinType2: 0.0274% missing values | MiscFeature: 0.964% missing values |

The result represents that for most features, the ratio of missing value is low and acceptable.

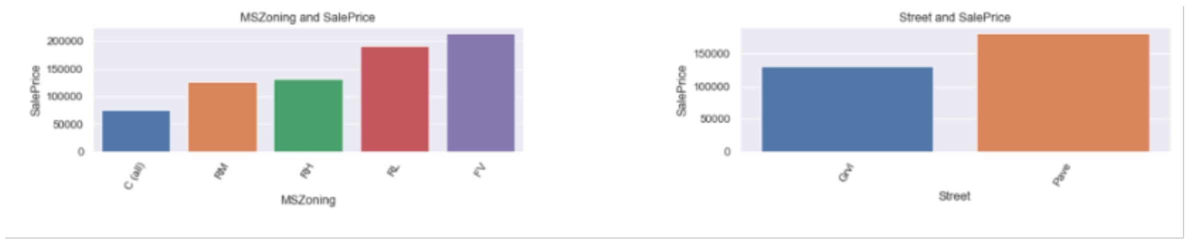
Then, we plotted different graphs to research the distribution of features. For example, the distribution represents that the sale prices are mainly in the range between 100k and 200k.



We created boxplots to research outliers in the dataset. We found that there are lots of outliers for some features such as LowQualFinSF, GrLivArea, LotArea.



There is a special part in our project, for categorical variables, we created barplots which illustrate the mean of sale price for different values in the column. This graph shows how different categories for those features might influence the sale price.



**Part II Data training and prediction**

- **Implement a linear regression class from scratch**

We implement a linear regression class based on least squares method, the functional model as follows. The same prediction is executed by the existing module to prove that this scratch is valid. (Since we only use one column of data as training input, the  result is not that precise.)

$$y = \beta_1 x + \beta_0 \quad \text{and} \quad \beta_1 = \frac{\sum x_i y_i - \frac{1}{n}\sum x_i \sum y_i}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} \; ; \; \beta_0 = \frac{1}{n}\sum y_i - \beta_1 \frac{1}{n}\sum x_i$$

- **Compare different regression models**

We import the scikit-learn package in the notebook. We use functions from it to encode non-value features, normalize the dataset and split the dataset into train set and test set.
Then we fit the model using the fit function in the regressor object, and then we calculate the MSE to evaluate the performance of the model with the score function.
We tested 4 models: Linear regressor, Linear SVR, NuSVR and XGBoost.
We tested linear regression and SVM regressors for the whole dataset. Firstly, the result shows that linear regression has better performance than SVM regressors. And for SVM regressors, Linear SVR has better performance compared to NuSVR.
We also tried the XGBoost model. This model has excellent performance in this dataset (lowest MSE and the score is higher than 95%).

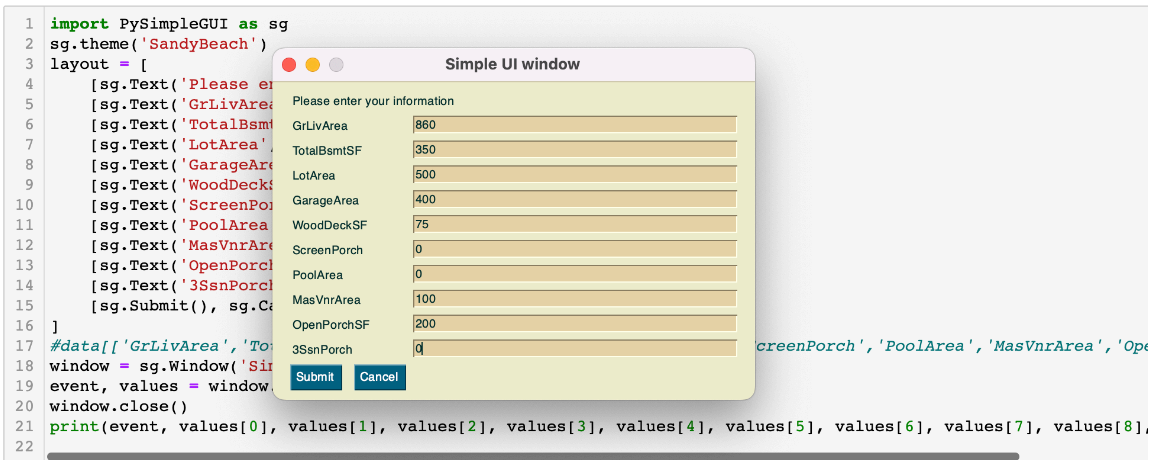
- **Use quantitative methods to pick features, descriptively compare the results to other data sets**

We tested for features that were picked in the above process, unfortunately, these features are not enough to train an accuracy model.

- **Simple UI creation**

After we completed the construction of the regression models, we created a simple UI window using the PySimpleGUI package, which allows people to input their house information and get their house prices estimates. In the figure below, users can input information towards GrLivArea, TotalBsmtSF, LotArea, GarageArea, WoodDeckSF, ScreenPorch, PoolArea, MasVnrArea, OpenPorchSF, and 3SsnPorch. Then, we created a "User" dataframe, which can store the information entered by the user and convert these variables into numeric variables for further calculations.

Simple UI



```
import PySimpleGUI as sg
sg.theme('SandyBeach')
layout = [
    [sg.Text('Please e
    [sg.Text('GrLivArea
    [sg.Text('TotalBsmt
    [sg.Text('LotArea',
    [sg.Text('GarageAre
    [sg.Text('WoodDeck
    [sg.Text('ScreenPor
    [sg.Text('PoolArea
    [sg.Text('MasVnrAre
    [sg.Text('OpenPorch
    [sg.Text('3SsnPorch
    [sg.Submit(), sg.Ca
]
#data[['GrLivArea','To                                creenPorch','PoolArea','MasVnrArea','Ope
window = sg.Window('Si
event, values = window.
window.close()
print(event, values[0], values[1], values[2], values[3], values[4], values[5], values[6], values[7], values[8],
```

Simple UI window

Please enter your information

| | |
|---|---|
| GrLivArea | 860 |
| TotalBsmtSF | 350 |
| LotArea | 500 |
| GarageArea | 400 |
| WoodDeckSF | 75 |
| ScreenPorch | 0 |
| PoolArea | 0 |
| MasVnrArea | 100 |
| OpenPorchSF | 200 |
| 3SsnPorch | 0 |

Based on the MSE's evaluation of the performance of Linear regression, Linear SVR, NuSVR and XGBoost, we decided to use linear regression model and XGBoost model to calculate and predict the house price input by users. The following figure is the predicted value of Linear regression model and XGBoost model.

```
In [56]:  UserPred_linear = regressor.predict(User)
          UserPred_linear

Out[56]:  array([[306076.08467536]])

In [57]:  UserPred_XGBoost = xg_model.predict(User)
          UserPred_XGBoost

Out[57]:  array([286416.97], dtype=float32)
```