The goal of this project is to build a machine learning algorithm for classifier problem. The Classifier should take any number of features i.e. multivariate dataset and will be predict the classes of multiclass labels. Here I have chosen to implement the Logistic Regression Classification Algorithm as it is easy to implement, and we can extend the concept of Linear Regression to build the same. To minimise the cost function, I am using classic Gradient Descent Algorithm.

Details of the Algorithm:

Binary Classification Implementation:

We will first try to build a binary classifier where there will be two types of dependent variable. As the Dependent variables are Binary in nature the linear regression model will not fit properly, so we need a model that can classify the binary data with the output probability. Logistic Regression uses the Logistic Function or the Sigmoid Function to fit the data and predict the probability. A sample has been shown in the following figure where the sigmoid function refers to the Probability of passing the Exam and it fits perfectly with the two type of binary result (0,1). To differentiate the data, we can set a threshold above which we will predict 1 and below that we predict 1, here the threshold might be 0.5.

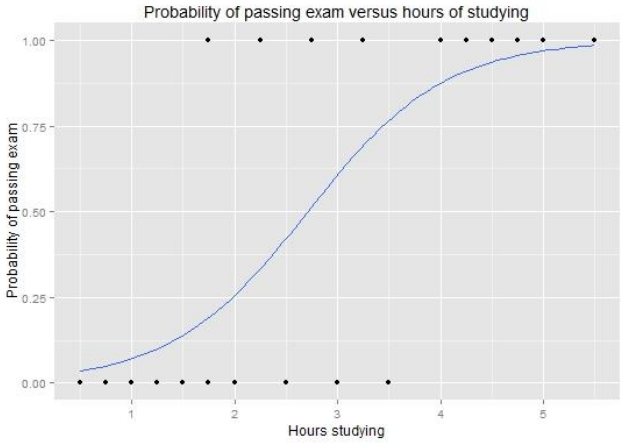


Figure 1 : Sample Sigmoid Function (En.wikipedia.org, 2018)

Mathematically the sigmoid function looks like :

$$g(z) = \frac{1}{1 + e^{-z}}$$

We know for Linear Regression the hypothesis Function for multivariate data looks like :

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_1 x_1 + \cdots \ldots\ldots\ldots\ldots\ldots + \theta_n x_n$$

,where x_0 is the bias term set to 1.

Here for mathematical simplicity we will use the Vectorised (Matrix) form.

So, if there are n variables and m rows of data $X = \begin{bmatrix} x_0^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_0^m & \cdots & x_n^m \end{bmatrix}$ And $\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ . \\ \theta_n \end{bmatrix}$

The hypothesis function in Vectorised form will looks like $H_\theta(X) = X\,\theta$

Now we will extend the Hypothesis Function for Logistic Regression using the Sigmoid Function as –

$$H_\theta(X) = \frac{1}{1 + e^{-X\theta}} = G(X\theta)$$

Note. – Here we have taken the Theta as a Column Matrix

Now to build the classifier we will try to minimise the error or difference between the actual Label Value (Dependent Variable) Y and the result from the hypothesis function, which is known as the Cost Function.

To get the minima we will take the First Order Partial Derivative and get the Gradient Function which in Vectorised form will looks like –

$$\text{Gradient} = (H_\theta(X) - Y)\, X = X^T\, (H_\theta(X) - Y)$$

Then we will try to get the Optimum Theta with Gradient Descent which looks like –

Repeat {

$$\theta := \theta - \frac{\alpha}{m} X^T\, (H_\theta(X) - Y)$$

}

,where α is the learning rate and m is the total number of training data.

After several Iteration we will get the θ_{Opt} which will give the best fit and will minimise the error between the hypothesis output and the actual Y value.

So, the output of our Model will look like –

$$H_{\theta_{OPT}} = X\theta_{OPT}$$

Which gives the probability for the independent variables, as it's a binary classifier we will classify using some threshold and compare the probability with the threshold.

Multi Class (One Vs All) Implementation:

As I am implementing algorithm in vectorised form it will work for multiple variable as well.

Till now we have to build our model for Binary Dependent Variables but for multiclass classification. For this we will use the One Vs All Algorithm. We will treat each class out as a binary i.e. for that class only the output is 1 and all others are zero and we will push the data through the model to get the individual probability of each class. Then for a new input we will assign the class which have the highest probability.

Mathematically -

$$h_\theta^i(x) = P(y = i|x, \theta) \quad i = 1,2,3, \ldots . No\ of\ Classes$$

Here P is the Probability for each class i. Here we will train the logistic classifier $h_\theta^i(x)$ for each class i to predict the probability that y = i . For a new input x to make the prediction we will pick the class i that maximises $h_\theta^i(x)$

Design Decision and Specifications :

Here I am implementing the Algorithm using Python Language. As I am implementing the classifier in vectorised form the classifier expects the input data in a matrix Form. Following are the Methods and their respective parameters.

Class : `LogisticRegressor(alpha = 0.5, iterations = 15000, tol=1e-8)`

Parameters:
alpha : learning rate
iterations : Maximum Number of Iterations for the Gradient Descent Iterations
tol : maximum tolerance level for the cost function

Methods:
Build the Model with the Training Data

fit (X,Y) –
Parameters :
X : ***{array-like, sparse matrix}, shape (m, n)***
- Training Feature Vector where m is the number of sample and n is the number of features

Y : ***array-like, shape (m,)***
- Target (Dependent Variables) Vector Relative to X

Returns :
self : object

predict(X) –
Parameters :
X : ***{array-like, sparse matrix}, shape (m, n)***
- Test Feature Vector where m is the number of sample and n is the number of features

Returns :
array, shape(m)
- Predicted Class Label per sample

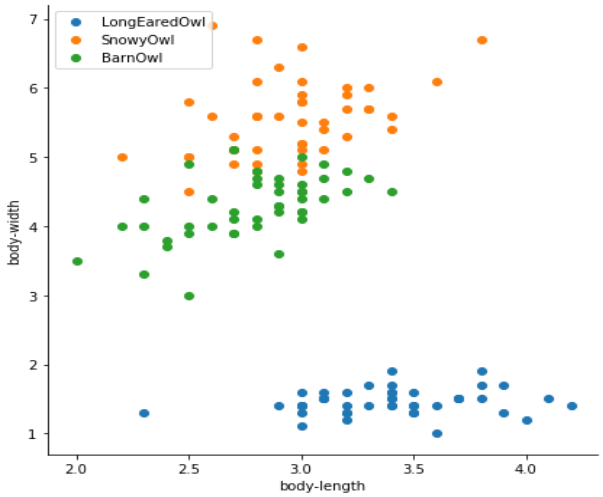
Pre-Requisites:
- Code needs to be run in python 3.5 Or above as few functionalities used is only available in it.
- Before running the Class python library numpy needs to be imported as np

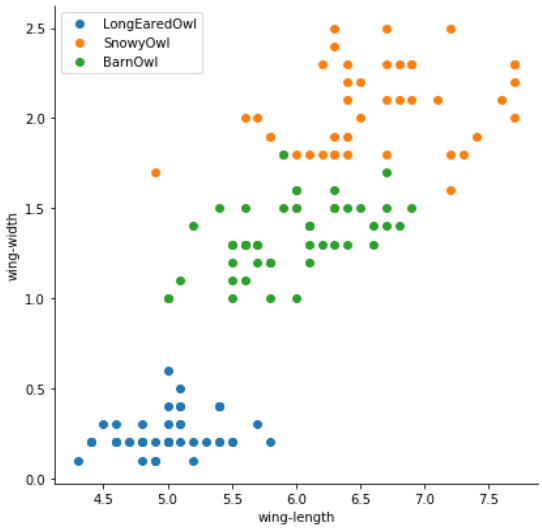
Assumptions :
- The Number of features should always be less than the number of examples
- Here we are assuming that Training Features(X) are independent and the correlation between them is very less
- The Class or the Dependent Variables are discrete
- There should not be many outliers in the features
- For the best result of the classifier the Input Feature Vector Should be Normalised before feeding to the system

Test Results :

Test with the data Provided as 'owls.csv' –

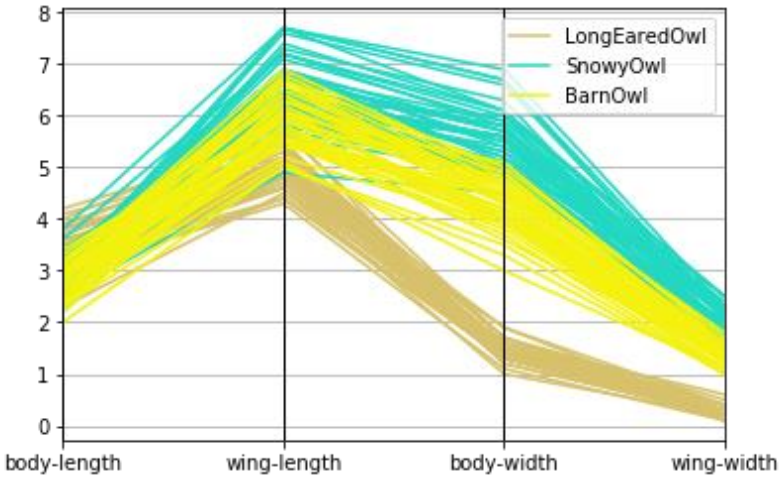
I first inspect the data to check if the data contains any anomaly like outliers and if the features has any correlations of if they are independent of each other, a way is to check scatter plot one feature with respect to another :





So we have found the data is more or less clustered out.

Another way to check the high dimensional multivariate data is to plot through Parallel Co-ordinates method –



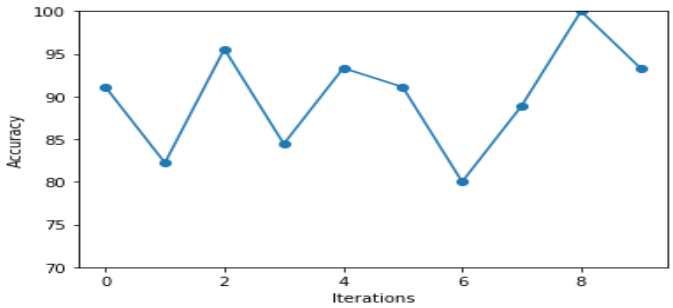
Here also we have found the data is quite well clustered and the Long-Eared-Owl Data is denser across all the attributes.

To Test the Model, I have first divided the data into Feature Vector and the Target Data(dependent variable).

Then Divide it to Train and Test in 2:1 Ratio and train the model with the Train Data.

Then to predict I pass the Test Feature Vector to our model and compare the Predicted Label with the Actual Label to calculate the accuracy.

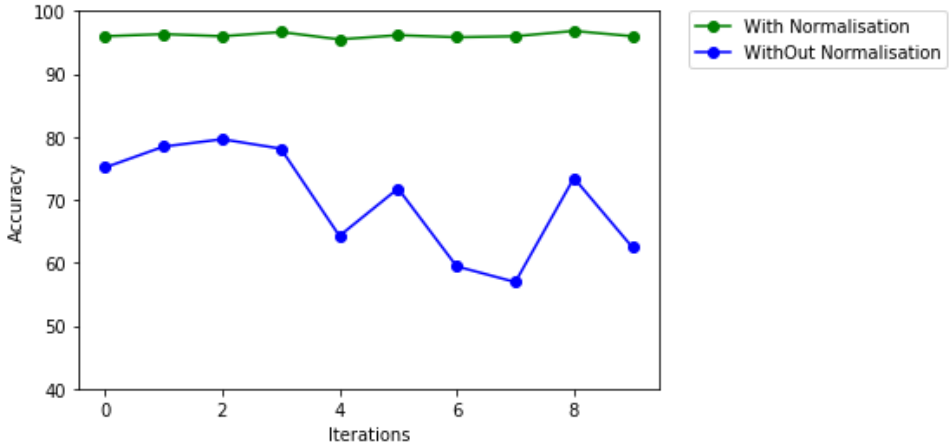
Then I repeat the above process for 10 times to get the accuracies for the 10 iterations –



Test with External Dataset:

I have also tested the same with an external dataset (The Digit Dataset from scikit learn) (Scikit-learn.org, 2018) The dataset has 64 attributes and 10 different classes. I have tested the Classifier with the raw data and with the normalised data separately

Following are the result :



Conclusion and Observation:

From the test executed on different dataset I have found the classifier gives higher accuracy while the feature vectors are normalised and passed to the classifier.

As we have used the vectorised form the same classifier will work for single variable and multivariate dataset.

If we have too many numbers of features the learned hypothesis may overfit the train set but behaves badly in Test cases. In this kind of issue, we can further enhance the cost function by adding a regularization term which will penalise the theta values and reduce the overfit.

References:

1. Medium. (2018). Python Implementation of Andrew Ng's Machine Learning Course (Part 2.1). [online] Available at: https://medium.com/analytics-vidhya/python-implementation-of-andrew-ngs-machine-learning-course-part-2-1-1a666f049ad6 [Accessed 28 Nov. 2018].
2. Ng, A. (2018). CS229 Lecture notes. [online] Cs229.stanford.edu. Available at: http://cs229.stanford.edu/notes/cs229-notes1.pdf [Accessed 28 Nov. 2018].
3. Www2.lawrence.edu. (2018). numpy for Linear Algebra. [online] Available at: http://www2.lawrence.edu/fast/GREGGJ/Python/numpy/numpyLA.html [Accessed 28 Nov. 2018].