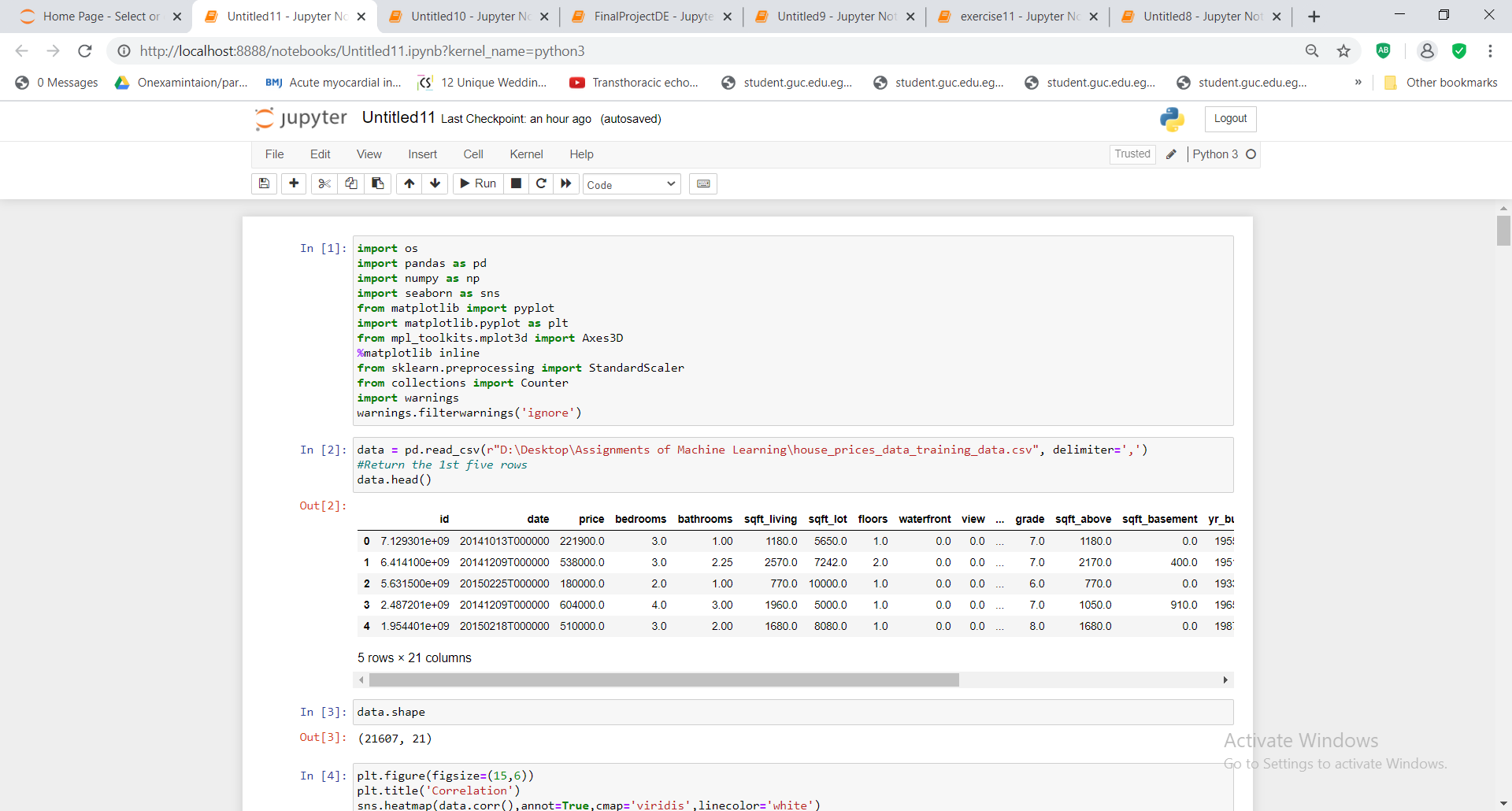
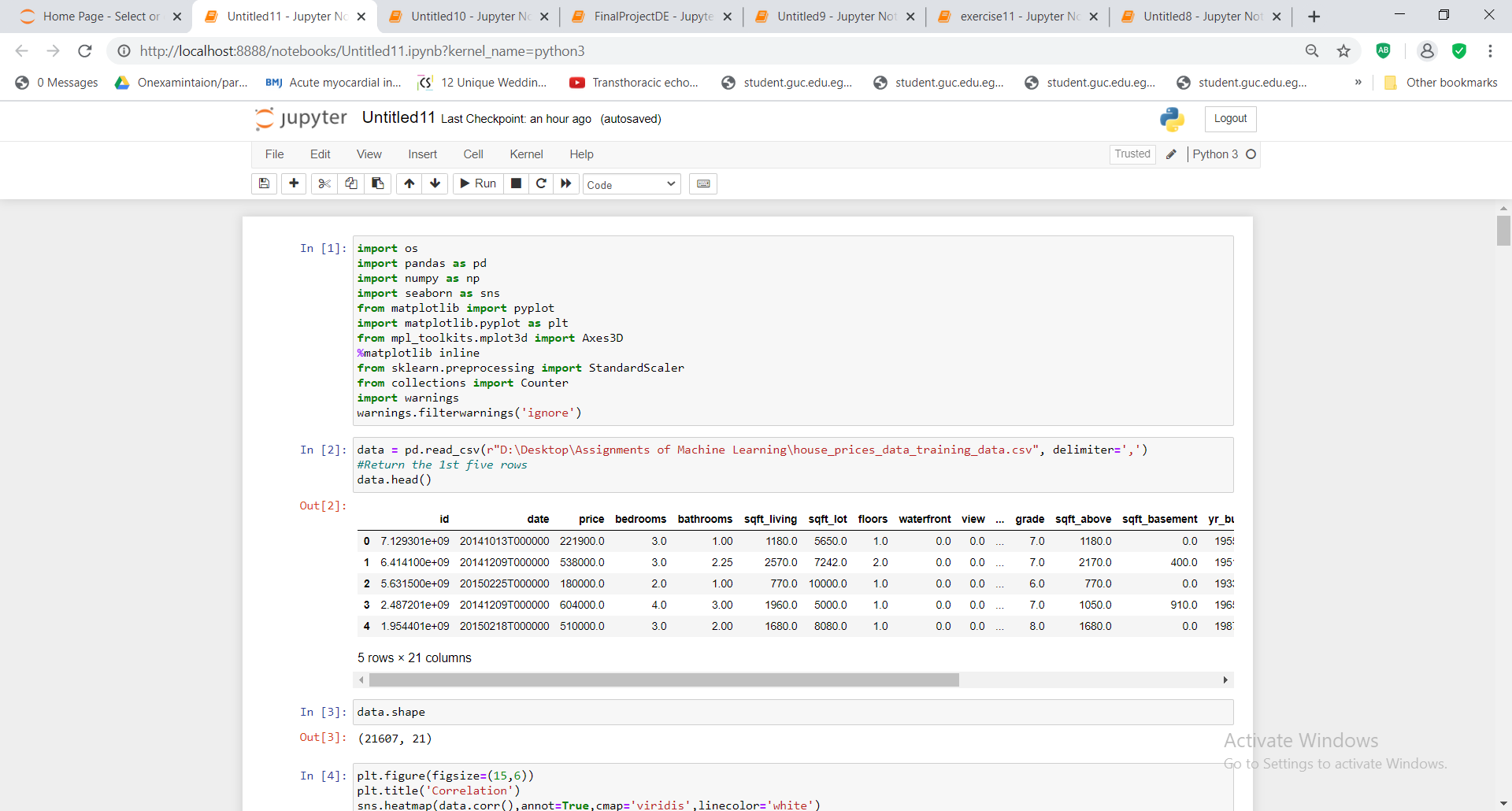
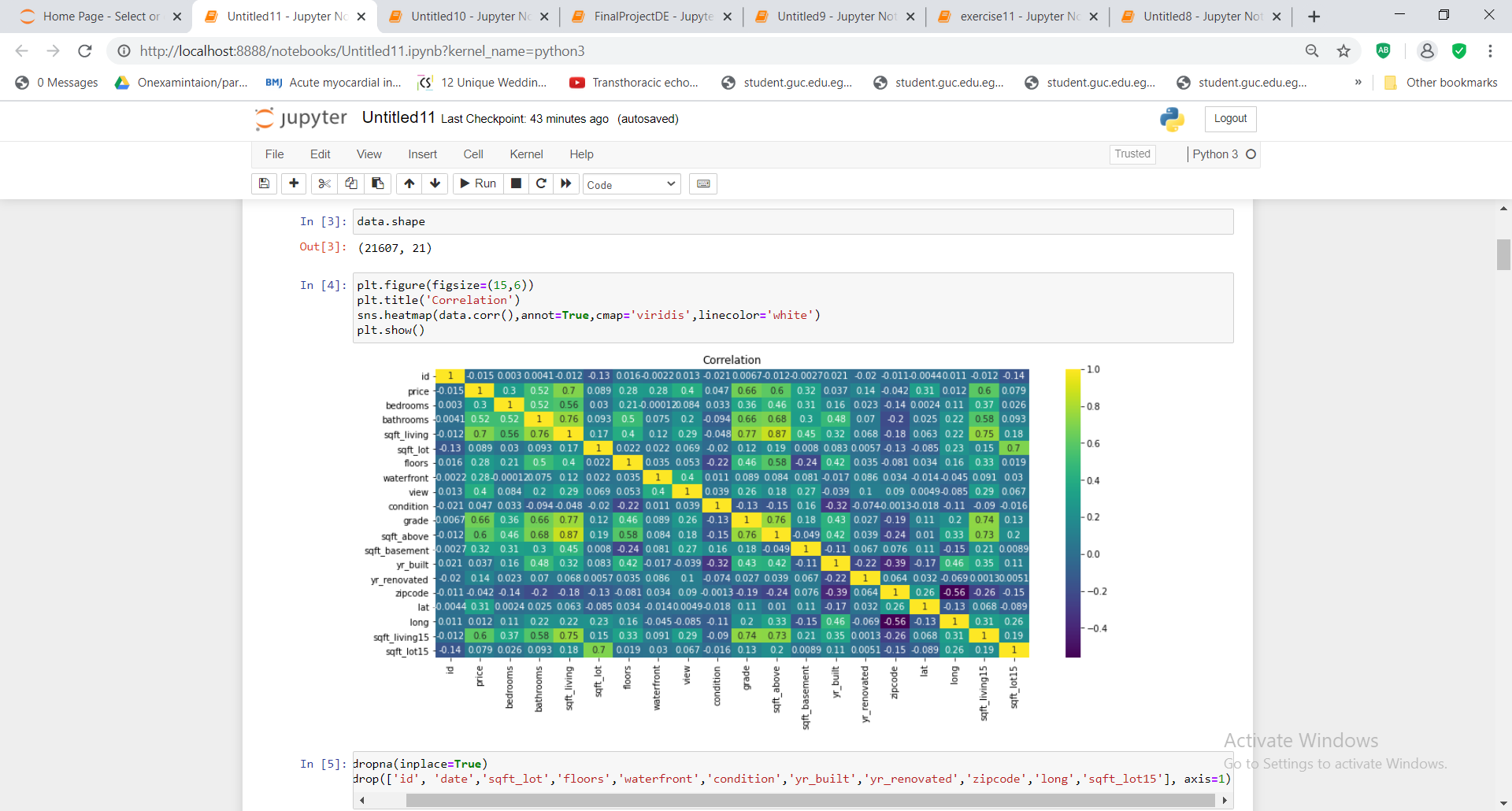
**Assignment 1 Report**

Import the useful libraries to be used in the code. Read the csv file and visualize the first five rows of the dataset using the head command and get the dimensions of the dataset using the shape command in order to know how many rows and columns.





Remember and use from the data engineering course, the heat map. Heat map helps to find correlation between the target variable (price) and the features affecting the target variable (other variables). The positive correlation ranges from ]0,1[ , the higher the value gets the more positively correlated the variables are. (Zero and one are not included).

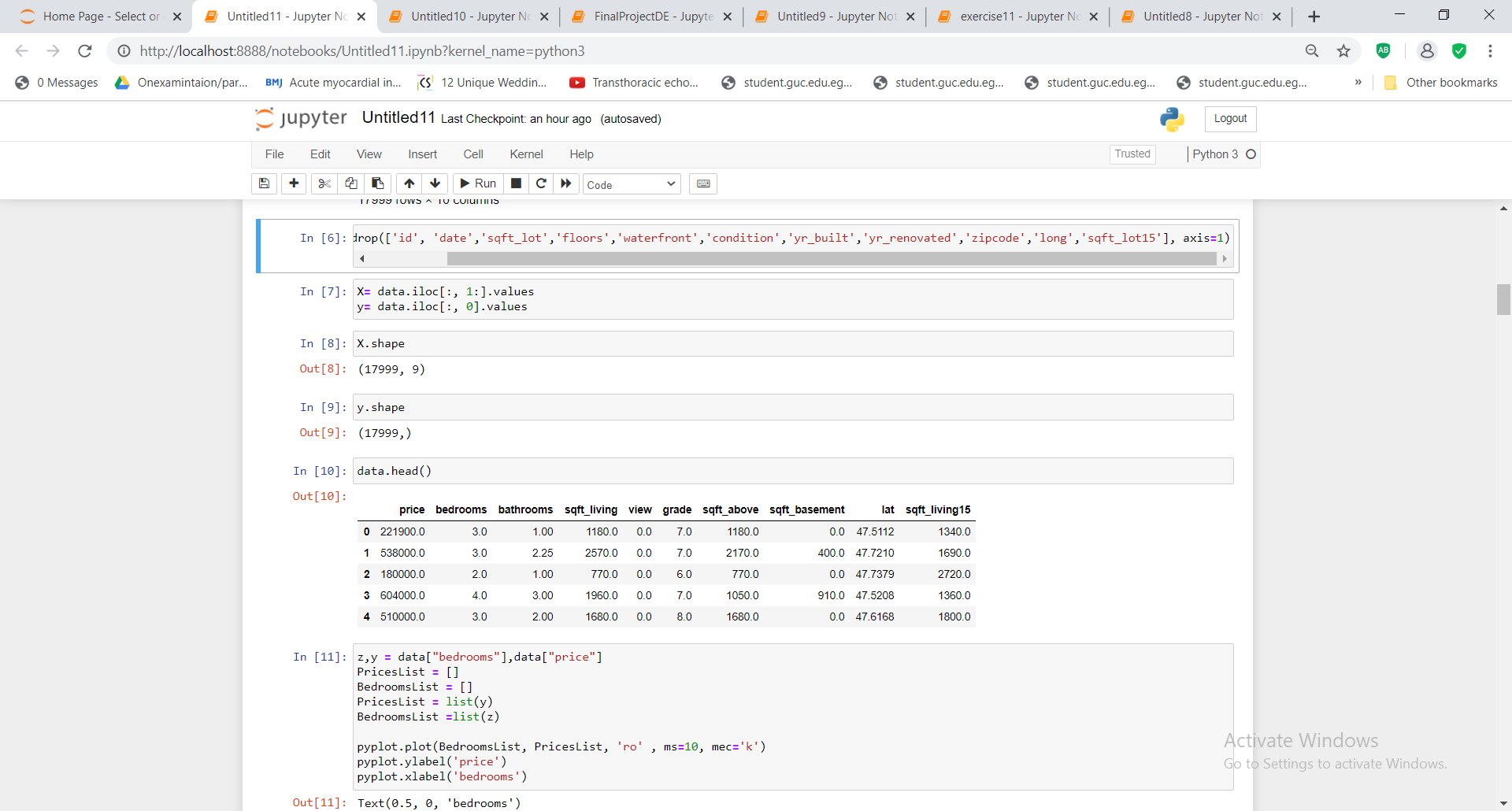


Drop the empty rows which contain no values.

After Putting a threshold level of 0.3 in the correlation produced by the heatmap, drop the columns of variables that their correlation values are less than 0.3. Adjust the dataset to be the data after removing the unnecessary variables in order to decrease the complexity.

data=data.drop(['id','date','sqft\_lot','floors','waterfront','condition','yr\_built','yr\_renovated','zipcode','long','sqft\_lot15'], axis=1)

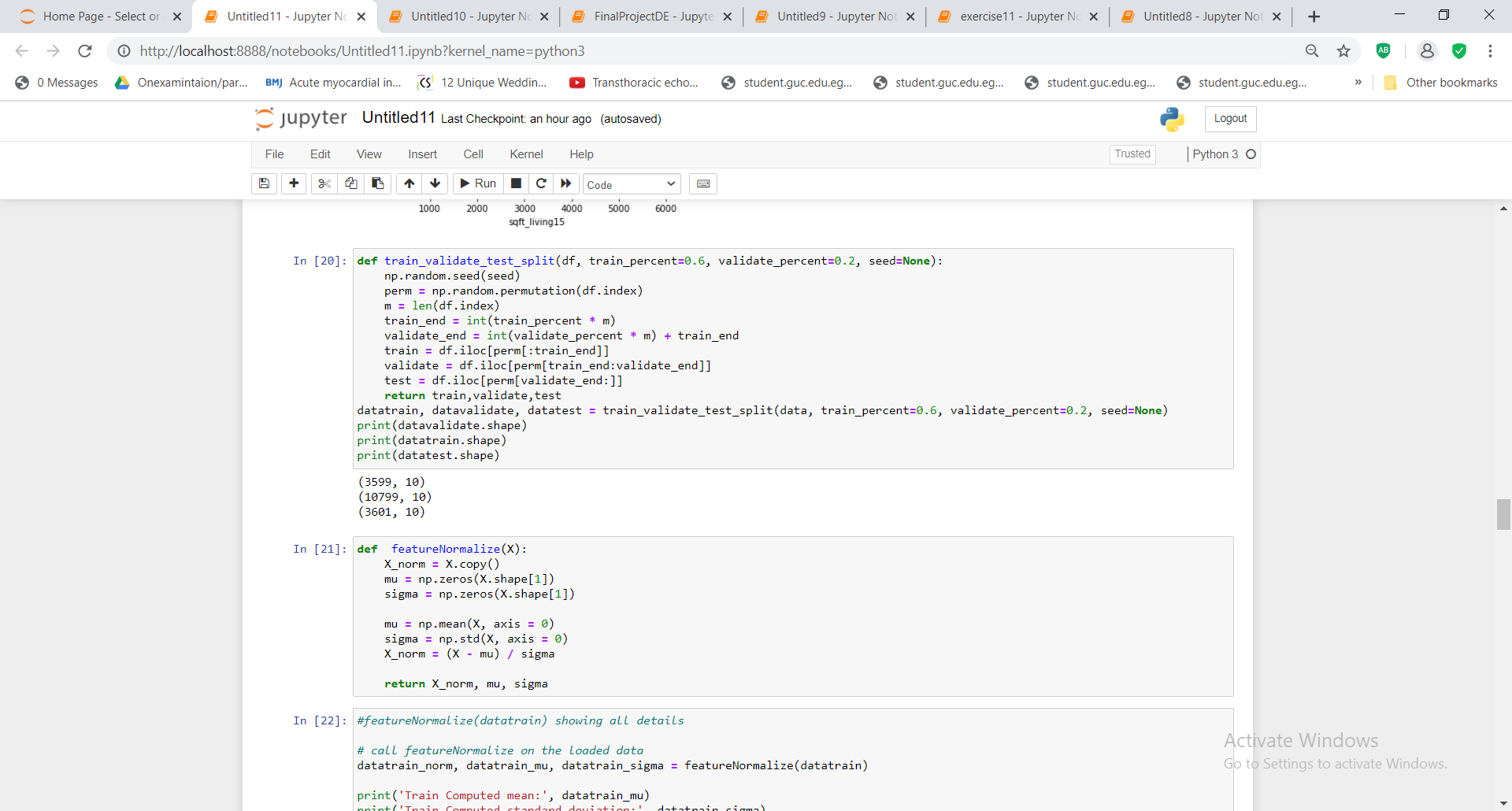
Let X be the features in the adjusted dataset that affect the target variable. Let Y be the target variable (Price) column and get the dimensions of X and the dimension of y. then, using the head command review the first five rows of the adjusted data set.



Plot each variable of X against Price(Y) using pyplot. (9 figures) shwn in the code.

In the model selection, split the data into a Training Set(60%), a Cross Validation (CV) Set (20%) and a Test Set (20%)

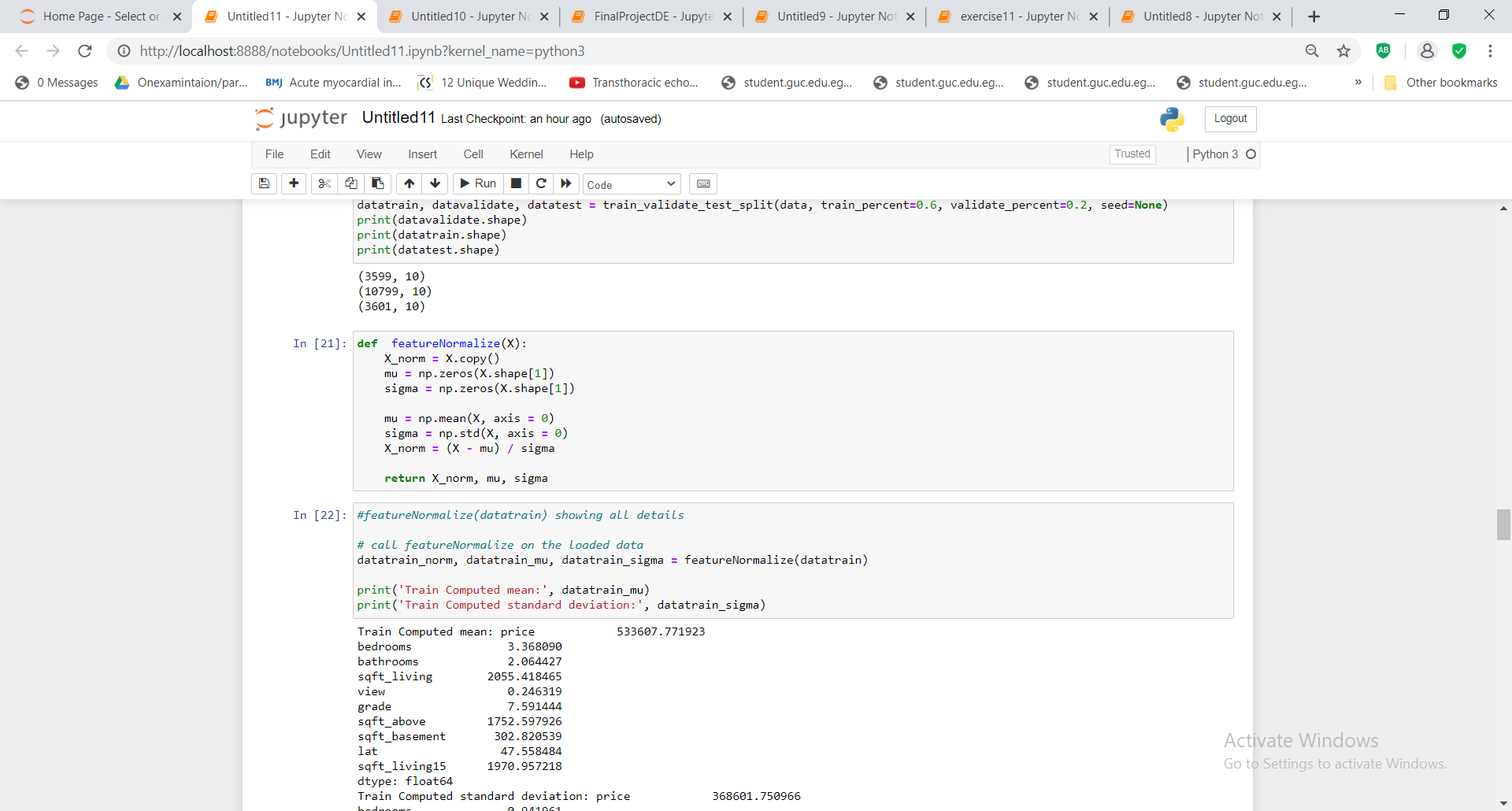
Check their dimensions (datatrain datavalidate datatest)



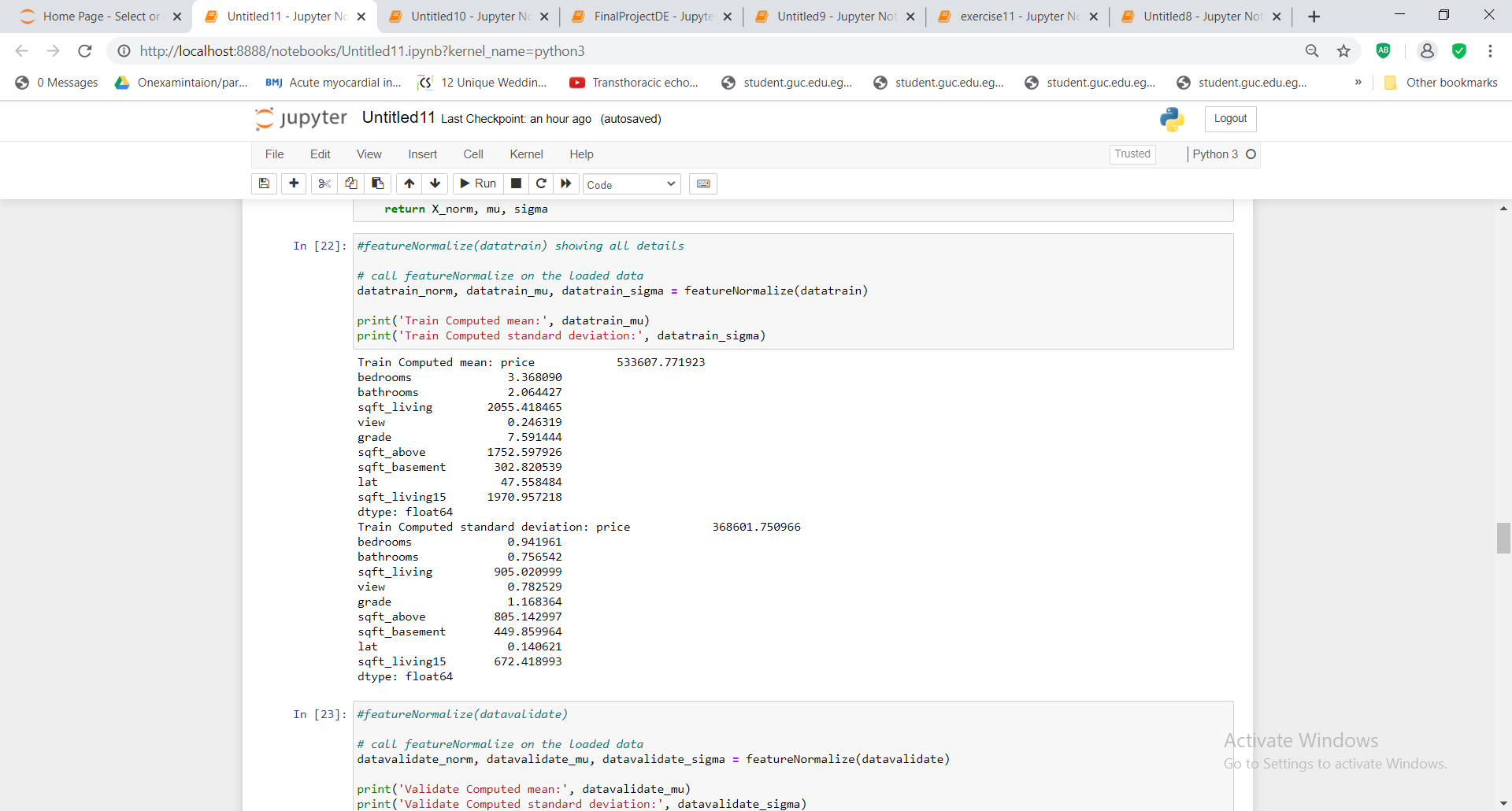
###### FEATURE NORMALIZATION #########

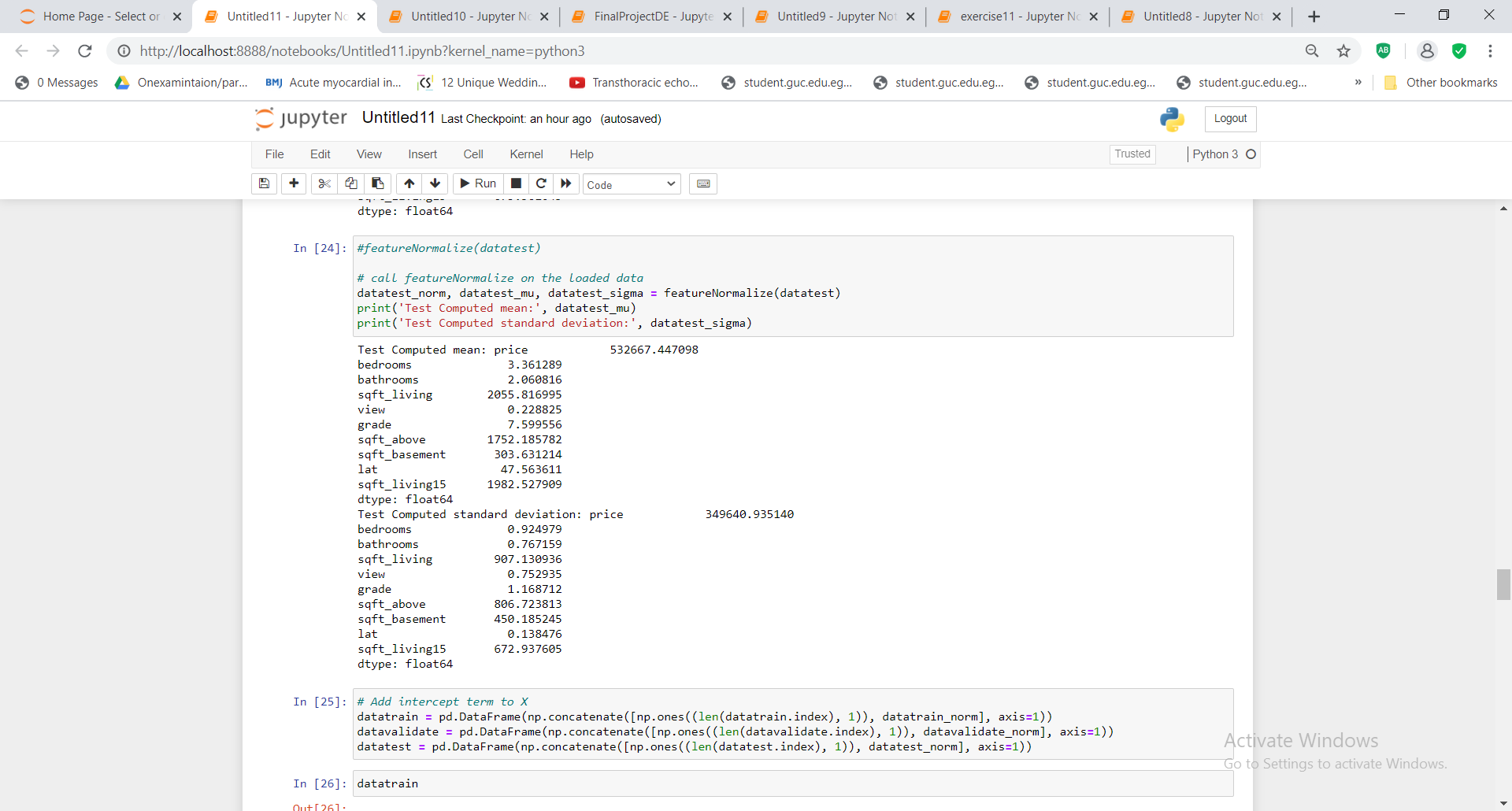
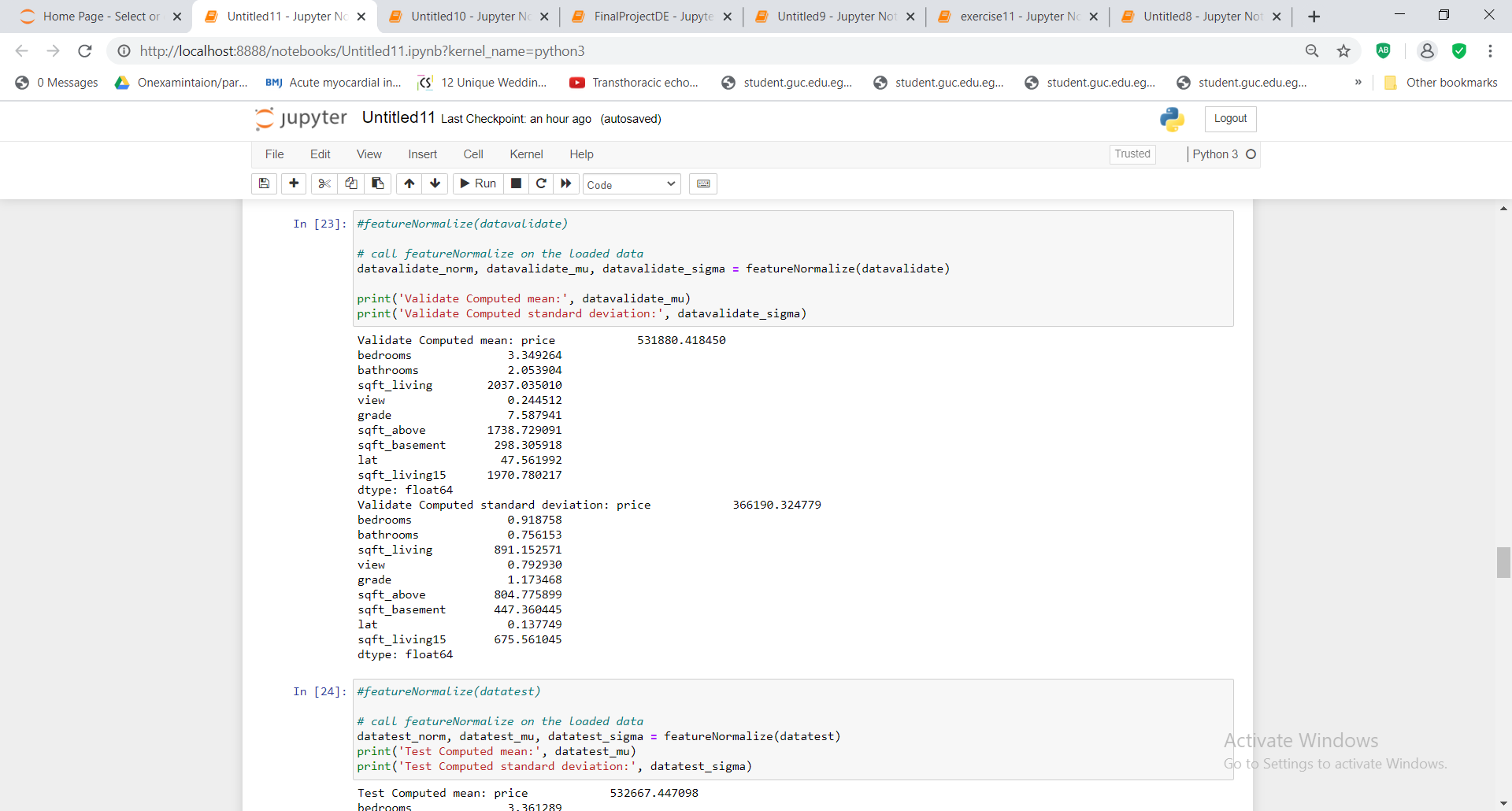
In feature normalization, normalize the features in X. return a normalized version of X where the mean value of each feature is 0 and the standard deviation is 1.

For each feature dimension, compute the mean of the feature and subtract it from the dataset, storing the mean value in mu. Next, compute the standard deviation of each feature and divide each feature by its standard deviation, storing the standard deviation in sigma.



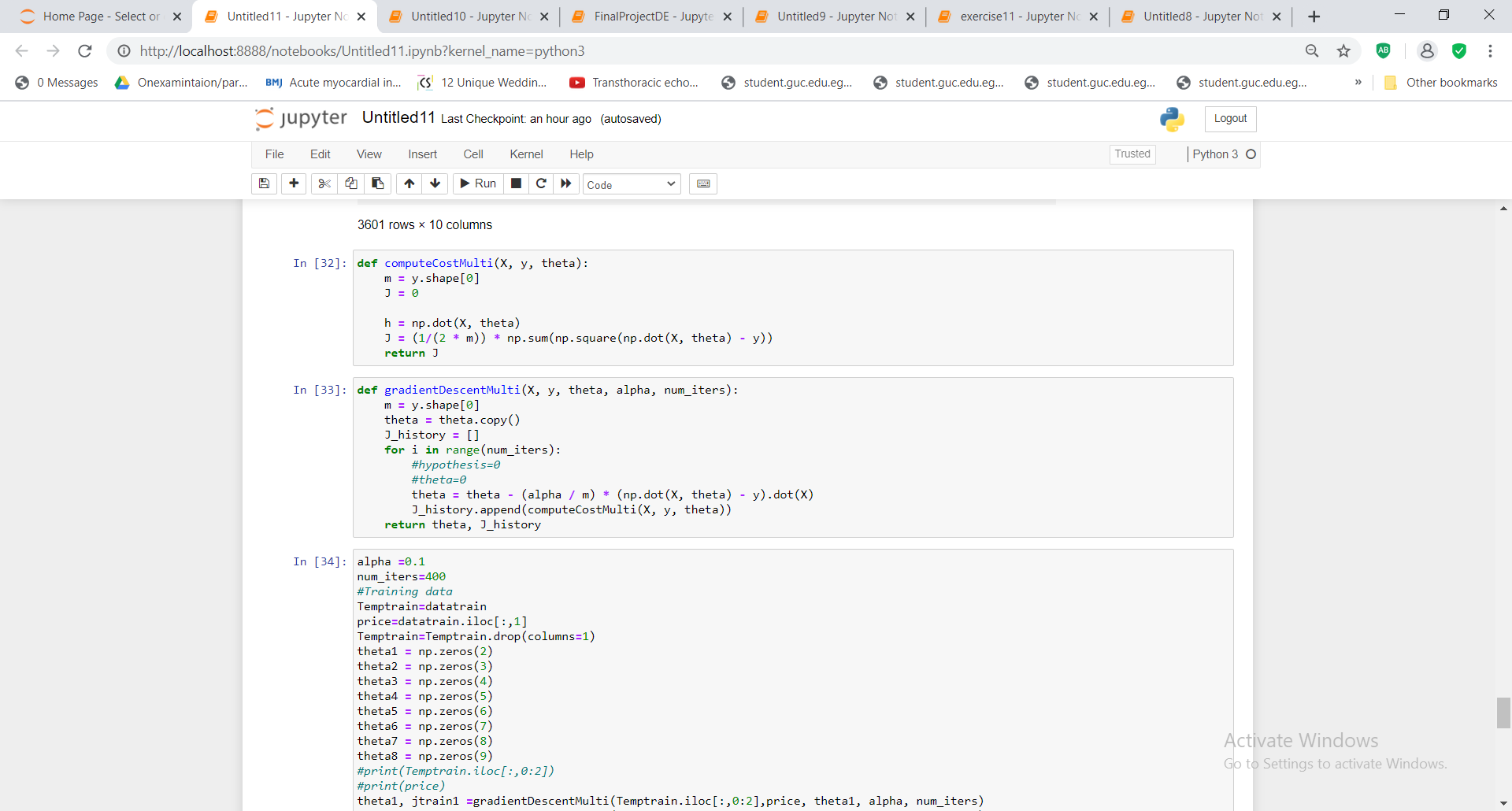
Perform the call of the featurenormalize function sepearetly on the datatrain, datavalidate and datatest and print the computed mean and standard deviation for each of the three splitted data. After calling the feature normalize function, add the intercept term (concatenate) to the datatrain\_norm, datavalidate\_norm, datatest\_norm.





###### COMPUTE COST #########

Compute cost for linear regression with multiple variables. Computes the cost of using theta as the parameter for linear regression to fit the data points in X and y. Set J to the cost, to compute the cost of a particular choice of theta where the thetas represent linear regression parameters.



###### GRADIENT DESCENT #########

the gradientdescentmulti function takes 5 parameters and returns two parameters,

the five parameters are

X ------> adjusted dataset features array

y-------> target variable (price column)

Theta --> linear regression parameters

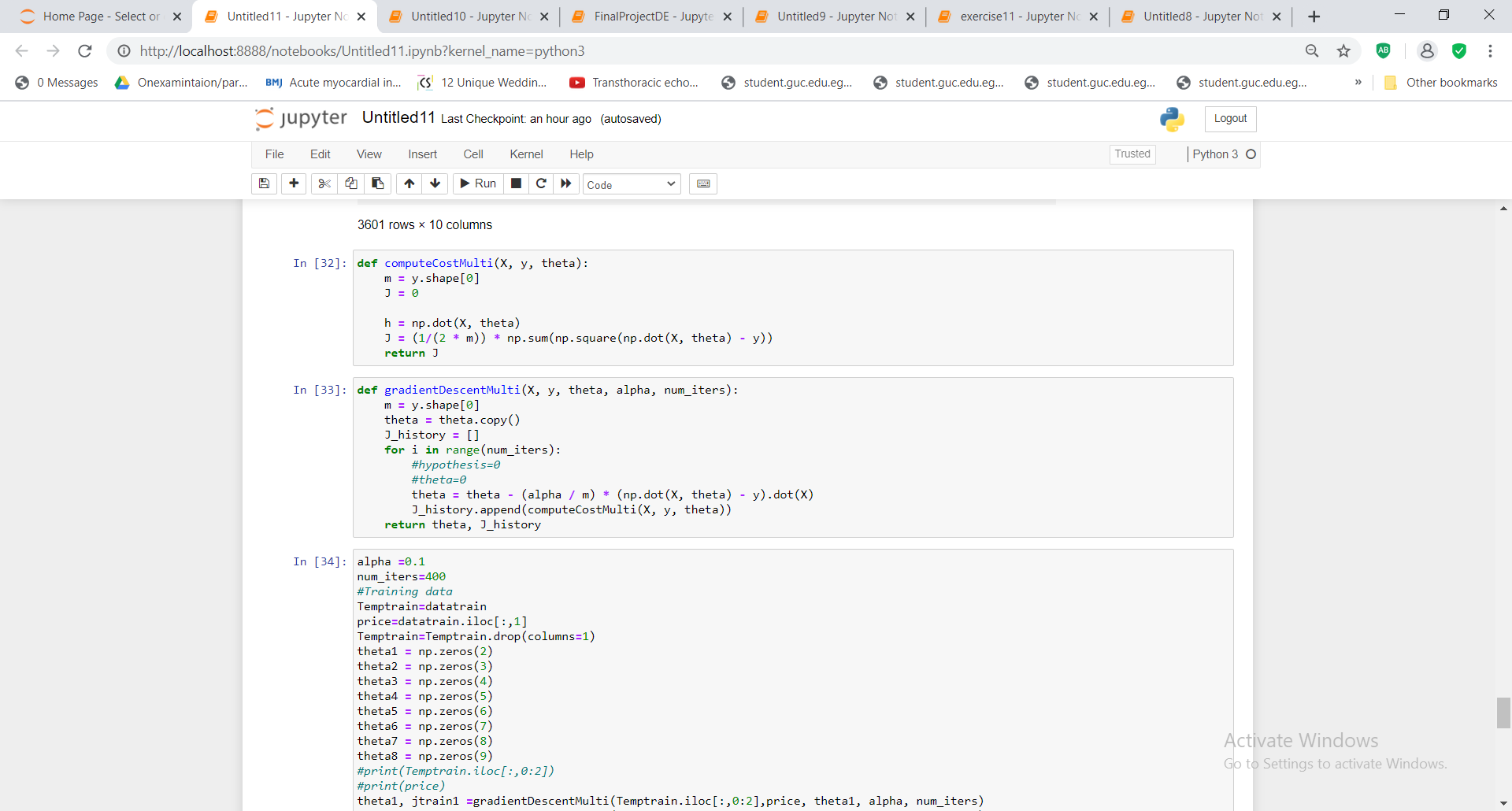
alpha---> learning rate for gradient descent

num\_iters --->number of iterations to run gradient descent

the returned parameters are

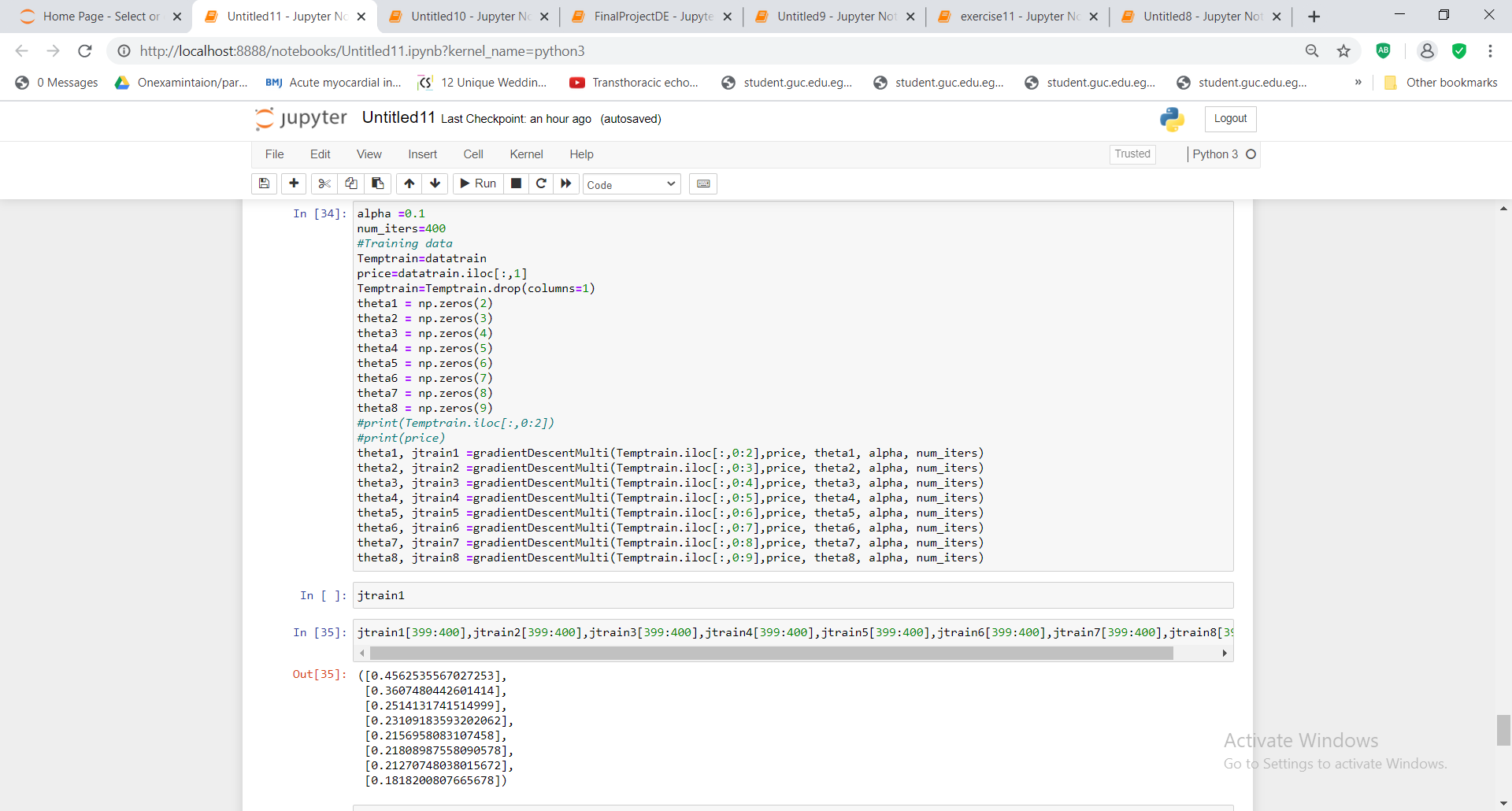
theta--> The learned linear regression parameters

J\_history --> A python list for the values of the cost function after each iteration.

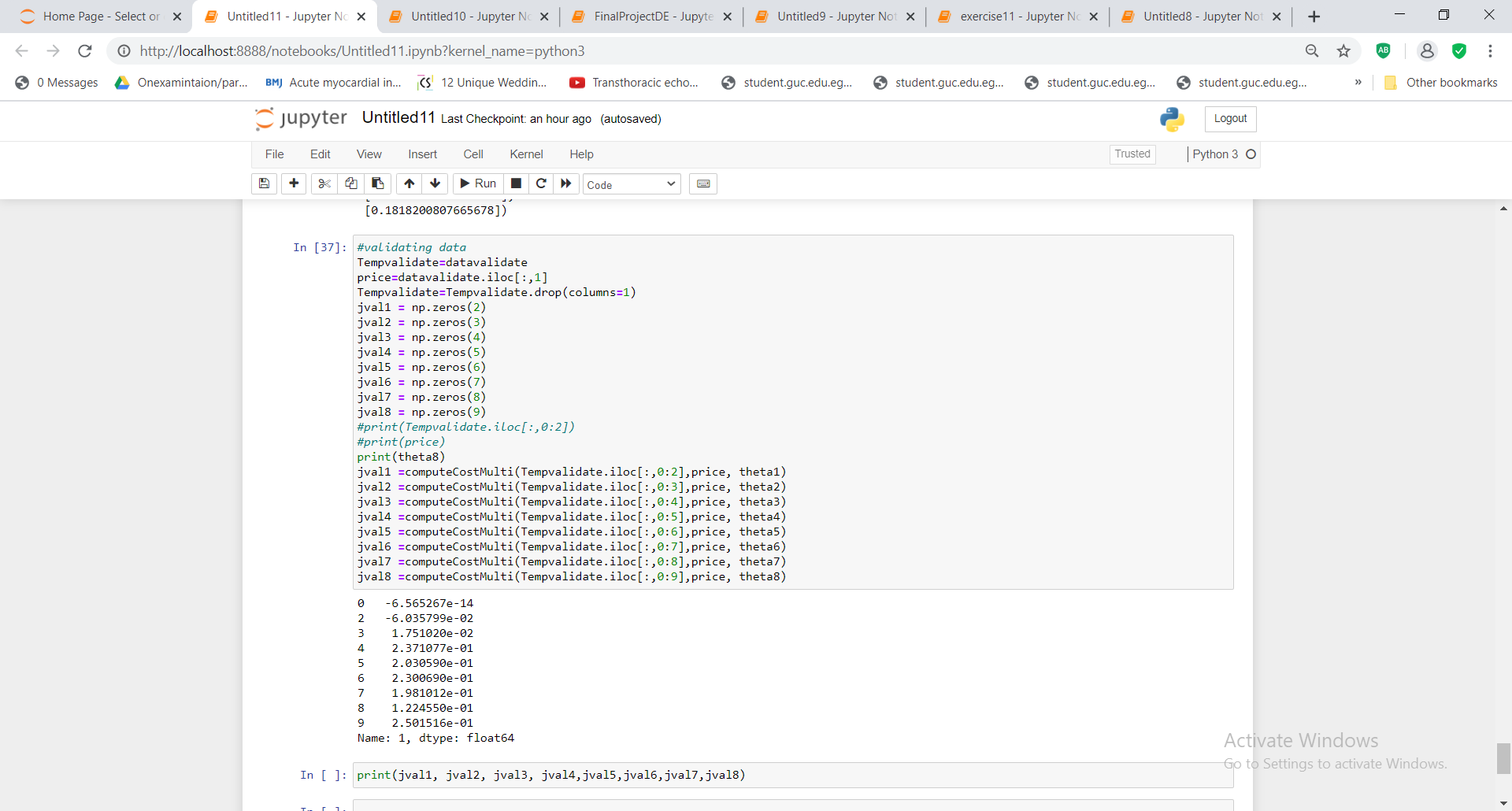


In training data, initialize thetas to have zeros in the columns of each feature. For example the first feature has zero in all its column where its column has an index equal to 2 and so on.

call the gradientdescent using the datatrain to get the trained thetas and jtrain. Where jtrain is a list of j\_history for the one feature iterated 400 times and taking into consideration alpha=0.1. where the last minimum theta between all thetas is taken as the minimum theta thus its degree represents it. (Degree 8)



In validating data, what we care about is the compute cost by each feature. Call computecostmulti function passing to it the X (feature) , y(target variable i.e: price), and the thetas that came from training the date (not the initial zero thetas). In return, this function gives me the J cost of validating data/error where the least j is the best as deducing the error. (Its degree is equal to 8) same degree of training data. Thus I have a degree 8.



After training and validating data, now Test data by calculating jtest by calling computecostmulti function now only once by using theta8 representing degree 8 which show the least error. Calculate price which is equal to the dot product of jtest and theta 8.

