Assignment 3 – Machine Learning and Big Data

Introduction

In this homework, I will be coding logistic regression.

Functions will have both unregularised and regularised implementations

With logistic regression, we can solve problems where linear regression fails.

* Because classification is better to find an answer which is a “value”
  + While logistic regression is better to find an answer which is a “class”
* And, if we try to divide the graph into 2 parts (by a threshold), model from linear regression fails
  + Because linear regression always looks to the continous values and tries to classify with continous values

How logistic regression works is like this:

We still have our model

* Differently from linear regression, our aim with model, is not to create “an accurate value”
  + Like predicting house prices… there is an actual price and our model generates an estimated price to the target price
* Our aim is to “classify. In other words, our goal is to create “an accurate seperation of classes”

Model (which generates continous values by itself) is being inputted to a sigmoid function (which creates a classification)

* Sigmoid function puts the values (which model generated) between 0 and 1

Only thing we do is to:

1. Calculating the cost of our model
2. Improving it with gradient descent algorithm

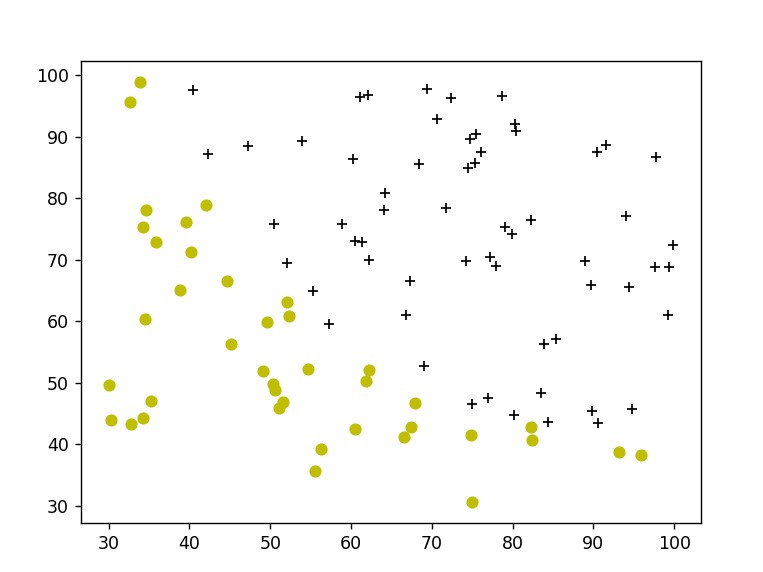
Part A: Logistic Regression

* It is for situations when we need to find a logistic regression model which is linear

Part B: Regularized Logistic Regression

* It is for situations when we need to find a logistic regression model which is not linear

Our initial data is like this:



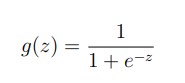
+ represents “admitted students”

o (yellow circle) erpresent “not admitted” students

Part A

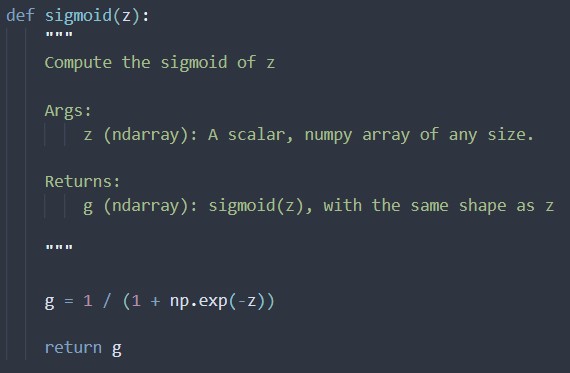
Sigmoid function

Sigmoid function is a function like below…



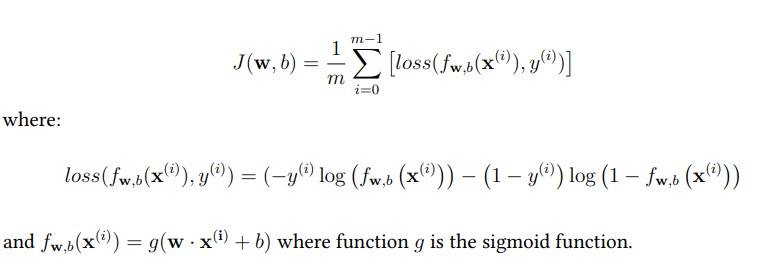
The z input is the results vector which our model obtained

This simple function is just to put the values (which our model found) in an interval between 0 and 1



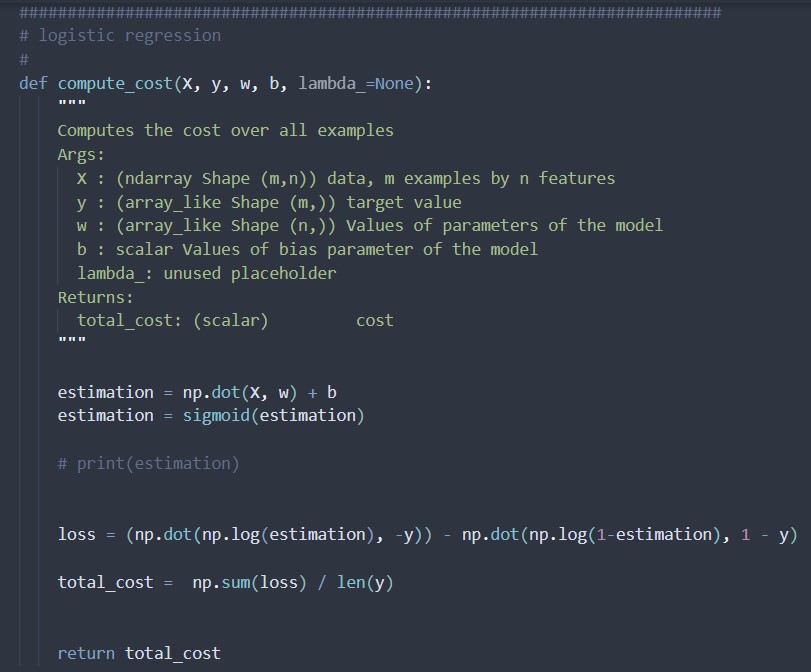
Compute cost

For logistic regression, we use a formula like this below:

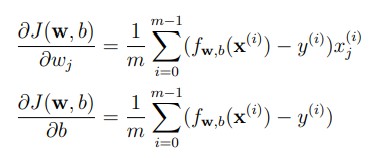


The reason that this formula is different than the “linear function” is because this:

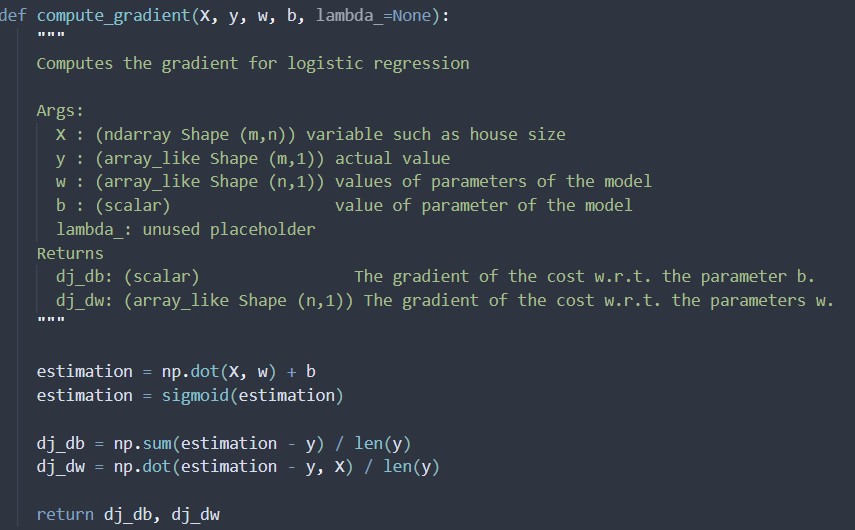
* If we use linear regression (to calculate the cost of our model) formula, we get a wiggly shape which is not convex
  + And since it is not convex, we can’t reach to the global minima
    - since there are multiple minimas and we can’t determine whether the minima we are at is a local minima or a global minima
      * Because, gradient descent function can only understand “whether it has reached to a minima/maxima or not”
      * And it will stop whenever it reaches a minima/maxima
* So, That is why we have a cost function like this
  + With this, we can get a convex shape for our cost function
* How cost function formula works is this:
  + We have our loss function
    - Our target class can be either 0 or 1
      * If our target is 0, left side of the subtraction becomes 0
      * If our target is 1, right side of the subtraction becomes 0
    - We calculate loss, with how far our estimation is from the target value
  + We just sum up all the loss values we got and then take their average



Compute gradient



We take the derivative of our cost function, which would be later used at gradient descent to improve our model



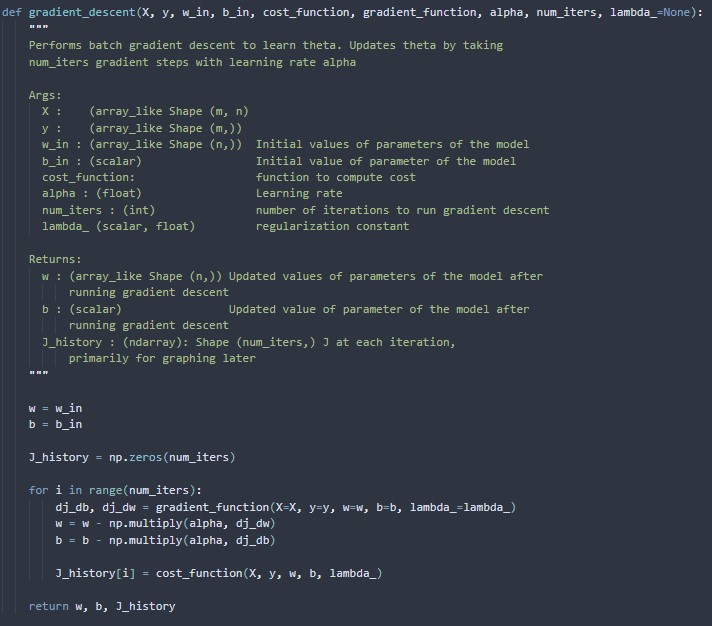
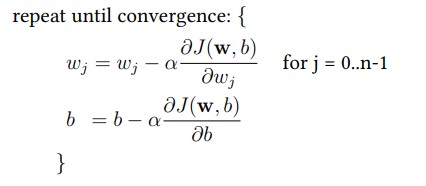
Gradient descent

Here, we update our model, by updating w and b

We take the derivative of our cost function

* Later, we will get closer to the minima
  + How fast and how steady we will get close is related with alpha

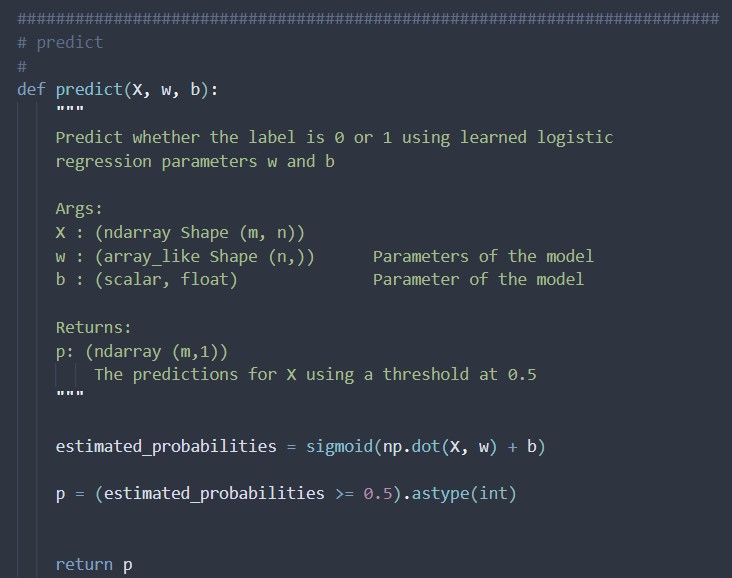
And then (by subtracting) we always get closer to the minima



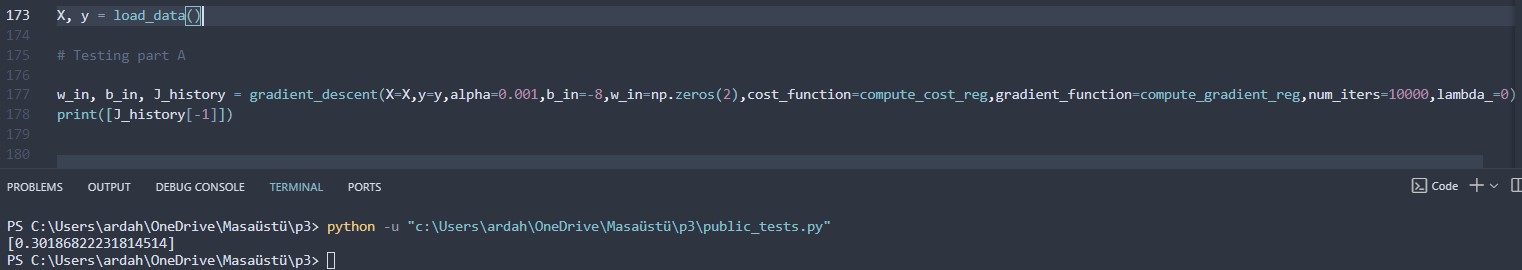
Predict

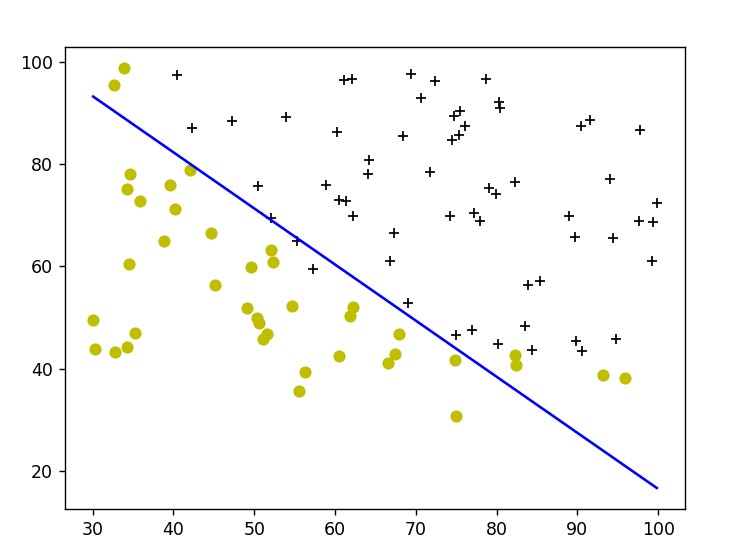
This function is just for testing the accuracy of our model

* If target and result match, then for that instance, result is accurate
  + If both target and result are 0 or 1
* If they do not match, they are not accurate



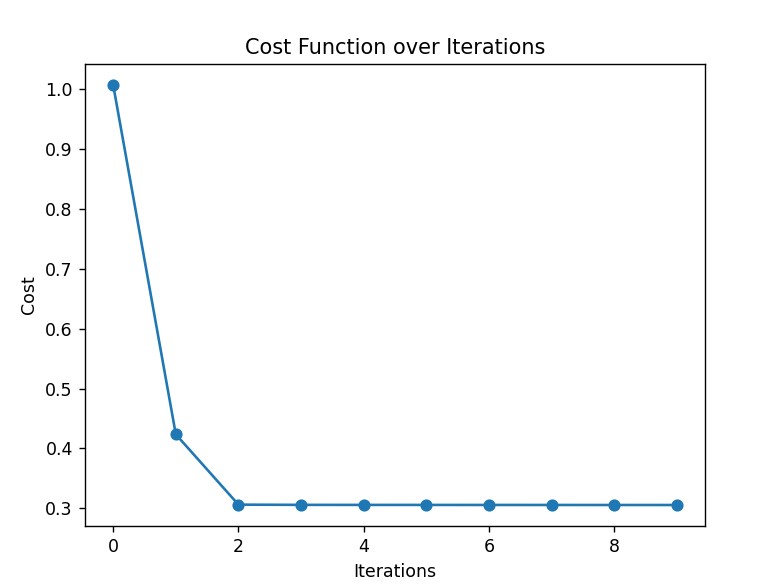
Here is the result I got:

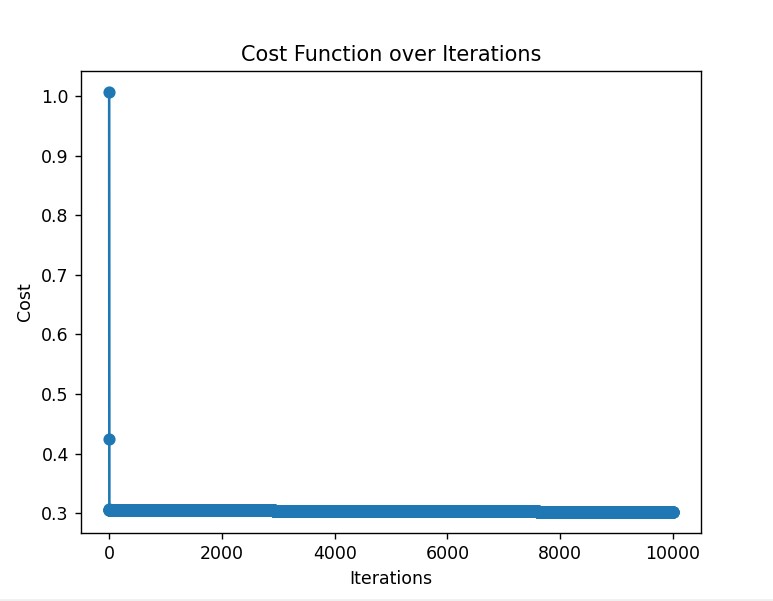




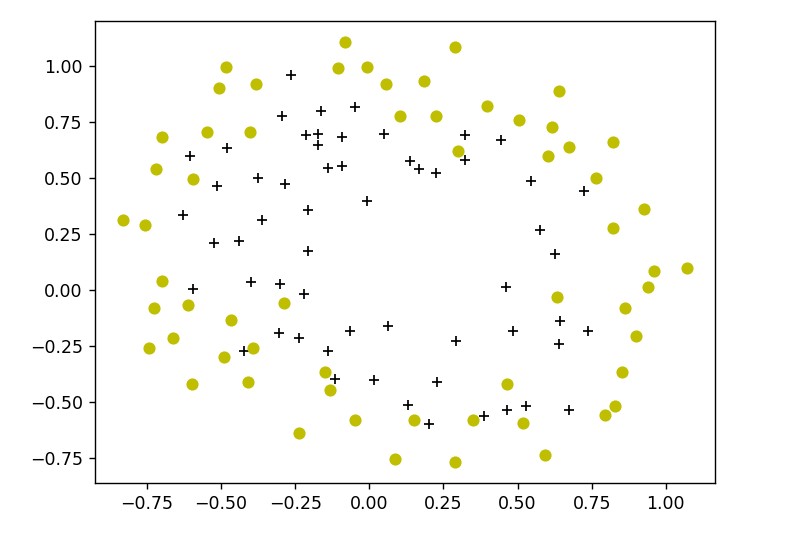
Here are my cost function graphs for part A:

(both graphs correspond to the “cost function of model at part A”)



I had also put the cost function for 10 iterations, because this cost function was converging really quickly (in just 2 iterations haha)  


Part B



+ => Accepted

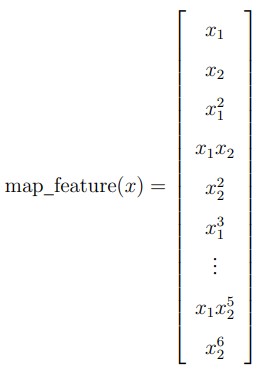
o => Rejected

Feature mapping:

Because that we need to find a non-linear model, we need to do “feature mapping”

For being able to create an accurate function, we need more characteristics and each of these characteristics should have the best weight that they can have

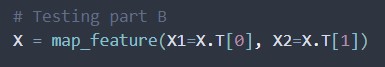
To achieve this, we apply feature mapping



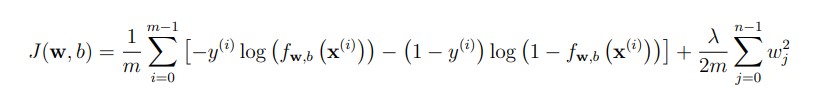
For example, in Part B we have 2 characteristics.

But we try every possible combination (to find the best weight)

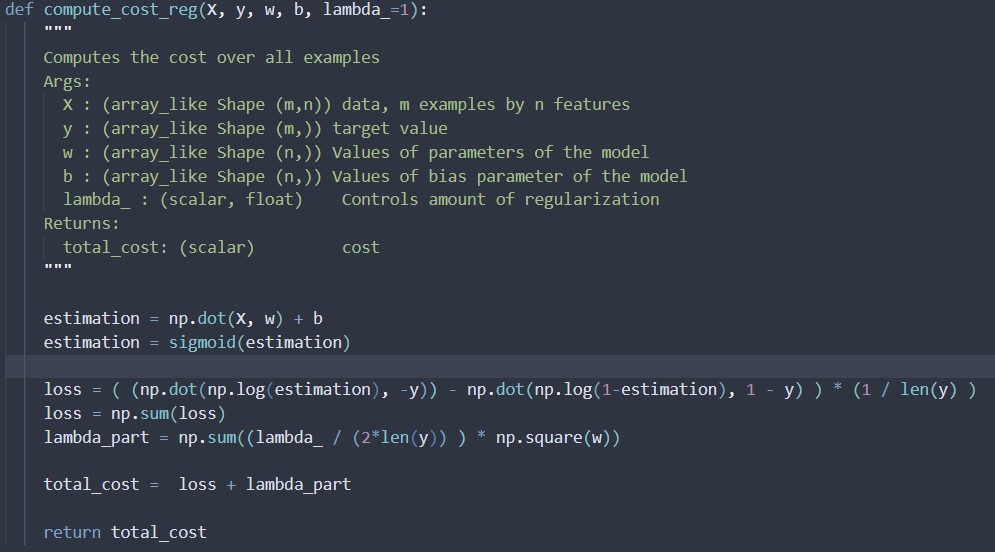
* So, we are adding more characteristics



Compute cost for regularized logistic regression

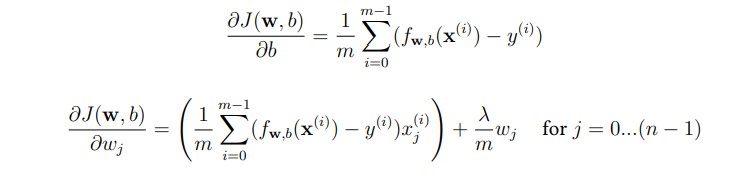


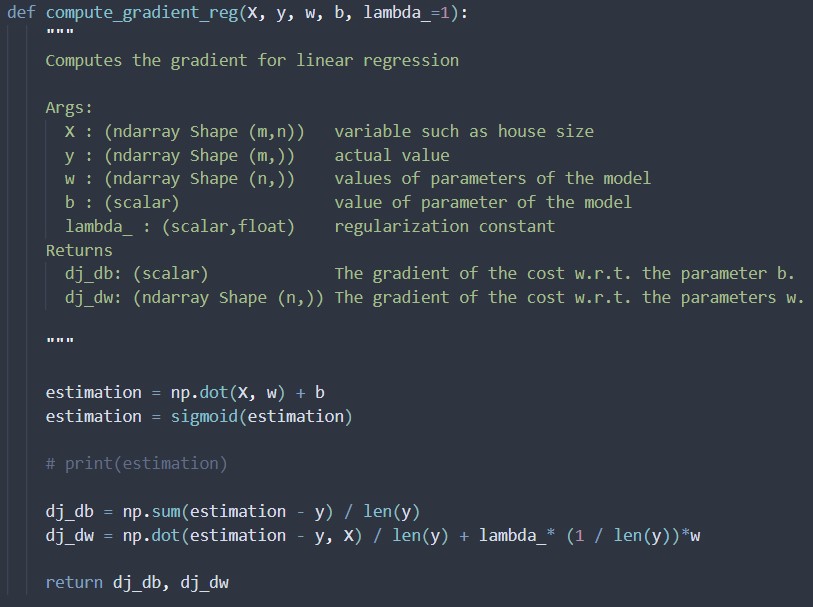
It is basically the same formula except now we use “regularization”



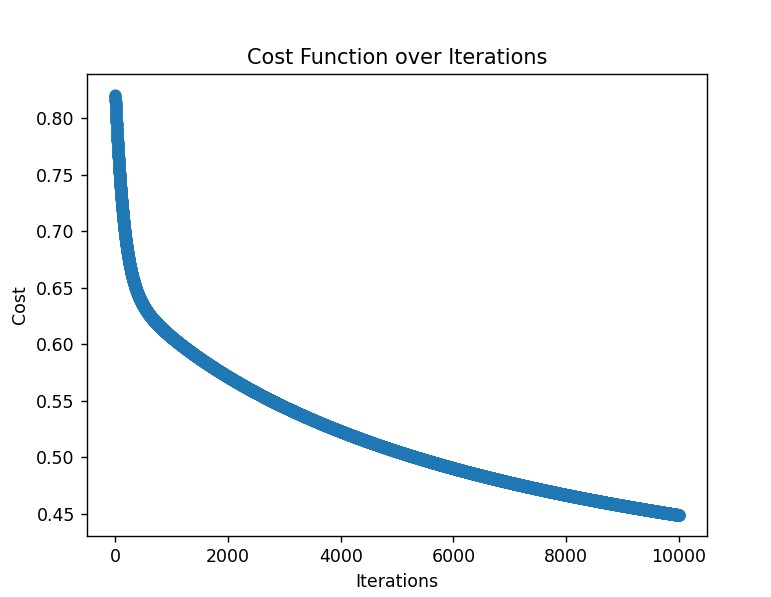
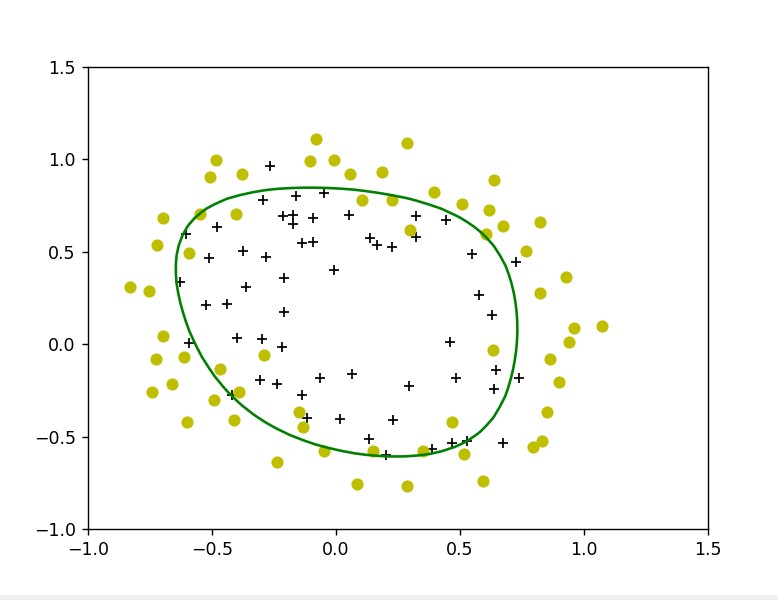
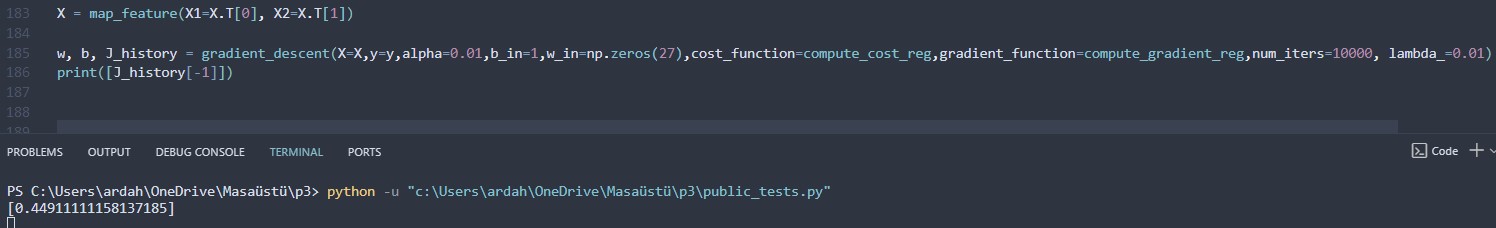
Gradient for regularized logistic regression

Again, same thing except the only difference is just we also regularize…





Here is the graph I got for part B:



Proof that all tests are passed:

