Machine Learning Lab

Experiment – 3

Ankith Motha

21BAI1284

Linear Regression

**Dataset:** Abalone Dataset

**Input Variables:** Length, Height, Viscera weight, Shell weight, Sex

**Target Variable:** Rings

Regression is a type of Machine learning which helps in finding the relationship between independent and dependent variables. Linear Regression is a type of Regression which assumes a linear relationship between the dependent and independent variables.

The objective of Linear Regression is to find the most efficient coefficients which describe the linear relationship the best,

Linear Regression can be solved through different methods such as Gradient Descent, Genetic Algorithm, Closed-Form Solution, etc. However, gradient descent is an iterative algorithm, whereas closed-form solution gives us the direct answer.

**Closed-Form Solution**

An optimization problem is closed-form solvable if it is differentiable with respect to the weights w and the derivative can be solved, but that is only true in the case of a minimum/maximum optimization problem.

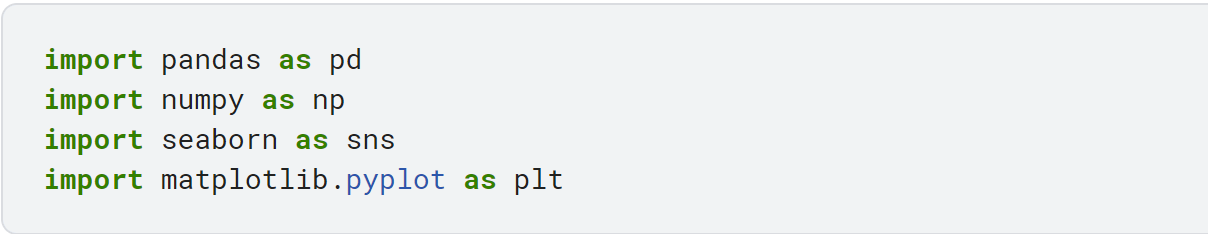
The optimization problem, aims to reduce the loss function used to calculate the deviation of the predicted value from the true value.

Given a set of N input vectors of D dimension and a set of output vectors of S dimension, we need to find a linear relation between the different input feature and the target.

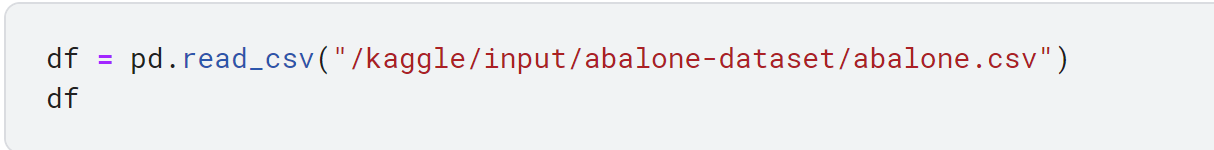
We use the linear relationship given below and find the optimum values of the coefficients, so as to get a linear model with least prediction error

**Code:**

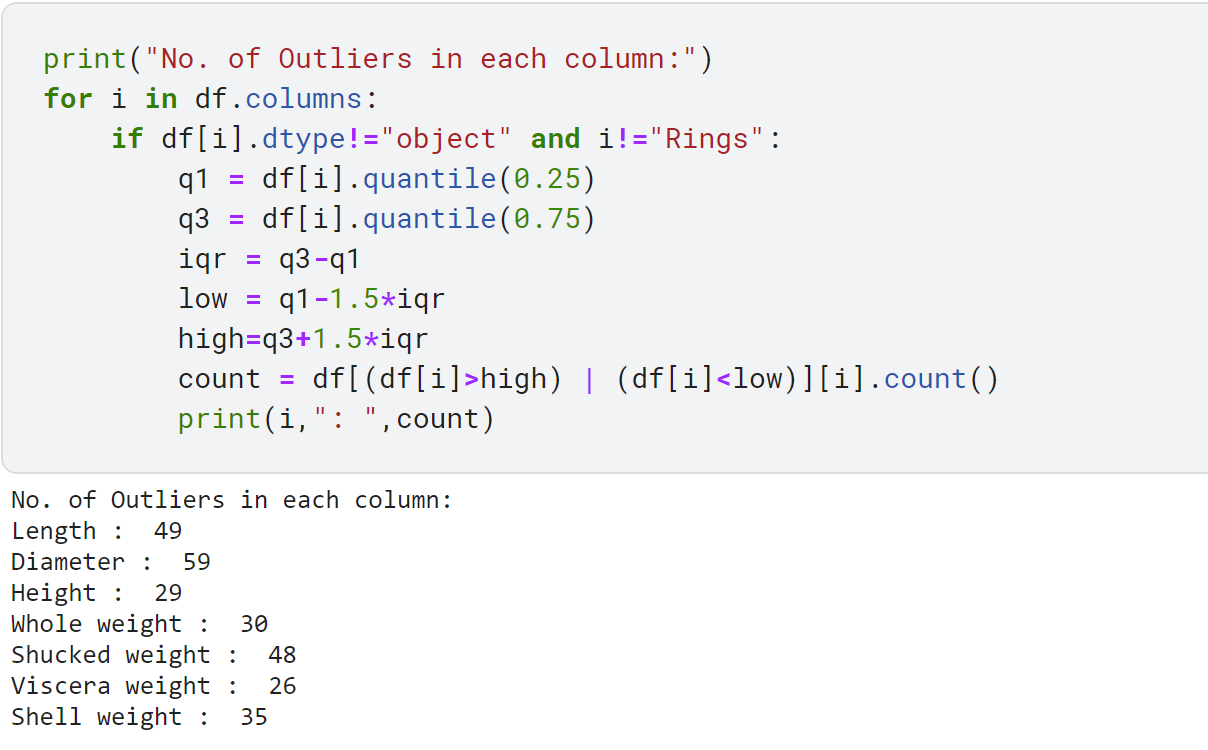
Import all the required packages



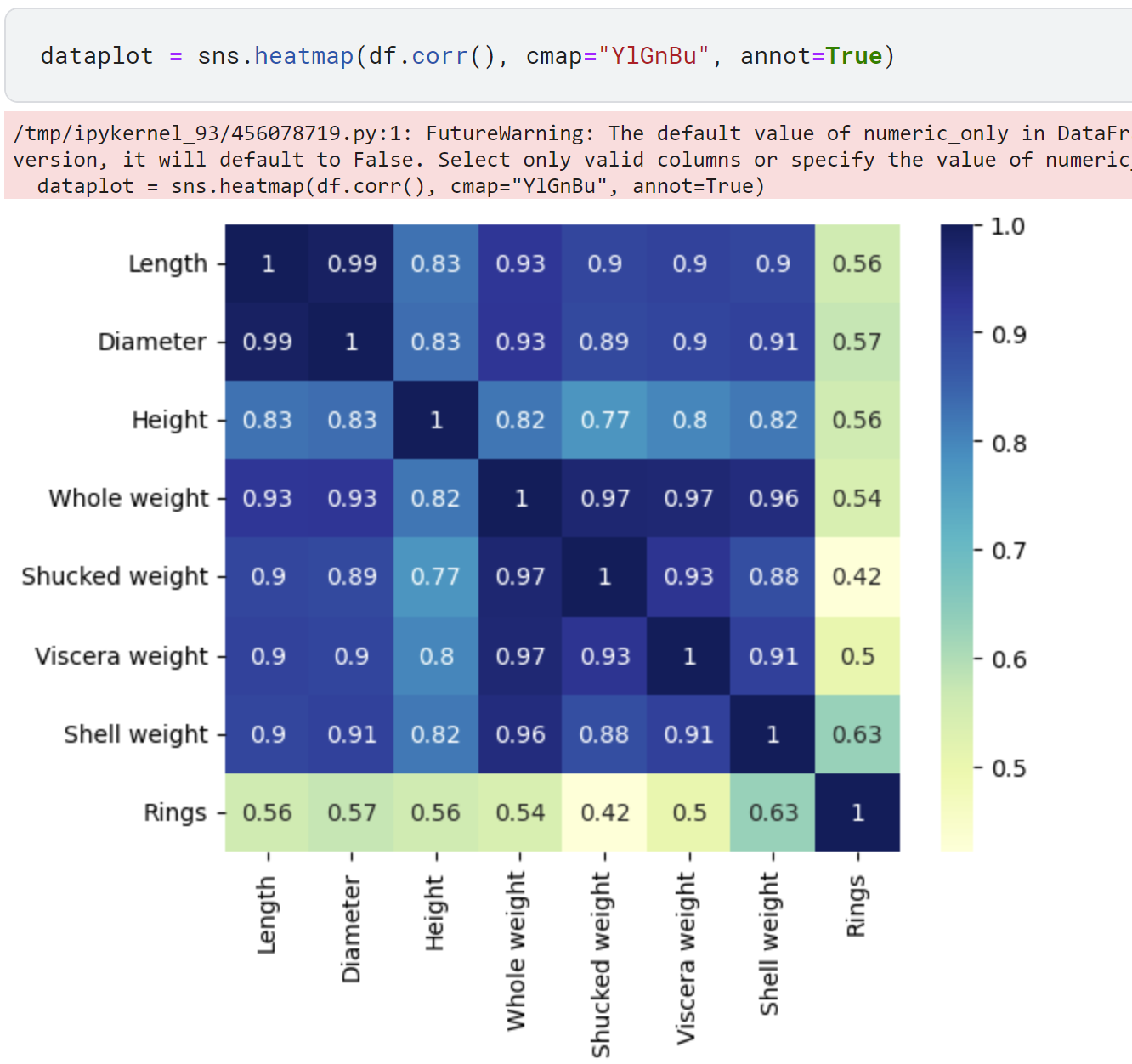
Load the dataset



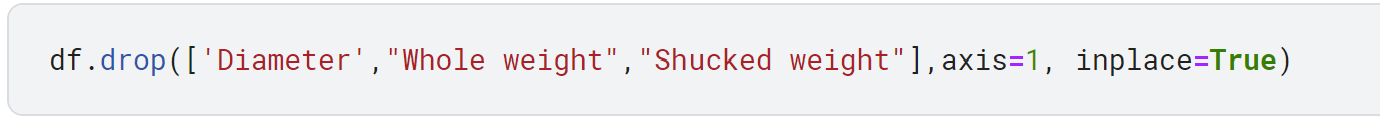
Find the outliers in the dataset (Only numeric columns)



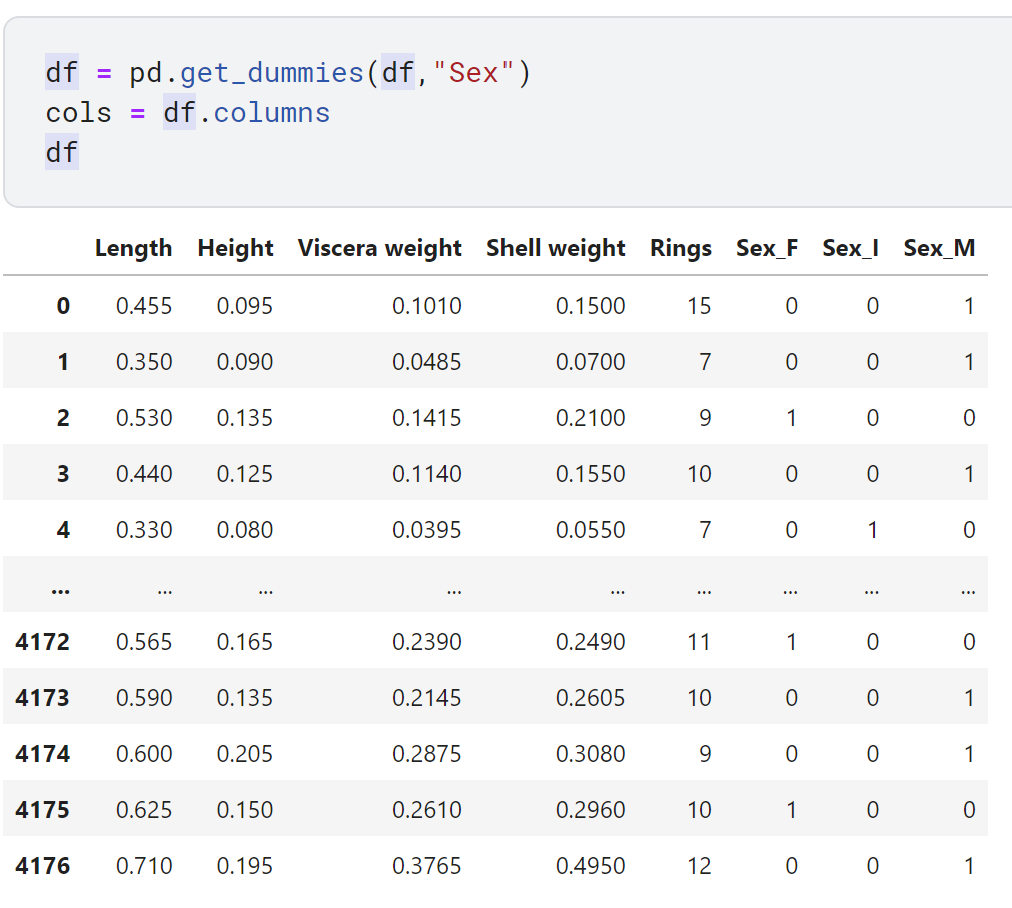
Display the correlation matrix and remove the extra columns with correlation more than 0.96, since these columns are highly related/similar and do not convey any meaningful features.



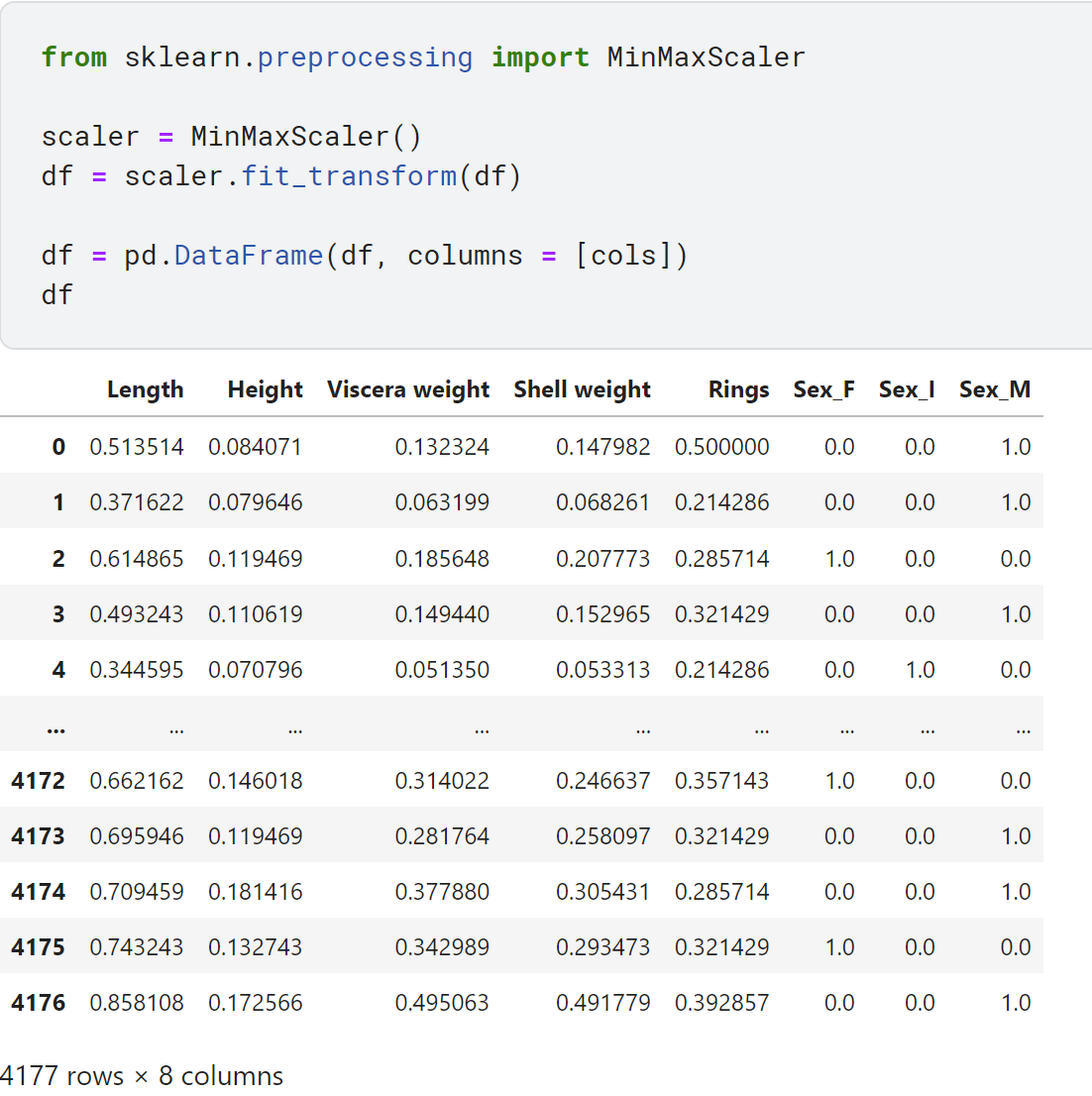
It can be seen above that, (whole weight, shucked weight and Viscera weight) are highly correlated and similarly, (diameter and Length) are also highly correlated, hence some columns can be forpped.



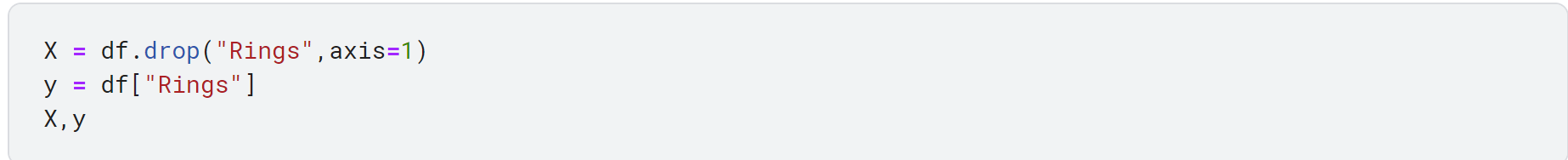
Next, we encode the columns with categorical data, since the ML model can’t process text data. We use One-Hot encoding for this.



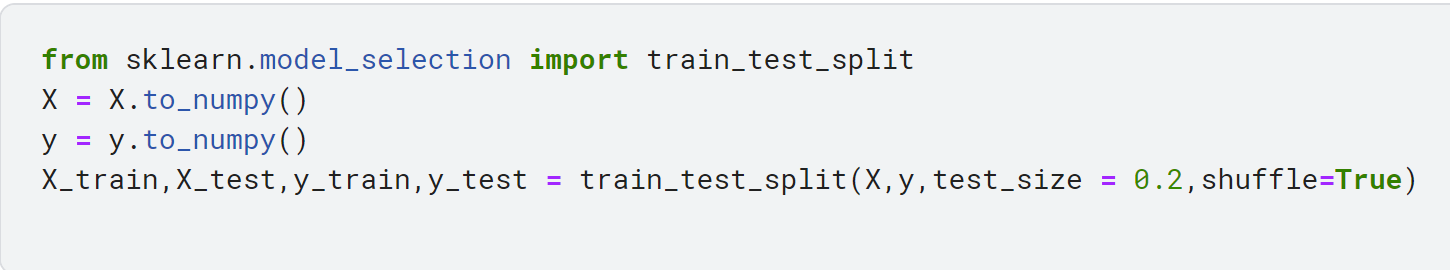
We then perform Normalization to bring the values of all the features within the range of 0 to 1. This will bring uniformity in data and prevents a biased prediction. We use MinMaxScaler to get the values within 0 to 1.



We finally split the data into the **Input Vectors and Target Vectors**



Now, split the X and y vectors for training and testing.



**Derivation of Closed Form Solution:**

We need to find a mapping that maps the multi-dimensional input to multi-dimensional output,

f: D ->S, where D is the dimension of our input and S that of the output.

One way to think of this is to find the weight vector m and bias b for each dimension of the output. It’s as if we’re stacking a bunch of these functions over N data points:

 where i = 1, 2, …, S.

If we formally stack these mappings together, the bias b is simply the stacked biases bi and the weight matrix W can be formulated by stacking all the m vectors so,



We rarely have just one sample in a dataset. We usually, as mentioned above, have a set of N data samples. We can add all these data samples into two matrixes, one for input and output.





We want to pass all these through the function at the same time, which means we need to extend the function to have the form,



The function at this step is:



We just need to absorb the bias. We’ll change the matrix X by adding a column of 1s to the end creating:



and the matrix W by adding the bias vector as the last row, so W becomes



Our new dimensions are:





Now we need to formulate the optimization. Since we want to get XW as close as possible to Y, the optimization is the minimization of the distance between f(X) = XW and Y or:



This is essentially the least-squares loss function, which classically looks like:



Now we find the derivative or gradient with respect to the weights w,







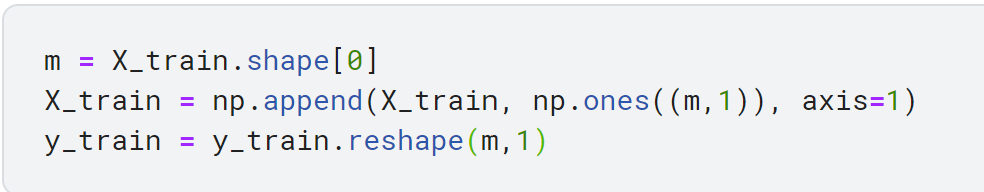
Since we need to find the values of W such that loss we least, we find the derivative and equate to 0

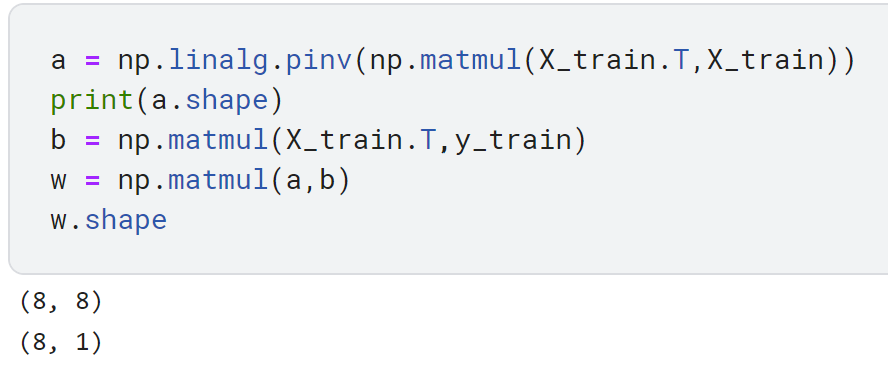


Hence to get the weights of the Linear regression line, we can use this closed form solution

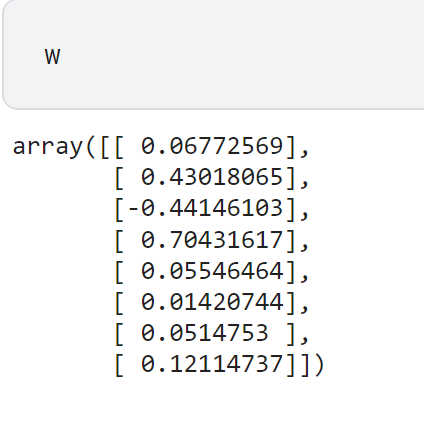
**Code:**

We now implement this closed form solution programmatically.

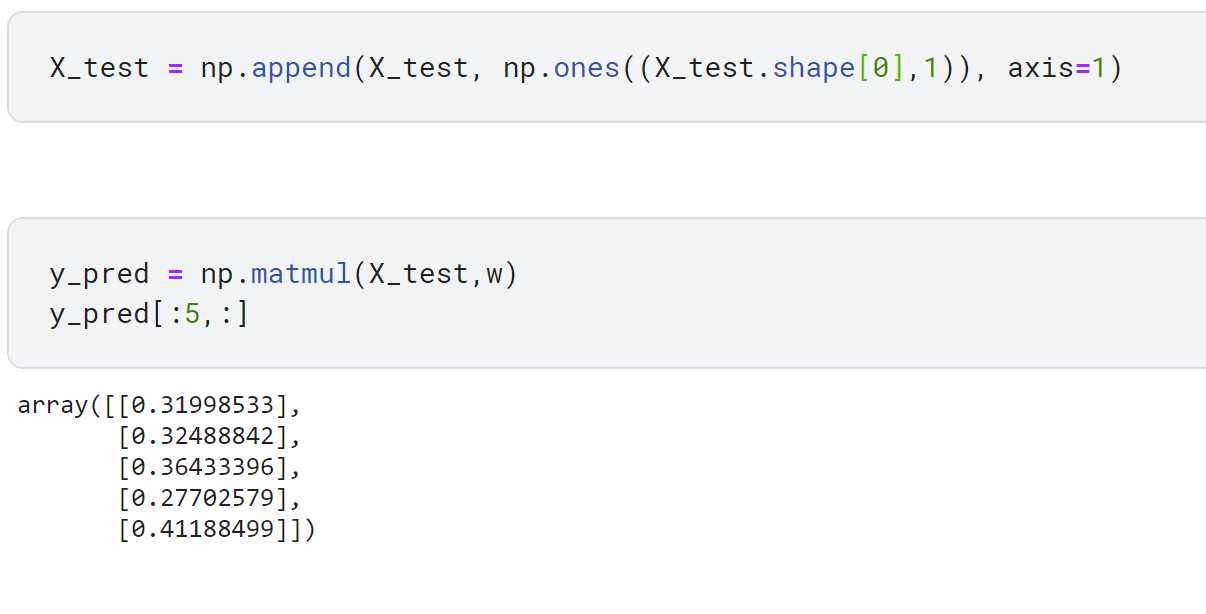


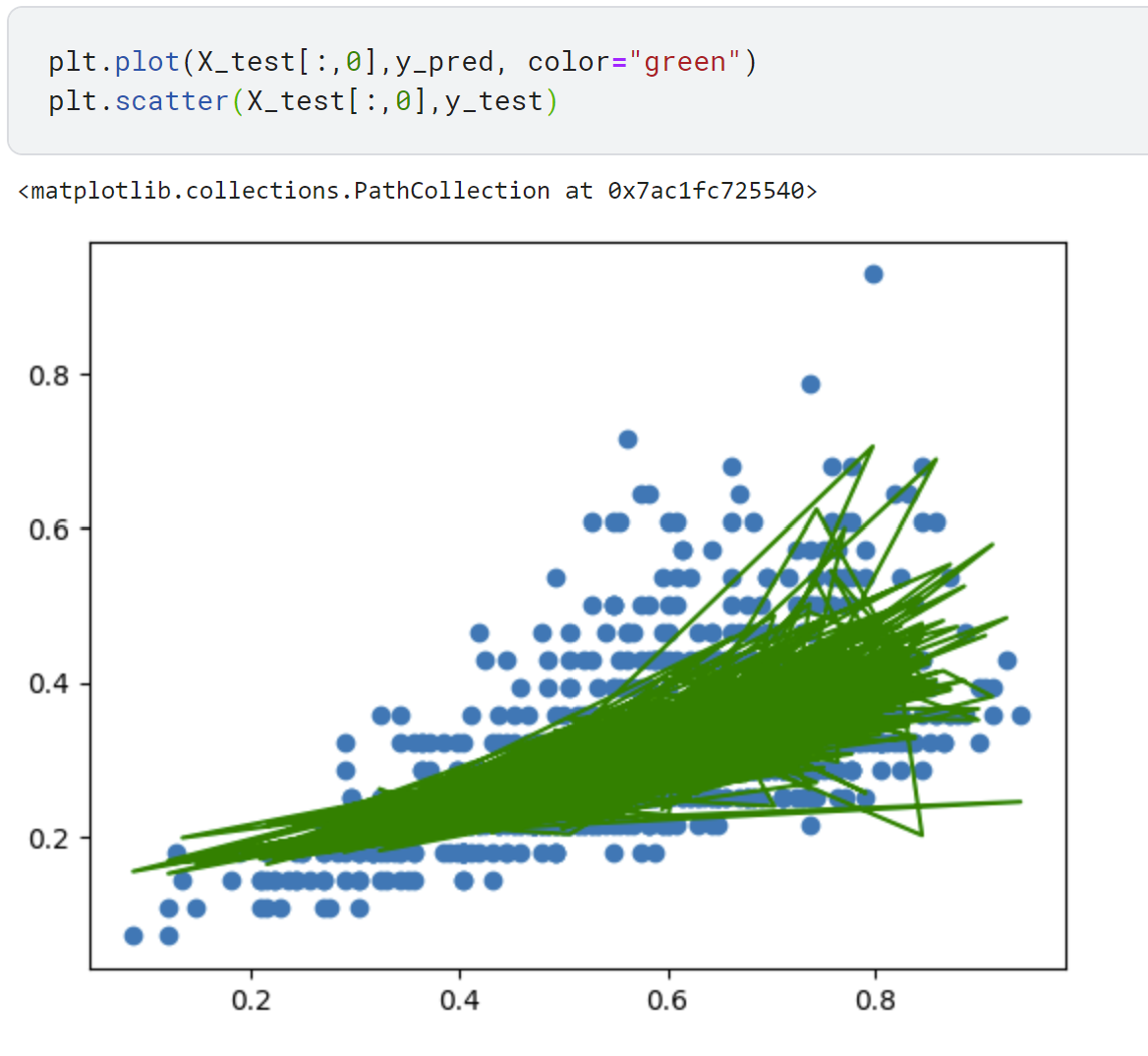


We have now found the weights which contain the coefficients and the bias (intercept)



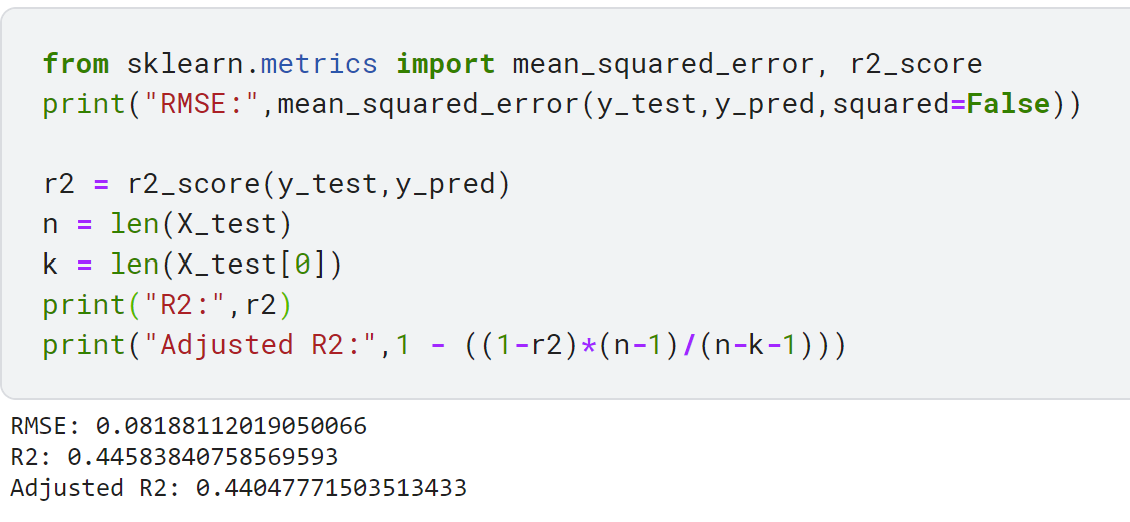
Now, we move on to testing our approach on the test data.



Given below is the plot of the regression line in 2D, however, this doesn’t exist as a line since we are only plotting the projection of the line from a higher dimension to two dimensions:  


Finally, we must evaluate our model’s performance, for which we use 3 metrics:

1. RMSE
2. R2 (R squared)
3. Adjusted R2



R2 score is a metric that tells the performance of your model.

In contrast, MSE depend on the context, whereas the R2 score is independent of context.

So, with help of R squared we have a baseline model to compare a model, which none of the other metrics provides. So basically R2 squared calculates how must regression line is better than a mean line.

Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.