Logistic regression

What's this algorithm?

This algorithm is a statistical analysis method that is used to predict binary outcomes. It is a supervised learning method meaning it must be trained on previously recorded data and for then to be applied to new data. The algorithm works with one or more independent variables and predicts a single binary dependent variable. The algorithm works by using the sigmoid function which has the nice property that it outputs a value between zero and one, this makes it ideal for binary classification as it assumes all rows with output above 0.5 to be 1 and below to be 0. The machine learning algorithm was implemented by using the sigmoid function and then through using gradient descent it continuously updates the weights and biases by a certain learning rate/alpha a certain number of iterations or epochs, once this is done it uses the final weights and biases in predict and dot-multiplies the X array to get the predictions. This algorithm is best suited for binary classification.

Inductive bias

The inductive bias to logistic regression is that there has exist a boundary between the binary outputs, a line that divides the 0's from the 1's. The other thing is that there can only be one line, you can't have two possible lines, then the algorithm would attempt to adhere to both, but would at best only work with one of the lines, or possibly none.

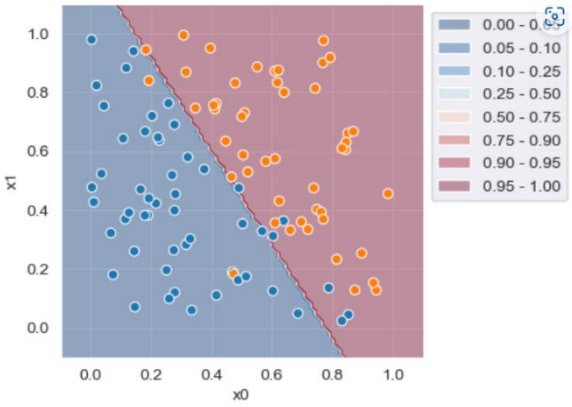


Figure 1

```
Accuracy: 0.920
Cross Entropy: 2.763
```

Results/Pre-processing

The results of the algorithm implementation for the first dataset led to the plot as you see above (Figure 1), which has a line drawn through on a diagonal axis that divides the points relatively well. This had relatively high accuracy, but naturally since there are some points that are out of place or don't fit the norm, the line is not as accurate. For the second dataset, the plot looked a little different (Figure 2). As one can see, there are two groups of blue points separated by a group of orange points. This dataset really accentuates the inductive bias in that there needs to be a simple line to divide the binary output. The initial accuracy was about ~61%, therefore some pre-processing had to be done for the same implementation of logistic regression to work on this dataset. From this dataset, one can draw a line following the function y= (-x), (the blue line in figure 2) and then what was done was to measure the distance from this line to the points and then you can "draw" a distinguishing line that divides the binary outputs. This was done by adding a column in the data that took the absolute value of the sum of x0 and x1, this effectively squeezed all the points towards the black line in figure 2 and then flipped the plot with the absolute value and this made the algorithm work very nicely with around ~98% accuracy.

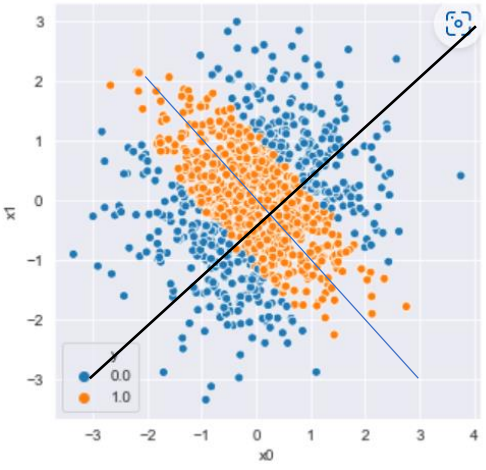
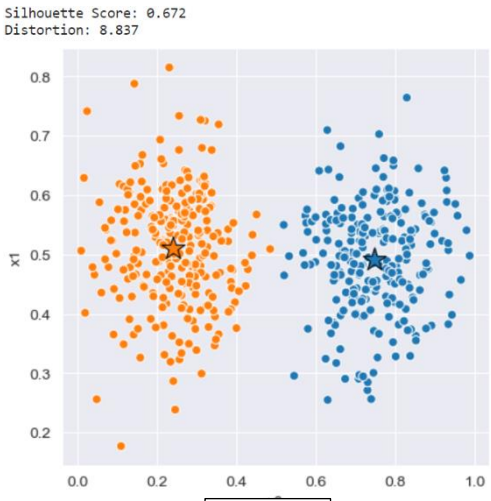


Figure 2

K-means clustering

What's this algorithm?

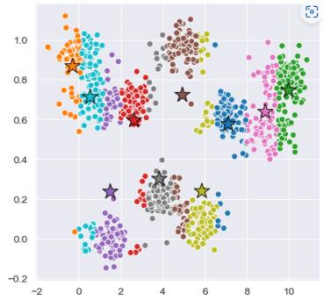
The k-means clustering algorithm is a fairly simple algorithm that is unsupervised and is great for finding groups within unlabelled data. It is also relatively simple to implement and follows four simple steps. The first step is to randomly initialize the centroids, this can be done numerous ways but a classic one is to randomly pick k points from the data and call them the centroids. The next step is to assign every point to a cluster, this is done by calculating the distance, in our case the Euclidean distance, between the points and the centroids and then assigning each point to the cluster of the closest centroid. Then, once each cluster is defined, then we recalculate the centroids so that they are in the middle of the clusters, for example by taking the mean of all the points in any given cluster. And once these new centroids have been defined, we repeat the last two steps, 1. Assigning each point to a cluster and 2. Recalculating the centroids until max iterations are reached or convergence. This means either until the centroids are the same as the previous ones or until they are sufficiently close.



Figur 1

Inductive bias

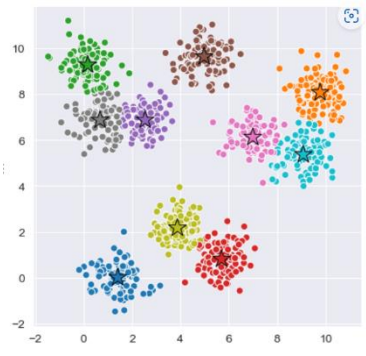
The inductive biases of this algorithm are the initial centroids. Basically, the results can vary depending on which initial centroids are picked, if you pick three points that are very close, then they probably wont split up, or if you pick a point that is completely isolated, it might never move far. Another thing that can be considered an inductive bias is how to determine the number of clusters, in our case it was given and fairly easy to see, but it might not always be.



Figur 2

Results/Pre-processing

For the first dataset, I simply implemented the k-means clustering and picked out two random points from the dataset as the initial centroids and it worked every time. So, the first dataset was very simple and straightforward (See figure 1). The second dataset, however, was not as successful from the get-go. The first thing I noticed was that the data was not scaled properly so the height(1-10) mattered way more than the width(0-1), see figure 2. So, the first thing I did was to feature engineer so that the x0 and x1 were weighted the same. Then I saw huge differences between each run due to the initial centroids being completely random, so I implemented the k-means++ initialization algorithm so that the starting points were spread out more. This was done by picking one random centroid and then for each new centroid pick one that is far away from the other centroids, this yielded very accurate results almost all the time. See figure 3.



Figur 3