CSE343/ECE343: Machine Learning **Name:** *Srishti Singh*
Assignment-2 Decision Trees, Random Forest and Perceptron
October 2022 **rollno:** *2020409*

1. Section A (Theoretical)

(a) Draw a decision tree. Illustrate all probabilities and outcome values.

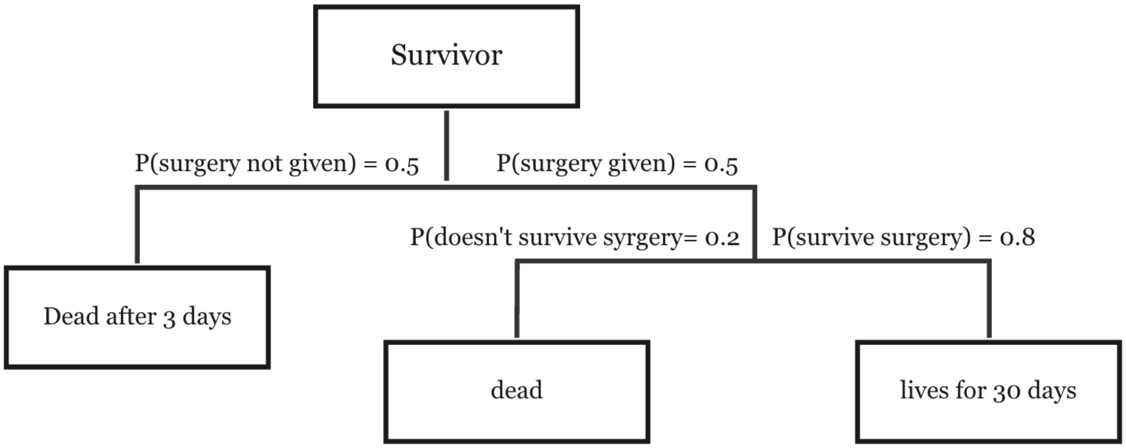
Assumption 1- since we do not the probability of whether the surgery is given or not so I am assuming it to be 50-50. Therefore using that we will have -

- P(survives surgery) = 0.8
- P(doesn't survive surgery) = 0.2
- P(surgery given) = 0.5
- P(surgery not given) = 0.5

Post surgery the patient either lives or 30 days or 0 day(given) Using all that we can make three cases -

(a) P(person lives for 30 days after surgery) = 0.5 * 0.8 = 0.4
(b) P(person dies after surgery) = 0.5 * 0.2 = 0.1
(c) P(person doesn't receive surgery) = 0.5

Decision tree made using all this will be -



(b) L(x) denotes the patient's living function, ... L(3)=0.8

Given -

- L(x) is patients living function
- L(30) = 1
- L(0) = 1
- L(3) = 8

Objective - Find minimum L(3) so that its equal to E(x) of LIVING function it the surgery is performed.
L(3) = E(L(x) — surgery performed)
= L(30) x 0.8 + L(0) x 0.2
= 1 x 0.8 + 0 x 0.2
= 0.8
Hence L(3) $>=$ **0.8**

(c) The doctor of Melon city finds a low-risk test procedure...

Given -

- $P(positive|survived) = 0.95$
- $P(positive|notsurvive) = 0.05$

Using Bayesian theorem $- P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

$$P(survive|positive) = \frac{P(positive|survived)P(survive)}{P(positive)}$$

$$P(positive) = P(positive|survived)*P(survived)+P(positive|notsurvive)*P(notsurvive)$$

$$P(positive) = P(positive) * [P(survive|positive + P(notsurvive|positive)]$$

$$P(positive) = P(positive) * [1]$$

Hence -

$$P(positive) = 0.95 * 0.8 + 0.05 * 0.2 = \mathbf{0.77}$$

Ans -

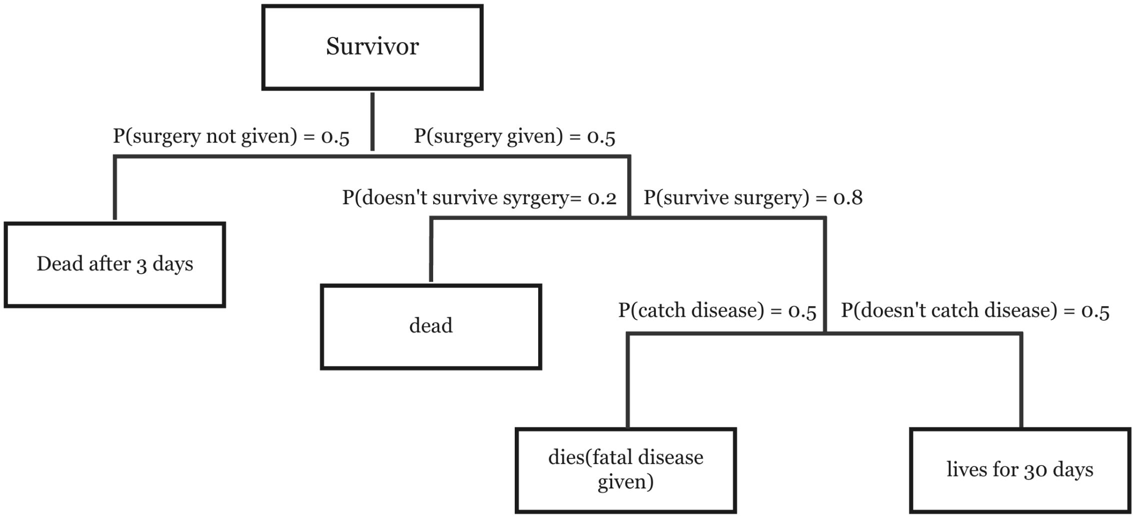
$$P(survive|positive) = \frac{0.95*0.8}{0.77} = \mathbf{0.987}$$

(d) Should the surgery be performed if the result of the test is positive?

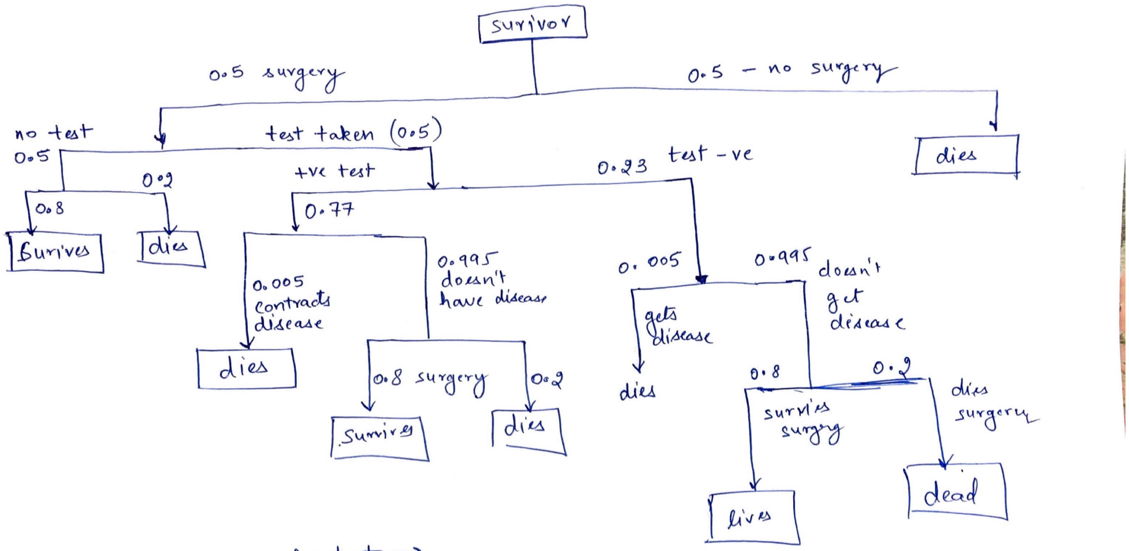
Yes
The probability of patient surviving the surgery if the test is positive is 0.987 (from part c). which is quite high. Therefore the surgery should be performed.

(e) Recently, it has come to the chief surgeon's notice that there is a ...

Assumption 1- since we are not given the probability of getting the disease I am assuming it to be 50-50.
P(getting disease and surviving) = 0.5*0.8*0.5 = 0.2
P(getting surgery no disease) = 0.5*0.8*0.5 = 0.2



(f) Suppose that the probability of contracting the new disease …



Survival with prior test →

$= 0.23 \times 0.995 \times 0.8 + 0.77 \times 0.995 \times 0.8$

$= 0.796$

Survival without test $= 0.8$

$0.8 > 0.796$

therefore Better to conduct test.

Section B

1. (2 marks) Create a dataset (with 10,000 points) using the circle equation

DONE IN CODE

2. Plot the dataset on a 2-D plot such that all the information related to the dataset i.e., x, y, and labels can be inferred from the 2d plot itself with add noise argument set to True and False.

Image of no noise data

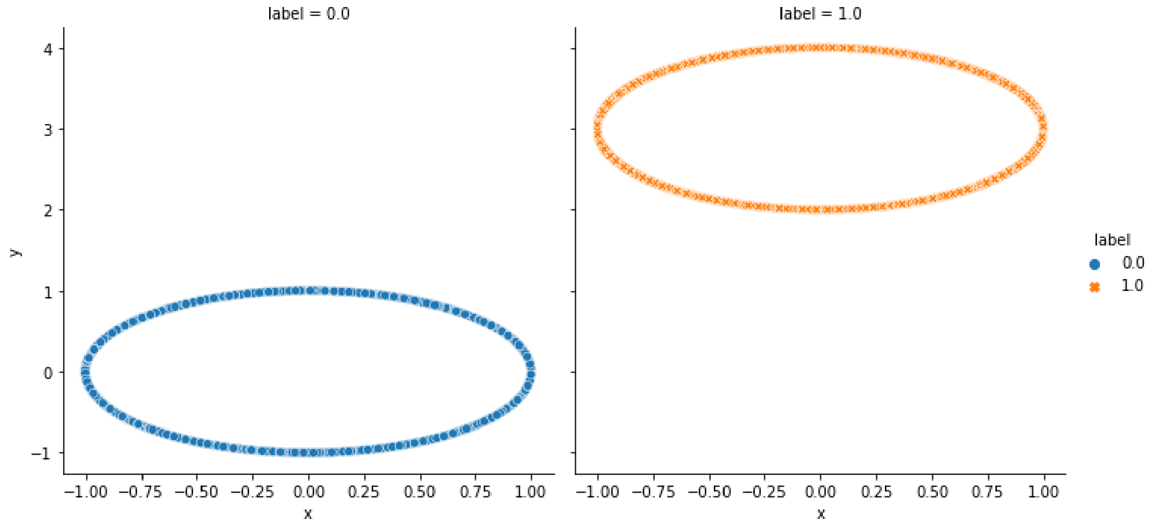
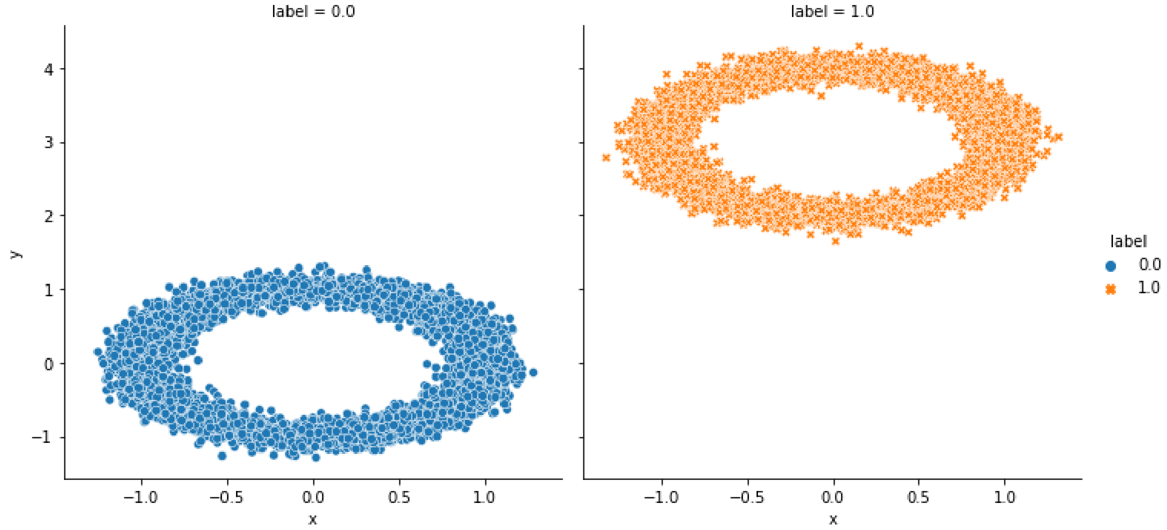
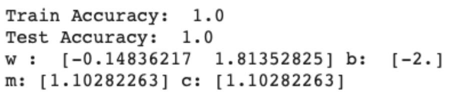


Image of noisy data

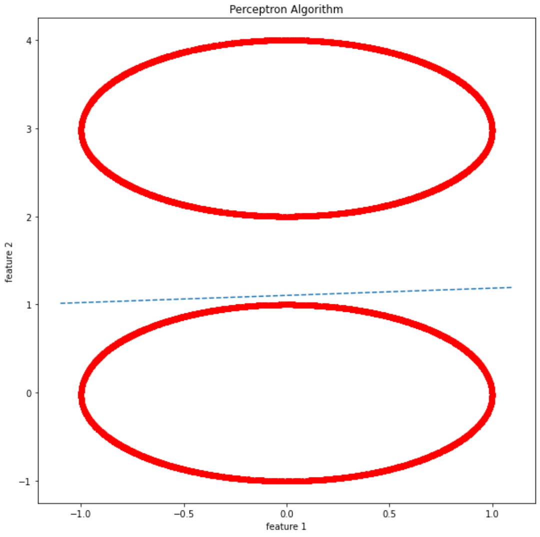


3. (5 marks) Train a classifier using the Perceptron training algorithm (PTA),

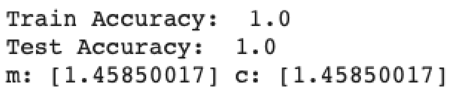
TRAINING NO NOISE DATA



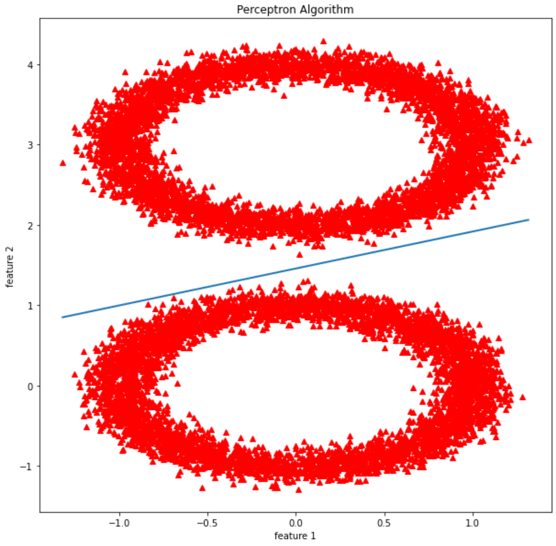
```
Train Accuracy:  1.0
Test Accuracy:  1.0
w :  [-0.14836217  1.81352825] b:  [-2.]
m: [1.10282263] c: [1.10282263]
```



TRAINING NOISY DATA

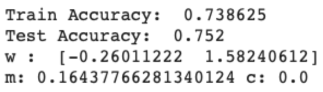


```
Train Accuracy:  1.0
Test Accuracy:  1.0
m: [1.45850017] c: [1.45850017]
```

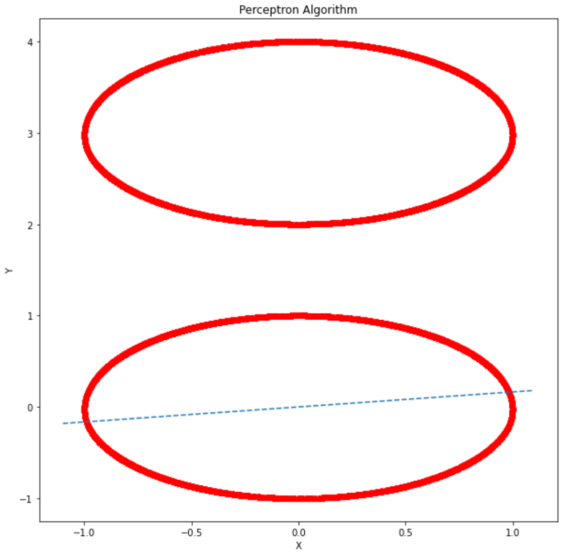


4. (3 marks) Train another classifier using the perceptron training algorithm (PTA)

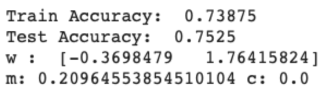
TRAINING NO NOISE DATA



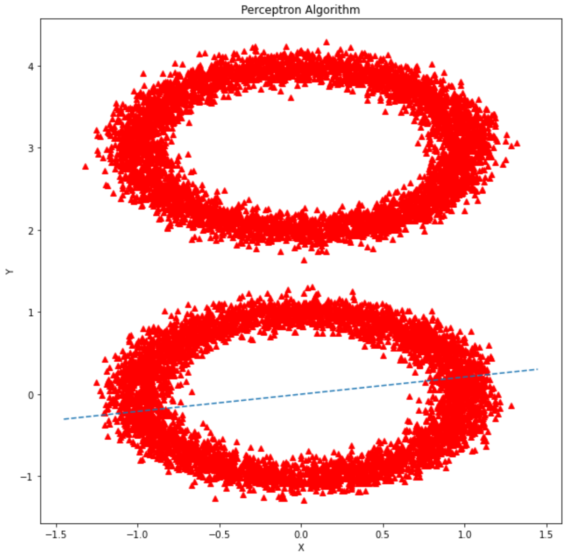
```
Train Accuracy:  0.738625
Test Accuracy:  0.752
w :  [-0.26011222  1.58240612]
m: 0.16437766281340124 c: 0.0
```



TRAINING NOISY DATA



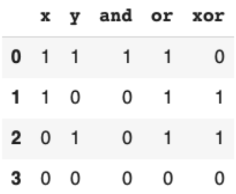
```
Train Accuracy:  0.73875
Test Accuracy:  0.7525
w :  [-0.3698479   1.76415824]
m: 0.20964553854510104 c: 0.0
```



The line that we have plotted is not the decision boundary. It is just the line which is trying to minimize the error. When we have the bias as 0, our data assumes that all the output depends on the current input and is a function(w1x1 + w2x2 + w3x3 … etc.) of these parameters only. When we add a biased term, then we are effectively adding another dimension to our function. Moreover, bias value helps us to shift the activation function to the left or right. For example, if we were using sigmoid as our activation function, the bias determines the curve's steepness. Another reason can be that if we have a bias as zero, then we are assuming that the decision line will always go through (0,0), which is clearly not the case with our data, as the circle with label 0 is present in all four quadrants.

5. (3 marks) Create a dataset (with 4 points) using the XOR, AND, and OR properties….

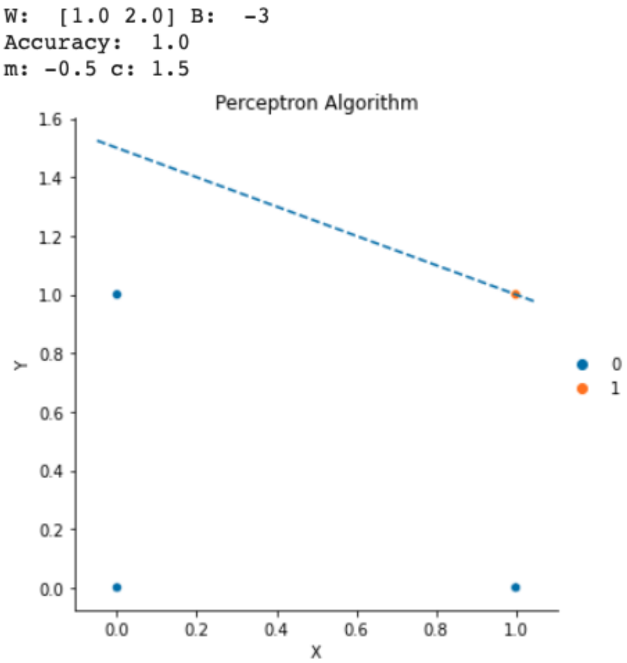
Dataset created



	x	y	and	or	xor
0	1	1	1	1	0
1	1	0	0	1	1
2	0	1	0	1	1
3	0	0	0	0	0

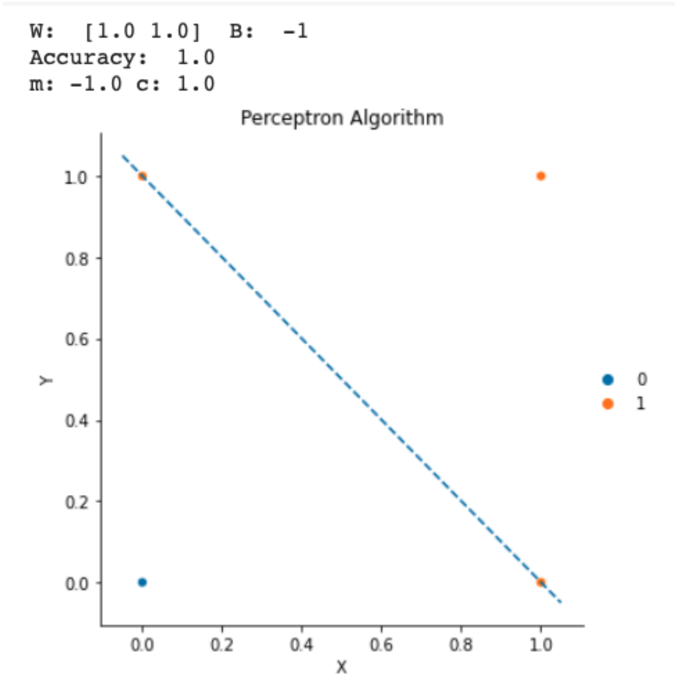
FLEXIBLE BIAS

AND



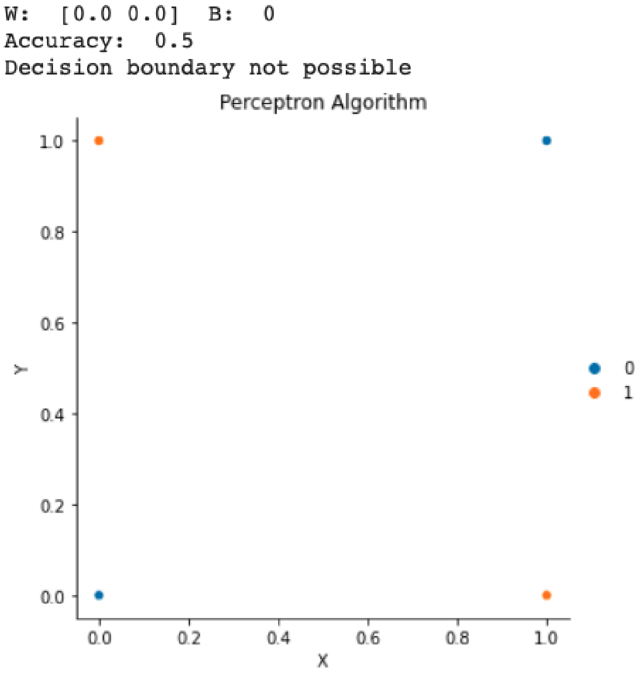
```
W:  [1.0 2.0] B:  -3
Accuracy:  1.0
m: -0.5 c: 1.5
```

OR



```
W:  [1.0 1.0]  B:  -1
Accuracy:  1.0
m: -1.0 c: 1.0
```

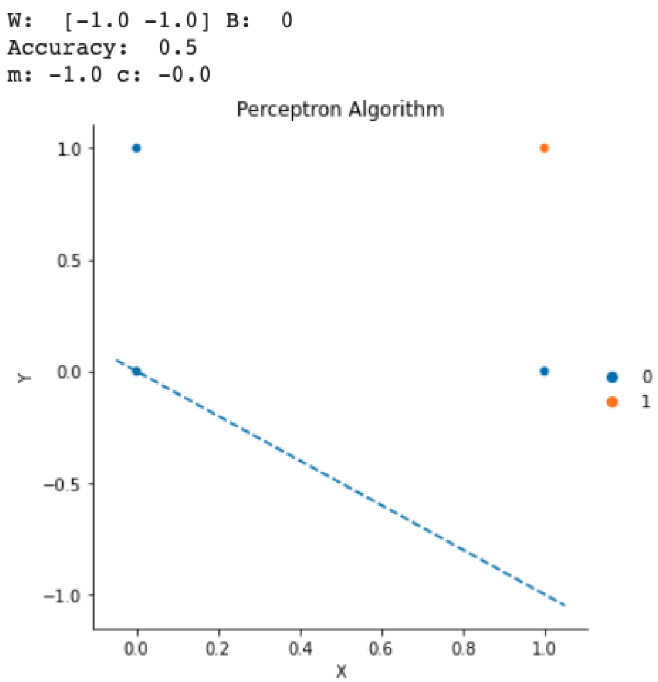
XOR



```
W:  [0.0 0.0]  B:  0
Accuracy:  0.5
Decision boundary not possible
```

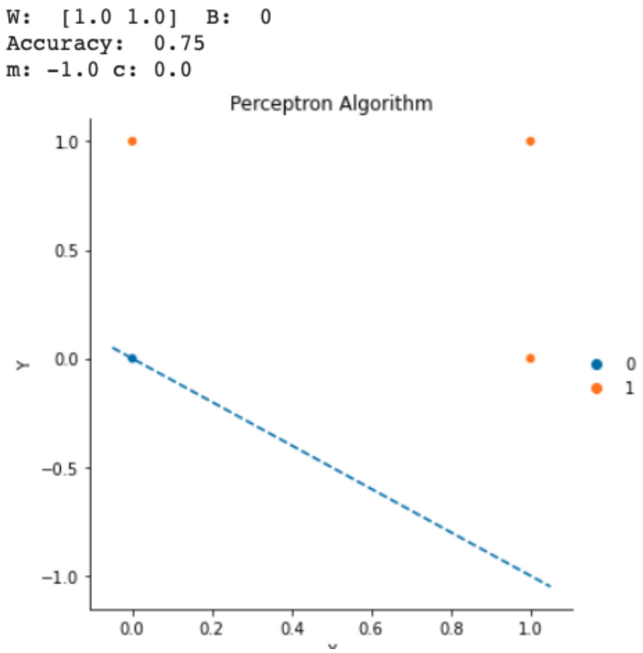
FIXED BIAS 0

AND



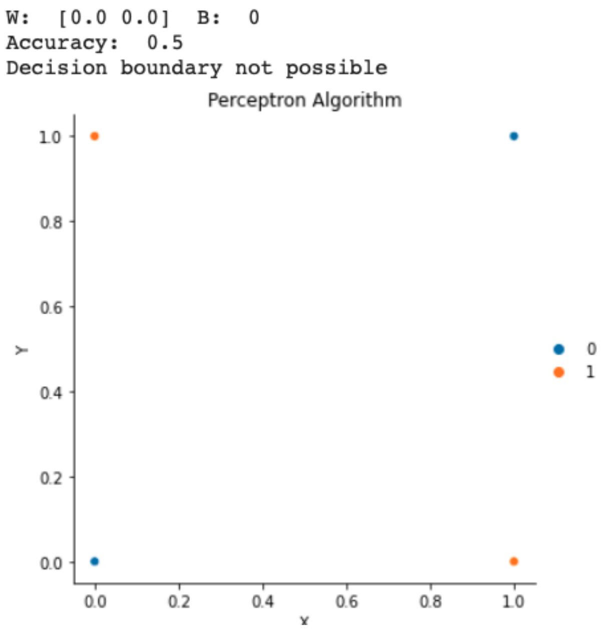
```
W:  [-1.0 -1.0] B:  0
Accuracy:  0.5
m: -1.0 c: -0.0
```

OR



```
W:  [1.0 1.0]  B:  0
Accuracy:  0.75
m: -1.0 c: 0.0
```

XOR



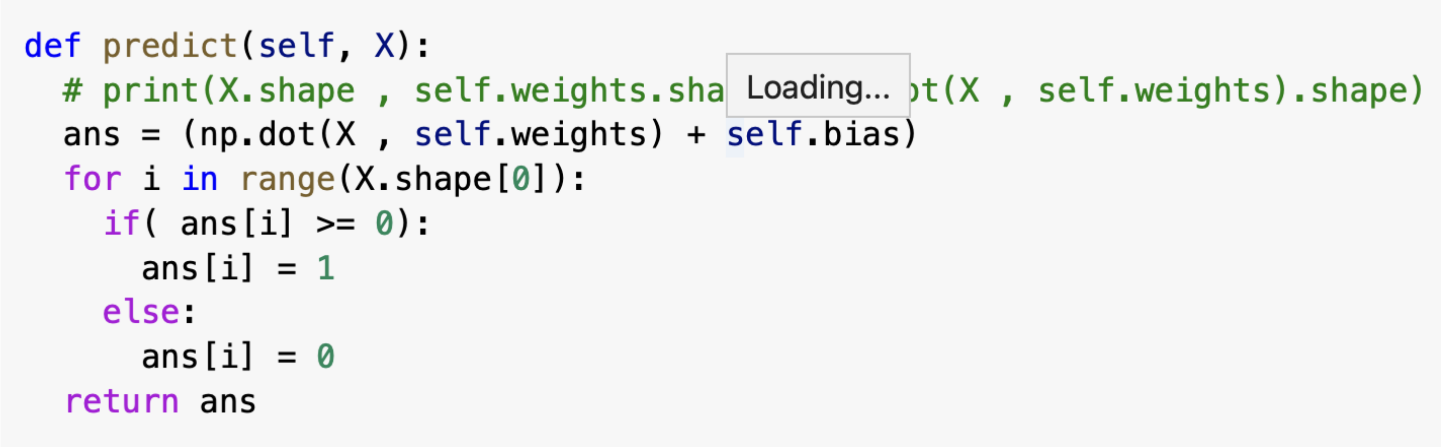
```
W:  [0.0 0.0]  B:  0
Accuracy:  0.5
Decision boundary not possible
```

For Flexible Bias - we can plot a decision boundary for and & or. However, it's not possible for xor

For Fixed Bias - we can plot a decision boundary for or only. However, it is not possible for xor & and

6. (1 mark) Given a hyperplane equation and a point, how would you predict which

We insert the point into the wT.x+b equation. If the value is greater or equal to 0, we assign class 0. Otherwise, we assign class 2. This ><= also depends on our activation function.

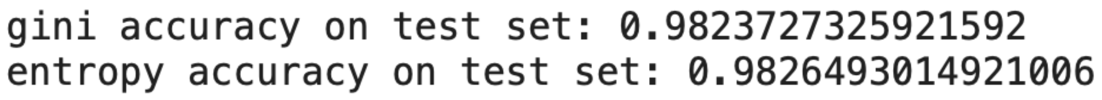


```
def predict(self, X):
  # print(X.shape , self.weights.sha Loading... t(X , self.weights).shape)
  ans = (np.dot(X , self.weights) + self.bias)
  for i in range(X.shape[0]):
    if( ans[i] >= 0):
      ans[i] = 1
    else:
      ans[i] = 0
  return ans
```

Section C

(a)Train a decision tree using the Gini index and the Entropy by changing the max-depth [4, 8, 10, 15, 20].

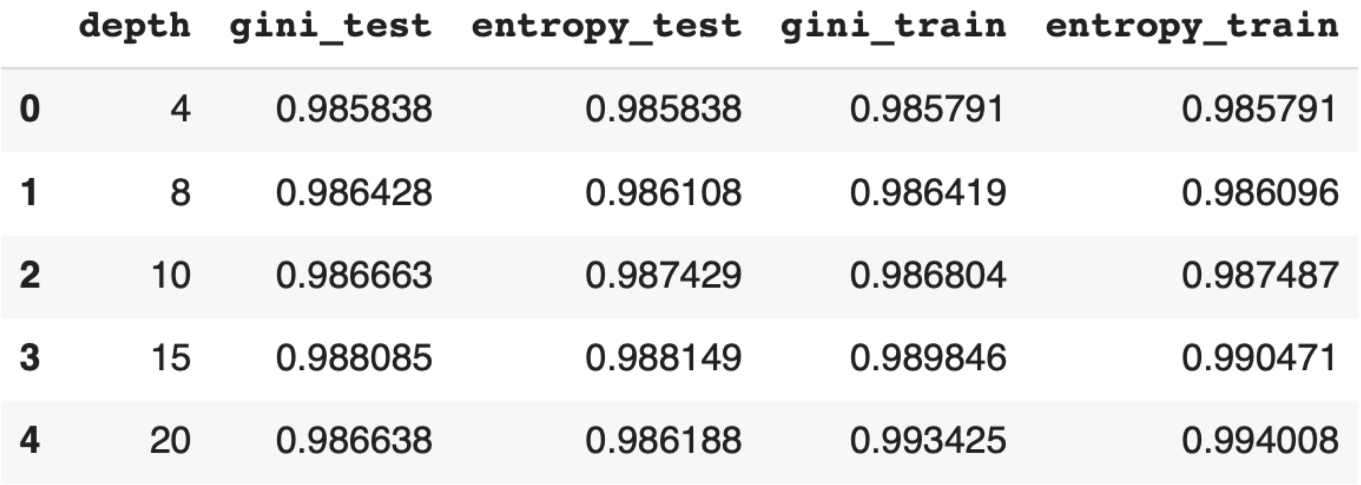
I ran a general decision tree with no depth mentioned first



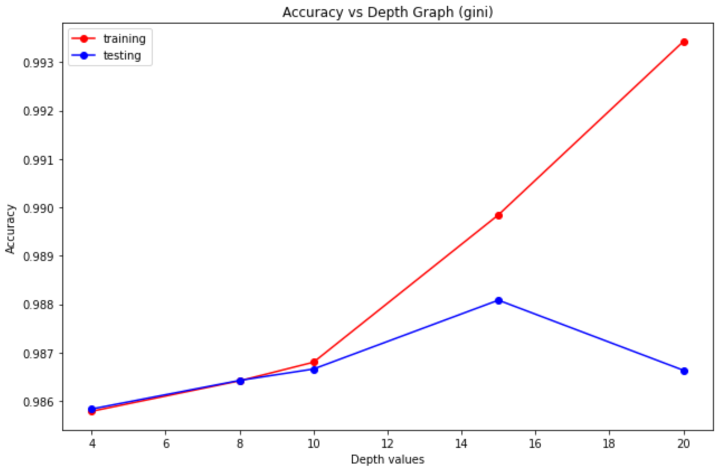
```
gini accuracy on test set: 0.9823727325921592
entropy accuracy on test set: 0.9826493014921006
```

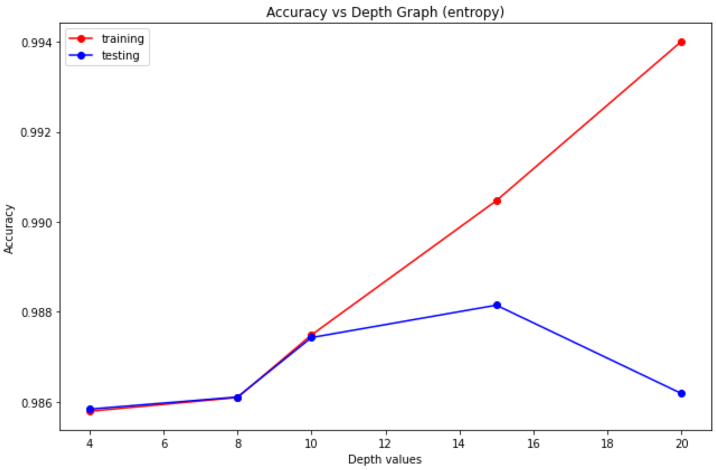
Gini & Entropy

depths = [4, 8, 10, 15, 20]



	depth	gini_test	entropy_test	gini_train	entropy_train
0	4	0.985838	0.985838	0.985791	0.985791
1	8	0.986428	0.986108	0.986419	0.986096
2	10	0.986663	0.987429	0.986804	0.987487
3	15	0.988085	0.988149	0.989846	0.990471
4	20	0.986638	0.986188	0.993425	0.994008

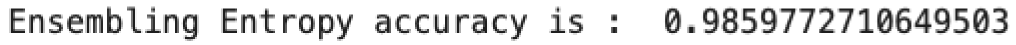




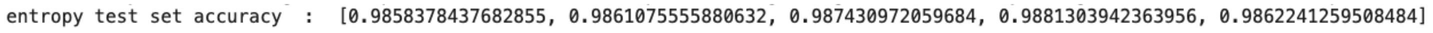
Entropy gives an overall better accuracy to me.

(b) (5 marks) Ensembling is a method to combine multiple ….

I used entropy decision trees and did bagging on them.
The Accuracy i got was -



```
Ensembling Entropy accuracy is :  0.9859772710649503
```



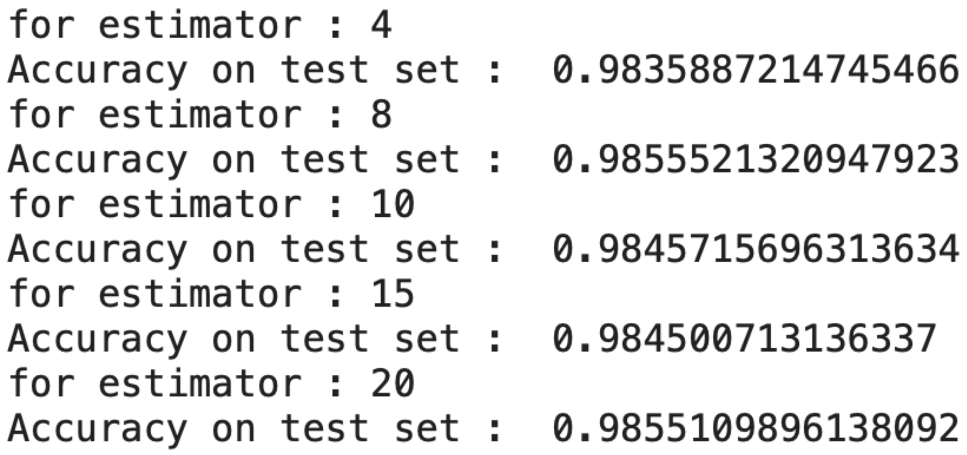
```
entropy test set accuracy  :  [0.9858378437682855, 0.9861075555880632, 0.987430972059684, 0.9881303942363956, 0.9862241259508484]
```

On comparing both, we can see that -

- bagging(depth 3) has better accuracy than decision tree(depth 4)
- However, its accuracy is still less than decision trees of depth 8, 10, 15, 20.

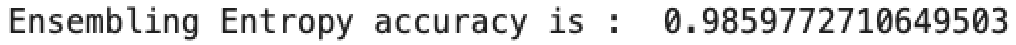
We can say that ensembling is good if we want to avoid making a tree of too much high depth. I can also confidently propose that if the depth of the ensembling decision tree was higher, like 10 or 8, then it would give a much better performance than just using the decision tree alone. (It can also backfire on us, however, if there is overfitting, like using a tree of too much high depth)

(c) (5 marks) Another popular boosting technique is Adaboost.



```
for estimator : 4
Accuracy on test set :  0.9835887214745466
for estimator : 8
Accuracy on test set :  0.9855521320947923
for estimator : 10
Accuracy on test set :  0.9845715696313634
for estimator : 15
Accuracy on test set :  0.984500713136337
for estimator : 20
Accuracy on test set :  0.9855109896138092
```

Random forest is giving better results



```
Ensembling Entropy accuracy is :  0.9859772710649503
```

Than all the AdaBoost accuracies.
Adaboost generally gives us better results as it focuses on handling incorrect observations and learns from previous models however it is also prone to overfitting, which is what I suspect happened in this case.
Random forest, on the other hand, creates 100 independent decision tree stumps and just combines their results. There is no learning from other stumps in this