Project3: Classification using Logistic Regression

Tamanna Baig

Problem Description:
The given dataset represents two types of fishes found in Lake Hartwell. TigerFish0 (represented as 0) which is bad for consumption and TigerFish1 (represented as 1) which looks identical but is a better choice. The main aim of the project is to develop classification algorithm using logistic regression.

The python file when executed, will prompt the user to input 2 values: body length of the sample fish in cms and dorsal fin length of the sample file in cms. This will be prompted till the user enters 0 for both the values. The final output is the predicted type of the sample fish.

Data Description:
The initial training data consists of 300 records, each record has 3 tab-separated entries: a float value 'Body Length' in centimeters; another float value, 'Dorsal Fin Length' in centimeters and the binary value for type of fish represented by 0 or 1. Figure 1 shows a plot for the initial data set given.

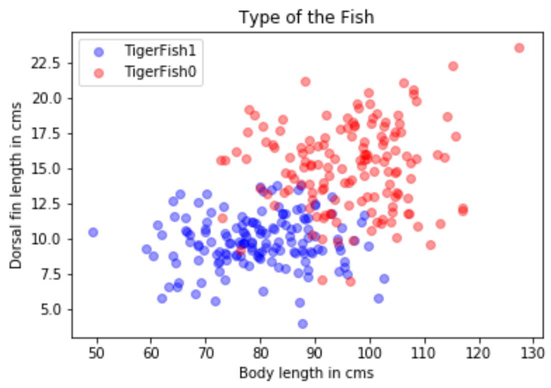


Figure 1. Initial plot of the data

Training the Logistic Regression algorithm:
The first step is to randomize the data. For **Feature scaling,** I chose to apply normalization on the dataset, then the dataset is split into 70:30 train and test sets respectively. Initial weights, learning rate, and number of iterations, the training set is subjected to sigmoid calculation, hypothesis function and Cost, J is calculated. Depending on the final J values, learning rate, weights and iterations need to be adjusted.

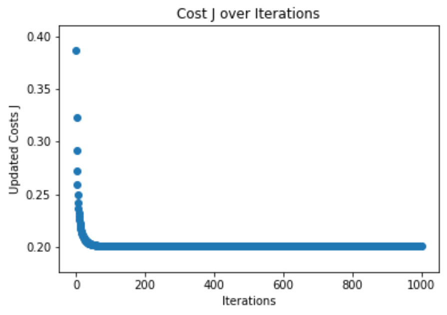


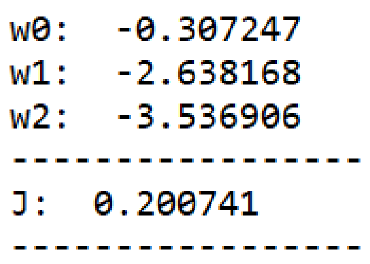
Figure 2. Cost with initial values

The initial plot (shown in Figure 2) was created by using iterations = 1000,
alpha = 0.01,
initial weights = [-1, 0, 1]
Final J value = 0. 200741

On close observations, I gathered making iterations as 400, would be sufficient enough and will give the algorithm enough time to learn and predict the correct fish type as the cost value does not change significantly after 400.

Obtaining Final Values:

Figure 3 shows the final values of weights, J and errors generated by the learning algorithm on the training set. Figure 4 shows the Cost vs Iteration plot.



```
w0:   -0.307247
w1:   -2.638168
w2:   -3.536906
-----------------
J:  0.200741
-----------------
```

Figure 3. Final Values after training

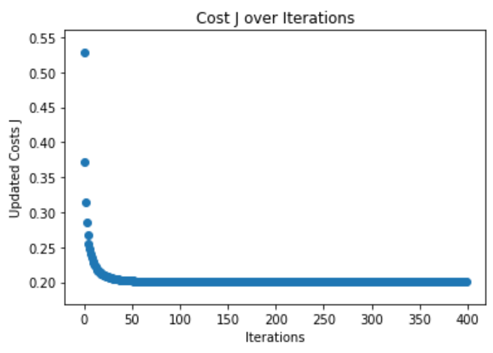
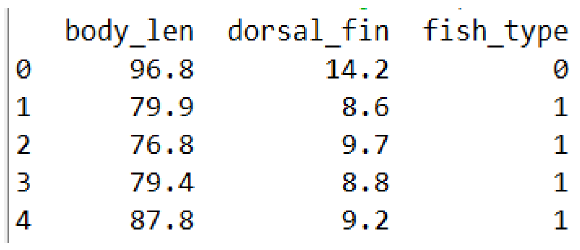


Figure 4. Final plot after training

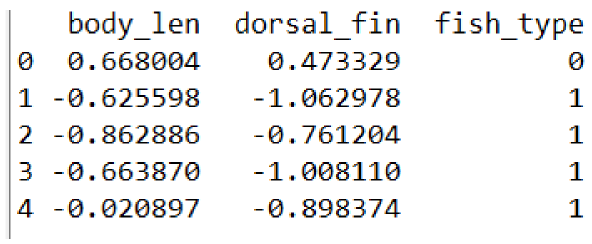
Feature Scaling:

The two features are in minutes and ounces respectively, therefore, to make them both into a similar scale, **normalization** was applied. Figure 5 and 6 show values before and after normalization respectively.



```
   body_len  dorsal_fin  fish_type
0      96.8        14.2          0
1      79.9         8.6          1
2      76.8         9.7          1
3      79.4         8.8          1
4      87.8         9.2          1
```

Figure 5. Values before normalization



```
   body_len  dorsal_fin  fish_type
0  0.668004    0.473329          0
1 -0.625598   -1.062978          1
2 -0.862886   -0.761204          1
3 -0.663870   -1.008110          1
4 -0.020897   -0.898374          1
```

Figure 6. Values after normalization

Value of J on Test set:

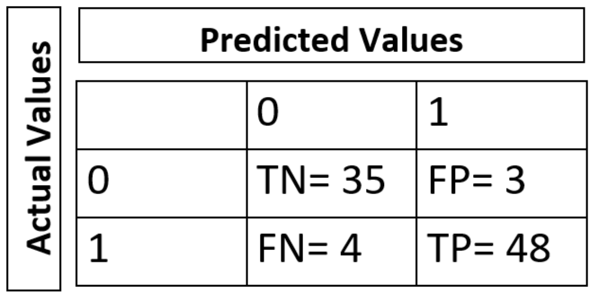
The value of J on test set = 0.200741

The Final Results:

Figure 7 represents the confusion matrix got from the predictions. Figure 6 shows the values for accuracy, precision, recall and F1 Score

Accuracy – 92.22%
Precision – 92.31%
Recall – 94.12%
F1 Score – .92

Figure 6. Final Values with Logistic Regression

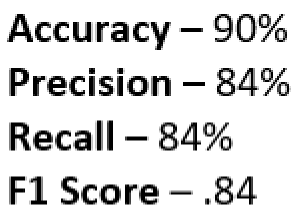


Actual Values	Predicted Values	
	0	1
0	TN= 35	FP= 3
1	FN= 4	TP= 48

Figure 7. Confusion Matrix

Comparison with kNN approach:

In my case Logistic regression did better compared to kNN approach by a little difference. The differences between the values is shown below. Even though the test set size was 60 in the knn method, the accuracy, precision, recall and F1 Score were better in case of Logistic regression.



Accuracy – 90%
Precision – 84%
Recall – 84%
F1 Score – .84

Figure 8. Final Values with kNN approach

		Predicted Values	
		0	1
Actual Values	0	TN= 38	FP= 3
	1	FN= 3	TP= 16

Figure 9. Confusion Matrix with kNN approach