CMPE462 Assignment 2

Meriç Üngör - 2017400156

Part 1:

In part 1 of the assignment, we were asked to implement logistic regression with full batch gradient descent and stochastic gradient descent. For stochastic gradient descent, since it wasn't specified, I split my data into 10 chunks and updated my weights after going through each chunk. Also we were asked to try out 3 different eta values and use 5-fold cross validation to see which one best fitted our model. The eta values I tried were 0.05, 0.2 and 0.8.

I implemented a combination of two different methods to realize when my algorithm needs to stop and yield a result. The main part of my stopping condition is that at each iteration, I take the l2-norm of my gradient and if it is below some hardcoded value, it means we are very close to a local minimum in our graph so we can return the last updated weights. Also I keep track of the iterations as well and stop the algorithm when it reaches a certain value. This addition prevents the algorithm from getting stuck in a loop.

Firstly, I executed cross validation for each of the eta values to come to a decision. The output log of the cross validation for full batch gradient descent with different eta values;

```
Cross validation on full batch gradient descent with eta 0.05:
Run 0: time taken 36.203235149383545, iterations: 10000, loss on validation: 3.5429726064300615, accuracy on validation: 0.963855421686747
Run 1: time taken 9.925652980804443, iterations: 2643, loss on validation: 7.682391059537611, accuracy on validation: 0.9759036144578314
Run 2: time taken 10.390380382537842, iterations: 2767, loss on validation: 18.825217034623275, accuracy on validation: 0.927710843373494
Run 3: time taken 11.986191511154175, iterations: 3246, loss on validation: 20.213670581099148, accuracy on validation: 0.927710843373494
Run 4: time taken 12.150766134262085, iterations: 3255, loss on validation: 7.063115194624914, accuracy on validation: 0.963855421686747
For eta 0.05, average time taken = 16.13124523162842, average iteration count = 4382.2, average loss = 11.465473295263001, average accuracy = 0.9518072289156627
```

```
Cross validation on full batch gradient descent with eta 0.2:
Run 0: time taken 35.546849966049194, iterations: 10000, loss on validation: 15.259526711314901, accuracy on validation: 0.9518072289156626
Run 1: time taken 9.855527400970459, iterations: 2657, loss on validation: 31.366669687938238, accuracy on validation: 0.9759036144578314
Run 2: time taken 10.831306219100952, iterations: 2906, loss on validation: 75.63440498856464, accuracy on validation: 0.927710843373494
Run 3: time taken 11.869563579559326, iterations: 3236, loss on validation: 79.42201044297208, accuracy on validation: 0.9397590361445783
Run 4: time taken 11.1198890209198, iterations: 2998, loss on validation: 32.78298155021142, accuracy on validation: 0.963855421686747
For eta 0.2, average time taken = 15.844627237319946, average iteration count = 4359.4, average loss = 46.89311867620026, average accuracy = 0.9518072289156627
```

```
Cross validation on full batch gradient descent with eta 0.8:
Run 0: time taken 34.95298480987549, iterations: 10000, loss on validation: 174.17647304220463, accuracy on validation: 0.9397590361445783
Run 1: time taken 7.75009298324585, iterations: 2114, loss on validation: 175.12000786930417, accuracy on validation: 0.927710843373494
Run 2: time taken 18.653603553771973, iterations: 5257, loss on validation: 207.7121810537269, accuracy on validation: 0.927710843373494
Run 3: time taken 12.74912166595459, iterations: 3401, loss on validation: 308.9429606287208, accuracy on validation: 0.9397590361445783
Run 4: time taken 12.628409624099731, iterations: 3133, loss on validation: 116.60176241910699, accuracy on validation: 0.963855421686747
For eta 0.8, average time taken = 17.346842527389526, average iteration count = 4781.0, average loss = 196.51067700261268, average accuracy = 0.9397590361445782
```

As we can see from the results, although they are pretty close in terms of average time taken and average iteration count, an eta of 0.8 has an advantage over an eta of 0.05 and 0.2. However in terms of average loss and accuracy, eta 0.05 has a

much bigger advantage over the other options. So I would sacrifice a little speed to gain a lot of accuracy and choose an eta of 0.05 to use with my model.

The output log of the cross validation for stochastic gradient descent:

```
Cross validation on stochastic gradient descent with eta 0.05:
Run 0: time taken: 3.427798271179199, iterations: 183, loss on validation: 93.83510843673479, accuracy on validation:0.927710843373494
Run 1: time taken: 1.705930471420288, iterations: 93, loss on validation: 197.55394877240968, accuracy on validation:0.8674698795180723
Run 2: time taken: 5.873953104019165, iterations: 321, loss on validation: 29.49726175875312, accuracy on validation:0.9518072289156626
Run 3: time taken: 0.9988296031951904, iterations: 55, loss on validation: 162.75200370314252, accuracy on validation:0.8554216867469879
Run 4: time taken: 2.0610909461975098, iterations: 113, loss on validation: 45.8192591904889, accuracy on validation:0.9036144578313253
For eta 0.05, average time taken = 2.8135204792022703, average iteration count = 153.0, average loss = 105.8915163723058, average accuracy = 0.9012048192771085
```

```
Cross validation on stochastic gradient descent with eta 0.2:
Run 0: time taken: 2.018573045730591, iterations: 110, loss on validation: 574.8535993284603, accuracy on validation:0.9036144578313253
Run 1: time taken: 3.627223014831543, iterations: 179, loss on validation: 744.5116853666556, accuracy on validation:0.8674698795180723
Run 2: time taken: 7.2392799854278564, iterations: 356, loss on validation: 93.04109441039544, accuracy on validation:0.9518072289156626
Run 3: time taken: 1.6631755828857422, iterations: 81, loss on validation: 560.0032708423535, accuracy on validation:0.8554216867469879
Run 4: time taken: 2.2268083095550537, iterations: 110, loss on validation: 220.94559139877157, accuracy on validation:0.891566265060241
For eta 0.2, average time taken = 3.355011987686157, average iteration count = 167.2, average loss = 438.6710482693273, average accuracy = 0.8939759036144579
```

```
Cross validation on stochastic gradient descent with eta 0.8:
Run 0: time taken: 3.21313214302063, iterations: 175, loss on validation: 1485.8576604245184, accuracy on validation:0.927710843373494
Run 1: time taken: 1.5077412128448486, iterations: 84, loss on validation: 3534.973466048024, accuracy on validation:0.8674698795180723
Run 2: time taken: 6.9665281772613525, iterations: 389, loss on validation: 351.23448358547904, accuracy on validation:0.9397590361445783
Run 3: time taken: 2.0154502391815186, iterations: 112, loss on validation: 2221.861843478746, accuracy on validation:0.8674698795180723
Run 4: time taken: 4.730157375335693, iterations: 264, loss on validation: 140.07457971153042, accuracy on validation:0.9518072289156626
For eta 0.8, average time taken = 3.6866018295288088, average iteration count = 204.8, average loss = 1546.8004066496596, average accuracy = 0.910843373493976
```

As we can see, in terms of all average time taken, average iteration count and average loss, eta 0.05 has a clear advantage over eta 0.2 and 0.8. 0.8 is slightly better than 0.2 in terms of accuracy, but it is not enough to look past other criteria. This advantage of 0.2 might be because when we choose higher values for our eta, we might start skipping the local minima and waste some of our time jumping back and forth. Also it makes sense that even though eta 0.2 and 0.8 was faster in full batch gradient descent, eta 0.05 is faster in stochastic gradient descent because in stochastic gradient descent, we don't have access to all of the data at the same time. So at each iteration, we have a larger chance of moving to a bit of a false direction. So a smaller eta would mean smaller mistakes that we can undo more easily. It is almost like taking smaller and more careful steps in rocky terrain compared to taking larger steps in walking on a regular road because we have a larger chance of tripping and falling.

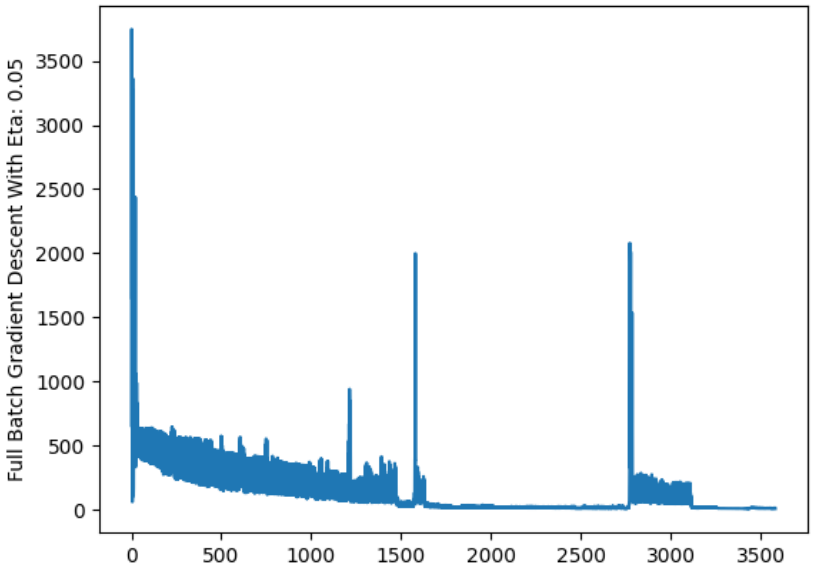
The reason why our average loss increases as eta increases in both full batch and stochastic gradient descent can be because of overfitting. With a larger eta, we always update our weights with gradients with larger absolute values. So it makes sense that the absolute values for our final weights would be larger as well. Although it works well on our training data, this would generally mean our model is more overfitted and more error prone to test data.

Finally, the reason why it takes less time to execute stochastic gradient descent compared to full batch gradient descent is that on stochastic gradient descent, even

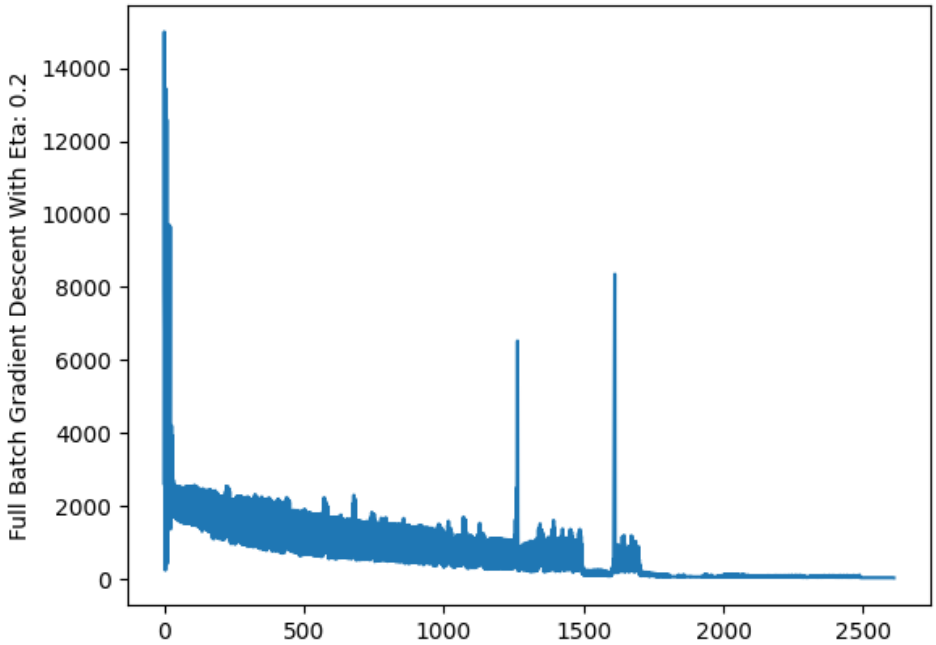
though each of our steps are a bit more deviated from the optimal one, we take 10 steps for every traverse over our dataset compared to the one on full batch. This means every time we traverse over our entire dataset, we take a larger step towards the local minima.

After cross validation, I executed both versions of logistic regression with all the eta values normally to plot their loss over iterations graph and to formally measure time it takes to execute the algorithm. The results:

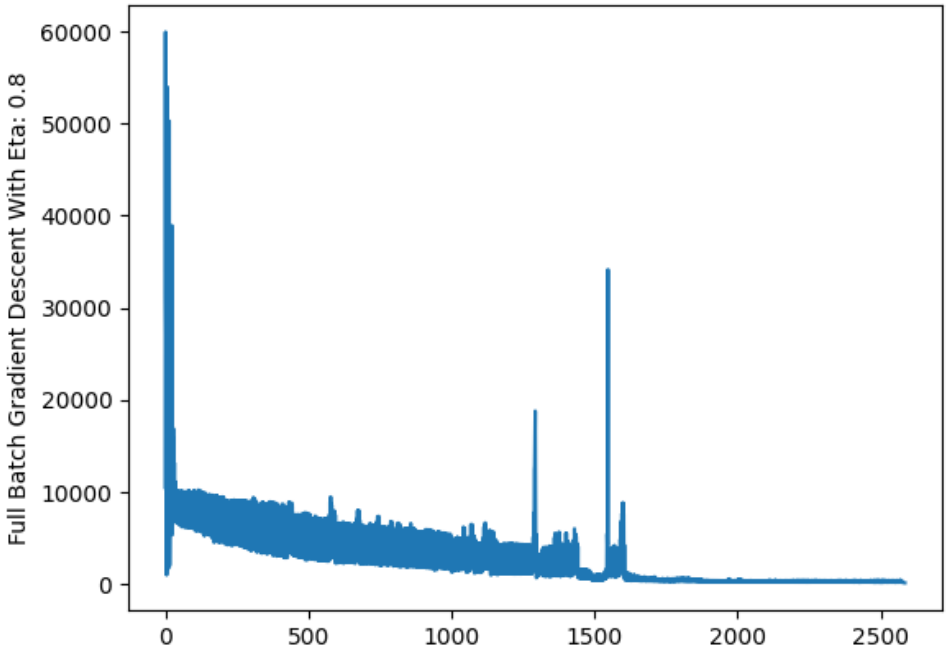
For full batch gradient descent with eta = 0.05;
Time taken = 15.677326679229736



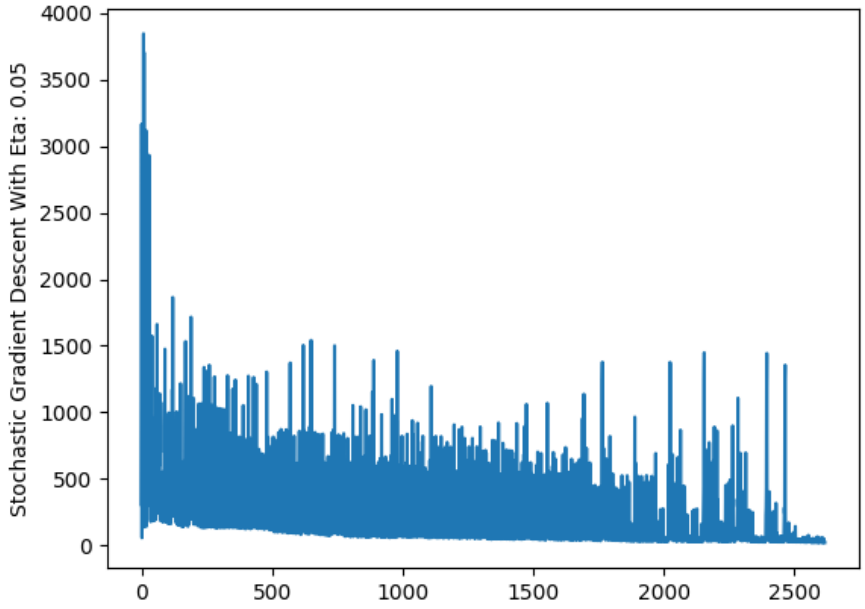
For full batch gradient descent with eta = 0.2;
Time taken = 11.678223133087158



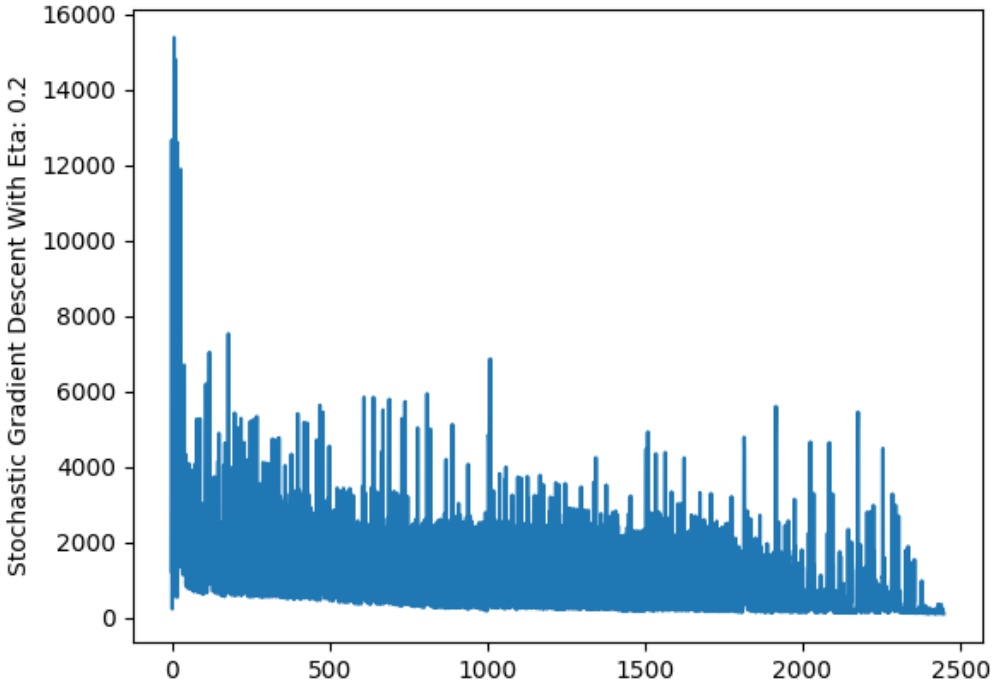
For full batch gradient descent with eta = 0.8:
Time taken = 11.393417358398438



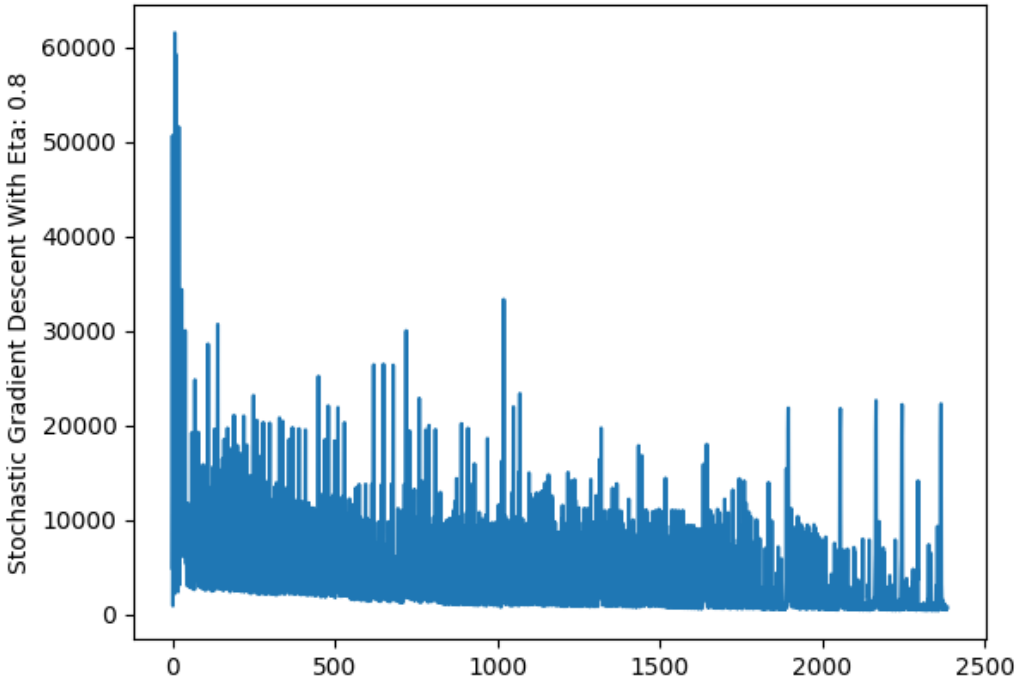
For stochastic gradient descent with eta = 0.05:
Time taken = 5.68282413482666



For stochastic gradient descent with eta = 0.2:
Time taken = 5.389808416366577



For stochastic gradient descent with eta = 0.8:
Time taken = 5.336887836456299



Part 2:

$$P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}$$

$$P(Y|GiveBirth = yes,\ CanFly = no,\ LiveInWater = yes,\ HaveLegs = no)$$
$$= \frac{P(yes, no, yes, no \mid Class) * P(Class)}{P(yes, no, yes, no)}$$

Since we are only comparing the probabilities for different Class values and their denominator is the same, we can eliminate the denominator for simplicity. Also If we assume conditional probability as well, the equation becomes;

$$= P(GiveBirth = yes \mid Class) * P(CanFly = no \mid Class) * P(LiveInWater = yes \mid Class) * P(HaveLegs = no \mid Class) * P(Class)$$

For class = mammals;

$$P(Class = mammals \mid X) = \frac{6}{7} * \frac{6}{7} * \frac{2}{7} * \frac{2}{7} * \frac{7}{20} = 0.020$$

For class = non-mammals;

$$P(Class = non - mammals \mid X) = \frac{1}{13} * \frac{10}{13} * \frac{3}{13} * \frac{4}{13} * \frac{13}{20} = 0.0$$

So, we can say that the probability for the class of last row being a mammal is bigger than the probability of the last row being a non-mammal. So we classify the last row as **mammal**