 Project: Predicting House Price using Linear Regression

1. **Introduction:**

This project demonstrates to predict the house price by using linear regression and try to understand how the dependent and independent variables relate to variable (‘askprice’) - the house sale price.

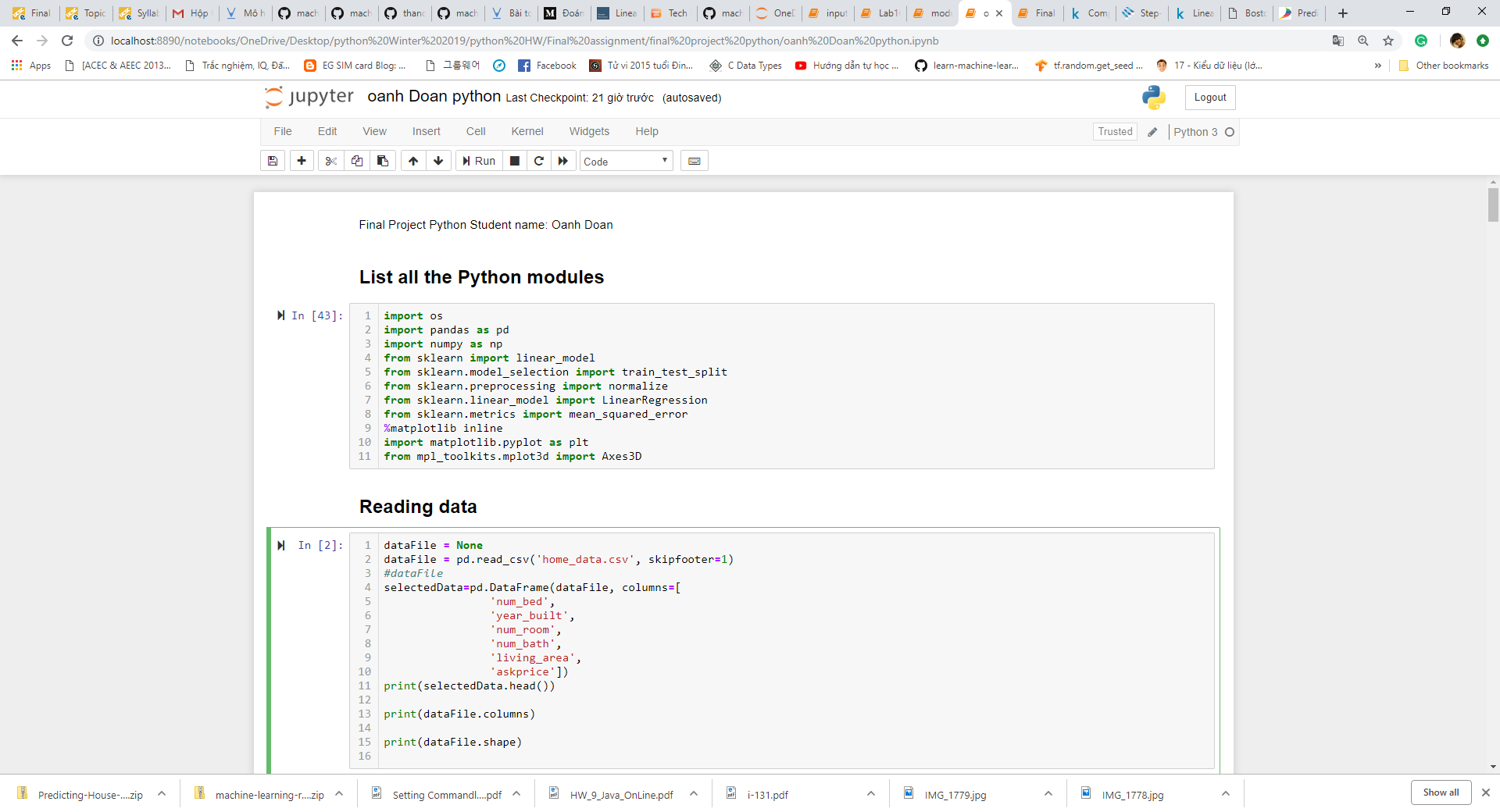
A small dataset of house data with 22 attributes of 555 house data points (555 rows). However, some of the most correlated variables such as 'num\_bed', 'year\_built', 'num\_room', 'num\_bath', 'living\_area', was used in Scikit-Learn to see the relationship with 'askprice' variable. Then the multivariate linear regression is used to uncover the relationship between the size ('living\_area'), number of bedrooms ('num\_bed') and the price of the house ('askprice'). In order to minimize the cost function of Multivariate Linear Regression, the project implemented 3 variants of Gradient Descent algorithm

* Batch Gradient Descent
* Stochastic Gradient Descent
* Mini-Batch Gradient Descent

<https://github.com/softporcupine/pandas-matplotlib-example/blob/master/home_data.csv>

1. **Requirements**

The python libraries are necessary for the rest of this mini project



1. **Description of the Python program**

***Step 1: Load and explore the data***

Using pandas to load the data from local directory ('home\_data.csv') as data frame.

Housing dataset has 555 data points with 22 variables each.

There are twenty two attributes (variables) as below

[ 'num\_bed', 'year\_built', 'longitude', 'latitude', 'num\_room',

'num\_bath', 'living\_area', 'property\_type', 'num\_parking',

'accessible\_buildings', 'family\_quality', 'art\_expos',

'emergency\_shelters', 'emergency\_water', 'Facilities', 'fire\_stations',

'Cultural', 'Monuments', 'police\_stations', 'Vacant', 'Free\_Parking',

'askprice']

***Step 2: Select only some variables for the model and Calculate Statistics***

Select some features (variables) such as

'num\_bed',

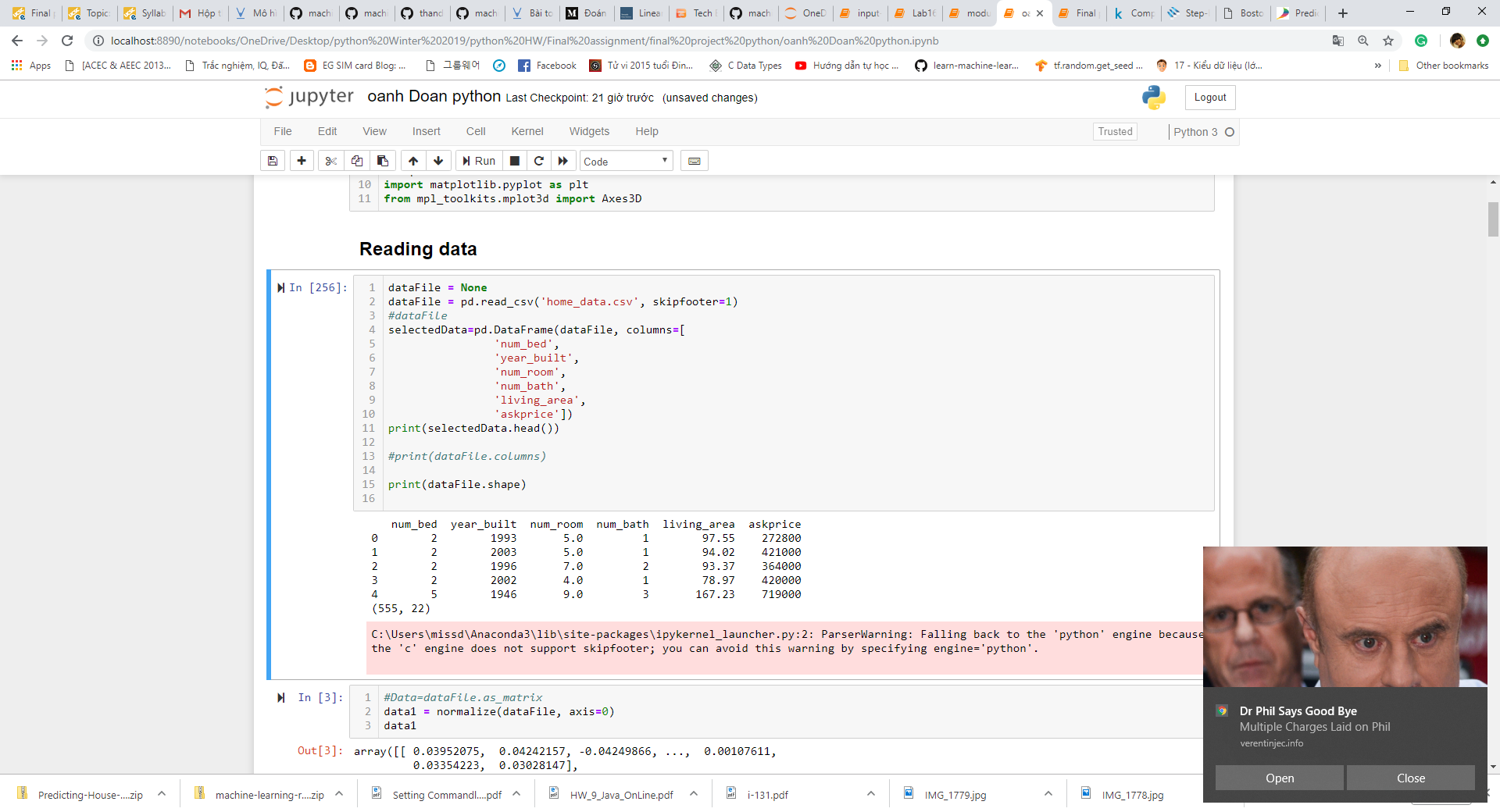
'year\_built',

'num\_room',

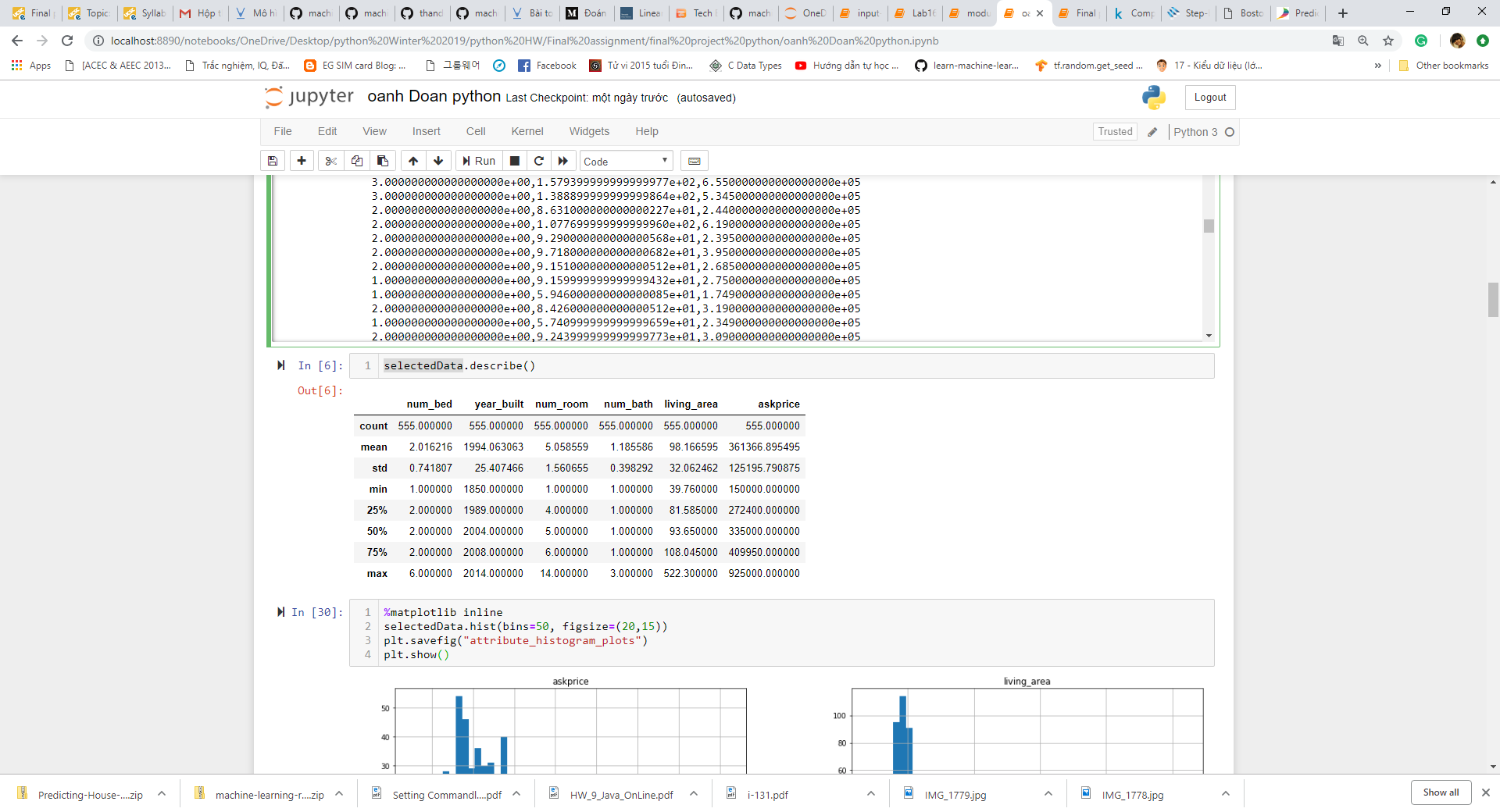
'num\_bath',

'living\_area'

Which I think that an increase or decrease in the value of that feature would lead to an increase or decrease in the value of 'askprice'.

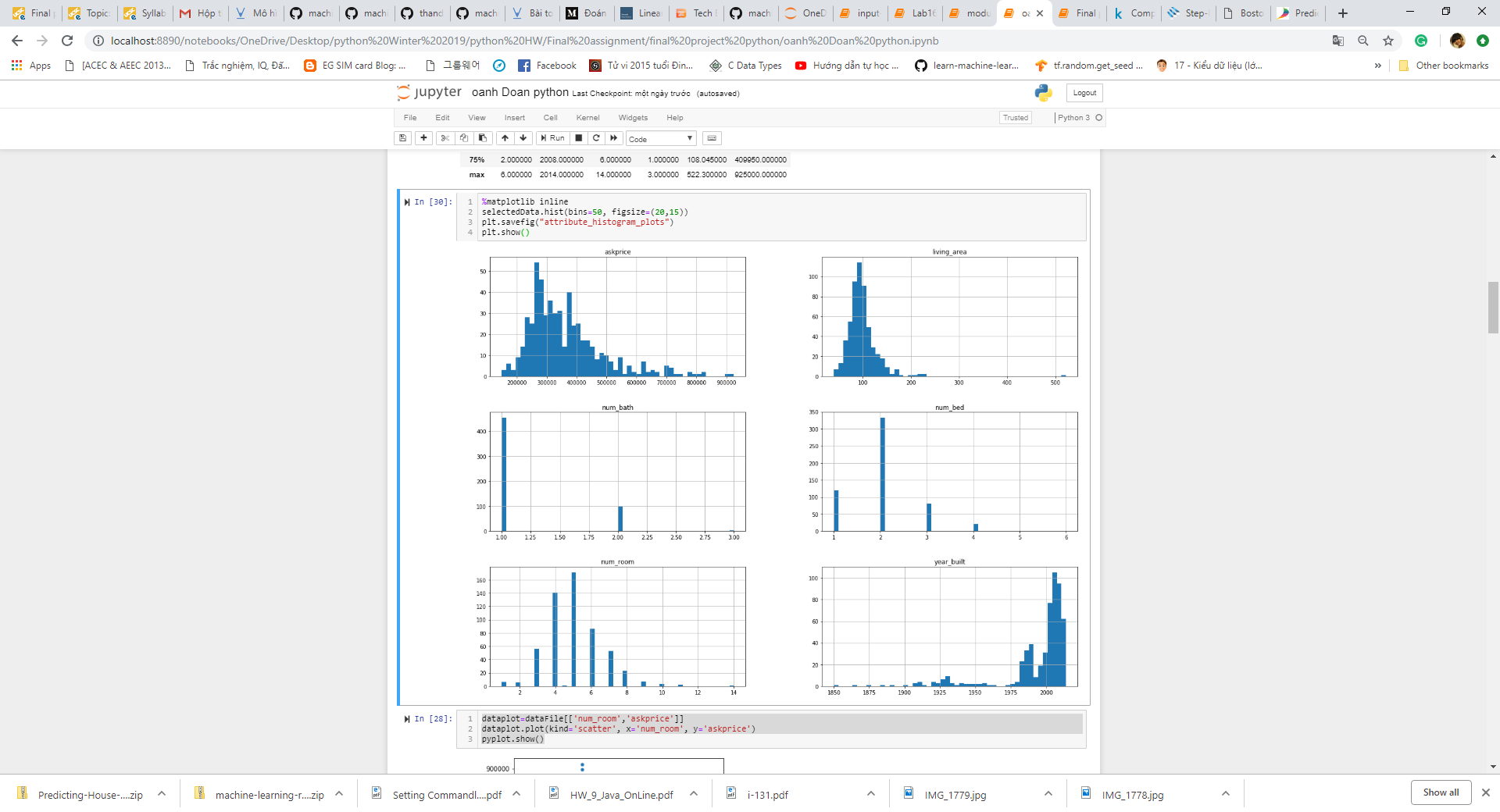


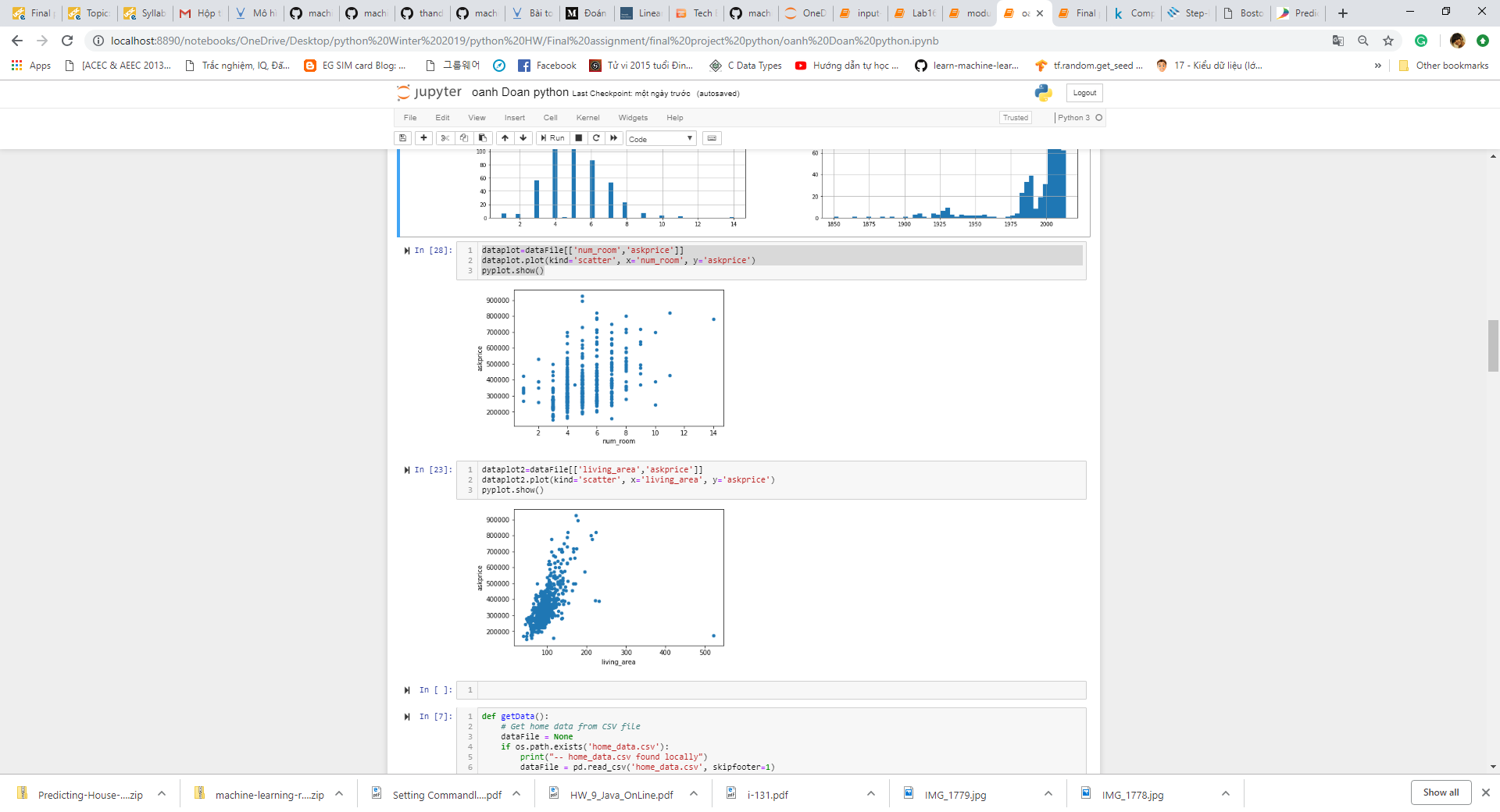
In order to better understand dataset, I take a look at statistics summary. However, I skip step of cleaning the dataset and handle the missing data in this report.



***Step 3: Visualize the data***

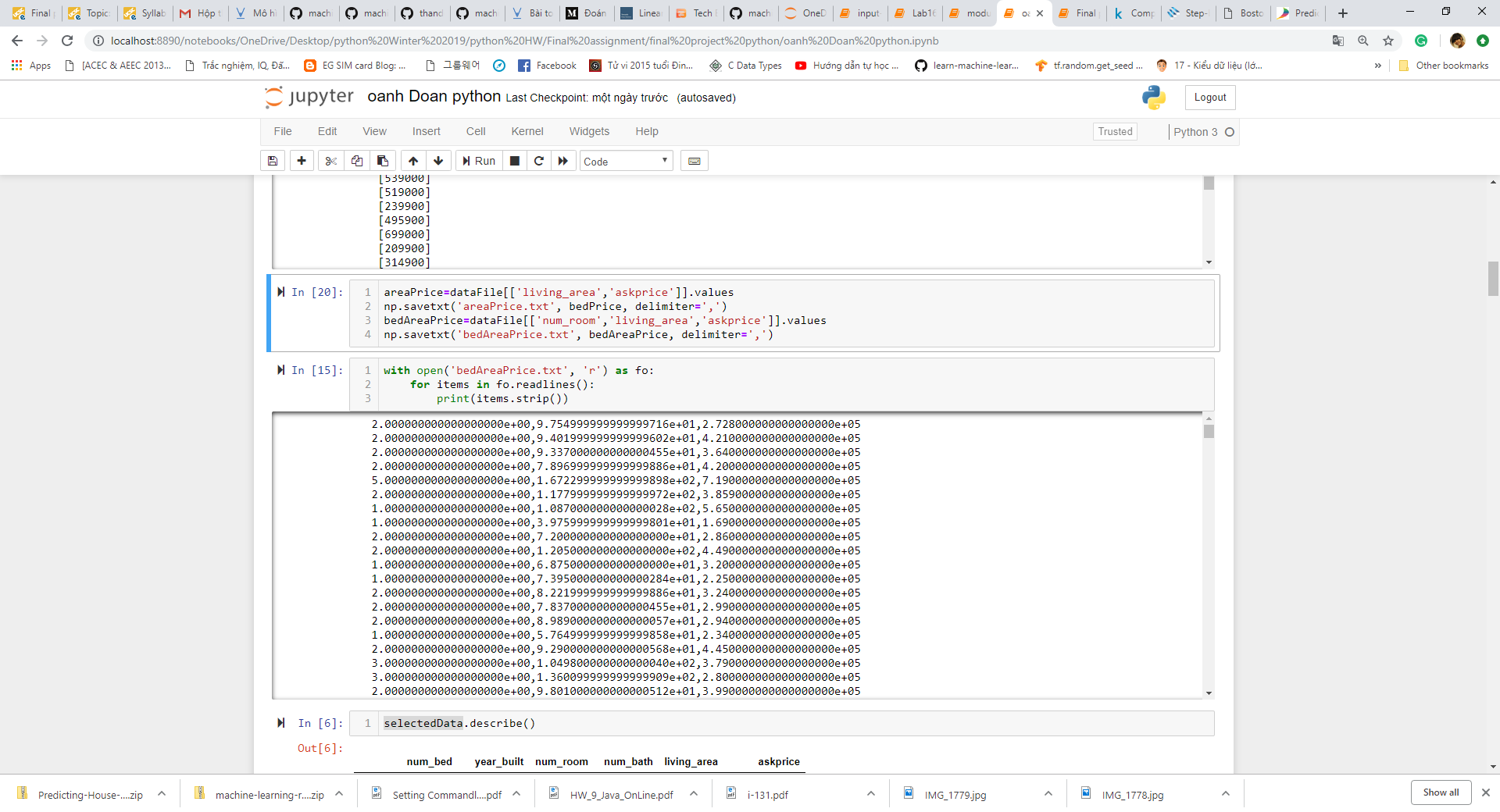
To see the distribution and relationship of these variables and model idea for the dataset.





***Step 4: Save new selected data set as text file***

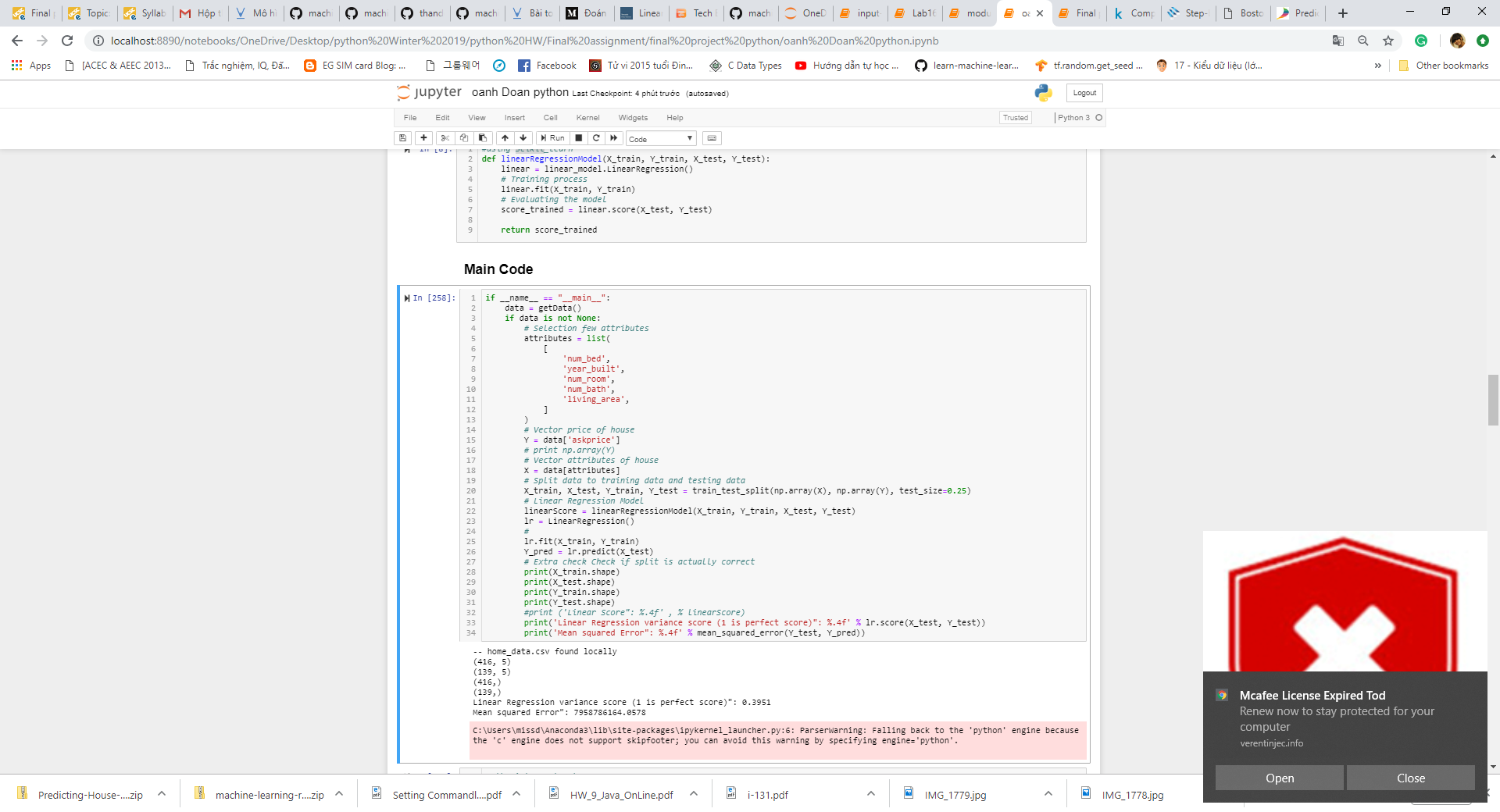
I want to save the selected data which contents only 3 columns (the size ('living\_area'), number of bedrooms ('num\_bed') and the price of the house ('askprice')) to the text file named 'bedAreaPrice.txt'. The new selected dataset is used to execute the multivariate linear regression and minimize cost later of this report.



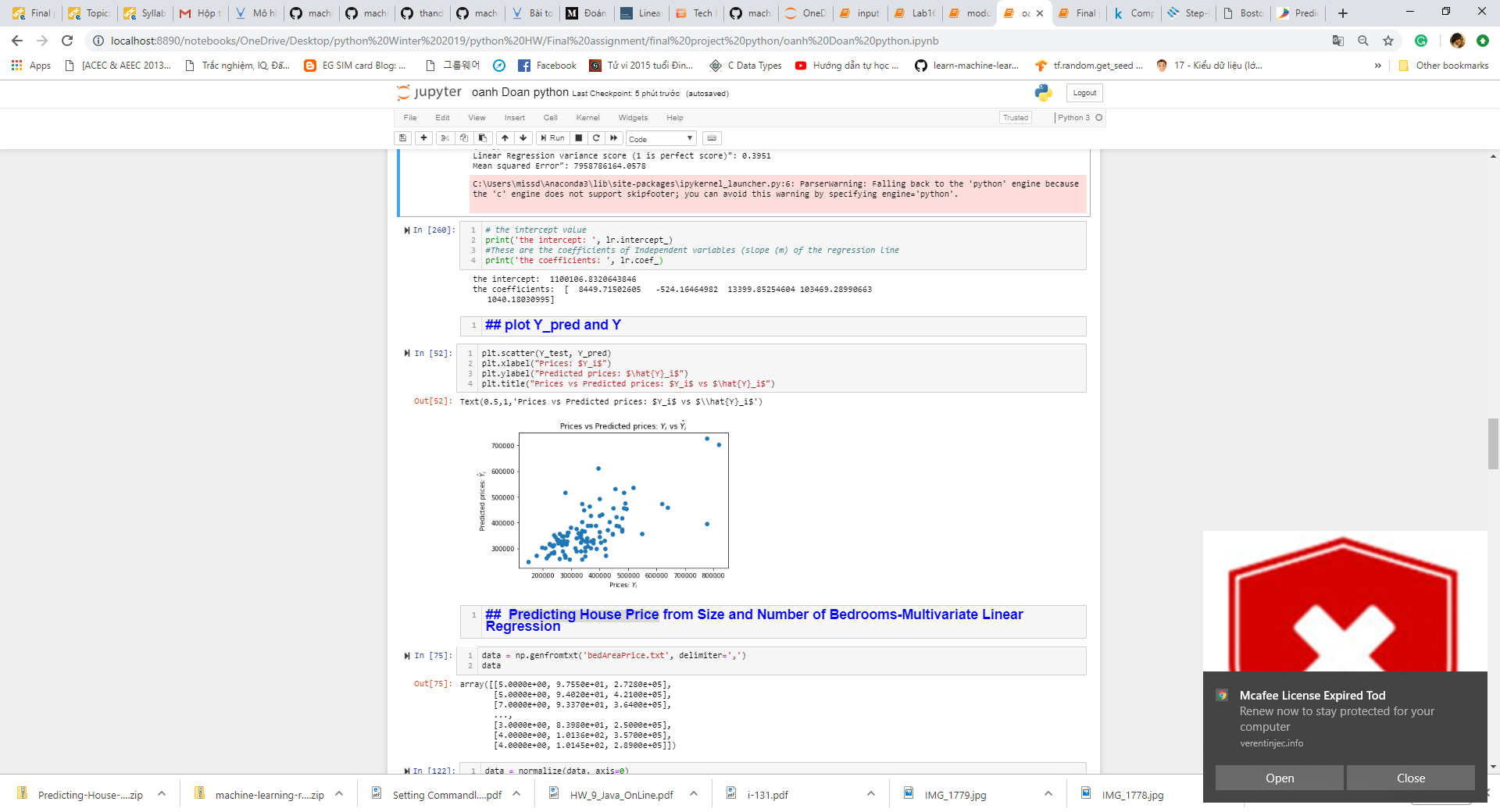
***Step 5: Use Scikit Learn to execute Linear Regression***

Scikit-Learn was used to see the relationship between 'askprice' variable and the most correlated variables such as 'num\_bed', 'year\_built', 'num\_room', 'num\_bath', 'living\_area'.

* + Calling a linear regression model.
  + Use train\_test\_split from sklearn.cross\_validation to shuffle and split the features and prices data into training and testing sets. Split the data into 75% training and 25% testing.
  + Fit the linear regression model to the training data set.
  + Assign the train and testing splits to X\_train, X\_test, y\_train, and y\_test.
* The coefficient of determination, R2, was calculated to quantify the model's performance. The values for R2 range from 0 to 1.
* calculate the intercept value, slope, mean squared error, coefficients, and the variance score.



* Plotting Y and Y\_pred



***Step 6: Predicting House Price from Size and Number of Bedrooms-Multivariate Linear Regression and display on 3D visualization.***

3 columns (the size ('living\_area'), number of bedrooms ('num\_bed') and the price of the house ('askprice')) was loaded from text file 'bedAreaPrice.txt'.

In this step, the main goals are:

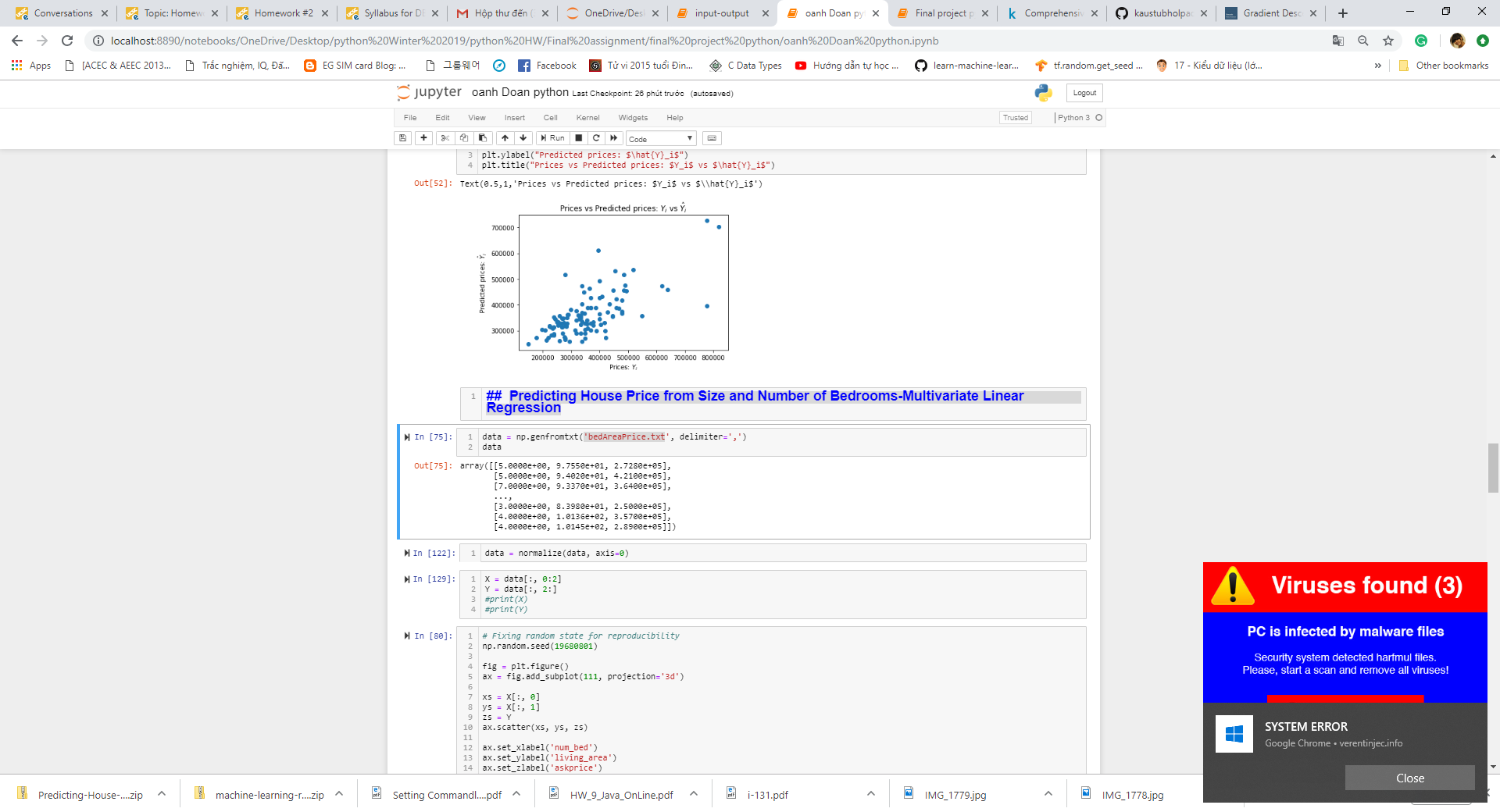
1. Using **Multivariate Linear Regression** to see the relation between 2 the most correlated variables (the size ('living\_area'), number of bedrooms ('num\_bed')) and the price of house.
2. Using 3 variants of Gradient Descent algorithm (Batch Gradient Descent, Stochastic Gradient Descent, Mini-Batch Gradient Descent) In order to minimize the cost function of Multivariate Linear Regression

(Theory reference: <https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3> )

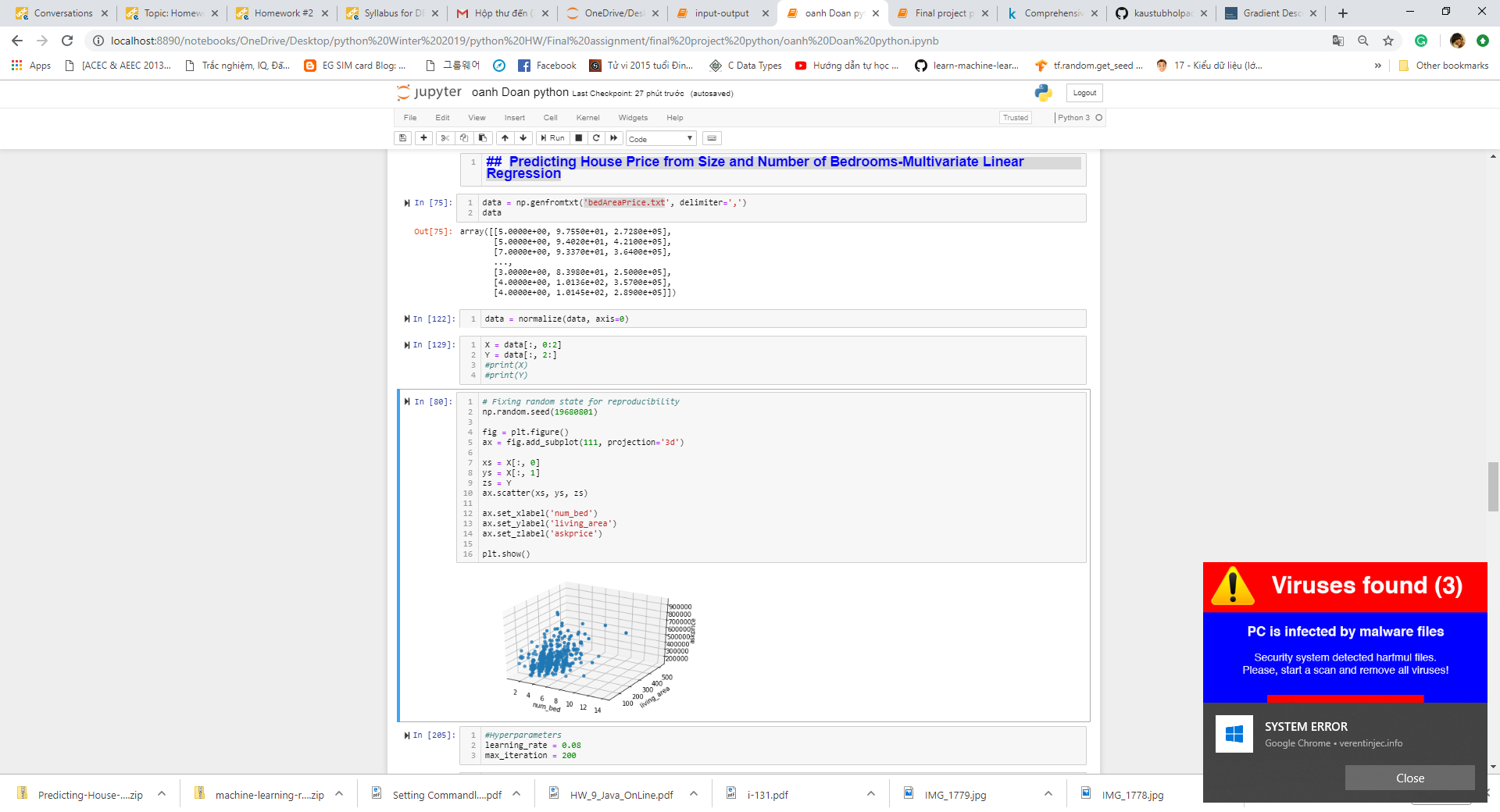
1. Then display in 3D to see the distributions.

* Get data and Normalize the data

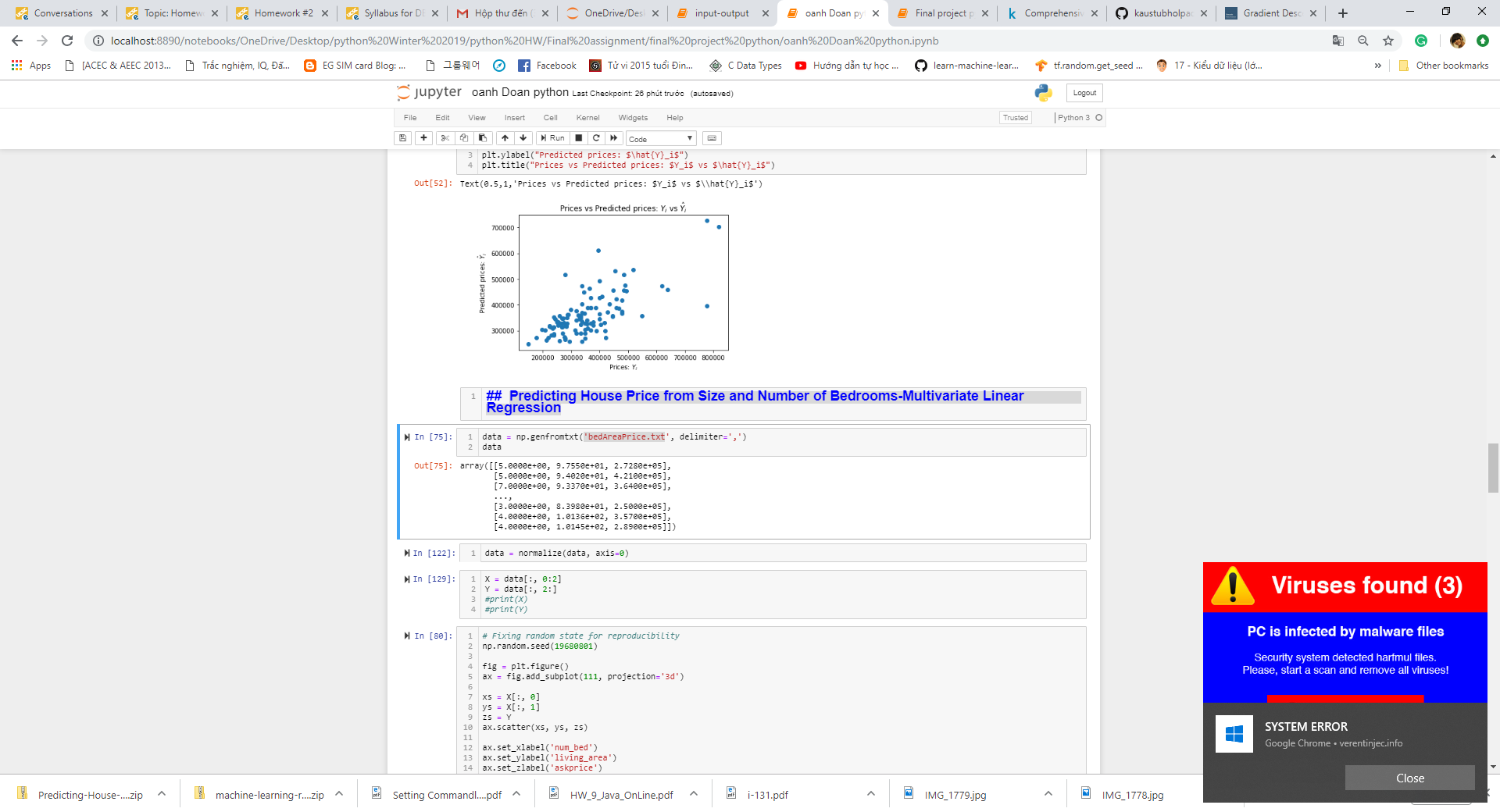
As you can see, the size ('living\_area'), number of bedrooms ('num\_bed') variables have different but comparable scales. Data was `normalized` .



* Plot data in 3D



* Create matrices and set hyperparameters:
* assign the first two columns as a matrix to X.
* assign the third column to y



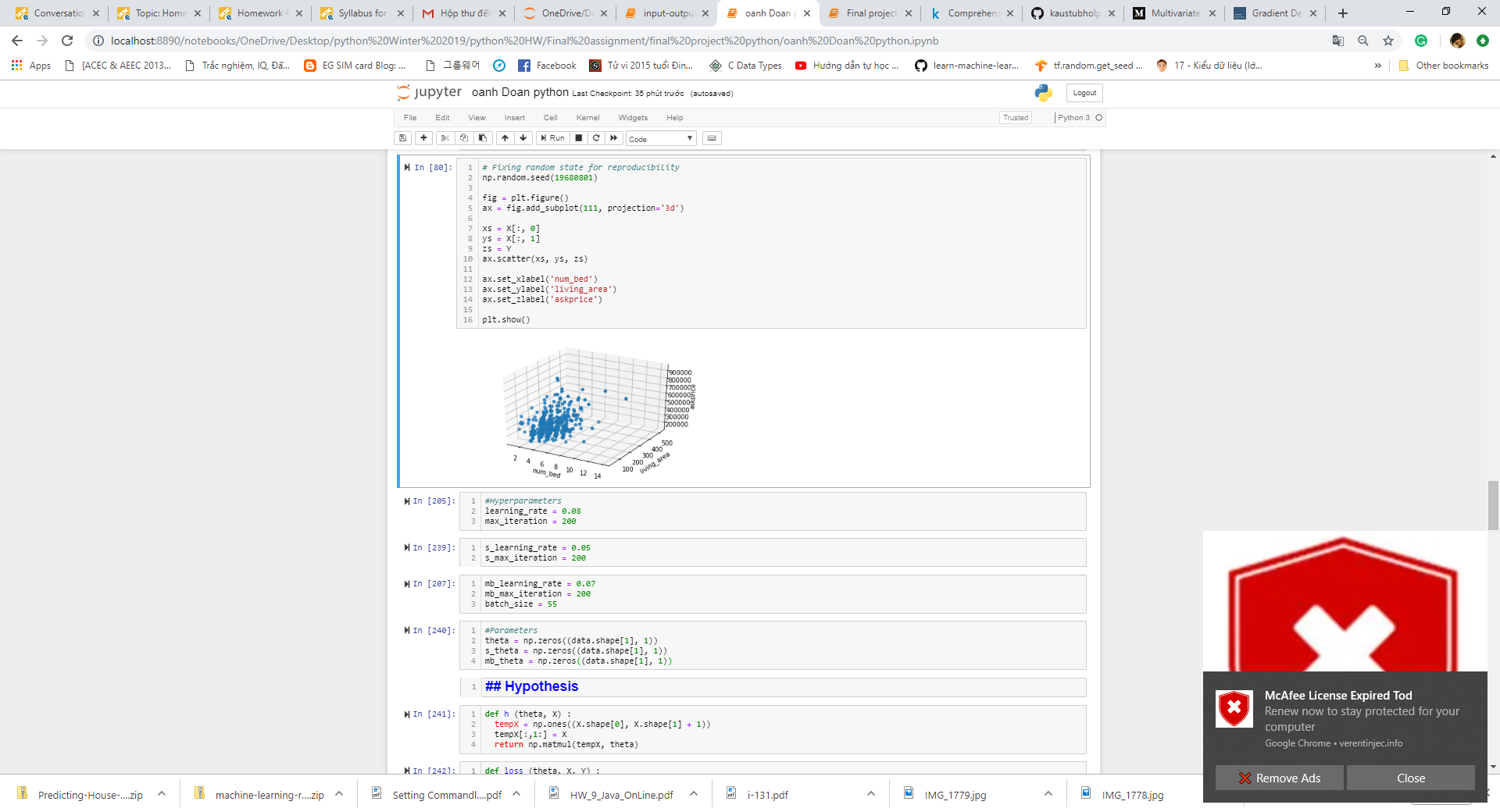
* set up the hyperparameters and initialize theta as an array of zeros

Initialize weight w and bias b to any random numbers.

Pick a value for the learning rate α. The learning rate determines how big the step would be on each iteration.

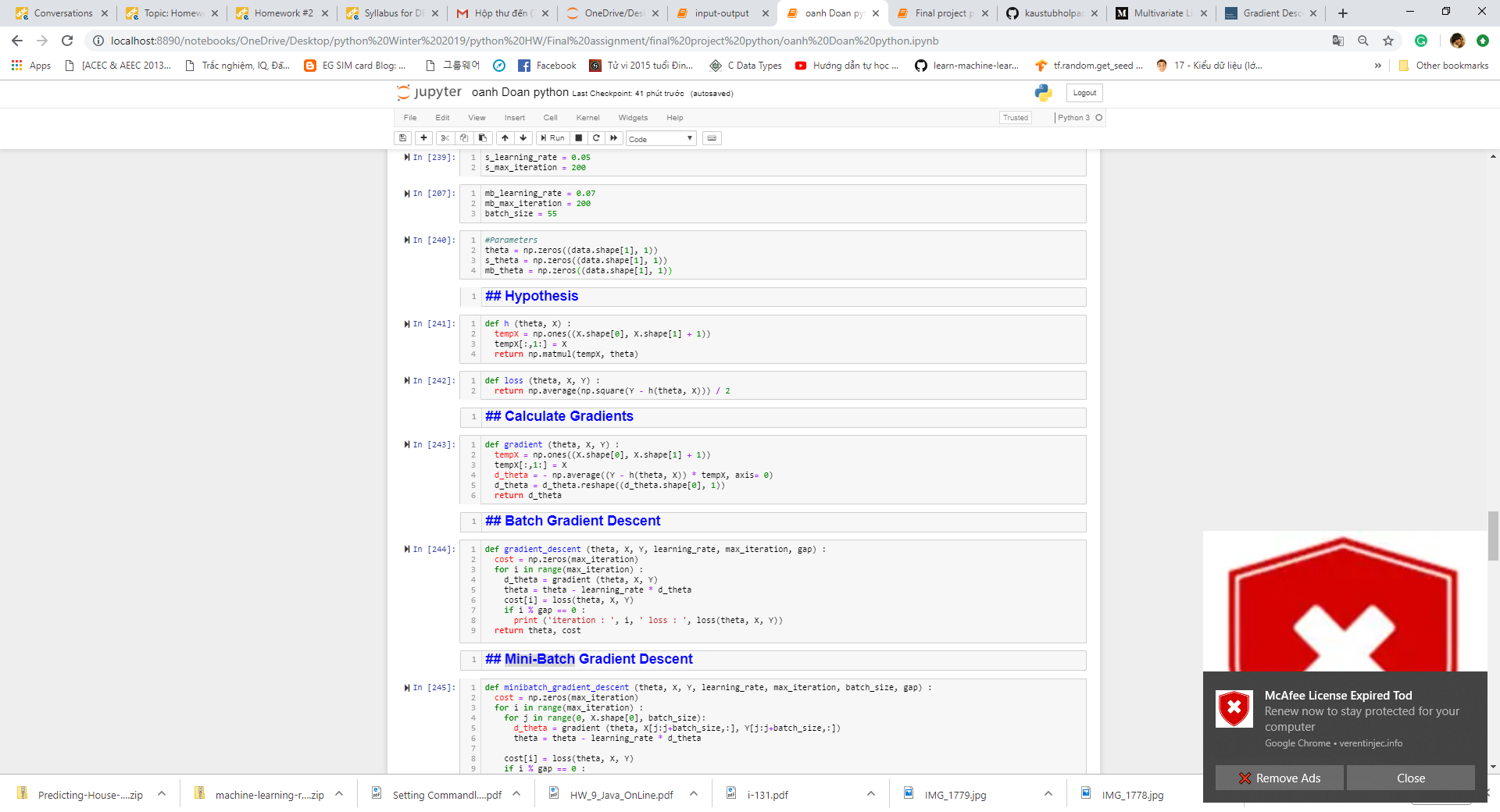
If α is very small, it would take long time to converge and become computationally expensive.

If α is large, it may fail to converge and overshoot the minimum.

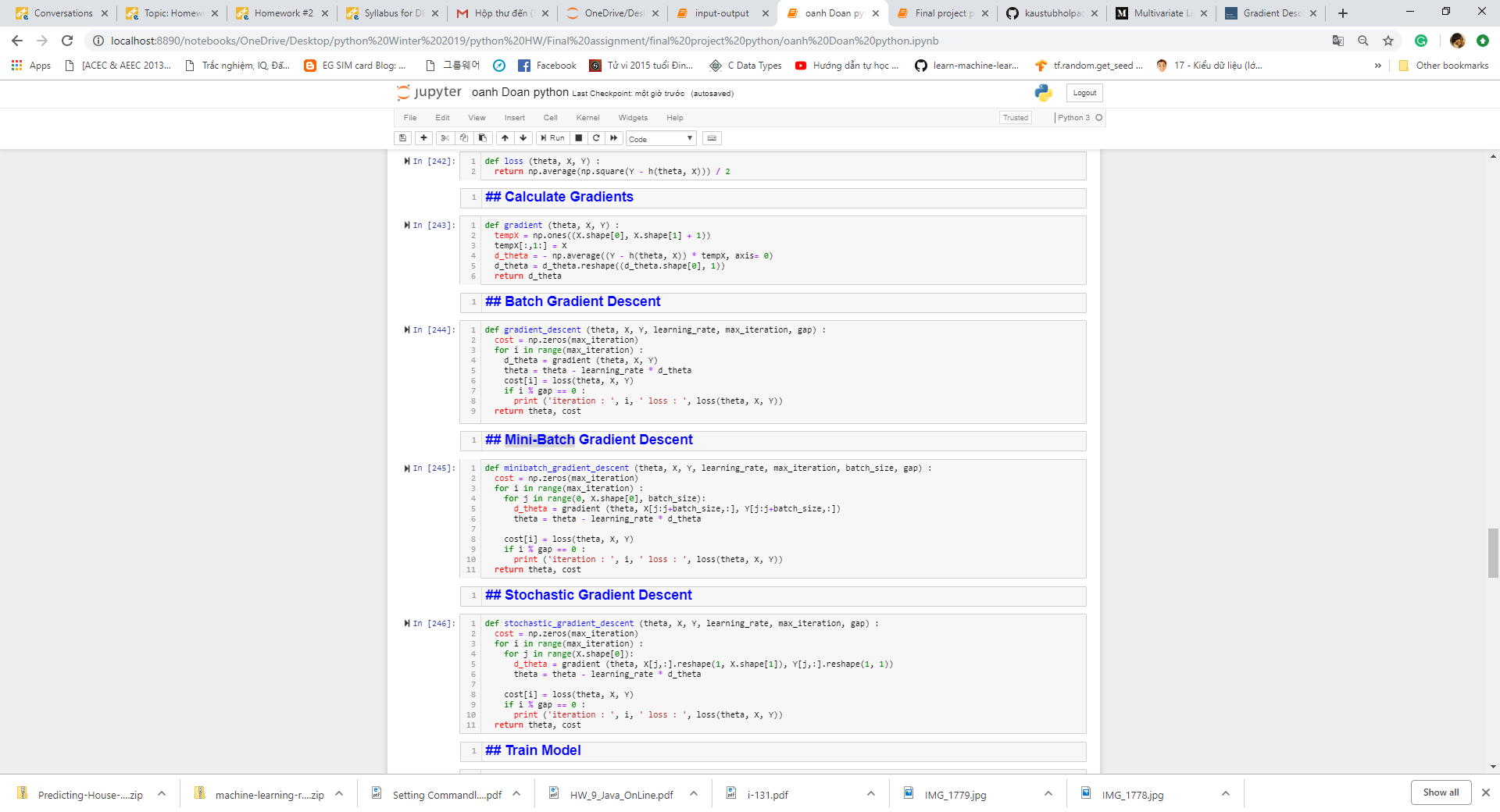


* Create the cost function

The loss function takes theta,x and and as parameters and computes the cost (loss). Then this cost is minimized by using gradient descent in next step.



* Create the Gradient Descent function:



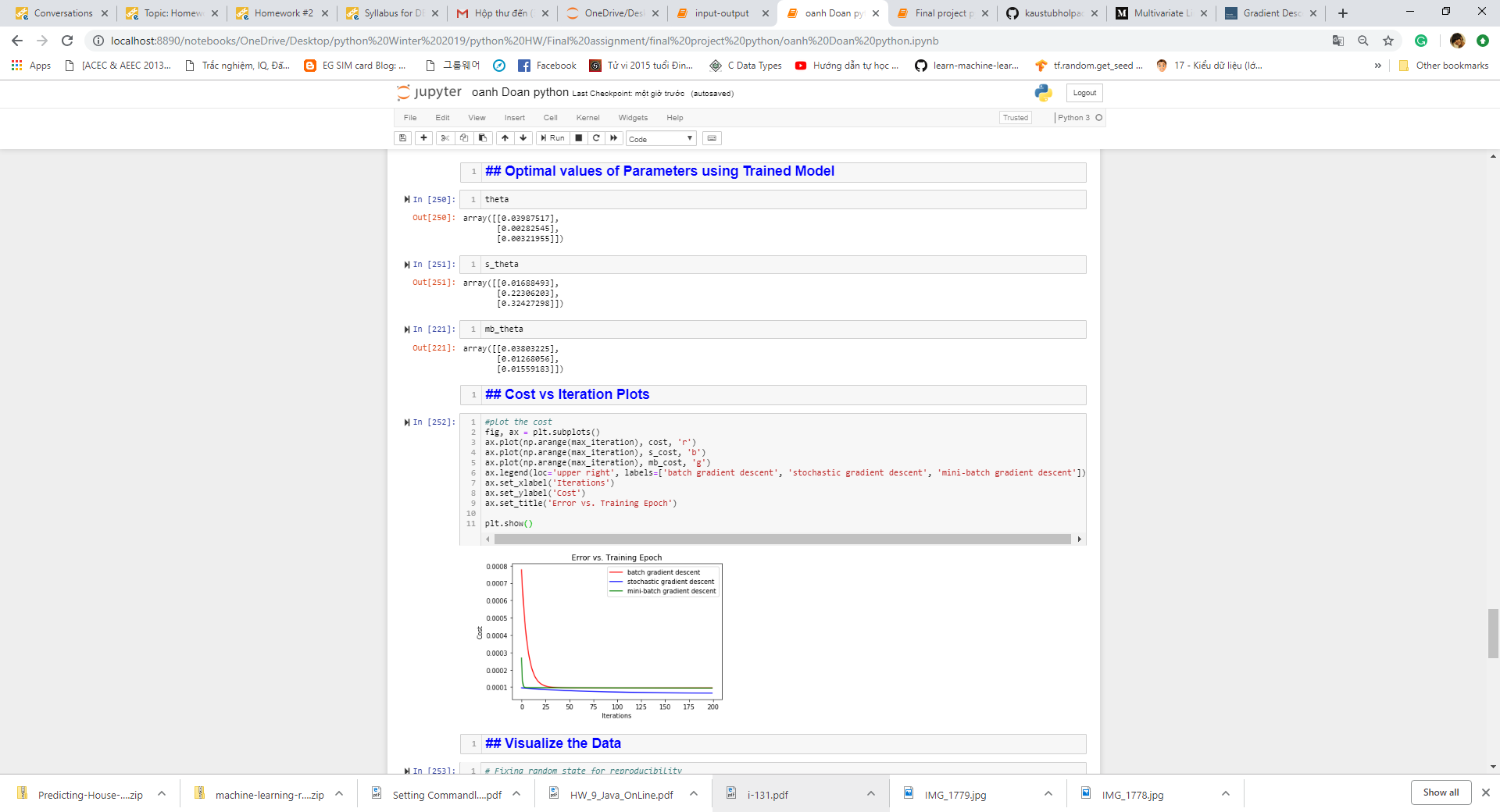
theta and X have the same number of columns the code will work.

**Batch Gradient Descent** : Each step of learning happens after going over all examples, therefore, the more the examples, the lower the standard error.

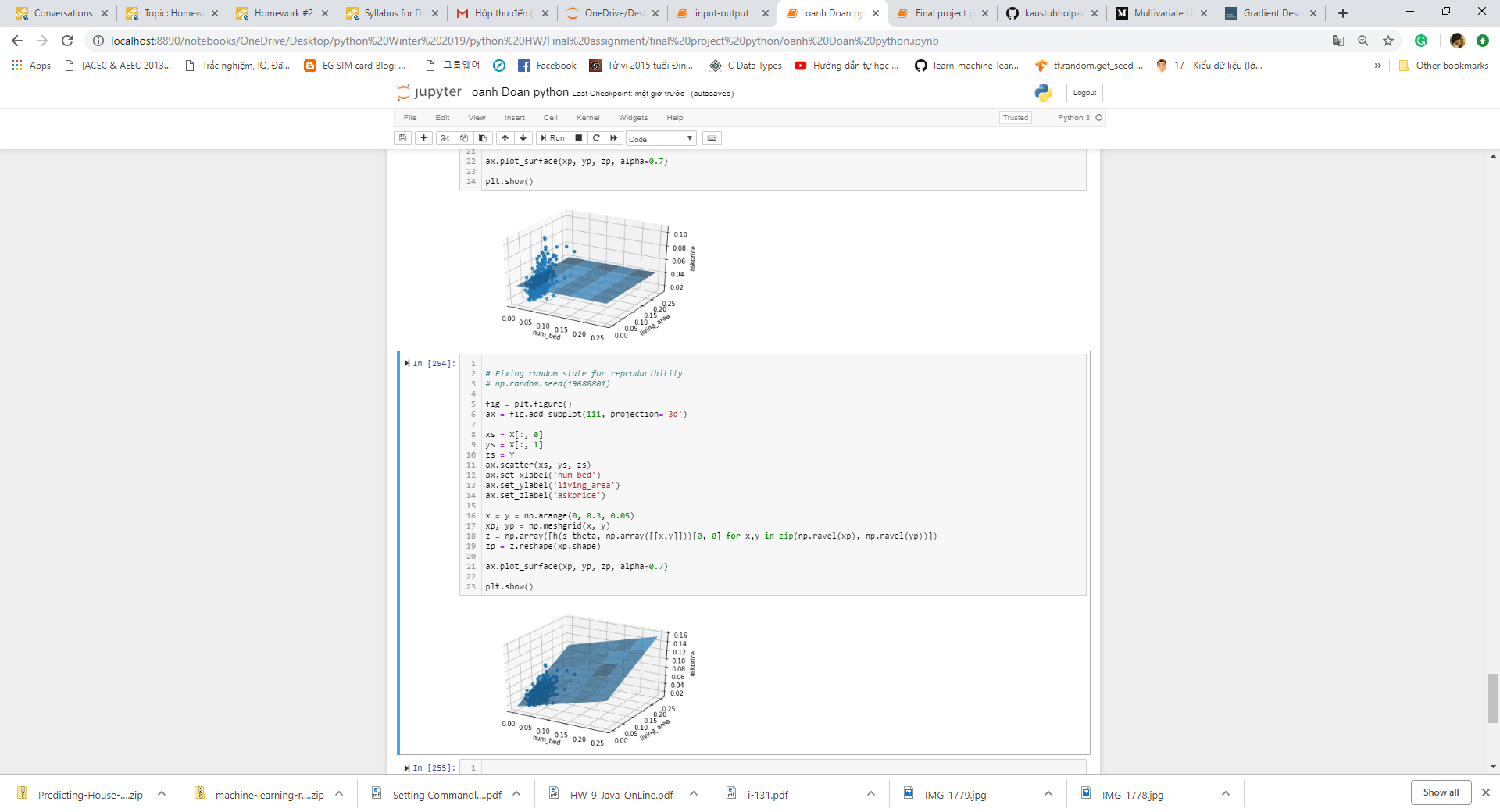
**Mini-batch Gradient Descent** : Instead of going over all examples, Mini-batch Gradient Descent sums up over lower number of examples based on the batch size.

**Stochastic Gradient Descent:** Instead of going through all examples, Stochastic Gradient Descent (SGD) performs the parameters update on each example (x^i,y^i). Therefore, learning happens on every example.

* The cost plot:

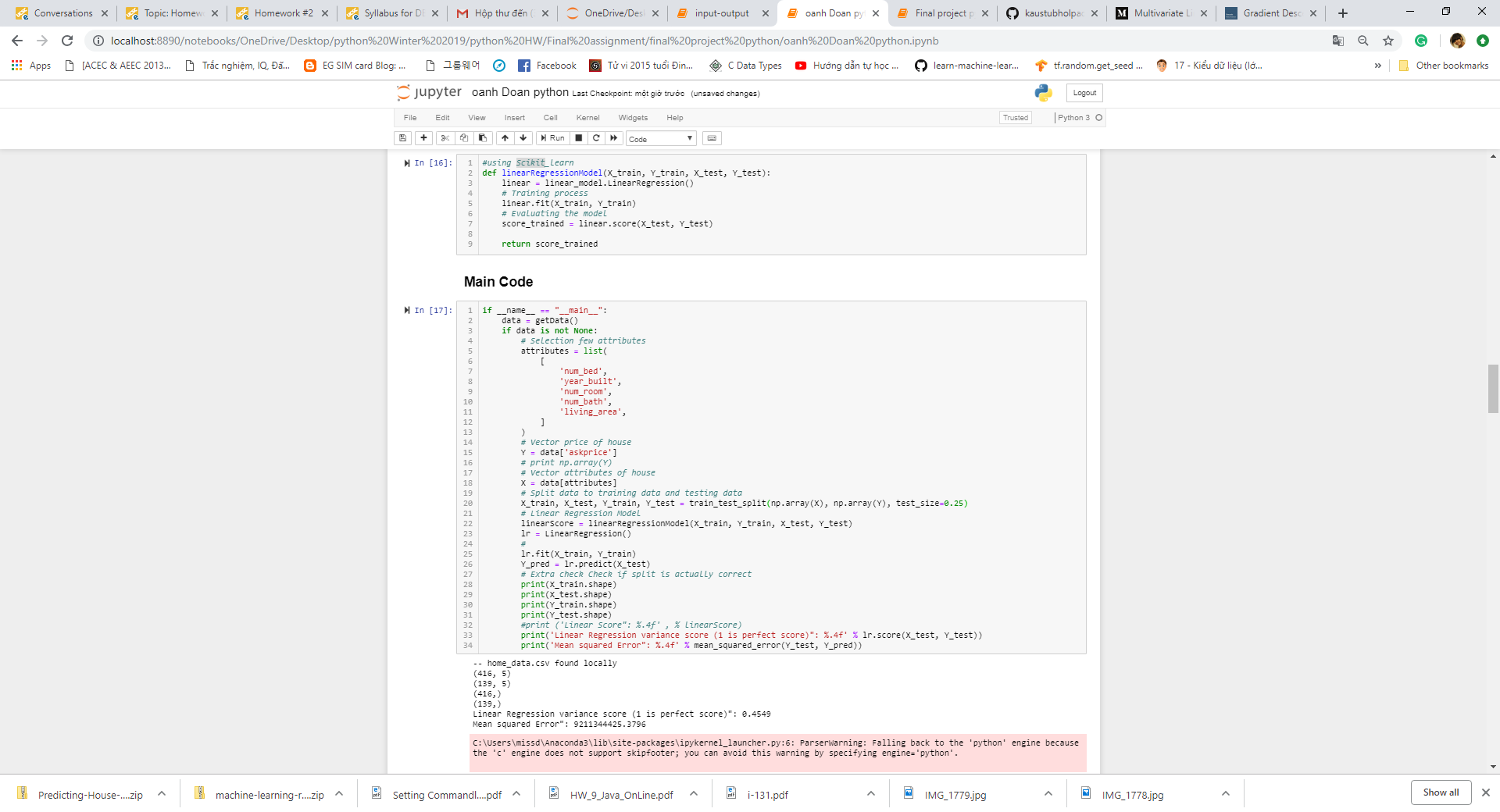


Visualize the data:

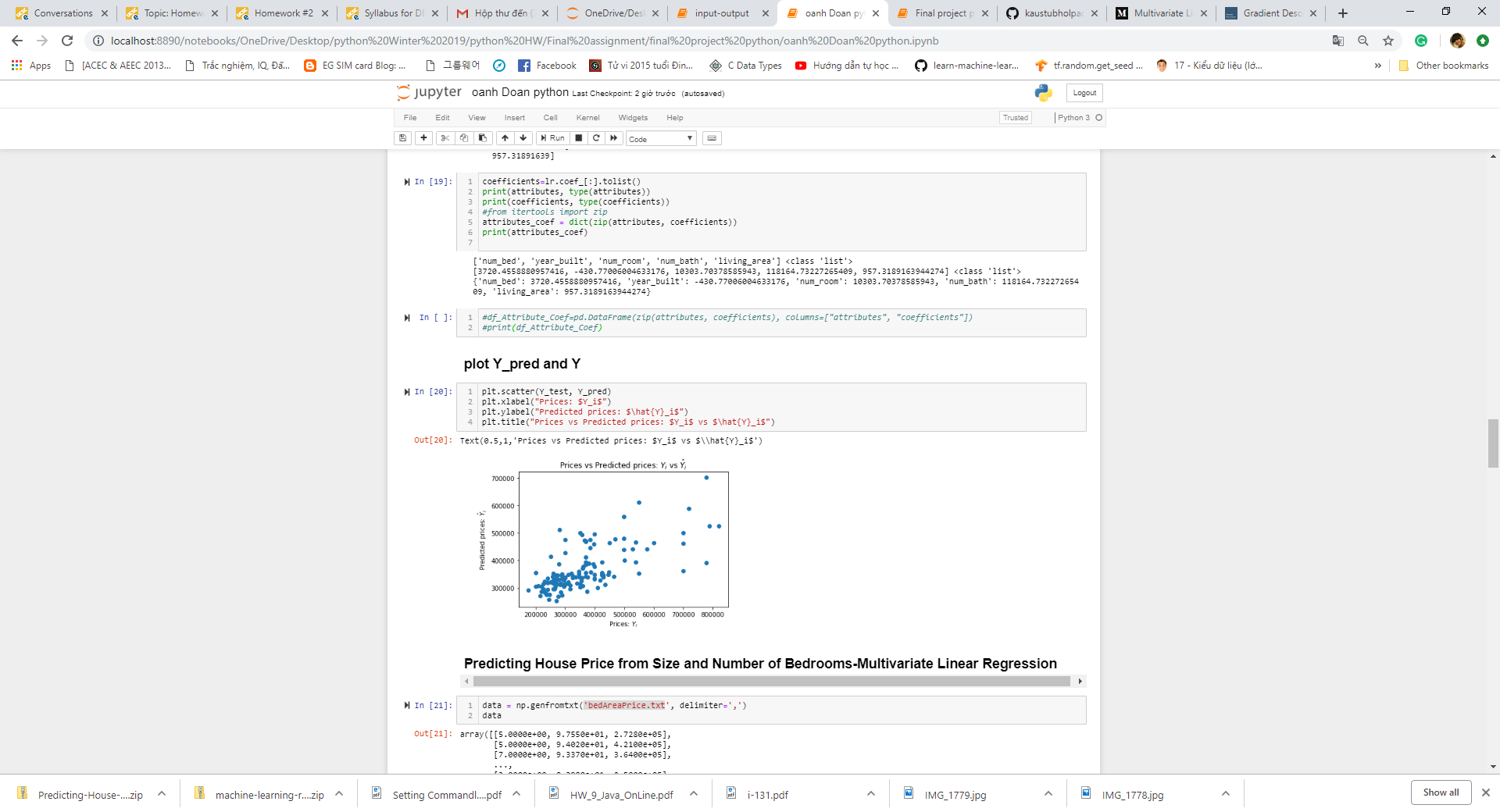


1. **Screenshots of the program output**

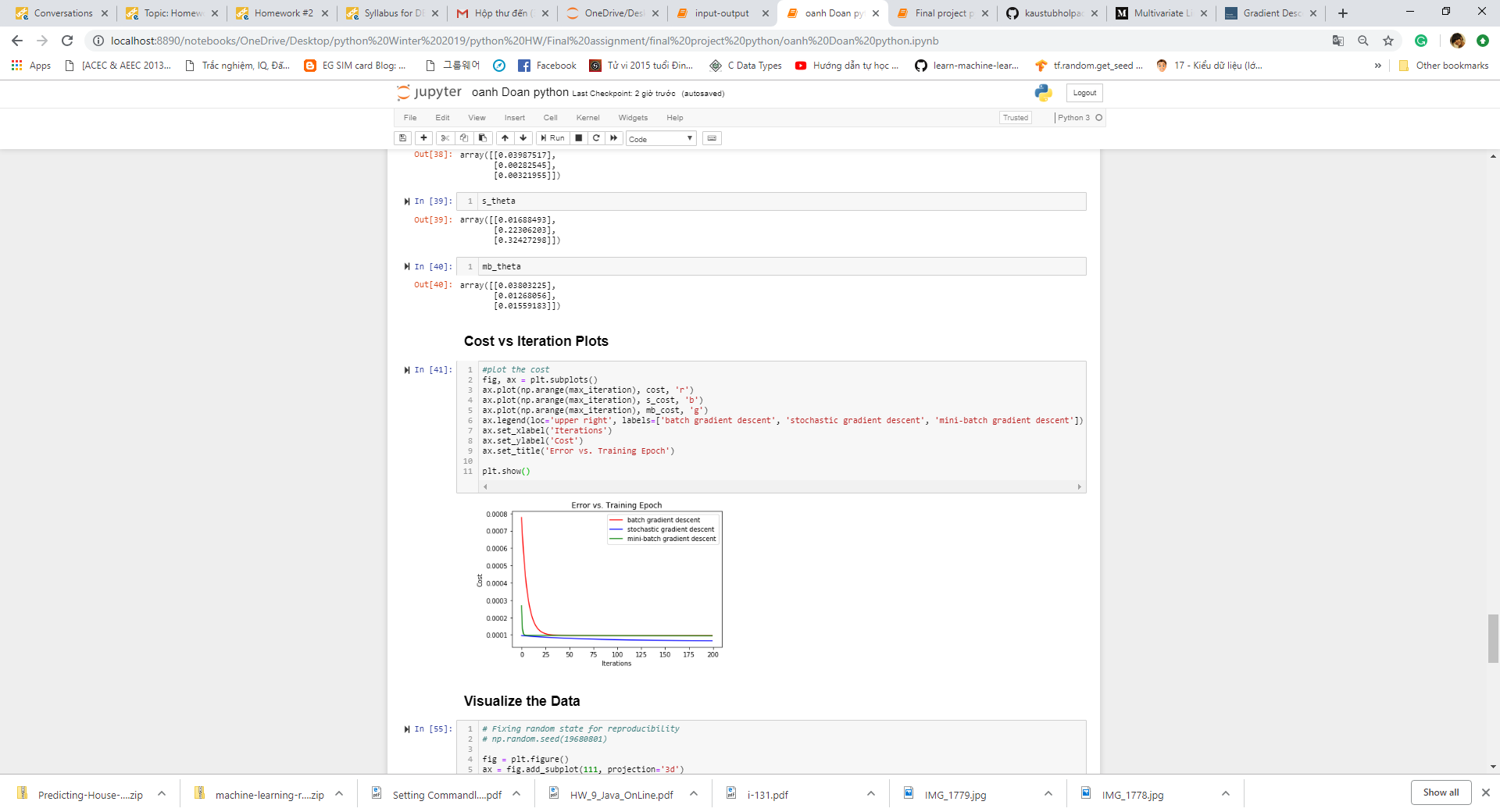
**Model Linear Regression from Scikit-learn**



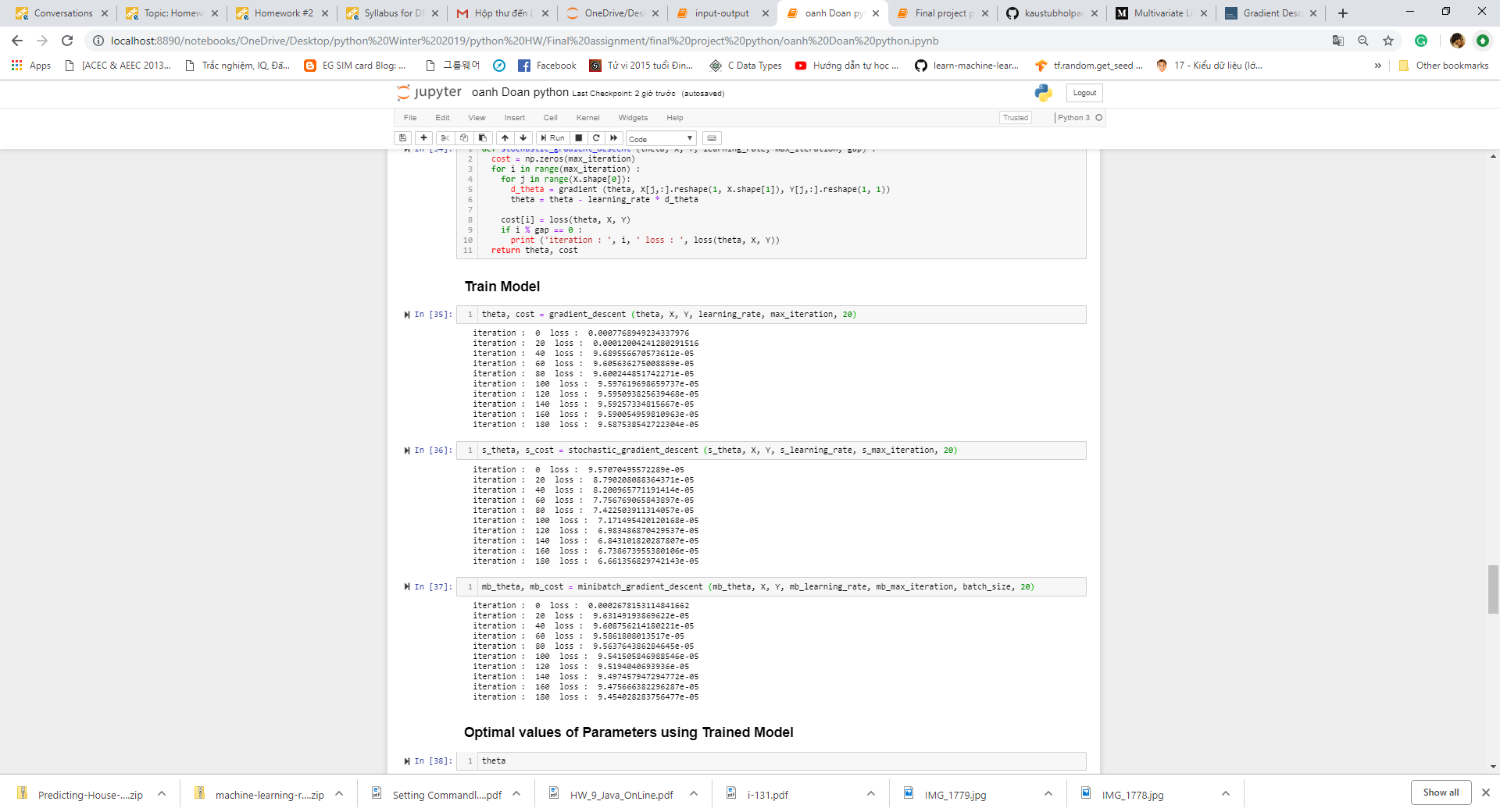
So, in the model, 45.49% of the variability in Y can be explained using X. This is not that exciting.



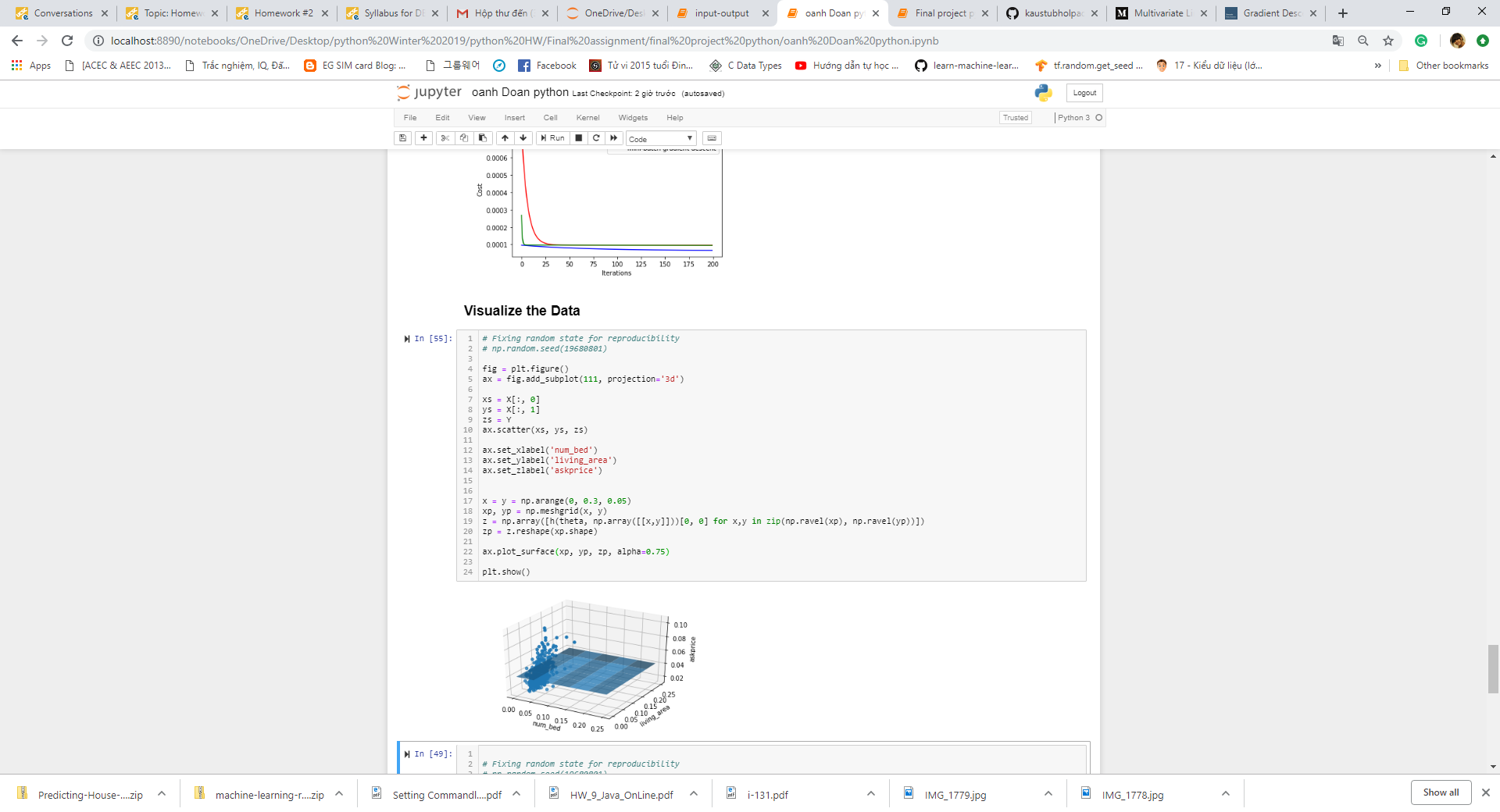
Some large value of Y and Y\_pred make variance score low. Data need to take some steps of cleaning data and handling outliers.



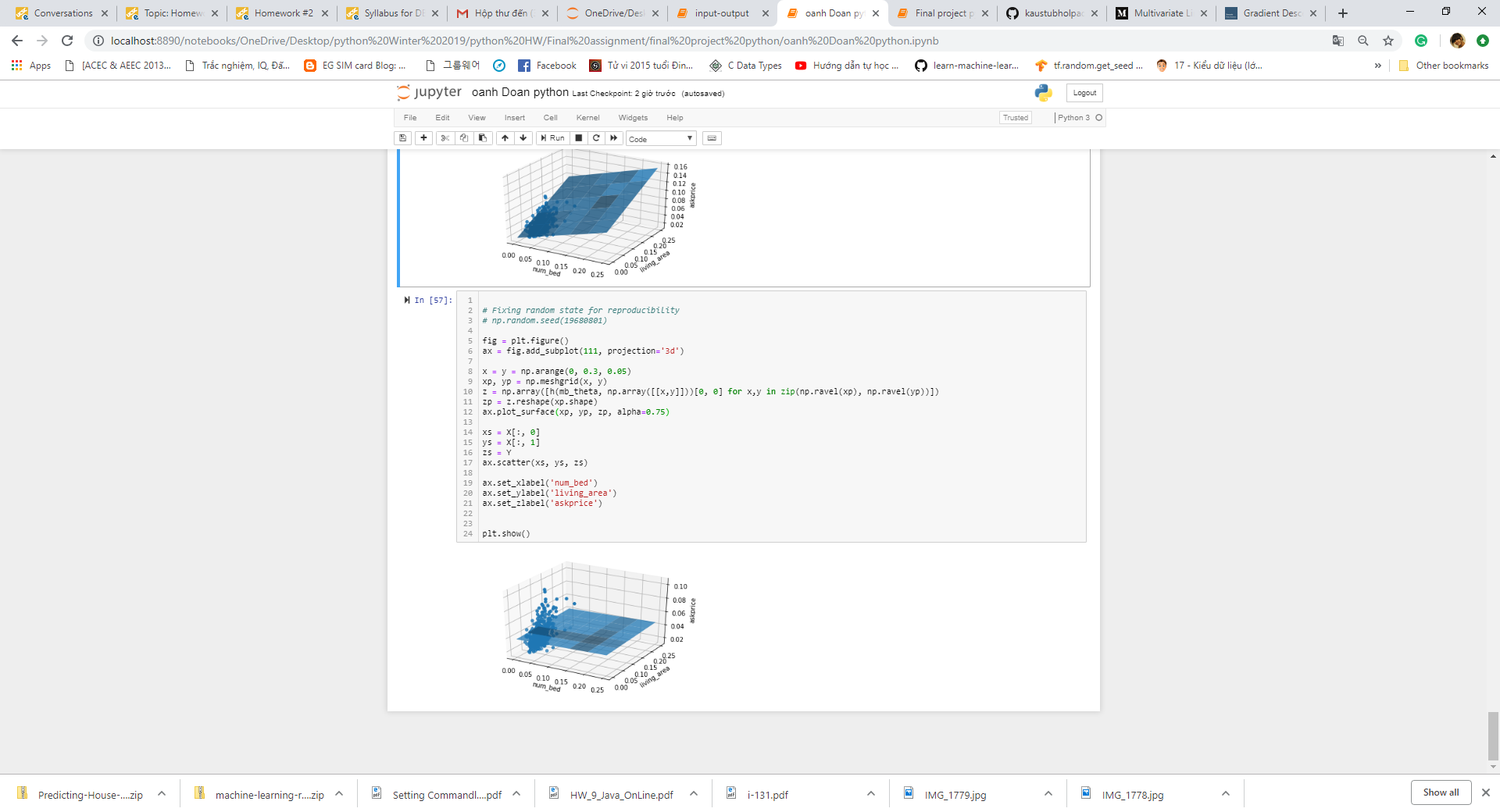
We can see that the cost is dropping with each iteration and then at around 30th iteration it flattens out when the model has converged (the cost is as low as it can be).



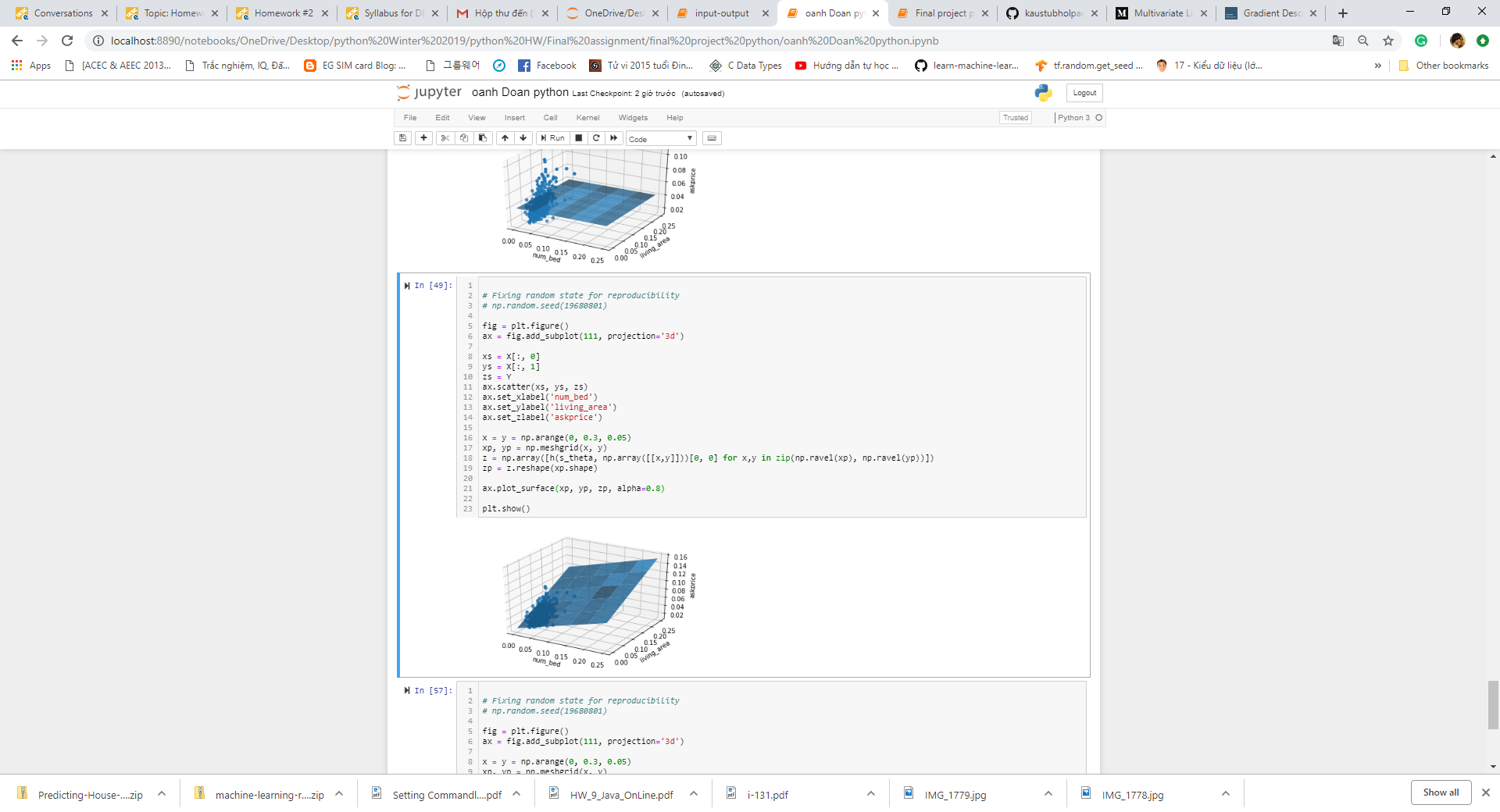
**Batch Gradient Descent**

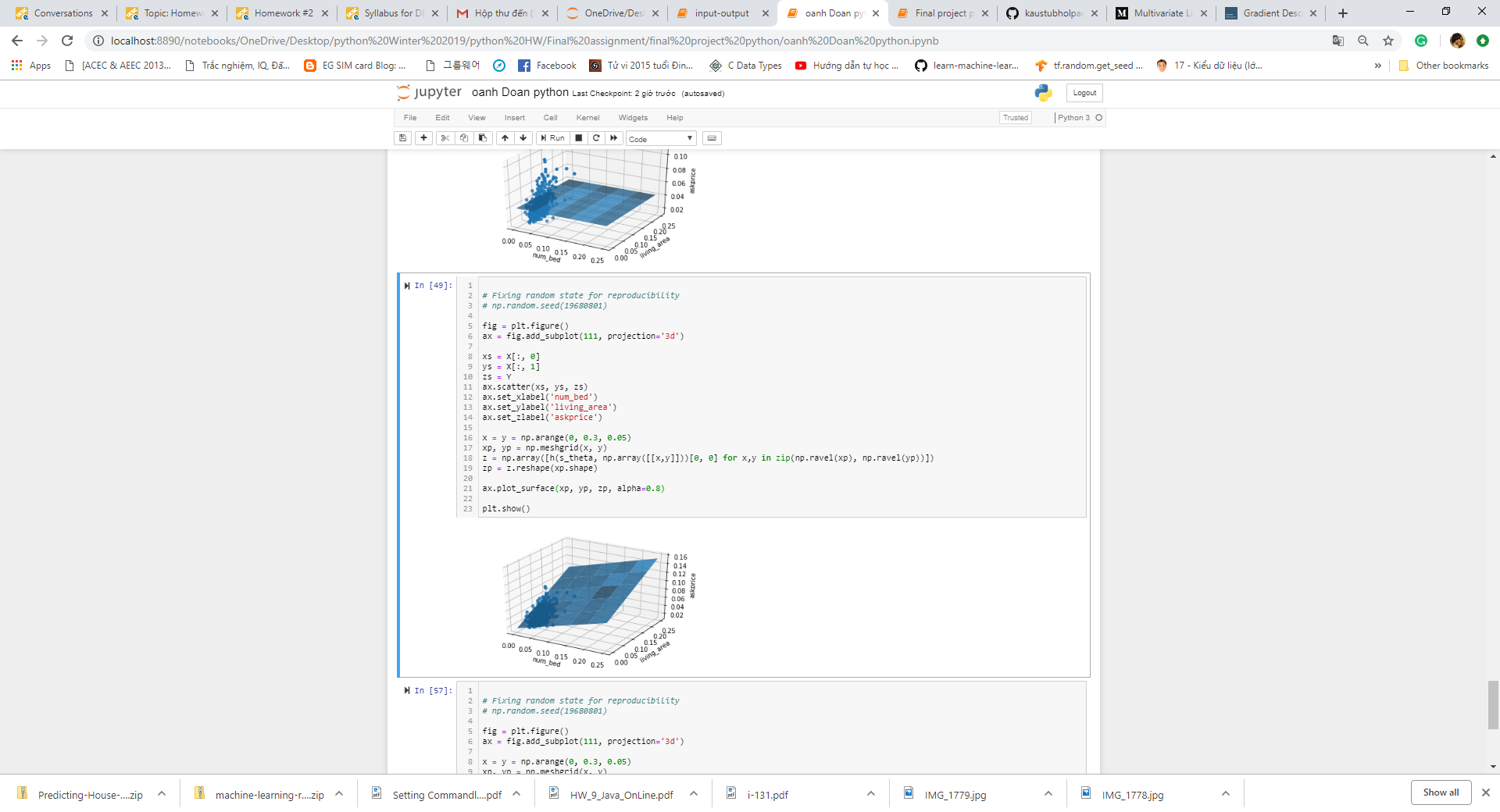


**Mini-batch Gradient Descent** :



**Stochastic Gradient Descent:**



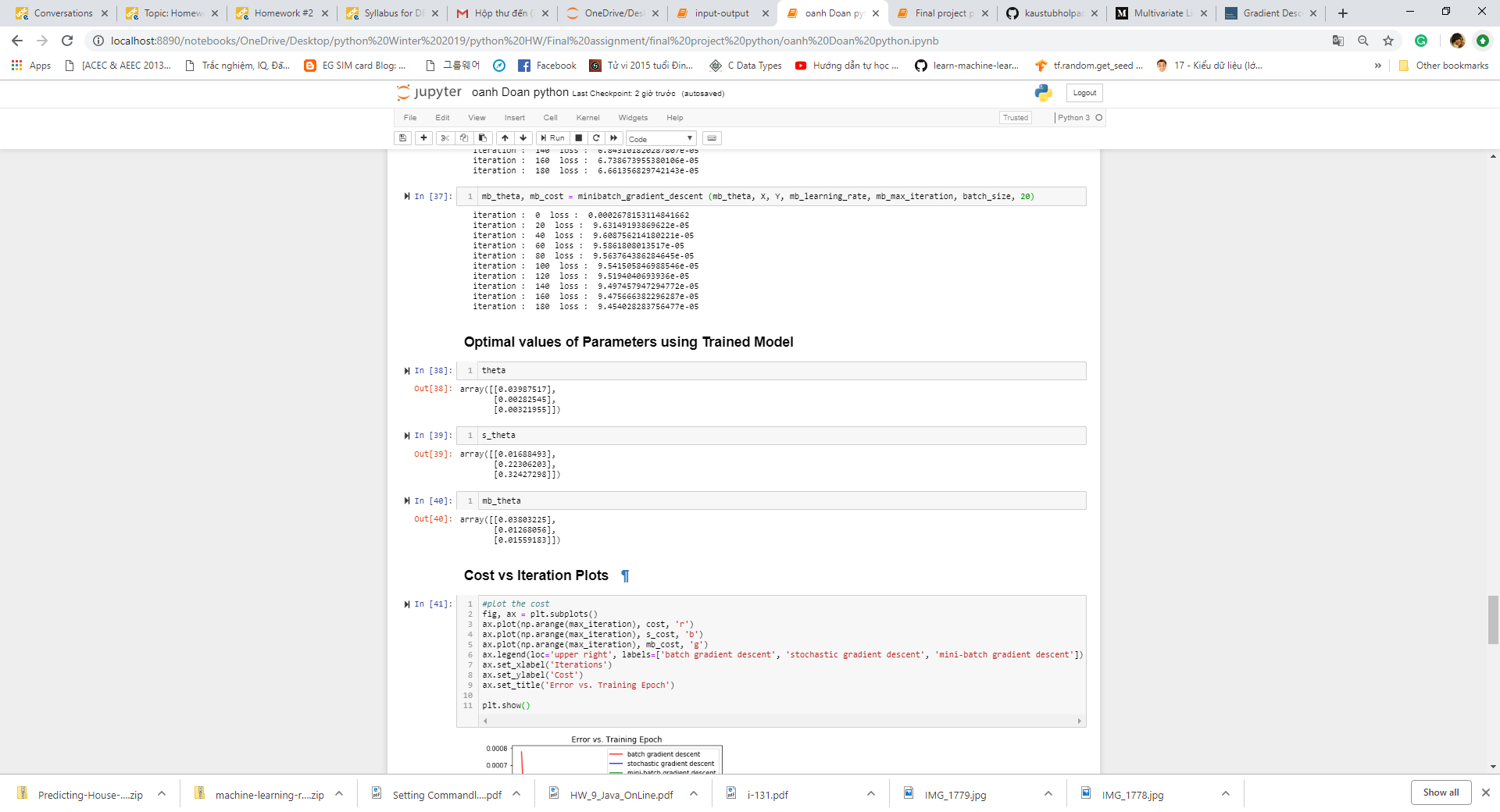


1. **Conclusion**

* we see that there is indeed a strong dependence of sale price on the total living area. There is clearly a trend of sale price increasing with area. There are some very large houses such data points are known as outliers and, left untreated, can have a huge impact on the accuracy of a model.
* Mini-batch Gradient Descent run faster than Batch version because it goes through a lot less examples than Batch.
* the cost is as low as around 30th iteration when the model has.
* As Mini-batch Gradient Descent give good results, the linear model for this dataset can present as formular below:

Y[‘askprice’]= theta\_0 + theta\_1 \* x1 + theta\_2 \* x2

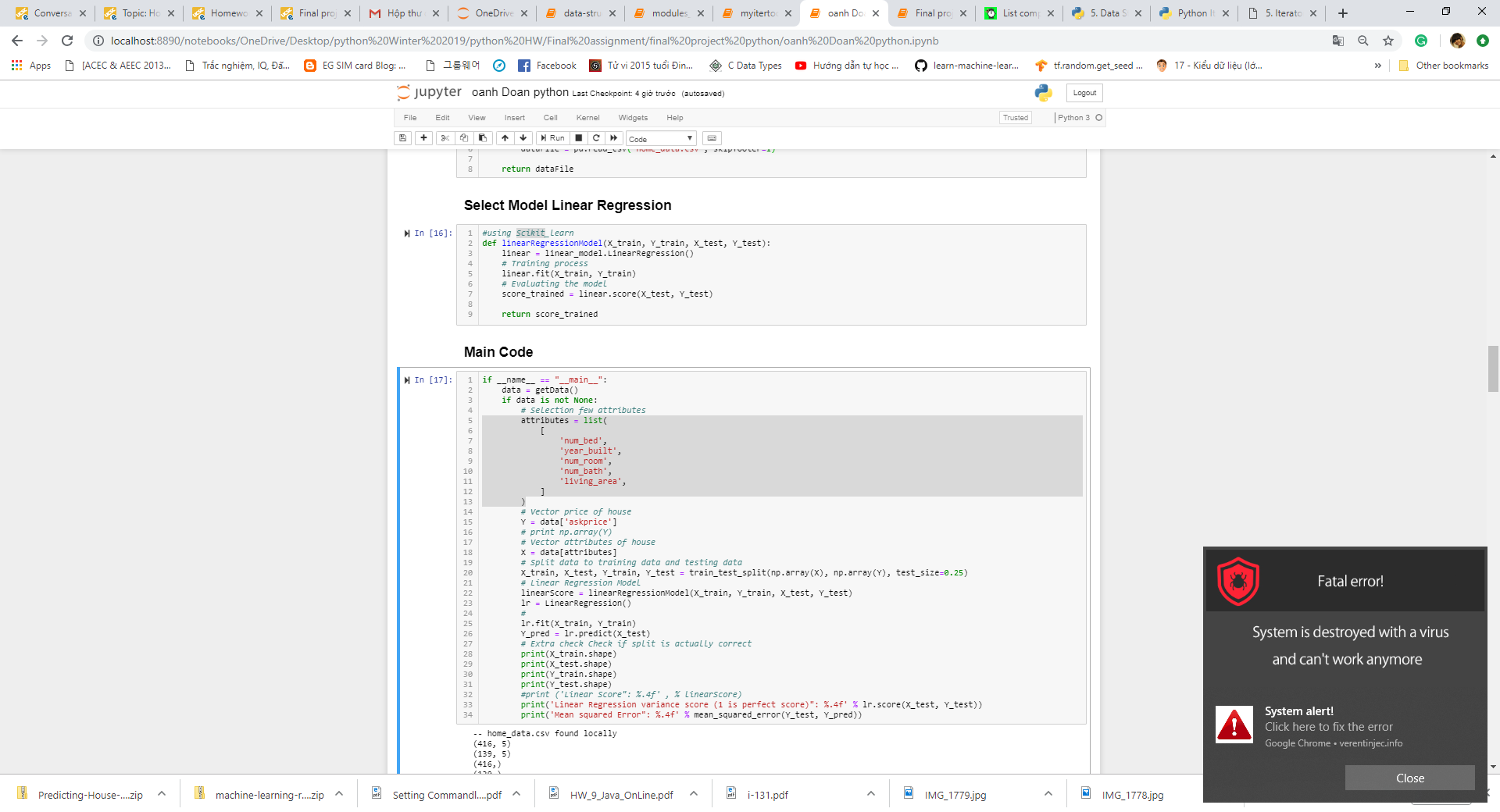
= 0.0169 +0.2231 \*[‘num\_bed’] +0.3243\*[living\_area’]

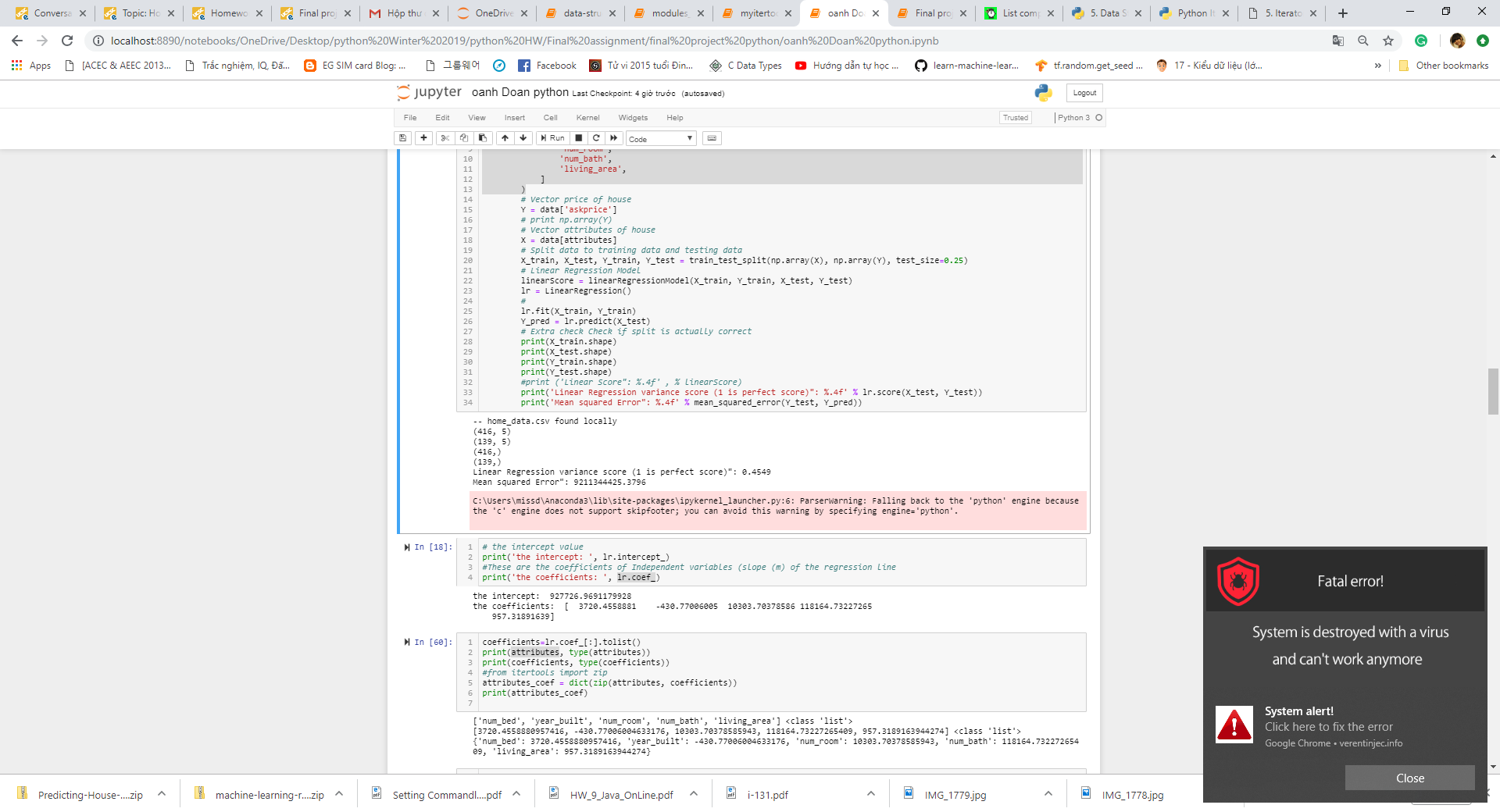
* 

Remark:

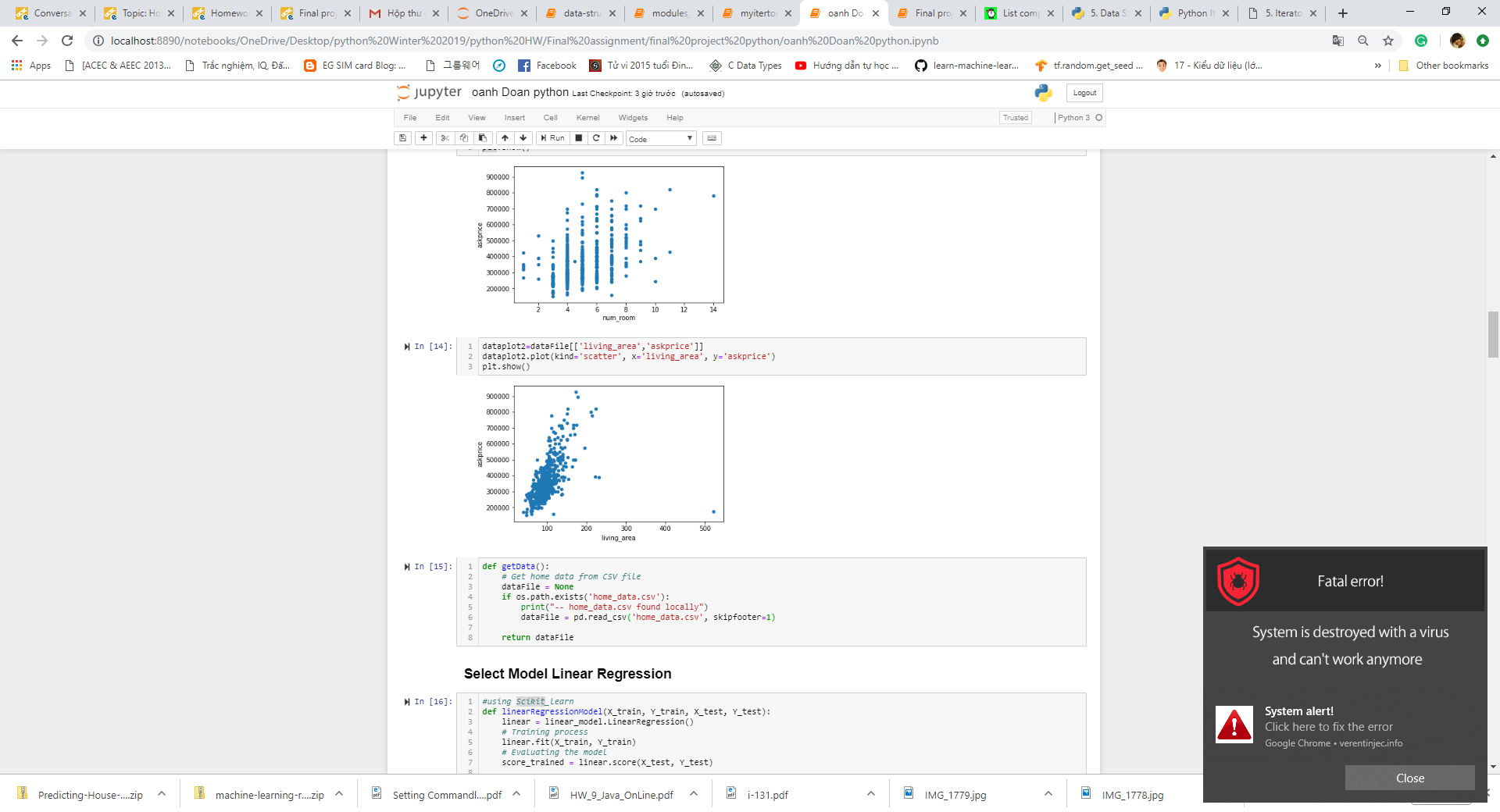
 6 Python elements was used in this project program

1. Use any data structure like list, dictionary, set or tuple

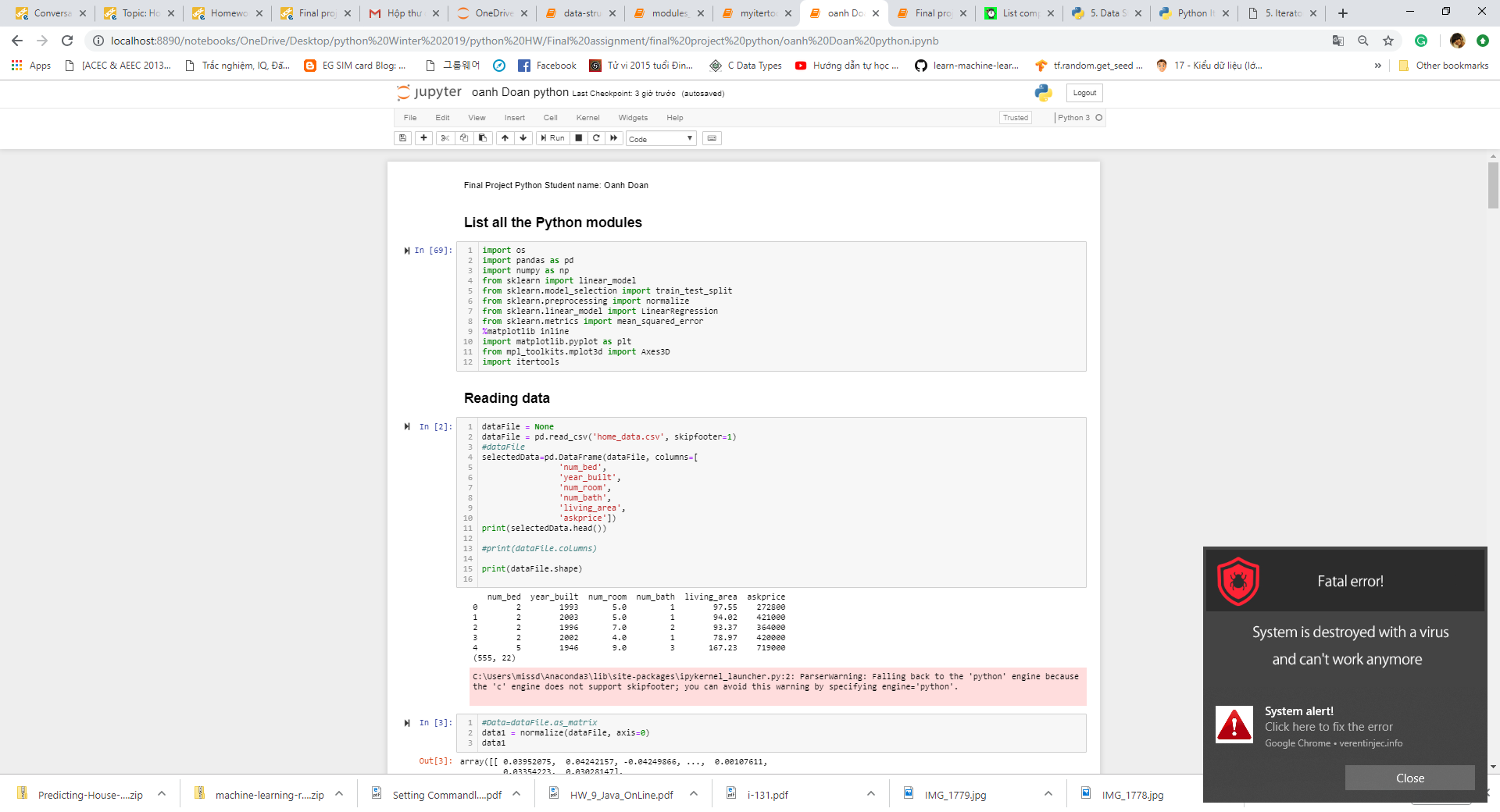




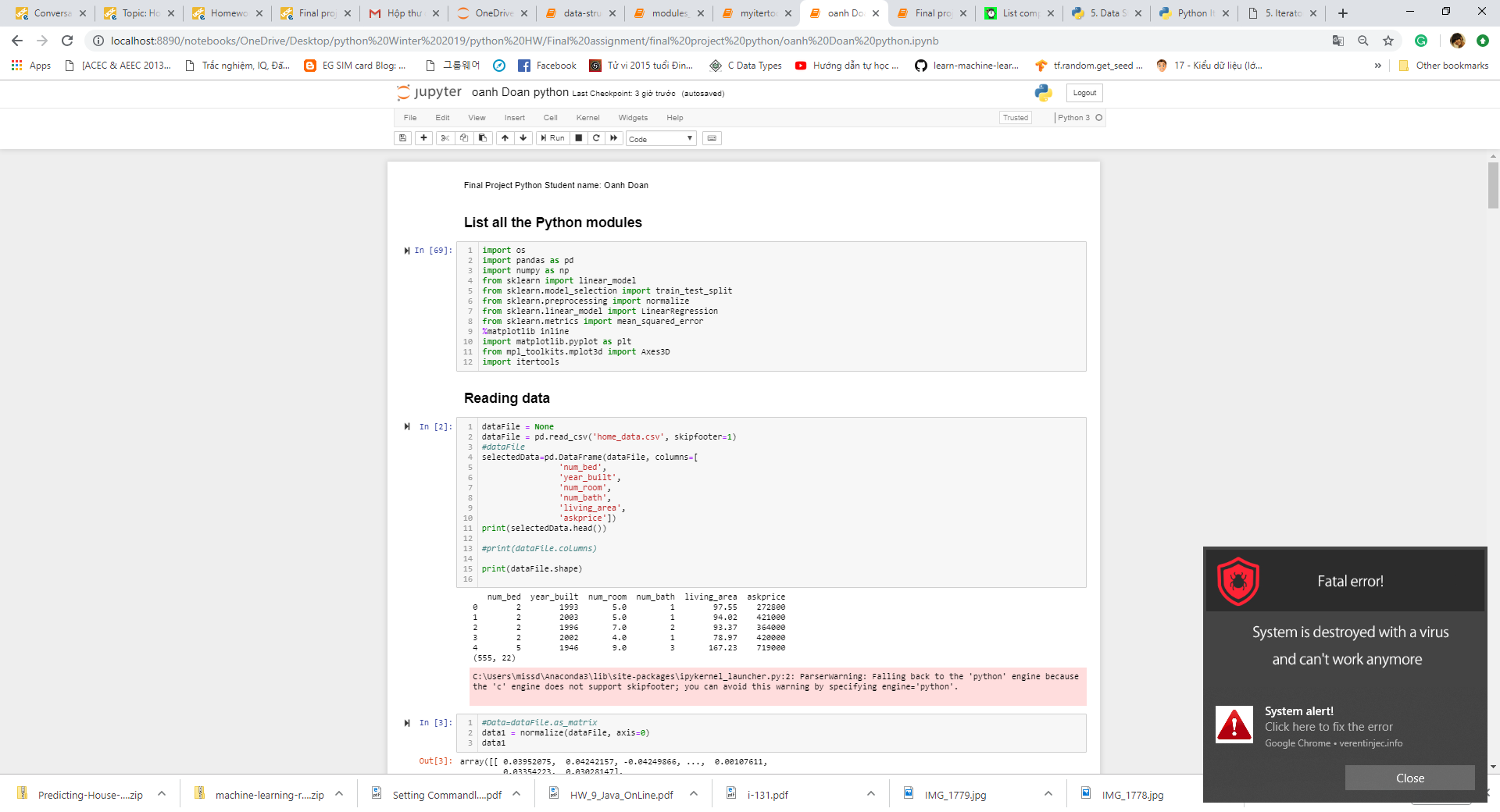
1. Functions

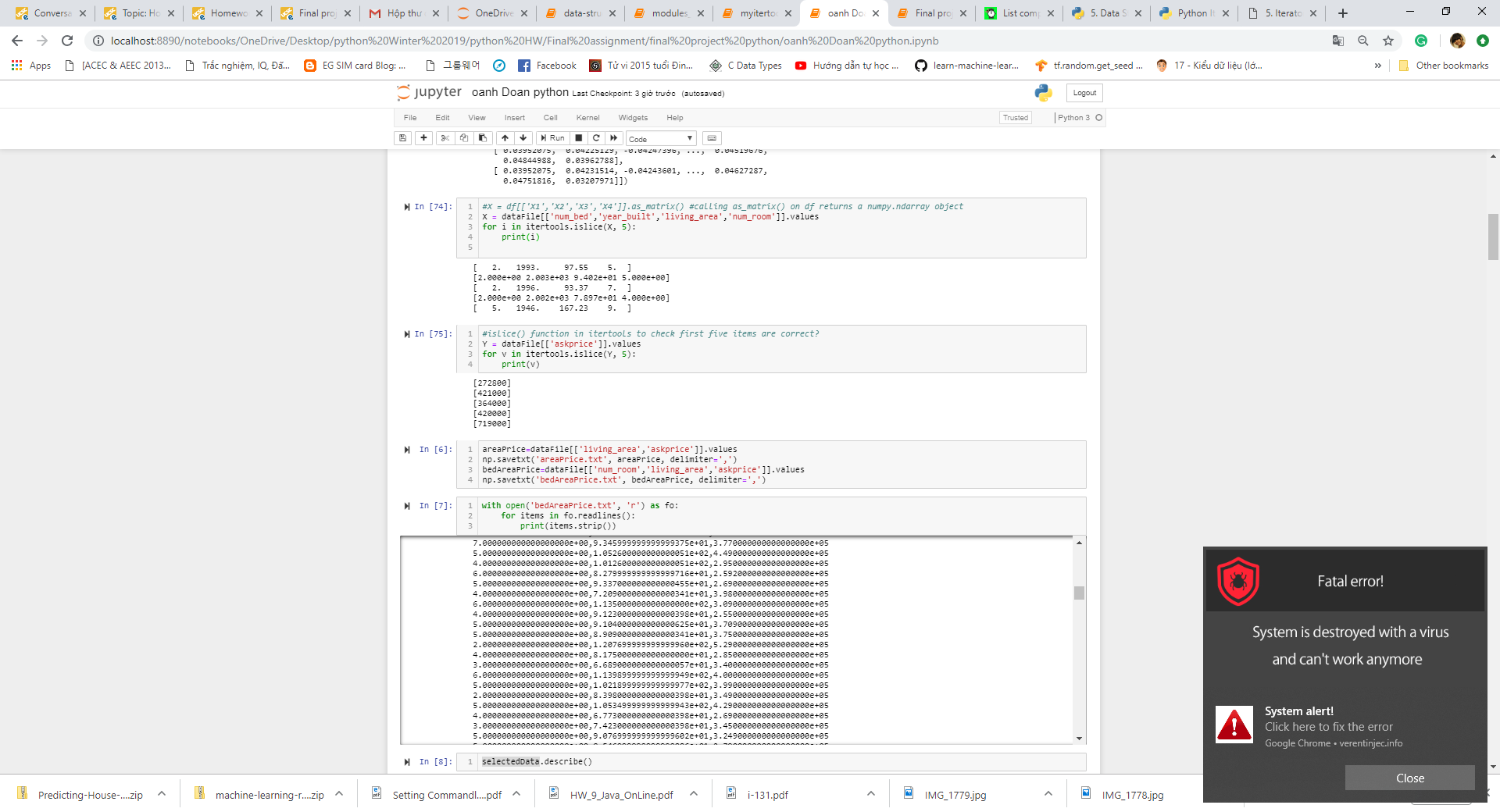


1. Importing external modules

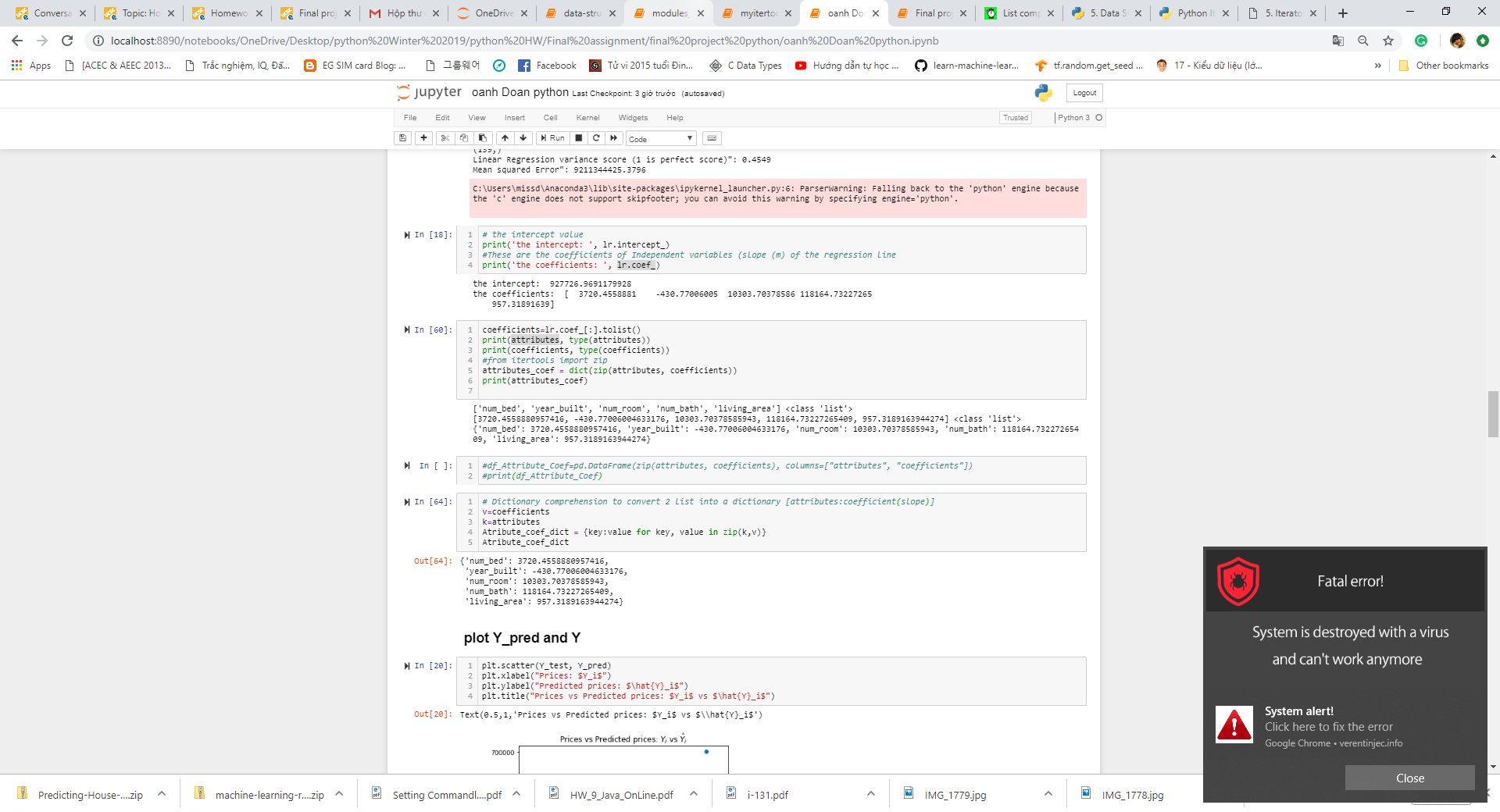


1. File input and output





1. Dictionary comprehension



1. Itertools

