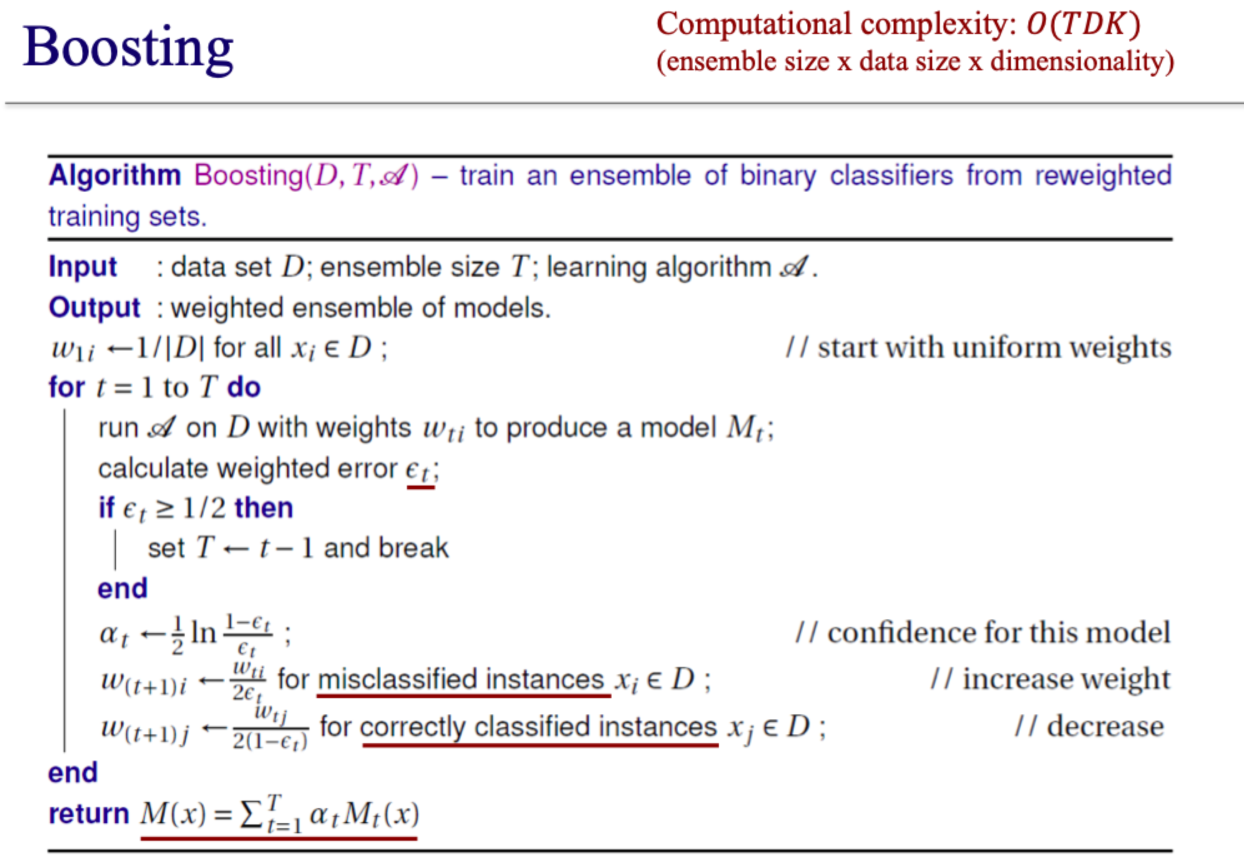
user name: lkim17

1. part 1: What did I do
    a. followed the below pseudocode



Boosting

Computational complexity: $O(TDK)$
(ensemble size x data size x dimensionality)

Algorithm Boosting($D, T, \mathscr{A}$) – train an ensemble of binary classifiers from reweighted training sets.

```
Input  : data set $D$; ensemble size $T$; learning algorithm $\mathscr{A}$.
Output : weighted ensemble of models.
$w_{1i} \leftarrow 1/|D|$ for all $x_i \in D$ ;            // start with uniform weights
for $t = 1$ to $T$ do
    run $\mathscr{A}$ on $D$ with weights $w_{ti}$ to produce a model $M_t$;
    calculate weighted error $\epsilon_t$;
    if $\epsilon_t \geq 1/2$ then
        set $T \leftarrow t - 1$ and break
    end
    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$ ;            // confidence for this model
    $w_{(t+1)i} \leftarrow \frac{w_{ti}}{2\epsilon_t}$ for misclassified instances $x_i \in D$ ;    // increase weight
    $w_{(t+1)j} \leftarrow \frac{w_{tj}}{2(1-\epsilon_t)}$ for correctly classified instances $x_j \in D$ ;    // decrease
end
return $M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$
```

    b. created the matrix of weights (different ensemble size, t, and line of X input as index) and Model (different ensemble size, t, and dimension of X input as index)
    c. class1 will have all the points that were classified as 1 and class 2 will have all the points that were classified as -1
    d. For each class 1 and 2, find the class_exemplar by adding all the multiplication of weights associated with point x and point x for each t
    e. using the formula of (p-n)x - (p-n)(p+n)/2 = 0, found the perpendicular bisector line of two points p and n. This would be the model
    f. count the number of errors by counting the number that M(x) > 0 but are labeled as -1 or M(x) < 0 but are labeled as 1
    g. divide those error count by the number of points in input X. If it's greater or equal to 0.5, break else find the confidence for the model and decrease or increase the weight
    h. add all the confidence with the associate model for each t and return it
2. part2: How is your code's performance locally
    a. dataset1.txt

```
(base) gim-ijin-ui-MacBookAir:submission peach$ python3
local_evaluation.py
```

```
Iteration  1 :
Error =  0.1875
Alpha =  0.7331685343967135
Factor to increase weights =  2.6666666666666665
Factor to decrease weights =  0.6153846153846154
Iteration  2 :
Error =  0.2375
Alpha =  0.5832174425034353
Factor to increase weights =  2.1052631578947367
Factor to decrease weights =  0.6557377049180328
Iteration  3 :
Error =  0.4625
Alpha =  0.07514110152466895
Factor to increase weights =  1.081081081081081
Factor to decrease weights =  0.9302325581395349
Iteration  4 :
Error =  0.5


Testing:
False positives:  13
False negatives:  0
Error rate:  0.16249999999999998
```

3. Report part3: Compare this and linear classifier:
    a. different values of T
        i. T = 1, Accuracy: 0.825, F-measure: 0.7941176470588236, Precision: 0.9642857142857143, Recall: 0.675, Final_Score: 80.95588235294119
        ii. T = 2 , Accuracy: 0.8625, F-measure: 0.8791208791208792, Precision: 0.7843137254901961, Recall: 1.0, Final_Score: 87.08104395604397
        iii. T = 3 , Accuracy: 0.8375, F-measure: 0.8602150537634409, Precision: 0.7547169811320755, Recall: 1.0, Final_Score: 84.88575268817203
        iv. T = 4, Accuracy: 0.8375, F-measure: 0.8602150537634409, Precision: 0.7547169811320755, Recall: 1.0, Final_Score: 84.88575268817203
        v. From T = 3, accuracy stays the same
        vi. The accuracy is highest as T = 2 for this data set
    b. The original linear classifier gives me a higher accuracy than using boosting classifier.