CS521 Final Project
House Price Predictor
Yuxiao Wu, Jisun Lee
December 8 2021

Introduction:

The purpose of this project is to create a linear regression model to predict house price using the data set 'House Prediction Data' which contains about 3000 rows of data for house details and their sale price. Python packages such as SKlearn, Pandas and Matplotlib were utilized during the project. At the end of the project, a simple user interface was created for users to input value and predict their house price.

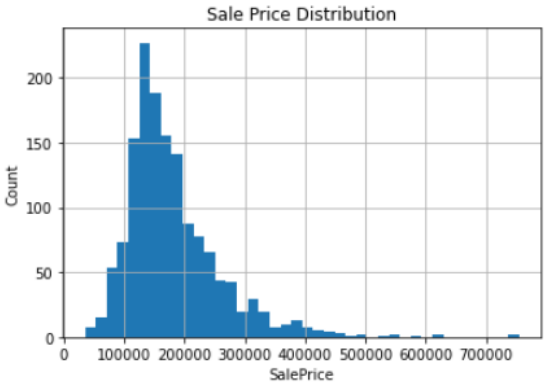
Import the Dataset:

following code was used to Making a network request for data, read the csv file online and use pandas to convert the csv file into a data frame.

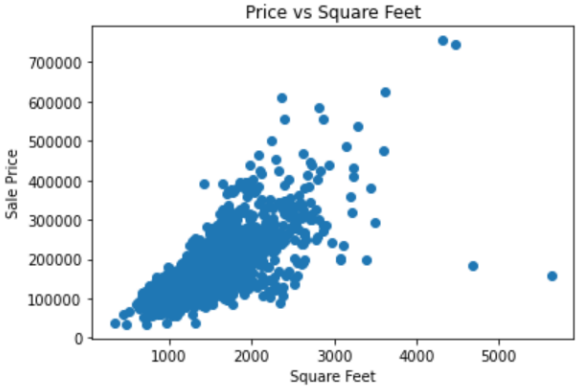
```
# import the data set
URL = 'https://raw.githubusercontent.com/bursteinalan/Data-Sets/master/Housing/House%20Prediction%20Dat
r = requests.get(URL, allow_redirects = True)
decoded_content = r.content.decode('utf-8')
decoded_content = StringIO(decoded_content)
df = pd.read_csv(decoded_content, sep = ',')
```

Dataset Exploration:

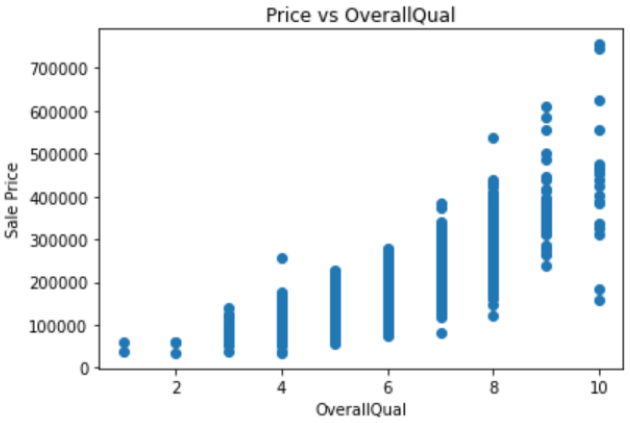
Before performing analysis for the data, some explorations for the data set was done by some plot. The below graph shows the distribution for the sales price using a histogram. It shows that most house prices are centered between 100k to 200k, with fewer above 300k.



Next, we want to see how some important factors can affect the price of the house. The graph below shows the scatter plot between living area and the sale price. As shown in the graph we can see the living area has a linear relationship with the sale price, which is reasonable, larger the house, higher the price.



Another important attribute is the overall quality of the house when sold, and the quality is scaled from 1 to 10. A scatter plot between sale price and overall quality of the house is shown below. The figure clearly shows that as the overall quality of house becomes higher, the sale price will be higher.



Handling missing data:

When looking through the SalePrice column in the data set, we found that almost half of the sale price is missing. Thus, we need to drop all rows whose sale price has a null value.
Then we continue to look for missing data for other columns and find attributes such as 'PoolQC', 'Alley', 'Fence', 'FireplaceQu' have many null values, which mean they are not useful for our analysis. So, we drop the columns that have multiple null values. Both steps were performed using the below code.

```
# Because SalePrice is our target attribute, drop all rows that SalePrice is Null
df = df[df.SalePrice.notnull()]

# Continue looking for missing data for other columns, columns with mutiple missing values are
# unusable and drop the column with many missing values
null_column = df.isnull().sum().sort_values(ascending=False)
print(null_column[null_column > 0])
null_value = ['PoolQC','MiscFeature','Alley','Fence','FireplaceQu','FireplaceQu','LotFrontage',
              'GarageYrBlt','GarageCond','GarageType','GarageFinish','GarageQual','BsmtFinType2',
              'BsmtExposure','BsmtQual','BsmtCond','BsmtFinType1','MasVnrArea','MasVnrType',
              'Electrical']
df = df.drop(null_value,axis=1)
```

Convert Categorical Data to Numeric:

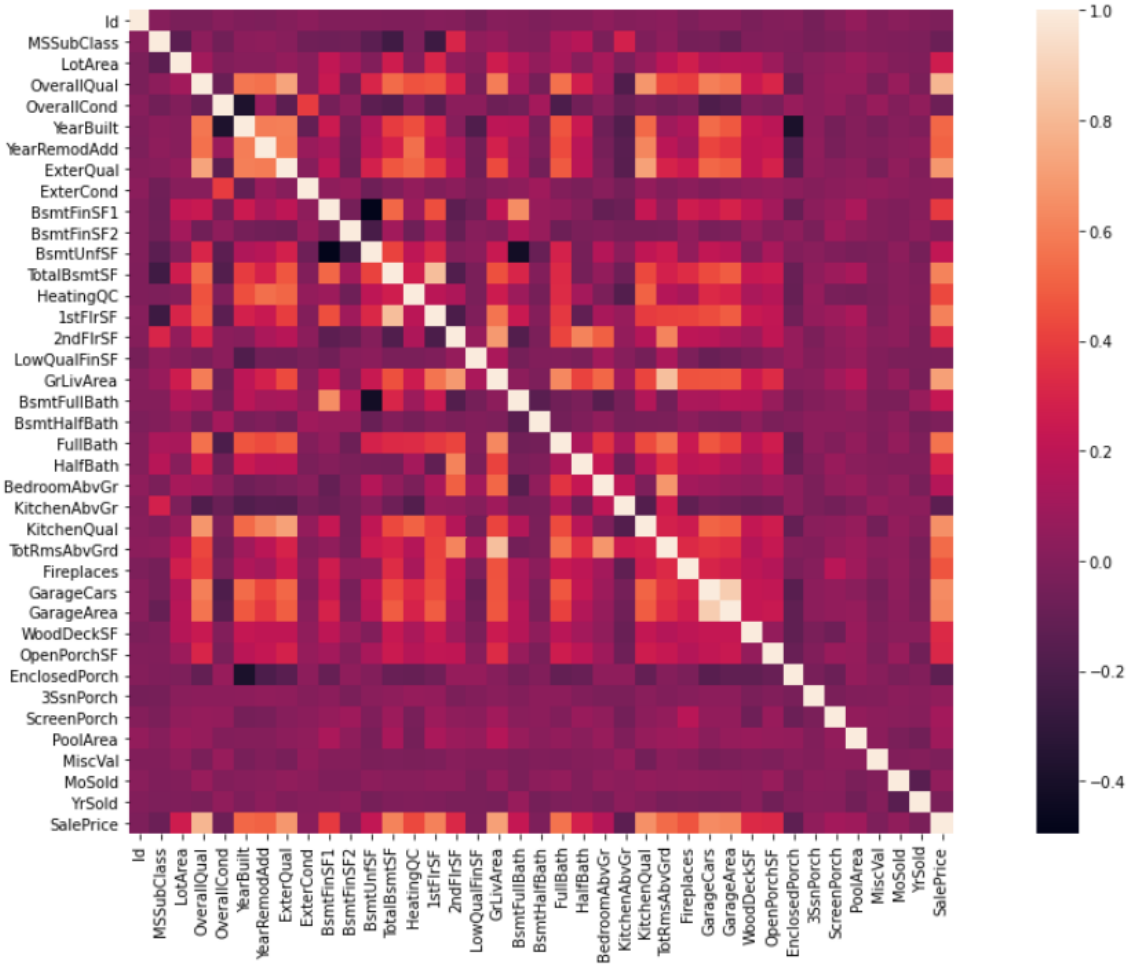
Another problem for the data set is that there are a lot of non-numeric values such as the Quality of the house, Type of road access to property, and etc. And these non-numerical values are unable to perform the linear regression, so we need to convert these categorical data into numerical data. The four columns were chosen to perform the conversion which are ExterQual, ExterCond, HeatingQC and KitchenQual, and these were done by convert the quality order ['Po','Fa', 'TA','Gd','Ex'] to numbers from 0 to 4 to show the quality level. Code was shown below.

```
# We found that many columns have non-numeric values that can not use for linear regression
# So we want to convert some categorical data to numerical data
quality_order = ['Po','Fa', 'TA','Gd','Ex']
df.ExterQual.replace(quality_order,[0,1,2,3,4], inplace=True)
df.ExterCond.replace(quality_order,[0,1,2,3,4], inplace=True)
df.HeatingQC.replace(quality_order,[0,1,2,3,4], inplace=True)
df.KitchenQual.replace(quality_order,[0,1,2,3,4], inplace=True)
```

Finding Correlations:

The first step of implementing linear regression is to find the correlations between each attribute and the target attribute SalePrice. This was done by using the built in function corr(), and a heatmap was created to show the correlations between each variable.

```
# Now Find the corelation between each varibale and the SalePrice
coorelation = df.corr()
f, ax = plt.subplots(figsize=(20, 10))
sns.heatmap(coorelation,  square=True);
```



As shown in the graph, we can see that each square shows the correlation between two variables, and lighter the color means higher positive correlation while darker color means higher negative correlation.

In order to determine which attributes should be used for performing a linear regression model, those attributes were sorted by the correlations with SalePrice, and attributes with absolute value of correlation higher than 0.5 were selected as code shown below.

```
# we want to chose correlation variable greatter than 0.5 to perfrom the linear regression.
price_corr = coorelation['SalePrice']
price_corr = price_corr.sort_values(ascending=False)
price_corr[abs(price_corr > 0.5)]
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
ExterQual       0.682639
KitchenQual     0.659600
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
Name: SalePrice, dtype: float64
```

Among those 12 attributes, we choose to use those variables as factors for predicting the SalePrice. ,'OverallQual' ,'ExterQual' ,'KitchenQual', 'GarageCars', 'GarageArea', 'TotalBsmtSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt', and 'YearRemodAdd'.

Split the Train and Test Data:

Next step is to split the data set into train data and test data. It means we use the train data to build the model and use the test data to see how the model performed. In this project, we set the train data to be 80 percent of the original dataset and test data to be the rest 20 percent. In order to avoid bias, all data was chosen randomly from the data set.

```
# Split the train and test data
df_X = df[['GrLivArea','OverallQual','ExterQual','KitchenQual','GarageCars','GarageArea',
           'TotalBsmtSF','FullBath','TotRmsAbvGrd','YearBuilt','YearRemodAdd']]
df_Y = df['SalePrice']
x_train, x_test, y_train, y_test = train_test_split(df_X,df_Y,test_size = 0.2,random_state = 2)
```

Fit the Linear Regression model:

After splitting the train and test data, we used the linear model from sklearn to fit the model and then used the test data to score the model. For multiple linear regression, a formula can be shown as $y = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \varepsilon$ where y is the predicted value, in this case is the SalePrice. β_0 is the y intercept, β_n is the coefficient for each variable, such as the OverallQual and GrLivArea. Code below shows the coefficient for the linear equation and the model test score. What we got for the score is 0.839 which is a reasonable value.

```
# fit the linear model and test the score
reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)
test_score = reg.score(x_test,y_test)
coef_arr = reg.coef_
print('coefficient = ', coef_arr)
print('model test score = ', test_score)
```

```
coefficient =  [ 4.86554410e+01  1.38101717e+04  1.50770437e+04  1.35293890e+04
  1.16560517e+04  8.10532285e+00  2.52511657e+01 -6.34571773e+03
  6.72186734e+02  2.25020906e+02  7.67565603e+01]
model test score =  0.8391619774105572
```

Compare Different Regression Models:

Other than the normal linear regression model, we implemented three other regression models which are Lasso, Ridge and Elastic Net models. They are all imported from the Sklrean package. We fit the test and train data to different models and calculate the mean squared error and r squared value to see how the different model performs. As shown in the results below, we can see that all models score very closely, while the lowest mean squared error model is Elastic net model and the model with highest r squared value is the normal linear regression model.

```
------Mean Squared Error------
Normal Linear Regression:  1065112062.5082631
Lasso Regression:  1065108853.3695242
Ridge Regression:  1064945038.4021134
ElasticNet Regression:  1056316593.534198
------R-Squared Value------
Normal Linear Regression:  0.7851854227567412
Lasso Regression:  0.7851853529345899
Ridge Regression:  0.7851762198404226
ElasticNet Regression:  0.7831063094076793
```

Use the Normal linear regression to Predict the Price:

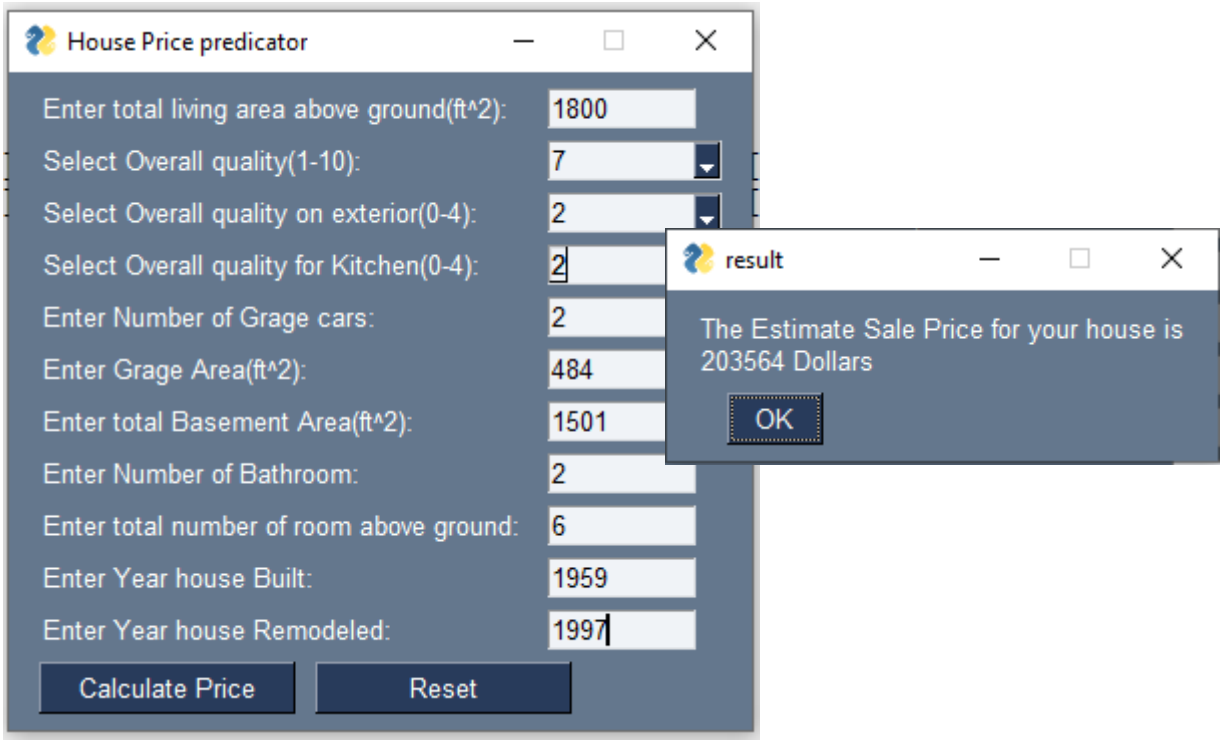
Now we can use the equation obtained above to predict the house price simply by plugging in each value for the variables. As we can see in the code below, by plug in the Area, overall quality, built year etc, we can get the predicted sale price.

```
predict_price = reg.predict([[1800,7,2,2,2,484,1501,2,6,1959,1997]])
print('predicted price = ', predict_price)
```

```
predicted price =  [203564.41443822]
```

User Interface:

We also created a simple application for people to enter information and get an estimate for their house price by using the PySimpleGUI package. As shown in the screenshot below, a window will pop out for people to either enter or select values. When they click the Calculate price button, the estimated price will be calculated using the linear model derived before.



Conclusion:

Through this project, we learned how to analyze the data, evaluate proper data to build a regression model, and develop using the model. Since the data has some null and different types, we needed to fix the data to standardize. Before that process, we did not realize that there are so much missing data. After cleaning some data, we built several regression models and visualized the data based on what we analyzed because it is easy to read the result to distinguish which variables are useful to predict. We also produced a simple application for users to calculate the house with the information they have. Therefore, people can predict the price of a house with the linear regression model.