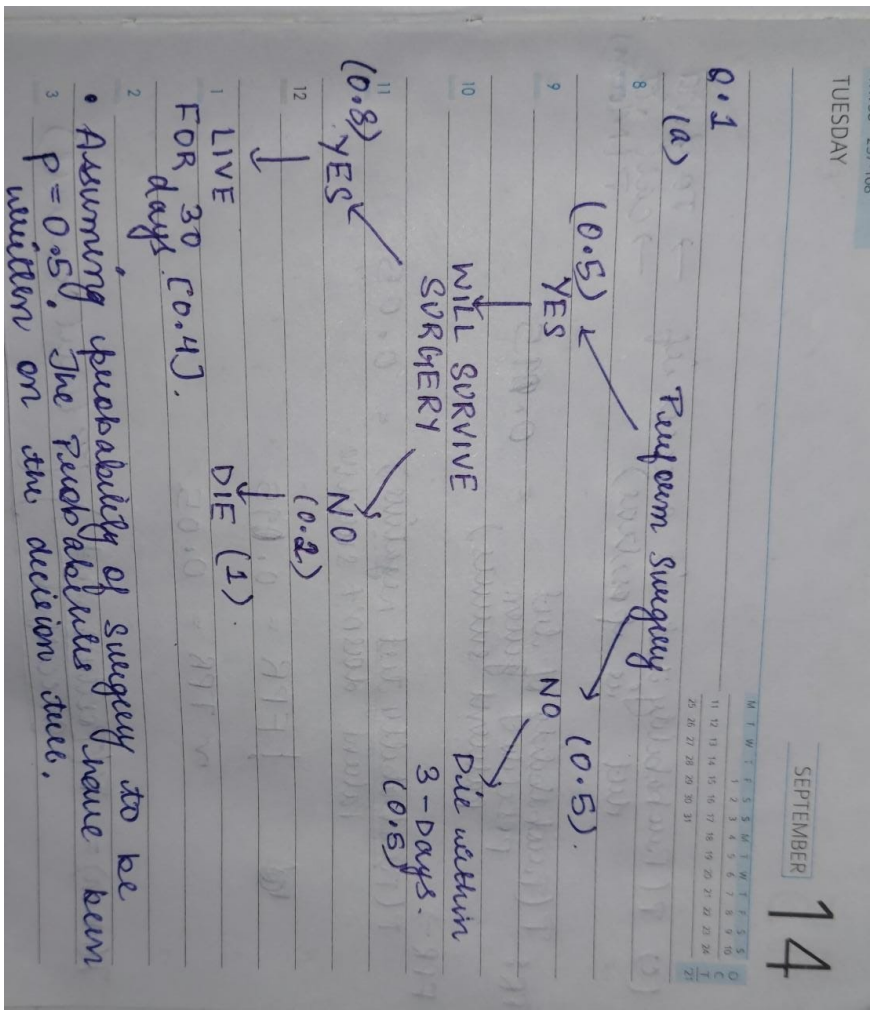
Assignment 2

Name : Vaibhav Jaiswal

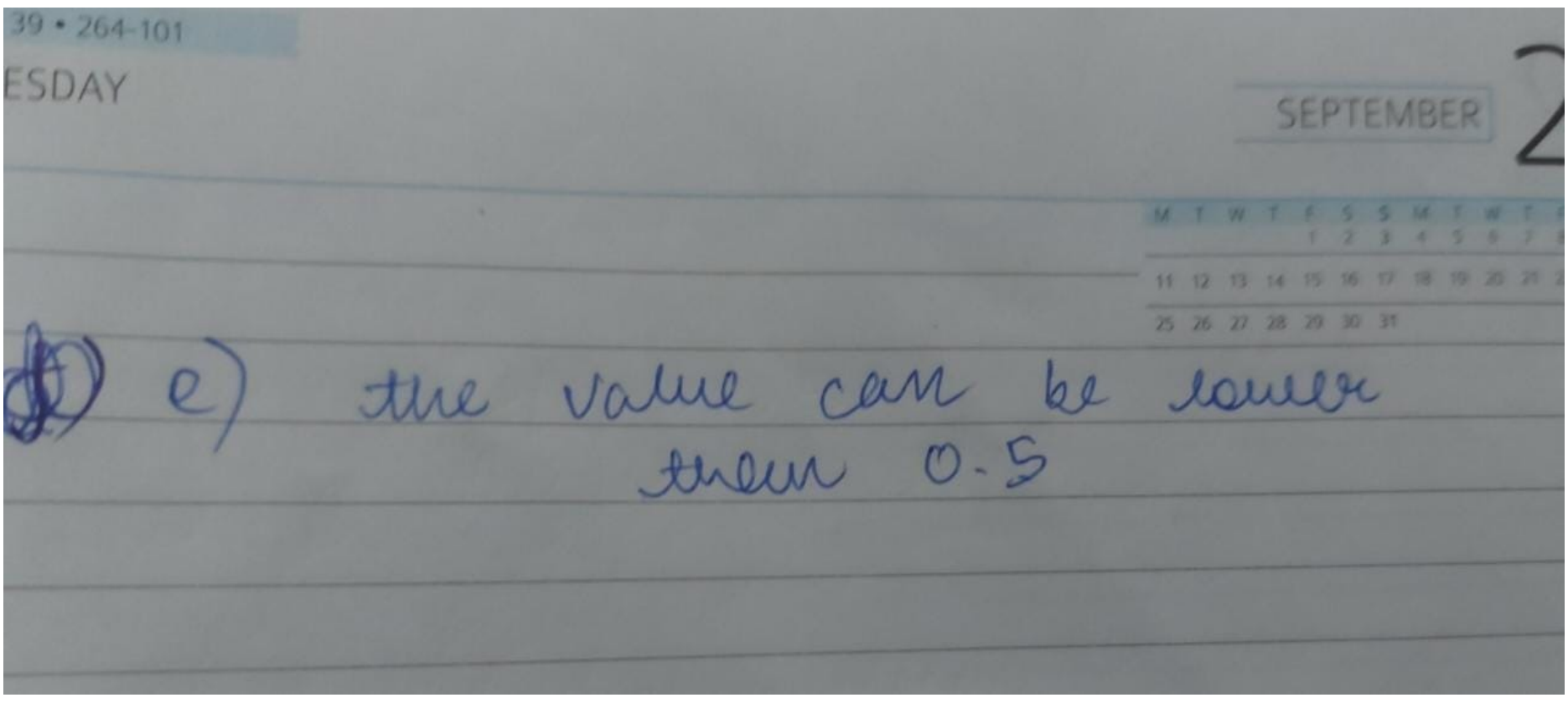
Roll Number : 2020547

Question 1:

A) →



B→



C→

(c) P(Probability test is success if test is positive) → To find
→ call it P(Main)

TPR: P(Probability of test positive given patient survives) = 0.95

FPR →

P(Probability test negative / Patient doesn't survive) = 0.05.

! FPR = 0.095

~ TPR = 0.05

$$P(Main) = \frac{P(\text{Positive test Successful}) \, (P(\text{Survive}))}{P(\text{Total})}$$

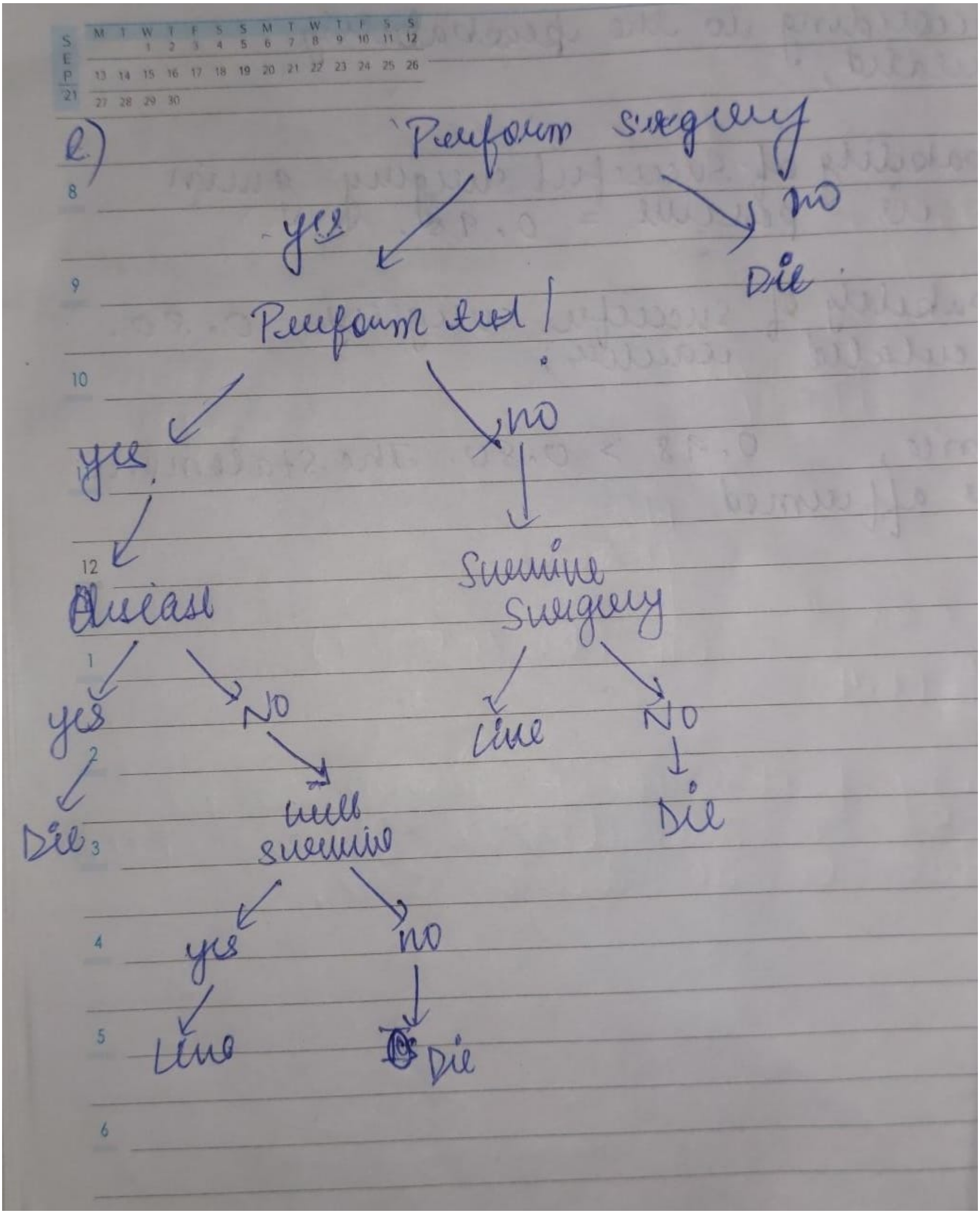
$$= \frac{(0.95)(0.8)}{(0.95)(0.8) + (0.05)(0.2)} = \frac{0.98}{}$$

∝

D →

d) According to the probability calculated, the Probability of Succesful surgery in Test is positive = 0.98. Probability of succesful surgery = 0.80. (calculated results)

Since, 0.98 > 0.80. The statement is affirmed.

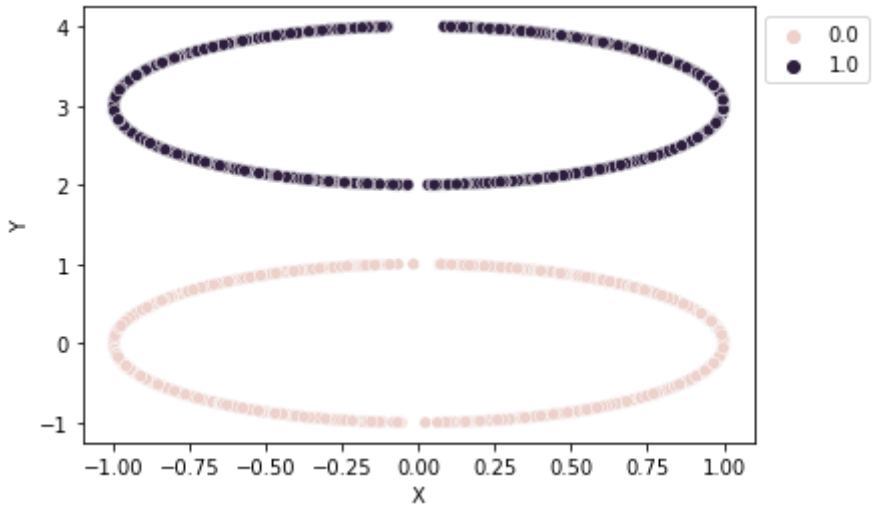

Perform surgery
yes
no
Die
Perform test
yes
no
Cancer
Immune Surgery
yes
no
Live
No
Die
yes
No
Die
Immune
yes
no
Live
Die

F→

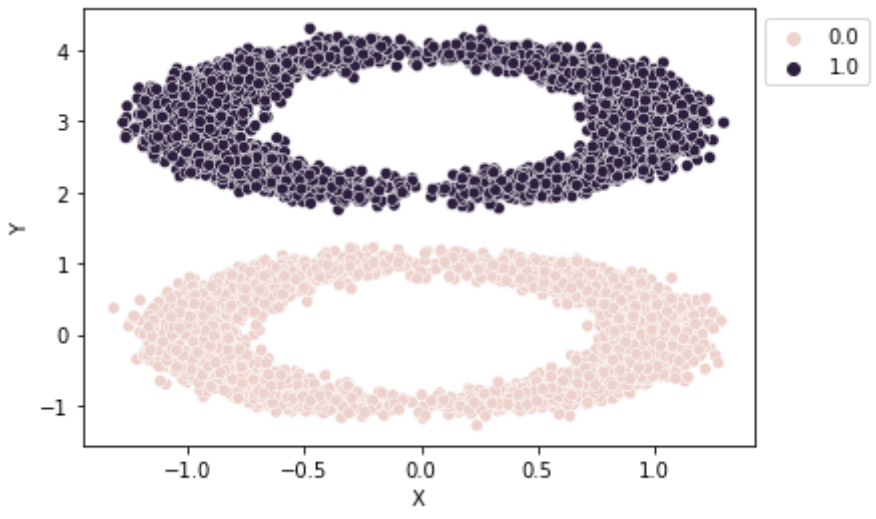
Question 2 :

A) The dataset was created for both without noise and with noise. The noise had been added for each individual value. Since, the data is large, the dataset will follow gaussian distribution.

B) Scatter Plot (2D) of the dataset without noise →



Scatter Plot (2D) of the dataset with noise →

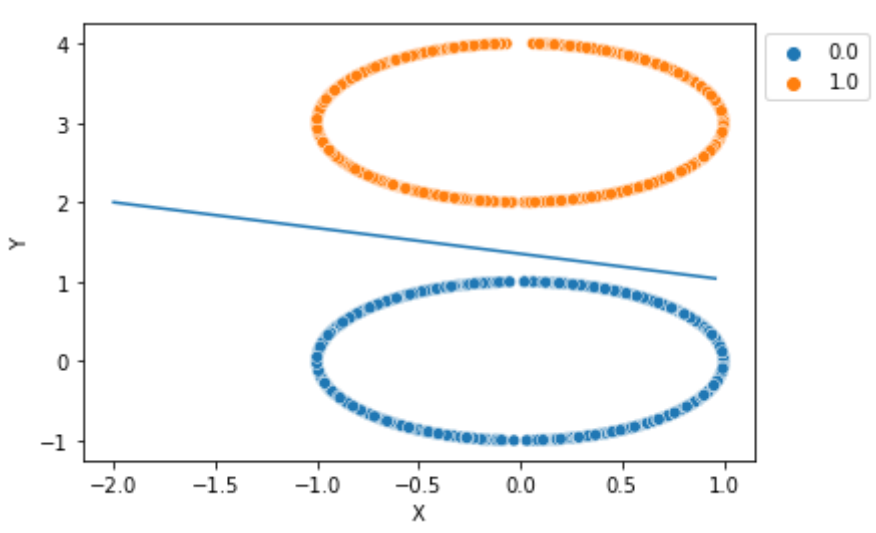


C) Decision Boundary with and without noise data was plotted : (PTA) (Please note that the values in the picture with the back background represent the values of W1, W2, and b in the order.

The accuracy when using testing1 and testing2 came out to be 99.96-100 percent over multiple different runs

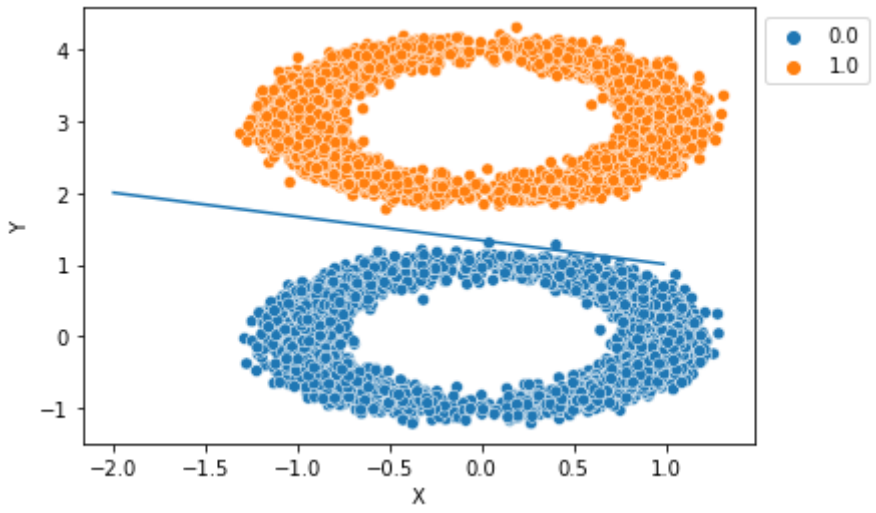
Without noise PTA1 →

```
-0.47093865224279363
90.91787521173788
-89.00000000000863
```

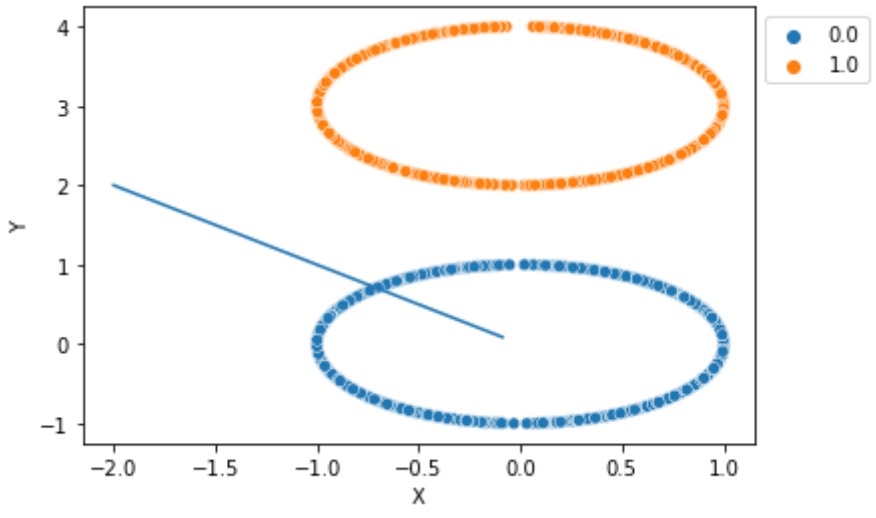


With noise PTA1→ (The accuracy comes out to be 99 percent). Due to noise generated some of the points lie over the decision boundary that has been created.

```
-0.4576275675062583
89.98426276693317
-89.9400000000091
```



D) Using Perceptron Training Algorithm on data created without noise, but fixed bias which will be equal to 0.

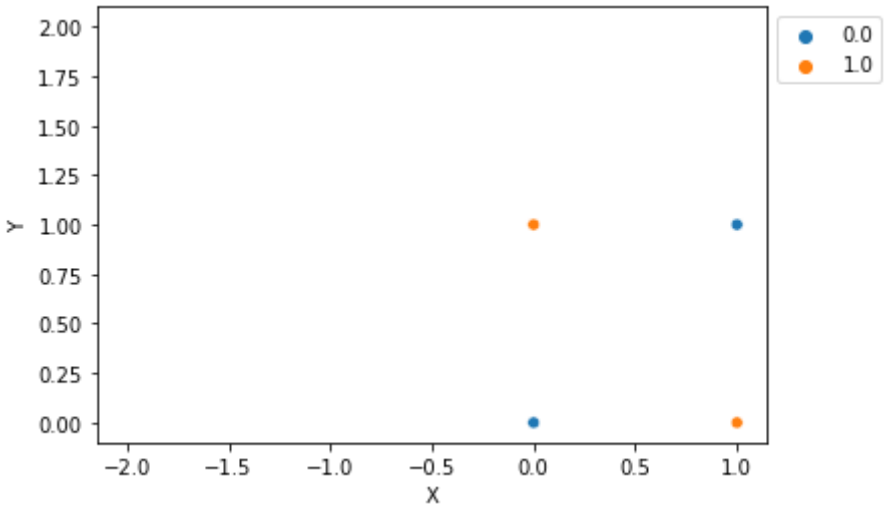



```
-3.7367245989198175
85.1446203075654
Accuracy of testing with fixed bias as zero is 75.6
```

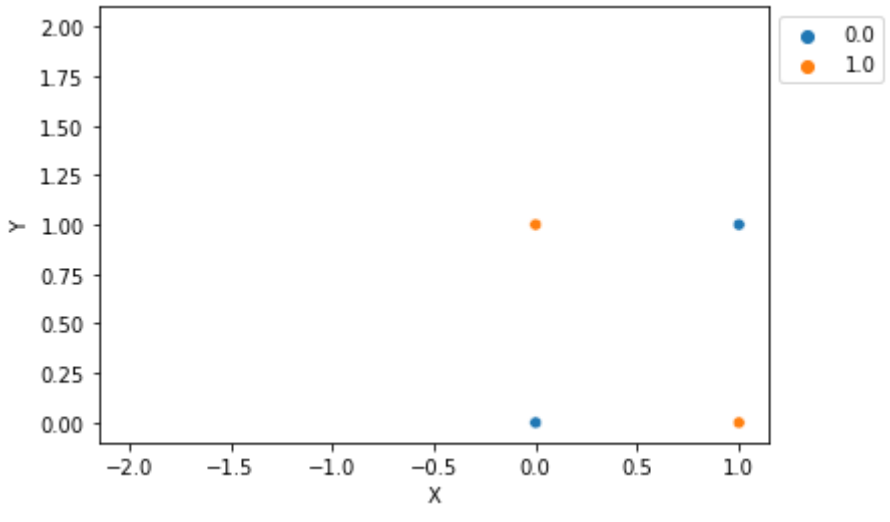
Comparison with Question 2.3 →

While comparing the perceptron training algorithm, we know that the decision boundary follows the equation $ax + by + c = 0$. In our case it becomes $W1$, and b becomes $W2$. On the oth hand c, becomes the bias b. Now, we know that, if the equation only be written as ax + by = 0, it will pass from origin. This is valid for the perceptron training algorithm in question 2.4, where we have fixed the bias to 0. Thus the decision boundary passes through origin. The circle x + y = 1, is present with origin as its centre, hence the decision boundary passes through the circle. This means that the accuracy drops since, the decision boundary would not be able to perform binary classification accurately. From the graph it's visible that the line passes for the label 0. On comparison with the noise dataset, the accuracy is still better because of bias. Bias helps the line as another parameter that will help train the model better, and hence create the decision boundary with more accuracy.

E) XOR →



XOR learnable

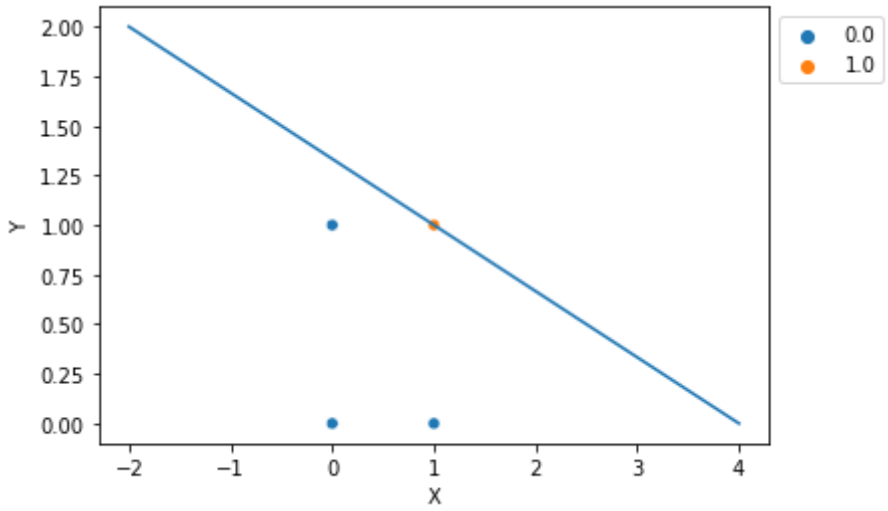


XOR fixed

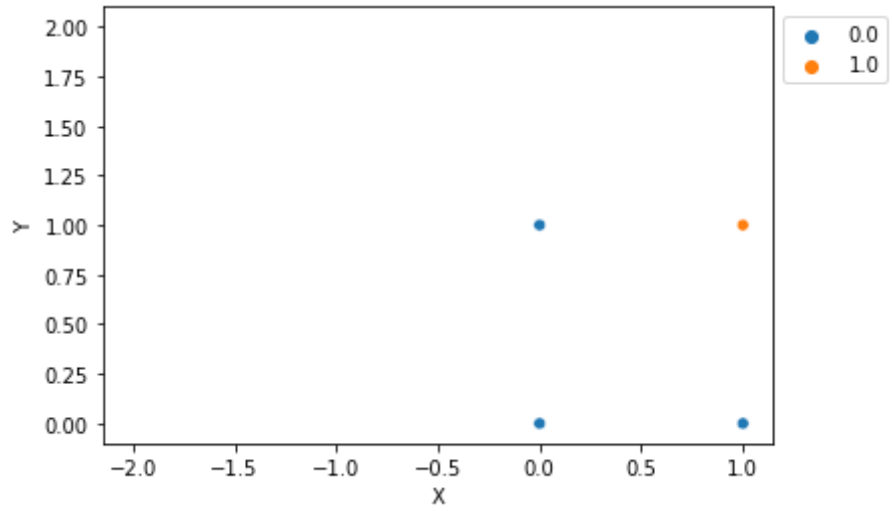
	X	Y	Label
1	0.0	0.0	0.0
2	0.0	1.0	1.0
3	1.0	0.0	1.0
4	1.0	1.0	0.0

AND BIAS →

	X	Y	Label
1	0.0	0.0	0.0
2	0.0	1.0	0.0
3	1.0	0.0	0.0
4	1.0	1.0	1.0

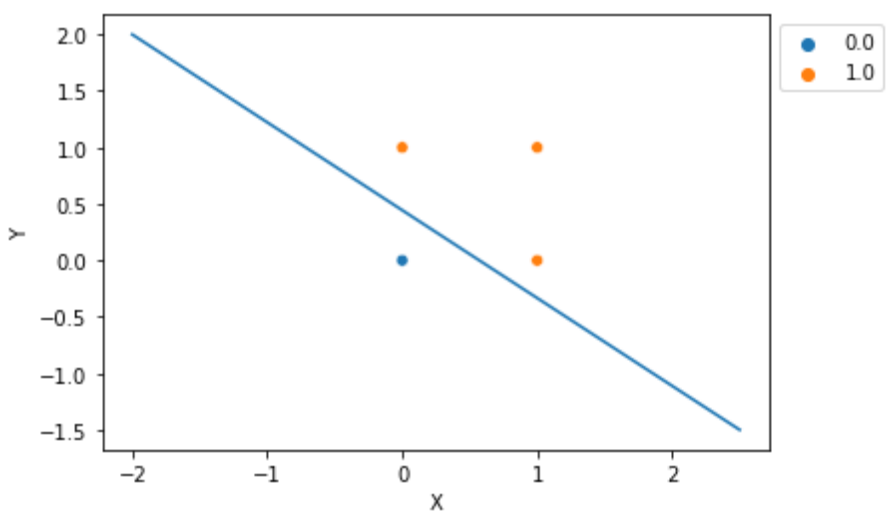


AND learnable bias



AND fixed bias

OR BIAS →



OR learnable bias

```
Decision Boundary exception
```

OR fixed bias

	X	Y	Label
1	0.0	0.0	0.0
2	0.0	1.0	1.0
3	1.0	0.0	1.0
4	1.0	1.0	1.0

F)

f) The hyperplane equation is represented as $w \cdot x + b = 0$, where w is vector normal to hyperplane and b is offset.

The equation can be expanded and written as $w_1x_1 + w_2x_2 + w_3x_3 + \dots w_nx_n + b = 0$

The equation works as a decision boundary to predict if a point "A" is in class (0 or 1). (since n → dimensional space)

• If, point A is in class 1, then (and has been accurately predicted)

$f(A) = w_1x_1 + w_2x_2 \dots w_nx_n + b_A$ will give the equation

• where, $f(A) \geq 0$. • Thus, we can now say if a point gives $f(\text{point}) \geq 0$. then it belongs to class 1. And if a point gives $f(\text{point}) < 0$, then it belongs to class 0.

SUNDAY 18

Hence, for a new point say P. if $f(P) \geq 0$ or $f(P) < 0$, then according to observation the point P can be in class 1 or class 0.

Question 3 :

Implementation of the train-test-validation split :

The split was implemented from scratch using basic of list functions in pandas library. The value of the total dataset length came out to be equal to the length of the $train - test - validation$ split.

```
The print value of the main count is 2916697 , whereas the print value of the all splits is equal to 2916697
```

Code implementation of the train-test-validation split

The Code Implementation :

```
def customTrainTestValidateSplit(df):
    n = df.label.count() #will give the size of the entire dataset
    trainnPoint = math.floor(n*0.7)
    testplusvalidaten = n - trainnPoint
    testplusvalidatePoint = trainnPoint + (math.floor(testplusvalidaten*0.5))
    shuffled_df = df.sample(frac = 1, random_state = 50)
    train_df = shuffled_df[:trainnPoint]
    test_df =  shuffled_df[trainnPoint:testplusvalidatePoint]
    validate_df = shuffled_df[testplusvalidatePoint:]
    return train_df, test_df, validate_df
```

The code implementation

Part A :

Training a Decision Tree →

Gini Index →

```
Accuracy  4 is 0.9859635889875544 (GINI INDEX) : Train
Accuracy  4 is 0.9858081621924321 (GINI INDEX) : Validation

Accuracy  8 is 0.9865852961680438 (GINI INDEX) : Train
Accuracy  8 is 0.9864367264374121 (GINI INDEX) : Validation

Accuracy  10 is 0.9869235780162512 (GINI INDEX) : Train
Accuracy  10 is 0.9867018662643855 (GINI INDEX) : Validation

Accuracy  15 is 0.9882789910972446 (GINI INDEX) : Train
Accuracy  15 is 0.9881555639364121 (GINI INDEX) : Validation

Accuracy  20 is 0.9868687215003257 (GINI INDEX) : Train
Accuracy  20 is 0.9866721523182592 (GINI INDEX) : Validation
```

Entropy index →

```
Accuracy  4 is 0.9859635889875544 (ENTROPY) : Test
Accuracy  4 is 0.9858081621924321 (ENTROPY) : Validation

Accuracy  8 is 0.9862447286316728 (ENTROPY) : Test
Accuracy  8 is 0.9861213014708403 (ENTROPY) : Validation

Accuracy  10 is 0.9875887132718483 (ENTROPY) : Test
Accuracy  10 is 0.987328144821202 (ENTROPY) : Validation

Accuracy  15 is 0.9882607055919361 (ENTROPY) : Test
Accuracy  15 is 0.987968137507 (ENTROPY) : Validation

Accuracy  20 is 0.9864412978137392 (ENTROPY) : Test
Accuracy  20 is 0.9863407275345425 (ENTROPY) : Validation
```

Entropy gives us better results as compared to “gini” index. The accuracy is slightly better, hence the it has been choosen for part b. It must be noted that gini index gave the results faster than entropy since entropy involves the calculation of logarithm.

PART B →

The aggregation function gives the accuracy of the function →

```
Accuracy of aggregation function will be 0.9859635889875544
Accuracy of aggregation function will be 0.9858081621924321 Validation
```

```
Accuracy of aggregation function will be 0.9859635889875544
```

The accuracy is better in case of ensemble models as compared to the single decision tree classifiers. The accuracy is more even though the depth is lower than the one used for decision tree. This proves the importance of random forest over decision tree, and in general how several models trained over the same set of data can give better accuracy.

PART C →

Adaboost and Random forest result comparison :

```
Accuracy 4 is 0.9849121724323151 (DECISION TREE WITH ADABOOST) : Test
Accuracy 4 is 0.9848893155506794 (DECISION TREE WITH ADABOOST) : Validation

Accuracy 8 is 0.9851247414315265 (DECISION TREE WITH ADABOOST) : Test
Accuracy 8 is 0.9849304579376236 (DECISION TREE WITH ADABOOST) : Validation

Accuracy 10 is 0.9837578999097153 (DECISION TREE WITH ADABOOST) : Test
Accuracy 10 is 0.9835224740288683 (DECISION TREE WITH ADABOOST) : Validation

Accuracy 15 is 0.9854538805270797 (DECISION TREE WITH ADABOOST) : Test
Accuracy 15 is 0.9853624530005372 (DECISION TREE WITH ADABOOST) : Validation

Accuracy 20 is 0.9865624392864082 (DECISION TREE WITH ADABOOST) : Test
Accuracy 20 is 0.9864275836847579 (DECISION TREE WITH ADABOOST) : Validation
```

Random forest →

```
Accuracy 4 is 0.9846013188420704 (RANDOM FOREST) : Test
Accuracy 4 is 0.9844481777351116 (RANDOM FOREST) : Validation

Accuracy 8 is 0.9871818607787339 (RANDOM FOREST) : Test
Accuracy 8 is 0.9869350064570691 (RANDOM FOREST) : Validation

Accuracy 10 is 0.9875475708849042 (RANDOM FOREST) : Test
Accuracy 10 is 0.9873098593158935 (RANDOM FOREST) : Validation

Accuracy 15 is 0.9879178523674015 (RANDOM FOREST) : Test
Accuracy 15 is 0.9877075690563536 (RANDOM FOREST) : Validation

Accuracy 20 is 0.9880664220980332 (RANDOM FOREST) : Test
Accuracy 20 is 0.9877761397012605 (RANDOM FOREST) : Validation
```

In theory, Adaboost gives better results than random forest algorithm. But during observation, they have comparatively same results on dataset. This can be correlated to the fact that the dataset is skewed and hence does not offere variability. (most of the labels in the dataset is of white)