J. Rodolfo Herrera Ruíz A01065894
Machine Learning
Linear regression implementation

Linear regression with fishes' dataset

In order to implement a linear regression model, it is first needed a dataset to fit this model. Normally this should be like this as according to the dataset and whatever it is wished to do with it is better to find a model which fits it the best. However, for a linear regression model this dataset was found about the fishes' market, and some of their characteristics, such as 3 different lengths, height and width and with those you'll be able to predict the fish weight.

The dataset was obtained from Kaggle and can be found at the following link. The file is a csv, comma separated value format document which is read through the pandas library. Thank you pandas. The database was quite short of data, so more data was added. With this it is possible to easily manipulate and display the data. This summary is shown below by the commands "print(data.info())" and "print(data.head())"

```
RangeIndex: 1113 entries, 0 to 1112
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Species  1113 non-null   object
 1   Weight   1113 non-null   float64
 2   Length1  1113 non-null   float64
 3   Length2  1113 non-null   float64
 4   Length3  1113 non-null   float64
 5   Height   1113 non-null   float64
 6   Width    1113 non-null   float64
dtypes: float64(6), object(1)
memory usage: 61.0+ KB
None
  Species  Weight  Length1  Length2  Length3   Height   Width
0   Bream   242.0     23.2     25.4     30.0  11.5200  4.0200
1   Bream   290.0     24.0     26.3     31.2  12.4800  4.3056
2   Bream   340.0     23.9     26.5     31.1  12.3778  4.6961
3   Bream   363.0     26.3     29.0     33.5  12.7300  4.4555
4   Bream   430.0     26.5     29.0     34.0  12.4440  5.1340
```

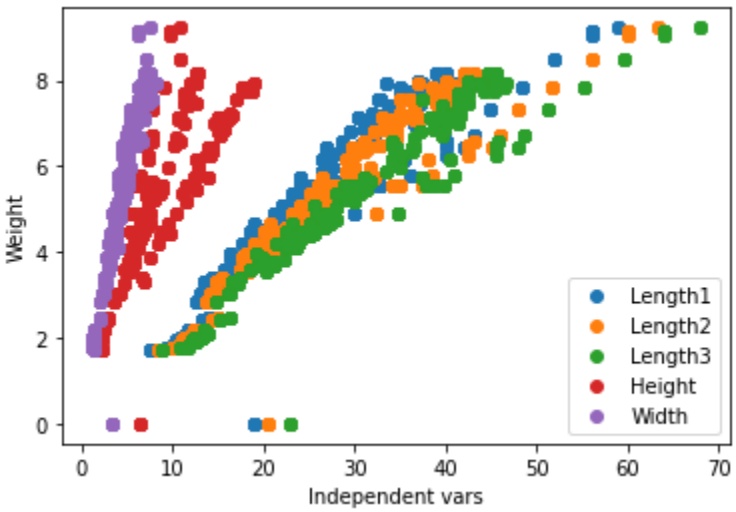
Img1. Data info and head

Now we need to separate the data into train and test data, however If we only take the first values for the train data it will only take some species and not all of them, so we randomly shuffle all the data, for that we can use another pandas function called "sample()" which randomly shuffles all the rows to random places. Thank you again pandas.

Know that this is done we can separate our train values from our test values, lets just grab the first thousand values for the train data and evaluate with one hundred. The independent train values are stored in "xtrain" and the dependent train values in "ytrain". The same happens with the test values, stored in "xtest" and "ytest".

J. Rodolfo Herrera Ruíz A01065894
Machine Learning
Linear regression implementation

Here comes a small trick. As the data is too disperse, we pre preprocess the data a little bit in order to be more linear, we elevate the output values to 0.3. Know we plot our train data, just to show its nicely trend to be linear.



Plt1. Independent variables linearity

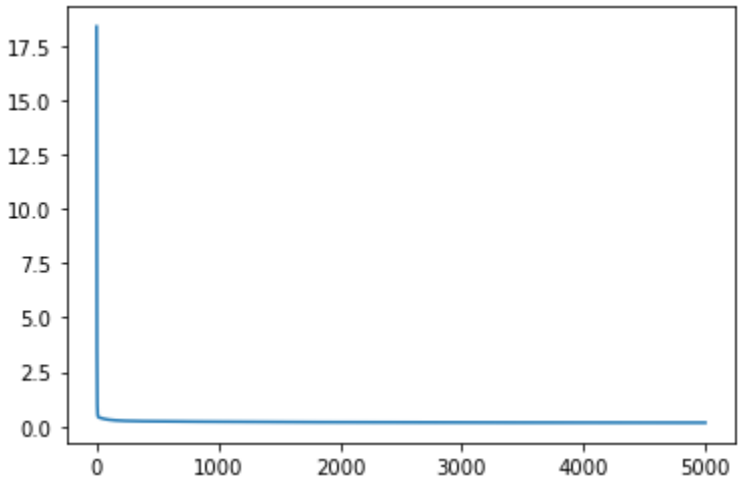
For the training implementation there needs to me certain functions that we'll be needing in the future, the "hyp" function will calculate the predicted values, therefore it needs some weights to evaluate the independent variables and return a respective output to the linear model which is represented by y=mx+b or in this scenario by h=b+w0x0+w1x1+w2x2…

In order to show improvement, there is a global variable which stores the accumulated error mean by generation, this will be finally plotted at the end to show how the error mean decreases through iteration. This is made through an error function which is amazing as is the one function which helps see what is happening, if your train model is doing something.

The following function is the one which makes everything move, the "gd" function contains the gradient descent method for the whole model to update the weights (for the predictions function) in order to do this, it uses the cost function. Therefore, receives the actual weight values and then modifies them according to the error between the predicted and actual outputs, then it is accumulated for lately generate new weights by a step of alfa.

For the overall running it is needed to define the alpha, maximum generations and initial weight values. The alpha works quite well with a value of 0.0006, if it its higher it will diverge and will keep going wrong making a huge error mean. That's It we run the code until there is no improvement or until it reaches the max generations. Each iteration the error mean will be displayed but the plot resumes it. The last error mean will vary depending on the selected train data, but it is generally under the 0.17. Same with the weights, they will vary according to the selected inputs.

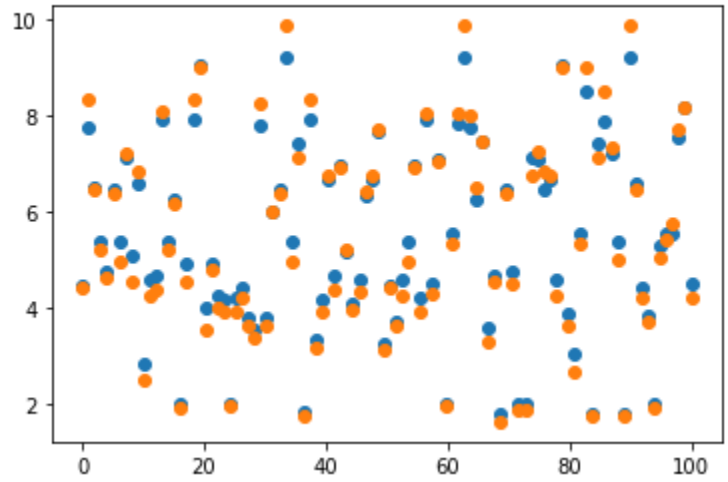
J. Rodolfo Herrera Ruíz A01065894
Machine Learning
Linear regression implementation



Plt2. Error per iteration

This seems nice, but its time to see if it works. To do this we compare each value of the train data against the actual true value. This is meant by the same prediction function but with the "xtest" values. The predictions are shown against the true values in the following plot. In order to get a relation, an error mean is calculated, and shown.

```
Test error mean=0.070089
```



Plt3. True vs predicted test values

Finally, there is a section for te user to try with different values and check how close the prediction is to the true value. And that's it, there is a trained linear regression model.