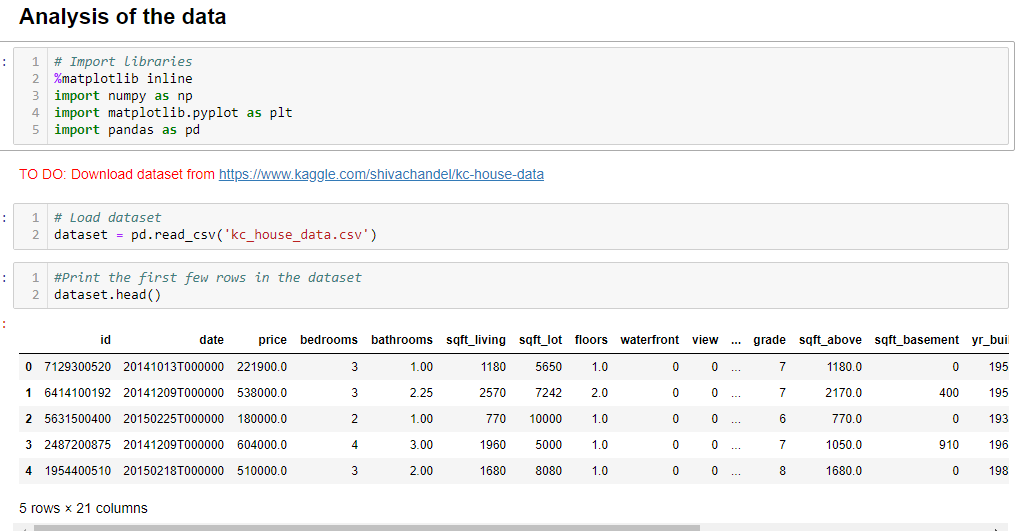
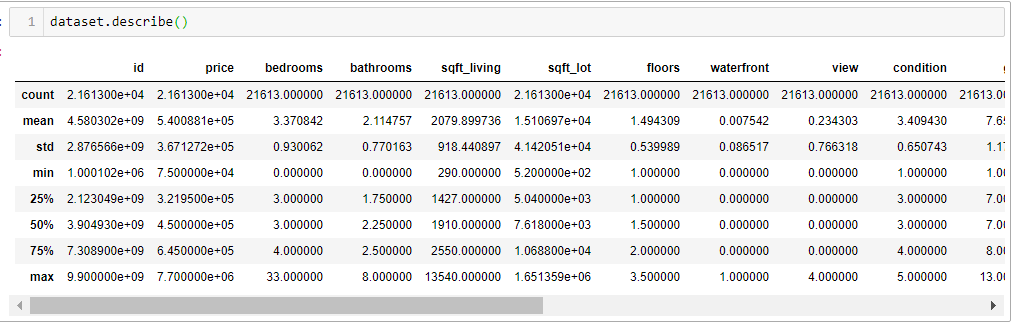
Loading Libraries and loading dataset, then looking some first examples of dataset.

Dataset.head(): shows the first n rows of dataset, default n=5

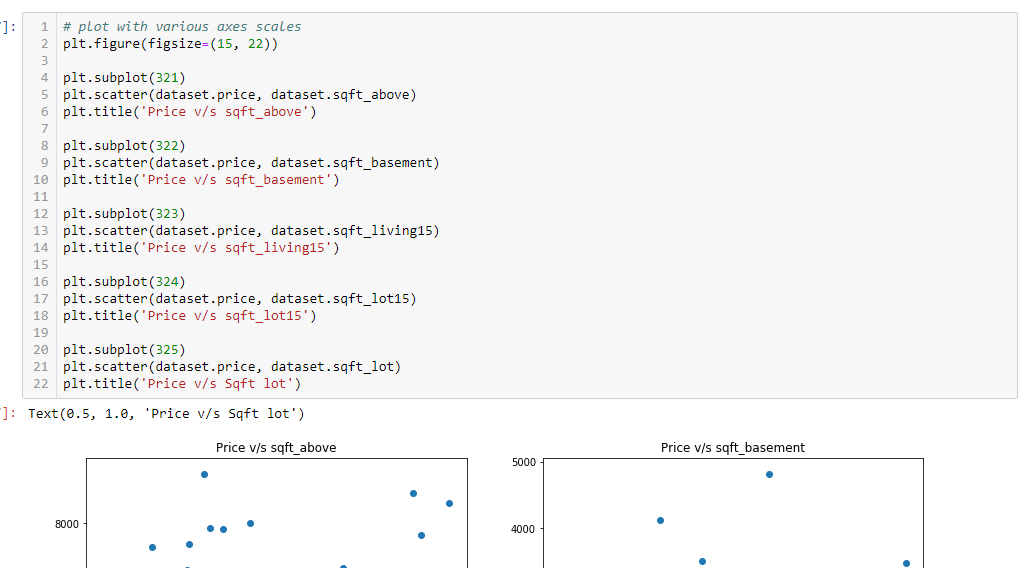
dataset.describe(): Generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding ``NaN`` values. Analyzes both numeric and object series, as well as ``DataFrame`` column sets of mixed data types. The output will vary depending on what is provided. Refer to the notes below for more detail.



plt.figure(figsize=(15, 15)): Define 15 by 15 images.

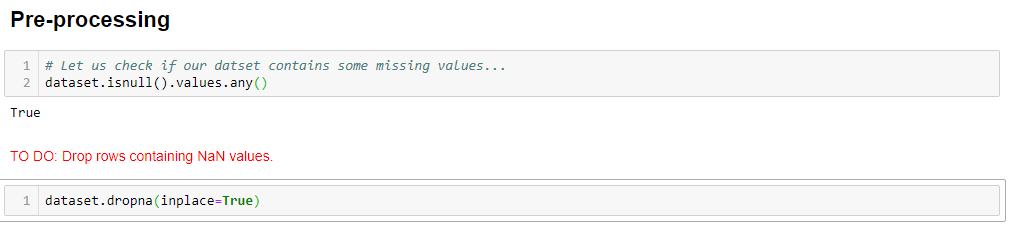
plt.scatter(dataset.price, dataset.sqft\_living): scatter plot like dot plot where x is price and y is living area.

For Multiple ploting we can use subploting method, where subplot(321) means nrows, ncols, index of image to plot.

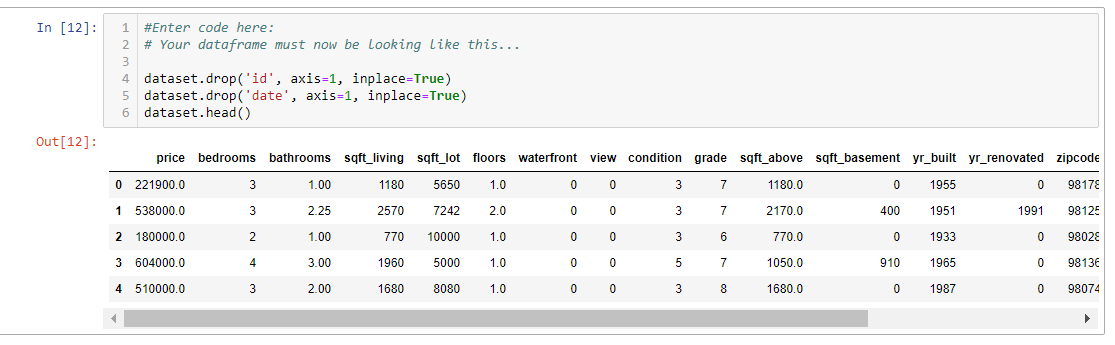


dataset.isnull().values.any(): check if our datset contains some missing values

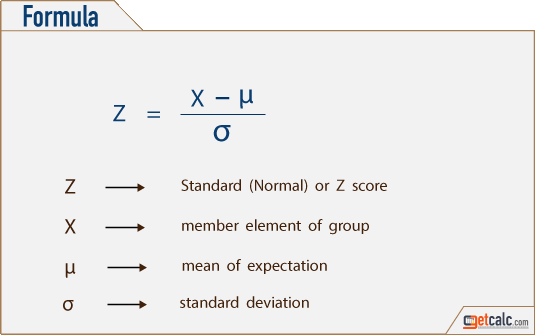
dataset.dropna(inplace=True): drop missing(‘Nan’) values. Inplace means that it will automatically overwrite in the dataset.

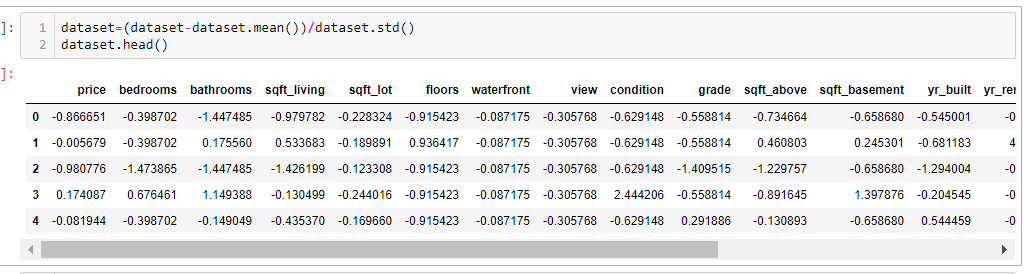


dataset.drop('column name', axis=1, inplace=True): drop the specific column in the dataset.

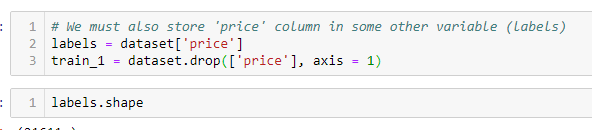


standard mean normalization on the dataframe.

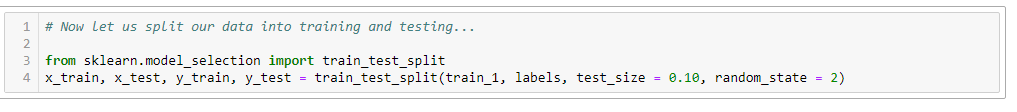




Using price as labels and then dropping price labels and using all other columns as features.

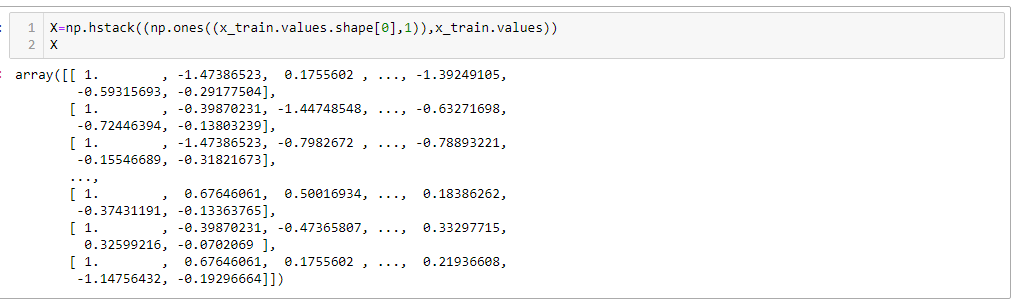


Splitting the dataset in training and splitting, test is 10% size and training is 90% size.

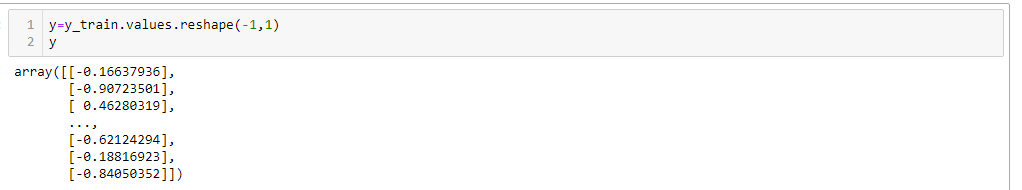


np.ones(x\_train.values.shape[0],1): making 1s column size of training data. And then stacking it with training features values.

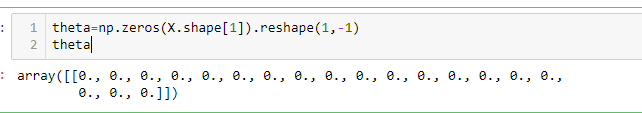
np.hstack: is for Stack arrays in sequence horizontally (column wise).



Reshaping the y\_labels, so it looks like 1 column matrix. So we can use it in matrix operation.



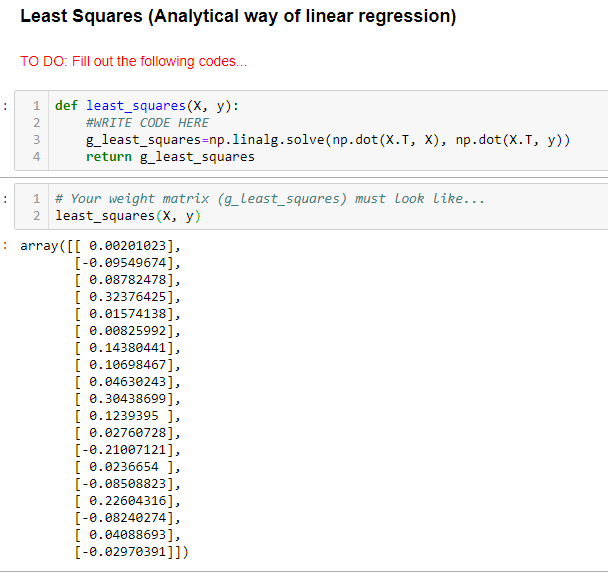
Making a array of of zeros, it will looks like 1 column and X.shape[1] size rows.



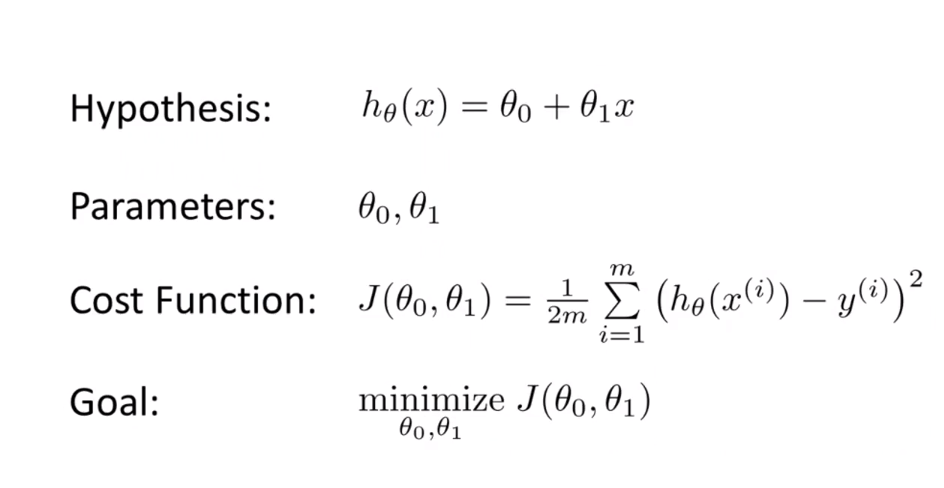
Least squares is analytical way of solving regression. Suppose X is features, theta is weights and y is predicted answers. For finding the weight we need to perform X inverse, for that we first perform sudo inverse. That explained below:



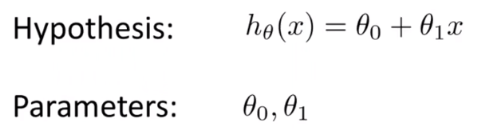
np.linalg.solve(): help us to solve  this equation. For finding theta.



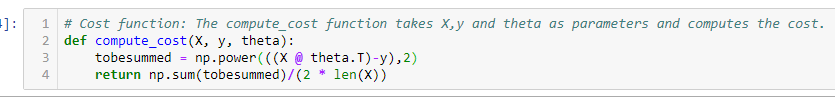
We define cost function which compute the cost by this formula.



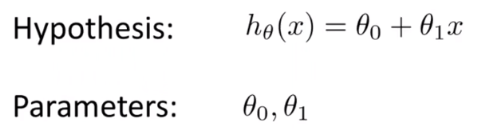
Where  is define below:

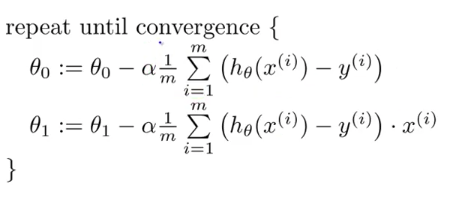


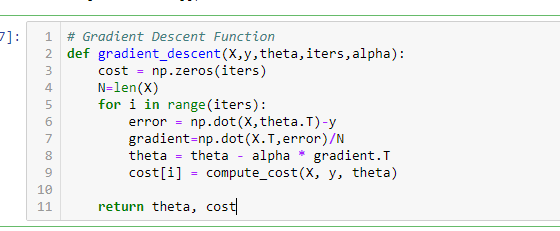
In the code X @ theta.T define the matrix multiplication. @ use for matrix multiplication.



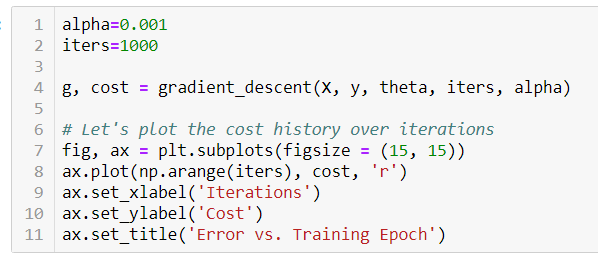
We define the gradient descent function.







Ploting the cost functions.



Mini-batch gradient descent:

In actual practice we use an approach called Mini batch gradient descent. This approach uses random samples but in batches. What this means is that we do not calculate the gradients for each observation but for a group of observations which results in a faster optimization.A simple way to implement is to shuffle the observations and then create batches and then proceed with gradient descent using batches.

For this

1. We first shuffle the dataset: np.random.shuffle(indices)
2. Then we assign a new training set from this batch parts:

end\_idx = min(start\_idx + batch\_size, len(X))

X\_batch=X.values[start\_idx:end\_idx]

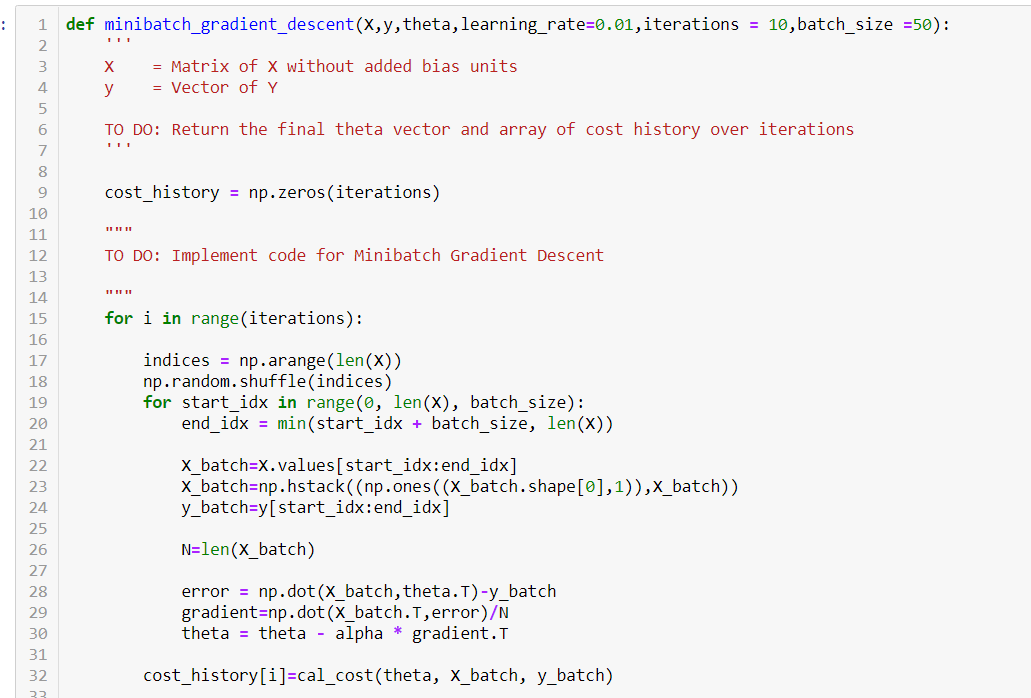
1. Then add 1 like above:

X\_batch=np.hstack((np.ones((X\_batch.shape[0],1)),X\_batch))

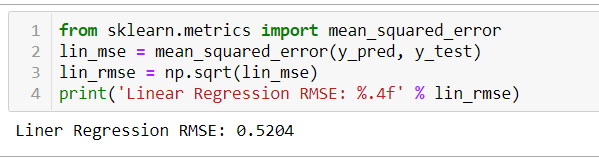
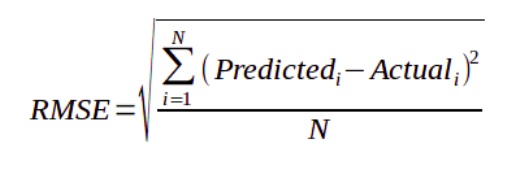
1. Also split the y like it

y\_batch=y[start\_idx:end\_idx]

other things are same as we have done earlier calculating gradient etc.



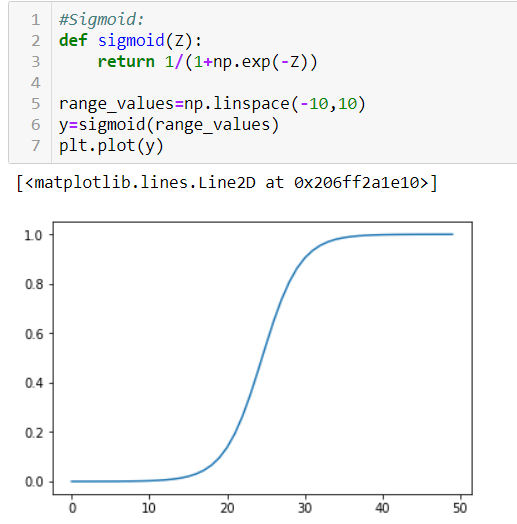
We find the Root mean squared error like this:

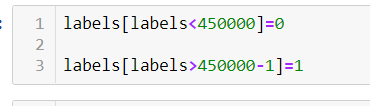


**Logistic Regression:**

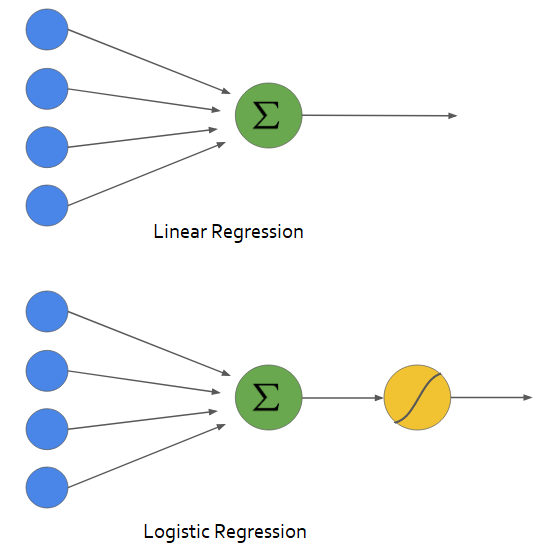
We define the sigmoid function and check the plot from values of -10 to 10.





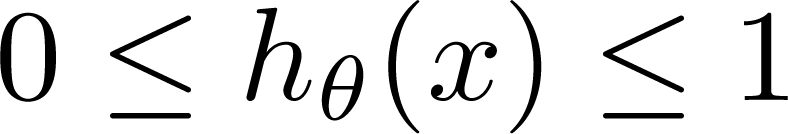
As we are doing logistic regression, so we need to fix number of classes for this house data. we have follow an approach to define that a house is costlier or not, for example if an house value is larger than respective value 450000 it is costlier, so we give it 1 tag, or other to 0.

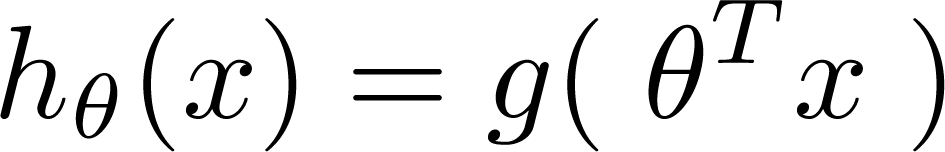
Logistic Regression is generally used for classification purposes. Unlike Linear Regression, the dependent variable can take a limited number of values only When the number of possible outcomes is only two it is called Binary Logistic Regression.

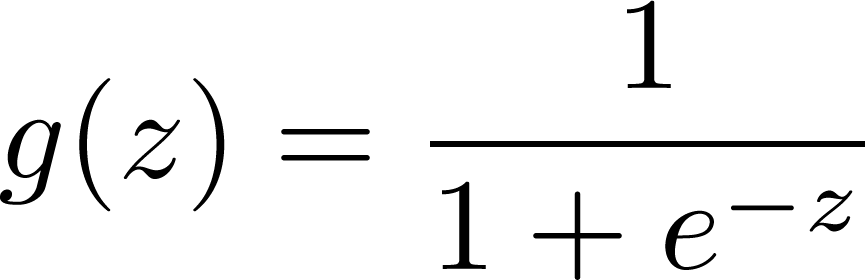


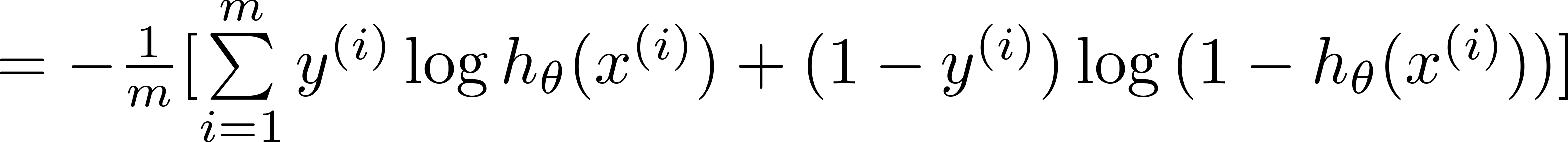
As we are doing logistic regression, so we need to fix number of classes for this house data. we have follow an approach to define that a house is costlier or not, for example if an house value is larger than respective value 450000 it is costlier, so we give it 1 tag, or other to 0.

Core idea: force hypotheses between 0 and 1

Logistic Regression: 

Hypothesis: 

Sigmoid Function: 

Cost Function: 

Gradient Descent:

