Machine Learning

Assignment 2

Sahil Goyal

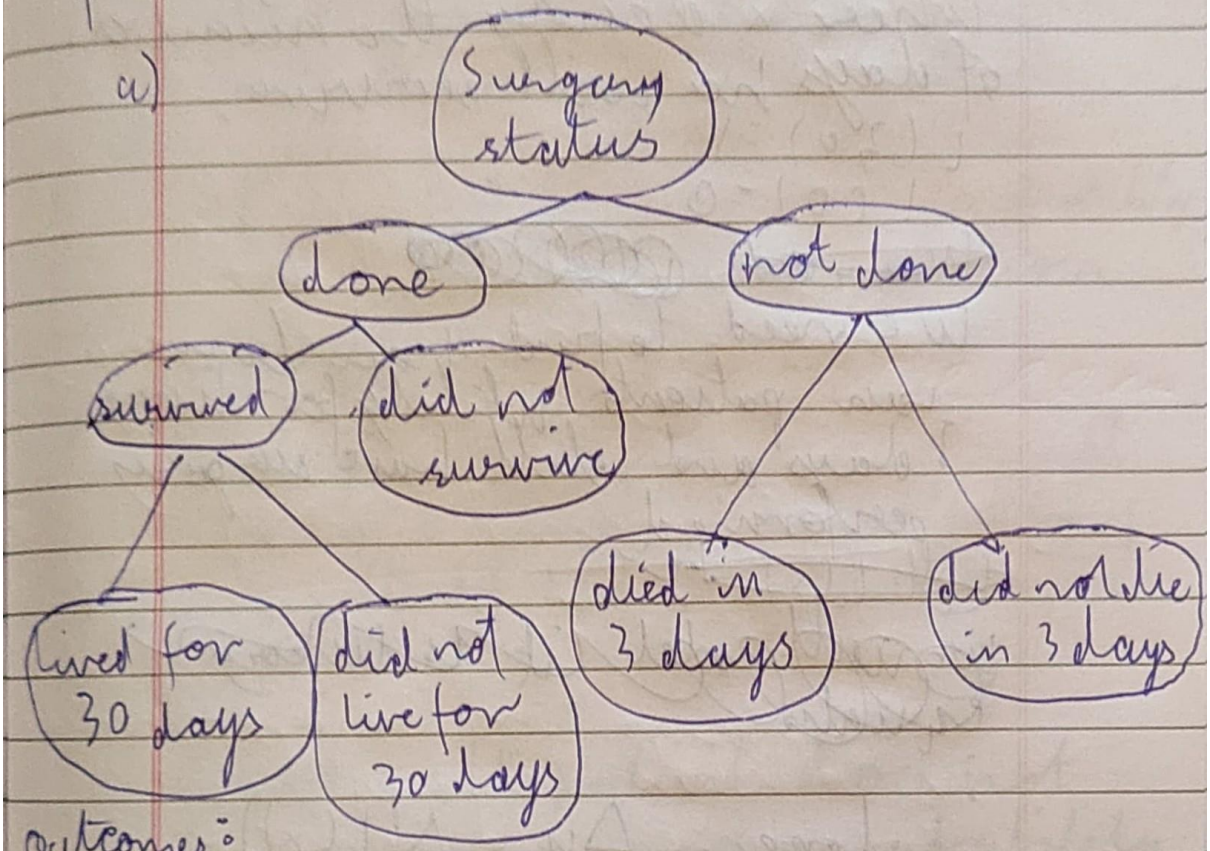
2020326

Section A:

1.

1

a)



outcomes:

P(lived for 30 days | survived surgery)

= 0.8

P(did not live for 30 days | survived surgery) = 0.2

P(died in 3 days | d not done surgery)

P(did not die in 3 days | not done surgery)

b) L(x) denotes the living function where x denotes the number of days he will survive.

L(30) = 1

L(0) = 0

~~We need~~

We need to find how long can patients utility for living 3 days and still have surgery performed

~~If L(x) be~~

~~would model it as linear equation~~

$$\therefore \text{slope} = \frac{\Delta y}{\Delta x} = \frac{\Delta(L(x))}{\Delta x}$$

$$= \frac{1}{30}$$

As L(0) = 0

$$\therefore L(x) = \frac{x}{30}$$

For x = 3

$$L(3) = \frac{1}{10}$$

∴ Patients utility for living 3 days and still having surgery is 1/10

c) The given procedure is a low risk procedure that predicts whether or not the patient will survive the operation. If the test is +ve, the probability of surviving the operation increases.

True +ve rate: If the patient will survive surgery probability of results of test being positive is 0.95

False +ve: If patient does not survive surgery, probability of test being positive is 0.05

$$\therefore P\left(\frac{\text{successful surgery}}{\text{Test is +ve}}\right)$$

$$\therefore P\left(\frac{\text{Test is +ve}}{\text{successful surgery}}\right) \frac{P(\text{successful surgery})}{P(\text{Test is +ve})}$$

$$= \frac{P(\text{Test is positive} / \text{successful surgery})\, P(\text{successful surgery})}{P(\text{Test is +ve} / \text{successful surgery})\, P(\text{successful surgery}) + P\left(\frac{\text{Test is +ve}}{\text{unsuccessful surgery}}\right) P(\text{unsuccessful surgery})}$$

$$= \frac{0.95 \times 0.8}{0.95 \times 0.8 + 0.05 \times 0.2}$$

$$= 0.987$$

∴ Survivors probability of having a successful surgery if the test is positive is 0.987

d) $P\left(\frac{\text{successful surgery}}{\text{Test is +ve}}\right) = 0.987$

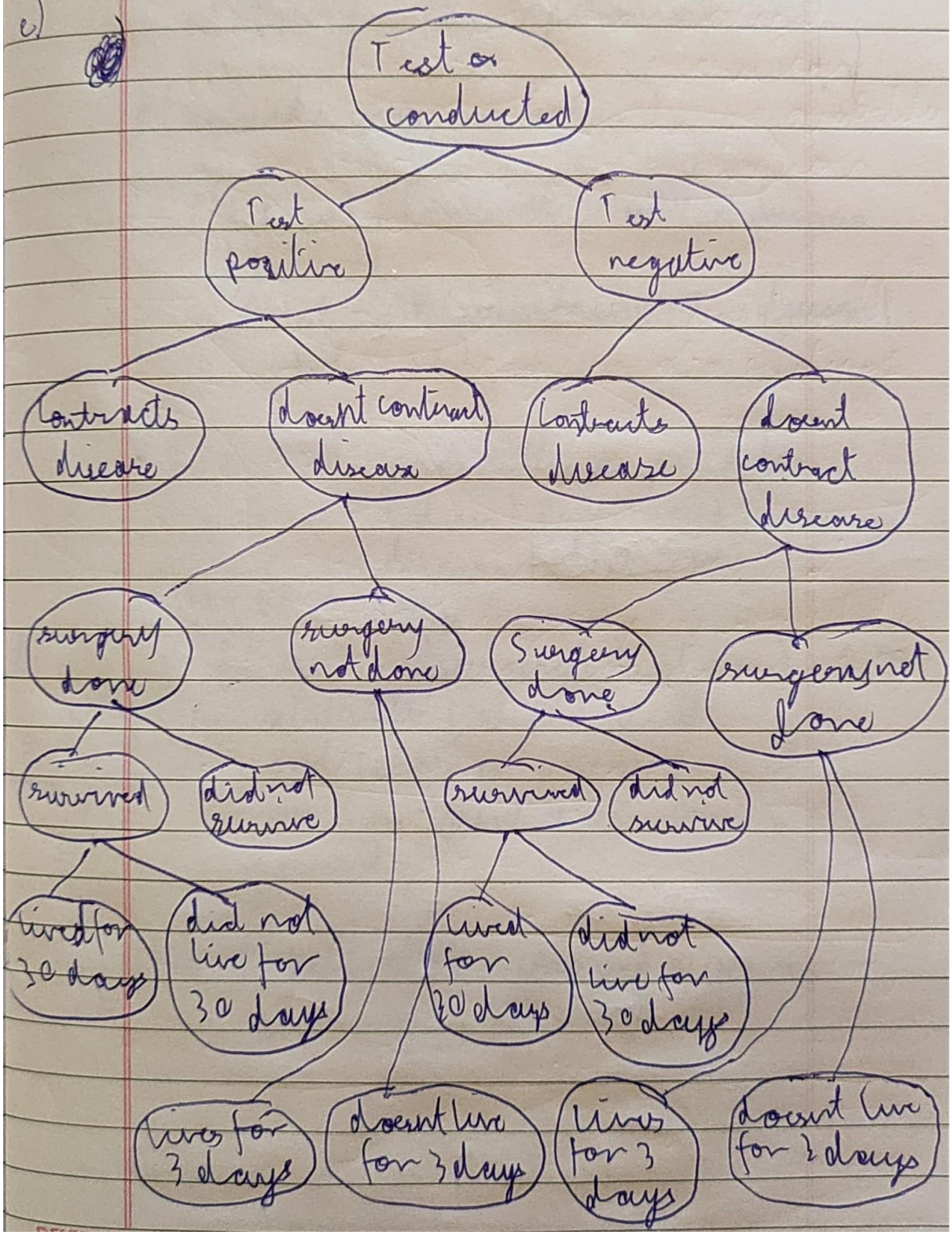
∴ Considering that the probability is so high, clearly the surgery should be performed if the test result is positive.

Earlier, the probability of the successful surgery was 0.8 but now it has gone up to 0.987 which means probability of failure has dropped from 0.2 to 0.013

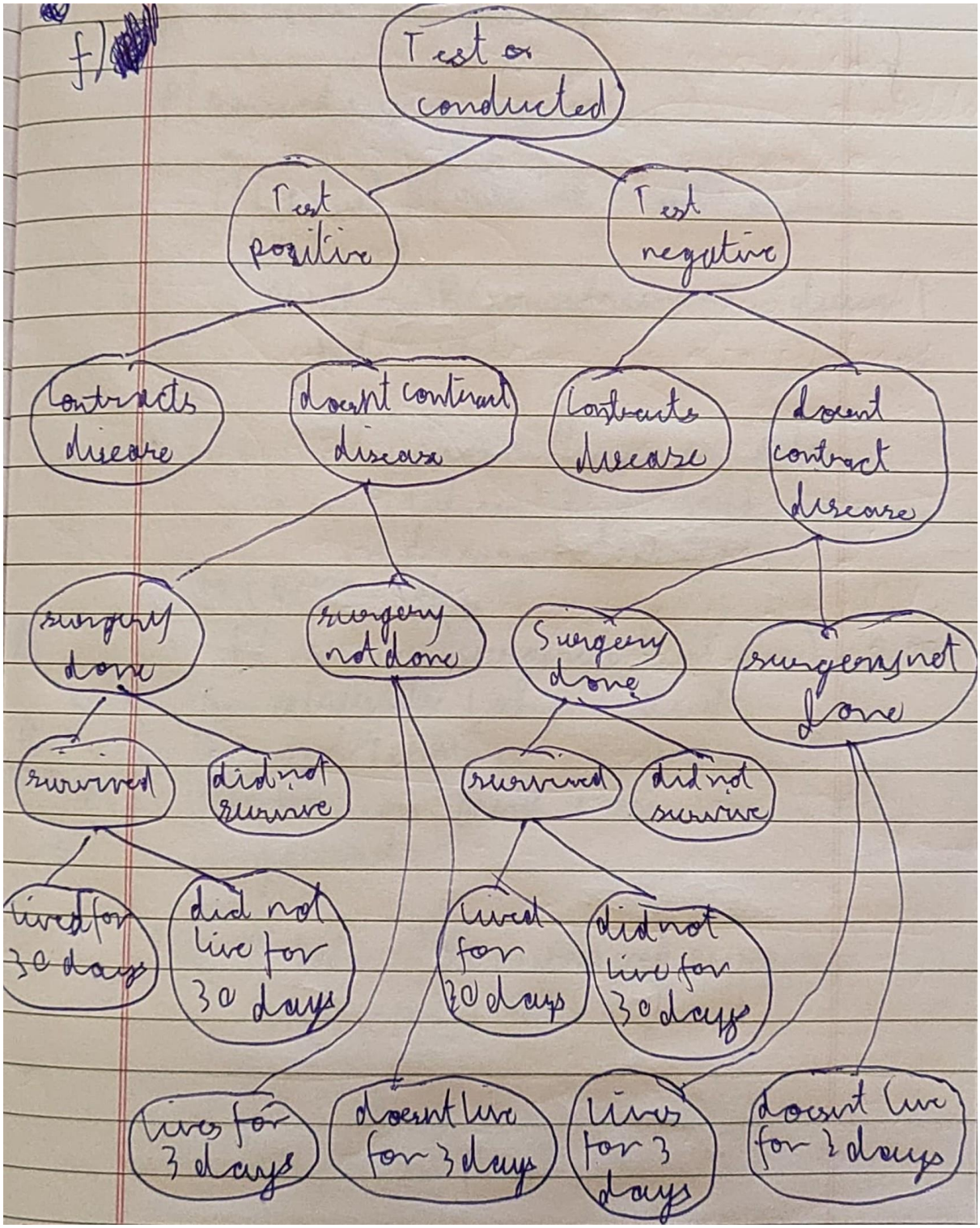
This means that chance of failure has reduced by approximately 15 times.

The surgery should be performed if the test result is positive.

c)



f)


Test is conducted
Test positive
Test negative
contracts disease
doesn't contract disease
contracts disease
doesn't contract disease
surgery done
surgery not done
Surgery done
surgery not done
survived
did not survive
survived
did not survive
lived for 30 days
did not live for 30 days
lived for 30 days
did not live for 30 days
lives for 3 days
doesn't live for 3 days
lives for 3 days
doesn't live for 3 days

We need to find whether or not the test should be conducted prior to the operation

P(survived surgery, no disease)

= P(test +ve) P(survived surgery, no disease | test +ve)
+ P(test -ve) P(survived surgery, no disease | test -ve)

We will ignore the second term
i.e P(test -ve) P(survived surgery, no disease | test -ve)
because
P(survived surgery | test +ve) = 0.987
is extremely large and second term is very small in comparison and can be safely ignored

⇒ P(test +ve) P(survived surgery, no disease | test +ve)

P(test +ve)

$= P(\text{test +ve} \mid \text{survived surgery}) \, P(\text{survived surgery})$

$+ P(\text{test +ve} \mid \text{didnt survive surgery}) \, P(\text{didnt survive surgery})$

$= (0.8)(0.95) + (0.2)(0.05)$

$= 0.77$

∴ Required probability

$= P(\text{test +ve}) \, P(\text{survived surgery} \mid \text{test +ve}) \, P(\text{no disease} \mid \text{+ve test})$

$= 0.77 \times 0.987 \times 0.995$

$= 0.756$

∴ Probability of survival after surgery is without doctors 0.756

∴ Test shouldnt be performed as probability of survival dropped from 0.8 to 0.756

Section B:

2)

1.

Explanation of Code:

- For this part, we have created a class named dataset, that takes the number of points as an input when initialising a new object of this class. This class has 3 variables, number(the number of points in dataset), data0(used to maintain the coordinates for circle with label 0), and data1(used to maintain the coordinates for the circle with label 1). This class has a function, get(), which takes a Boolean value add_noise as input. This function gives us random points on a circle, with label 0 and 1 and the label for each point is also chosen randomly. The points are chosen by first taking x as a random uniform float between -1 and 1, and then getting the value of y by substituting the value of x in the circle equation. If we set add_noise as True, gaussian noise with mean 0 and standard deviation 0.1 is added to all the points.

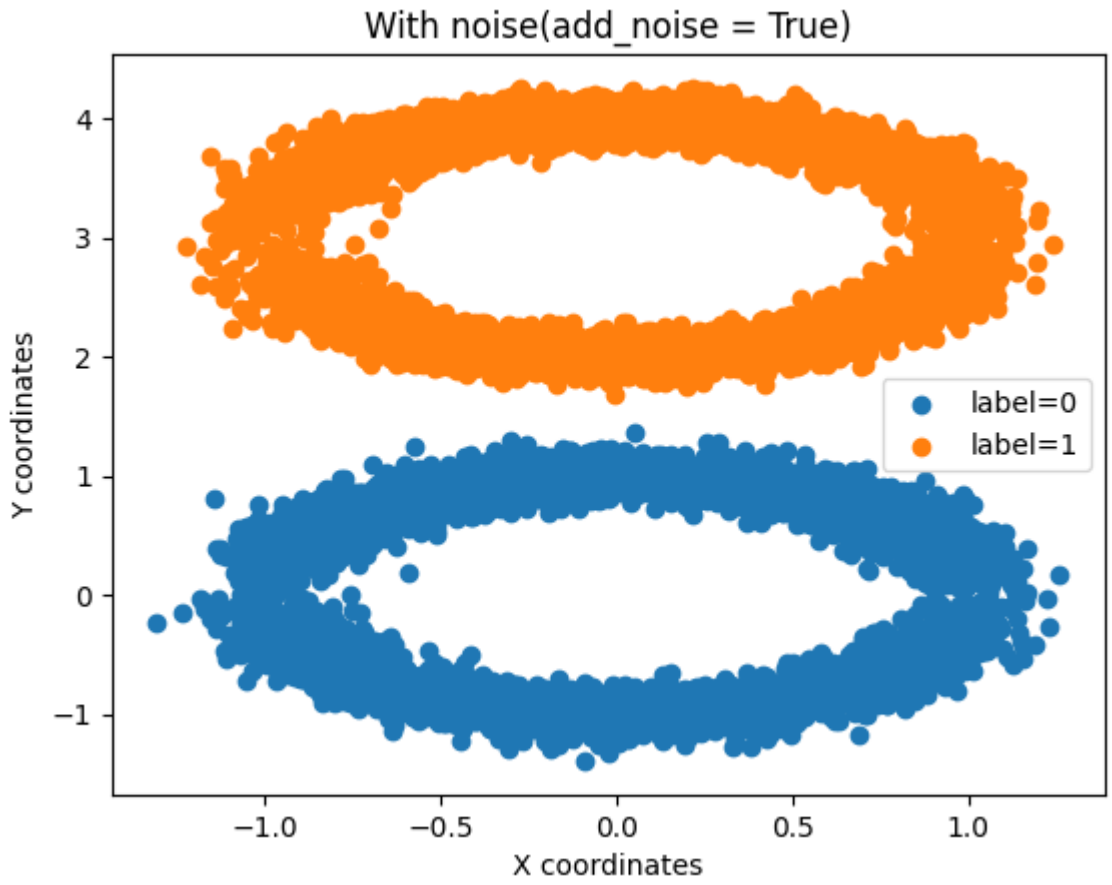
2.

Explanation of Code:

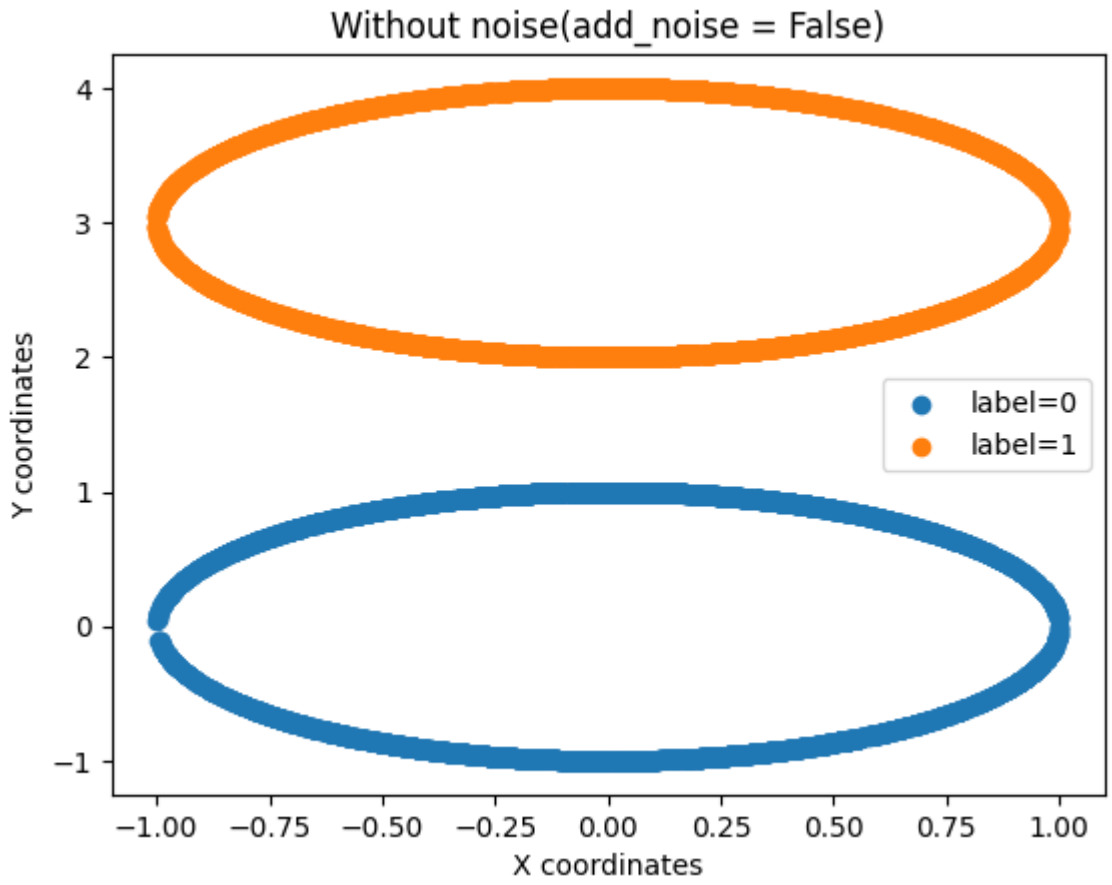
- After creating the object o dataset class, we use the function get to get the required data with labels 0 and 1, and use matplotlib library to plot the data. The X axis is labelled as X coordinates, the Y axis is labelled as Y coordinates, and there is a legend to show the label for both the circles we have plotted.
- After this, we use get unction again, but with add_noise set to false. This way, we do not get any gaussian noise, but otherwise the process for plotting the dataset is the same.

Relevant Graphs:

With Noise:



Without Noise:



Results:

- For first graph, we can see that as there is noise, the points are not on the exact line of the circle, but they are slightly moved because of the noise.
- For second graph, there is no noise so we can see that the points are exactly on the circles as given.
- We can see that in both the graphs the circles are linearly separable.

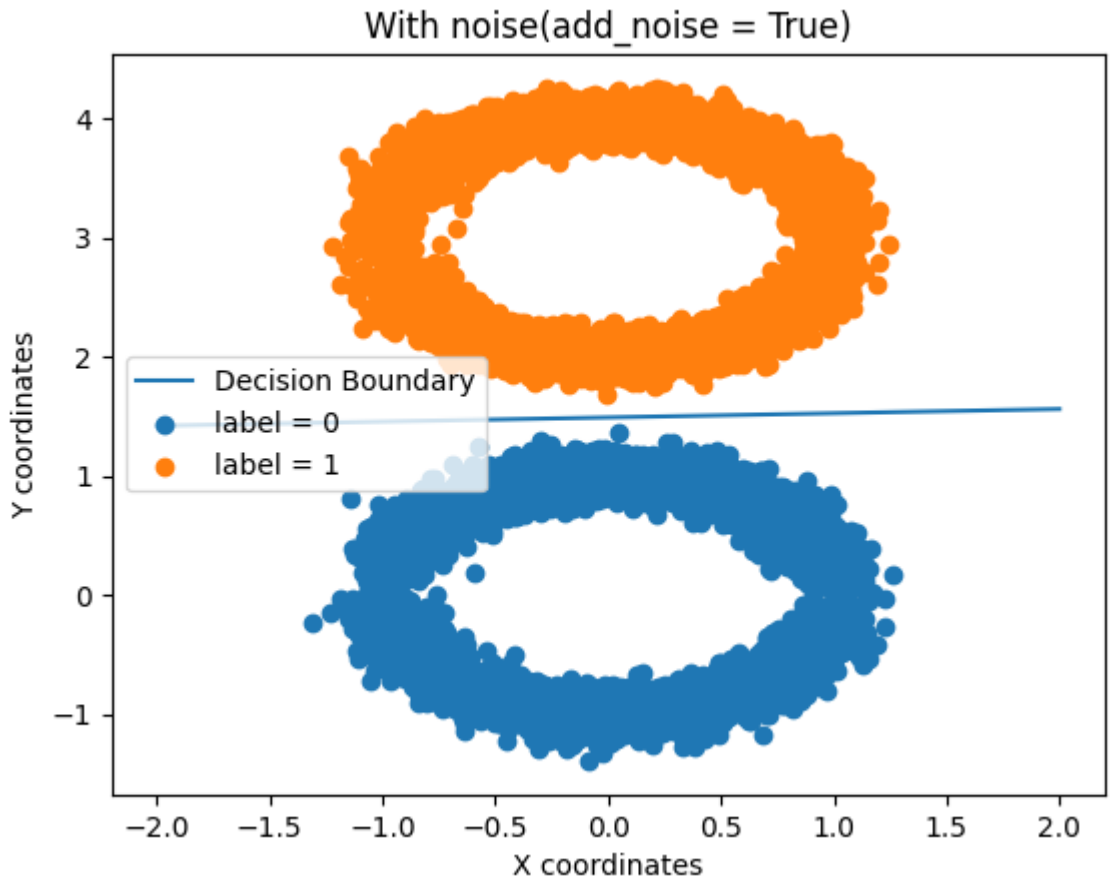
3.

Explanation of Code:

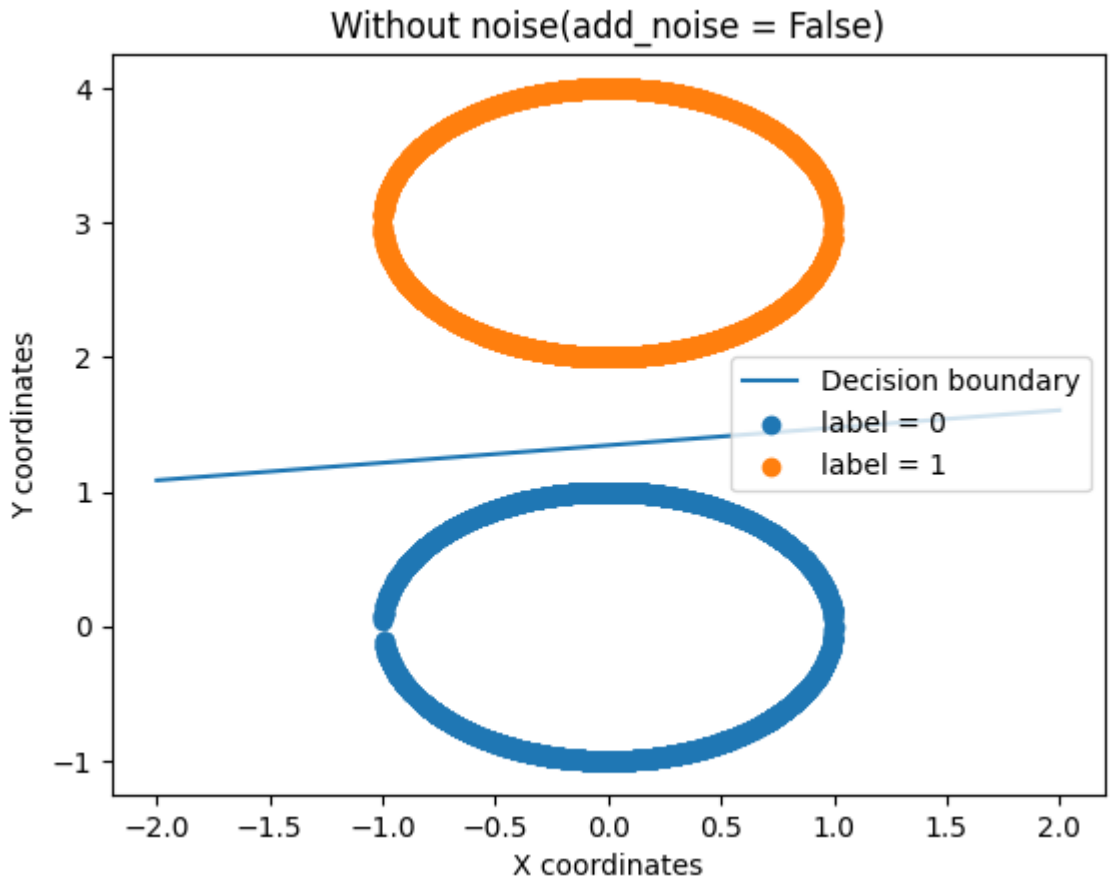
- For this part, we have implemented the perceptron_Training_Algorithm_customed class that implements the perceptron training algorithm on the given data and returns the weights and bias.
- The function takes the data as input and a Boolean value bias, that tells us whether or not the bias should be included. The code runs for 10000 epochs.

Graphs:

With Noise:



Without Noise:



Results:

- We can see that we have a decision boundary in both the graphs that separates the data properly.
- Therefore, we can see that the decision boundary exists for both the datasets, be it with noise or without noise.

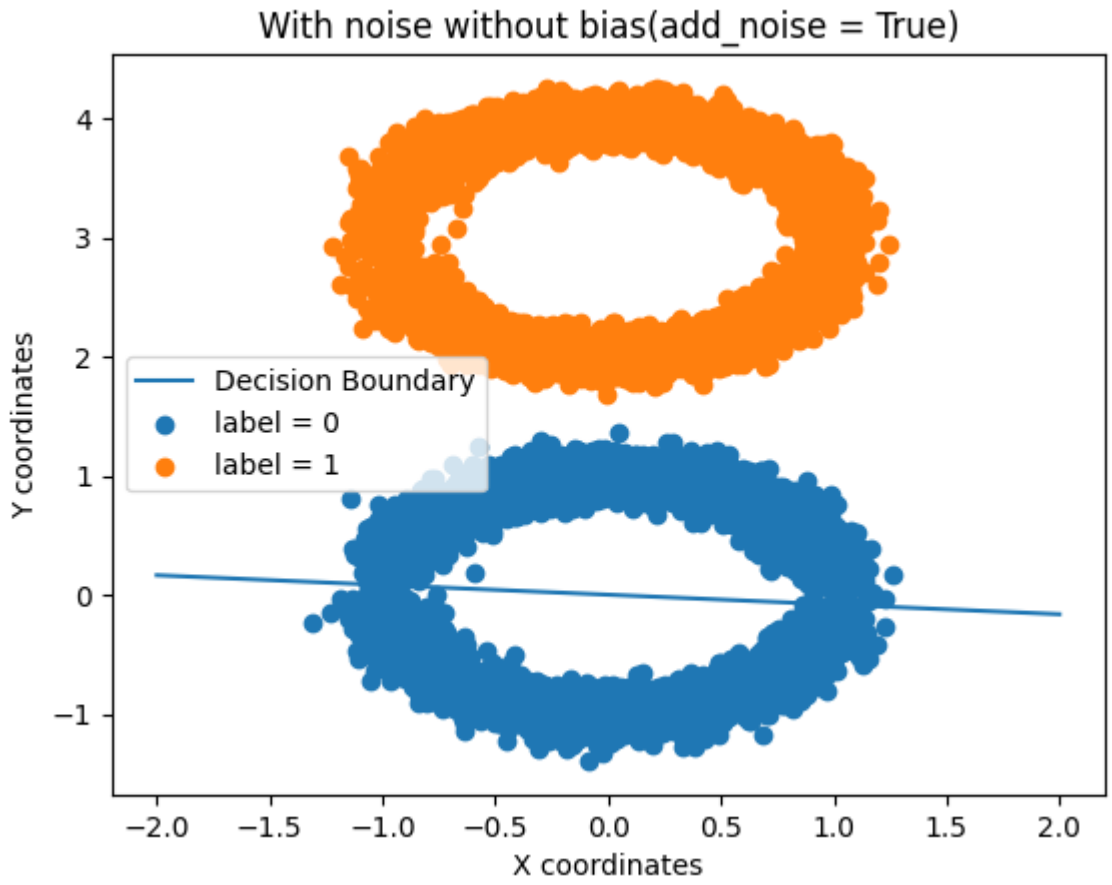
4.

Explanation of Code:

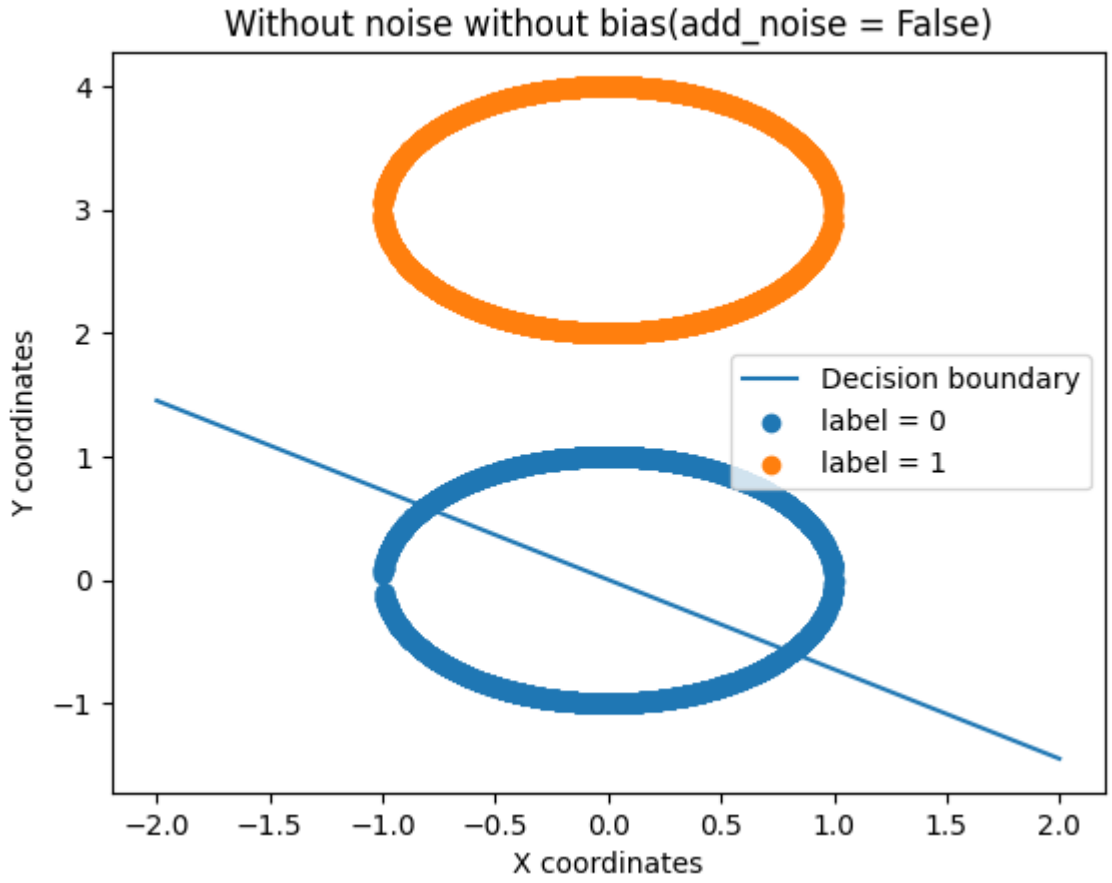
- For this part, we have set the bias to 0, so that the bias is not included. The only difference in the code compared to earlier is that bias is not included in the code.
- Without bias, the decision boundary will always pass through the origin.

Graphs:

With Noise:



Without Noise:



Results:

- We can see that the decision boundary does not separate the two circles in either case. It passes through the circle with label 0.
- This is because with bias 0, the decision boundary will always pass through the origin, and the first circle also has origin as centre, because of which it passes through the circle, and it cannot separate the two circles.
- We can see that the data does not get linearly separated in either case.

Comparison of the results with bias and without bias:

- We can observe that when we trained the model with bias, we got a decision boundary that linearly separated the data, whereas when we trained the model without bias, the data wasn't linearly separated.
- This is because when we include bias, the equation of the decision boundary can be ax+by=c, where a, b, c are constants, whereas when we do not include bias, the equation of the decision boundary becomes ax+by=0.
- We can see that when we remove bias, the constant term gets removed, which means the decision boundary is constrained to always pass through the origin.
- Since the circle with a label 0 has the origin as its centre, the decision boundary that passes through the origin will never be able to linearly separate the data, as it will not be able to classify all the points in the circle with label 0 correctly.
- Therefore, for the given data, if we set the bias as 0, we cannot get a decision boundary that linearly separates the data no matter how many epochs we run the algorithm for.

- Thus, for bias = 0, a decision boundary does not exist for this dataset.

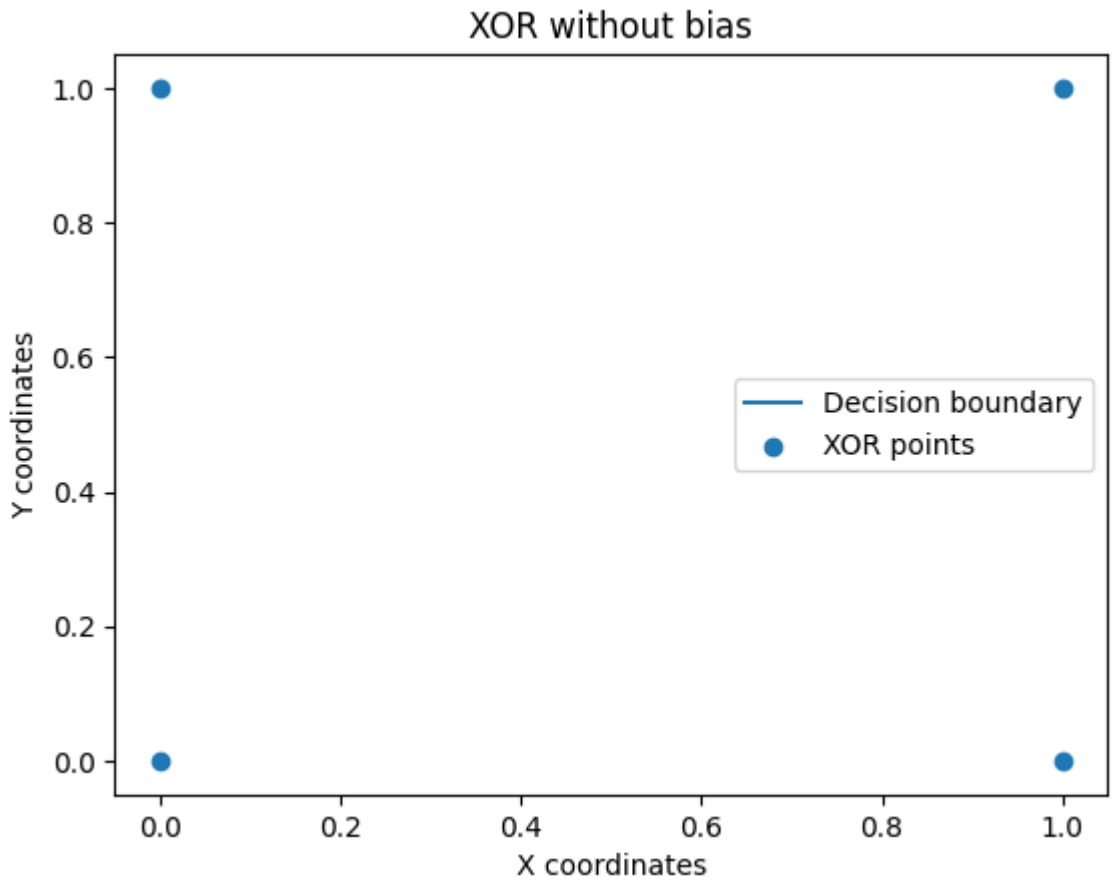
5.

Explanation of Code:

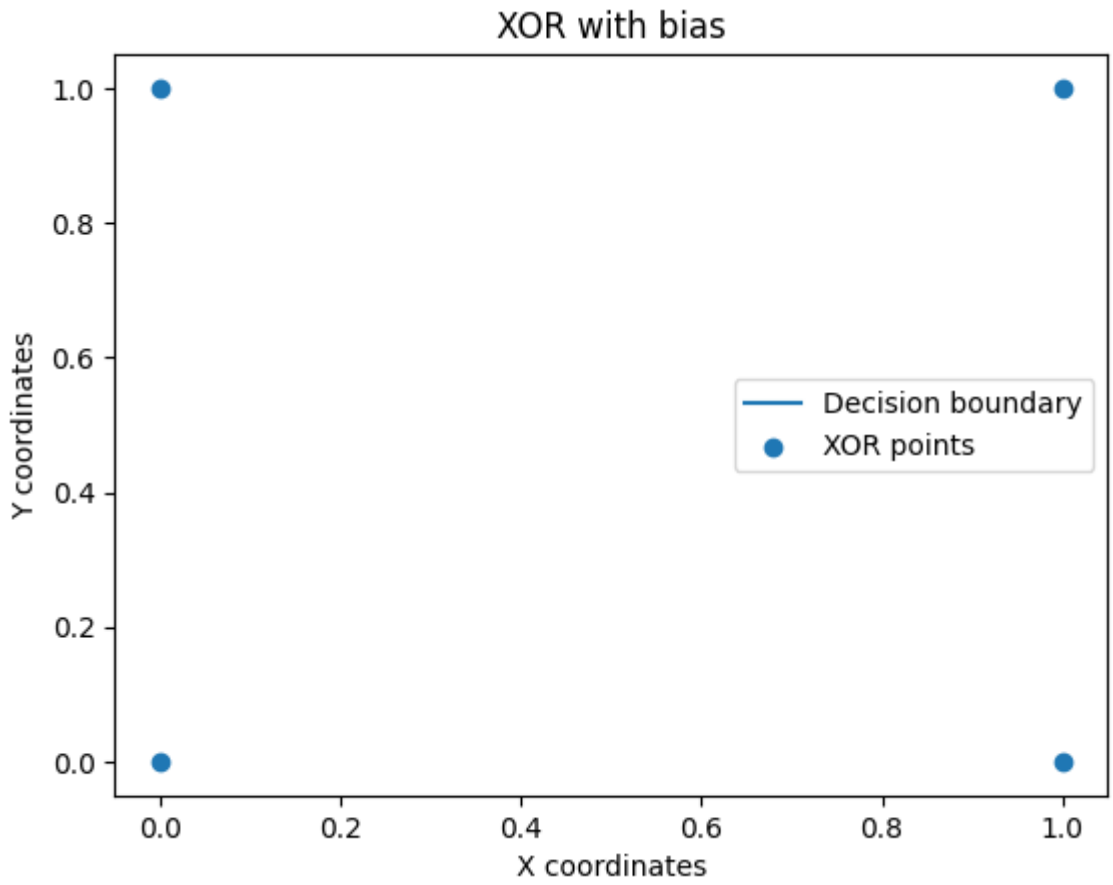
- First, we have created our numpy array, that contains all the required coordinates.
- For each property (XOR, AND, OR), we create the labels for the points, and then implement the perceptron training algorithm on them.
- If a decision boundary exists for these points, it will be plotted, but if the decision boundary does not exist, it will not be plotted.

Graphs:

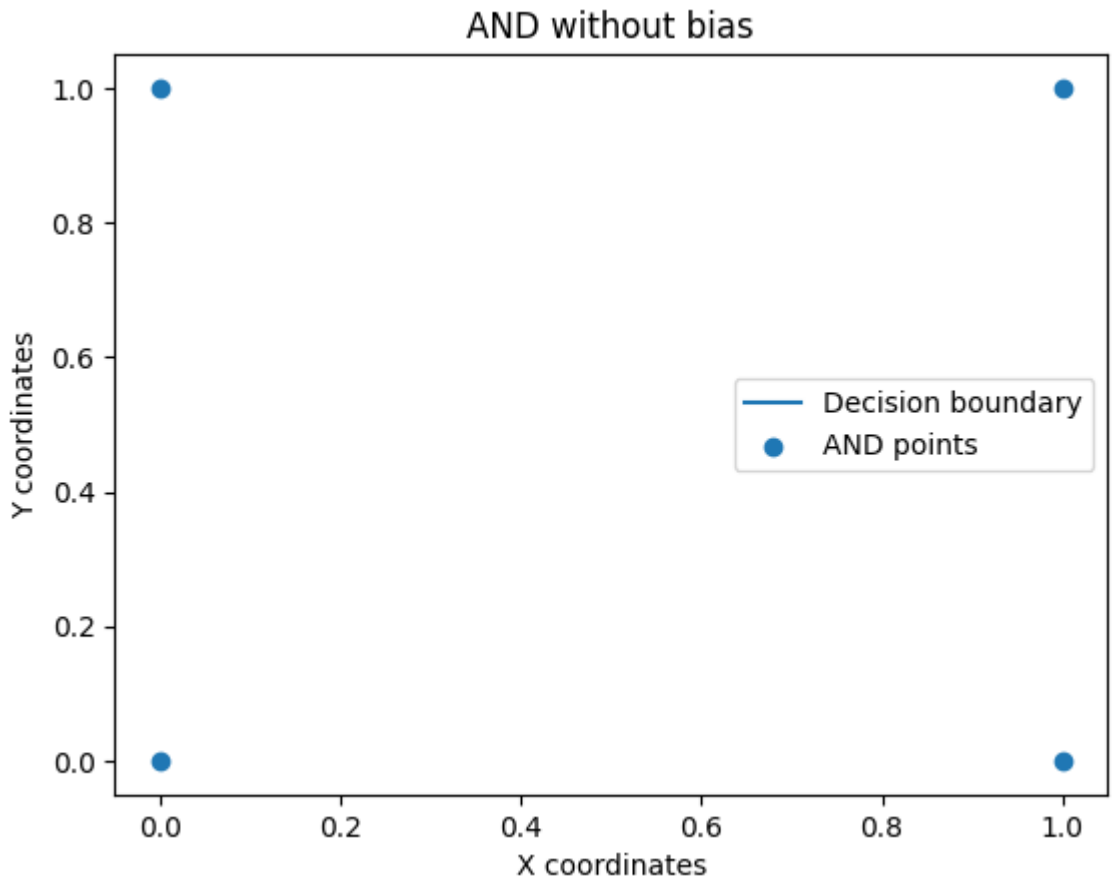
XOR without bias:



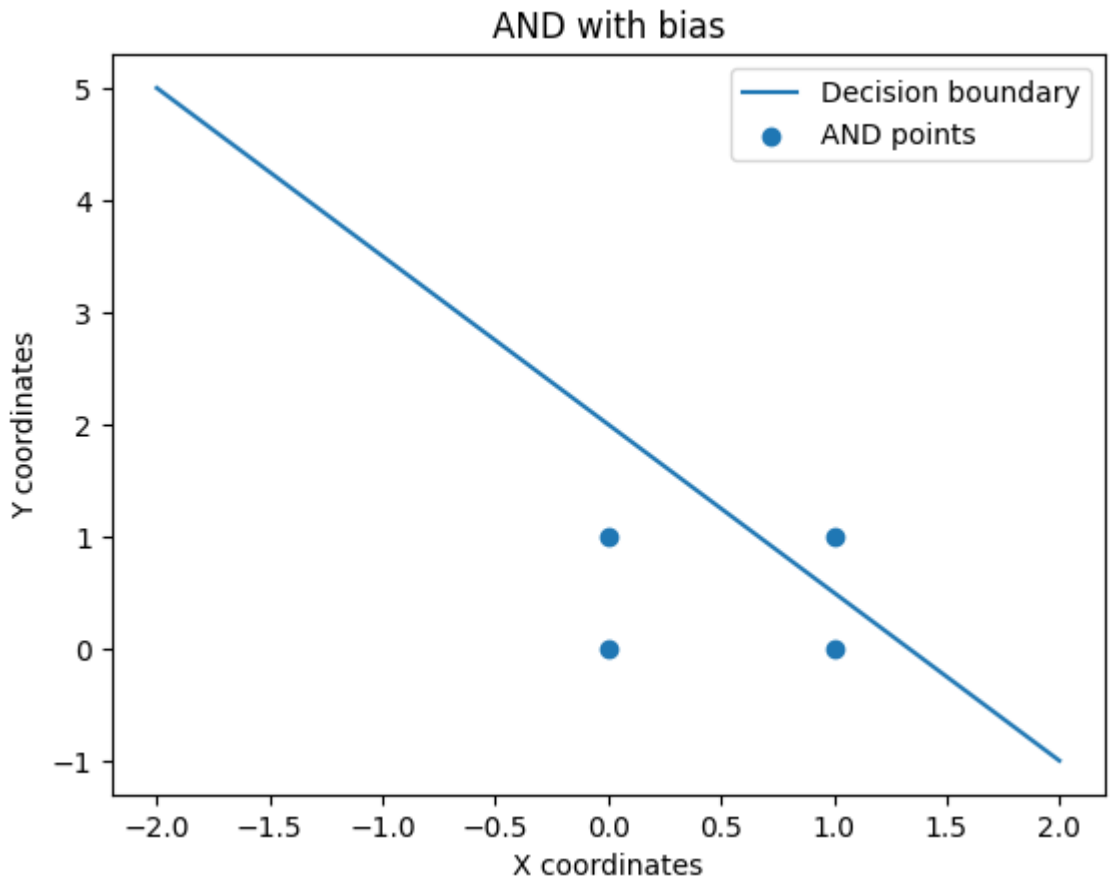
XOR with bias:



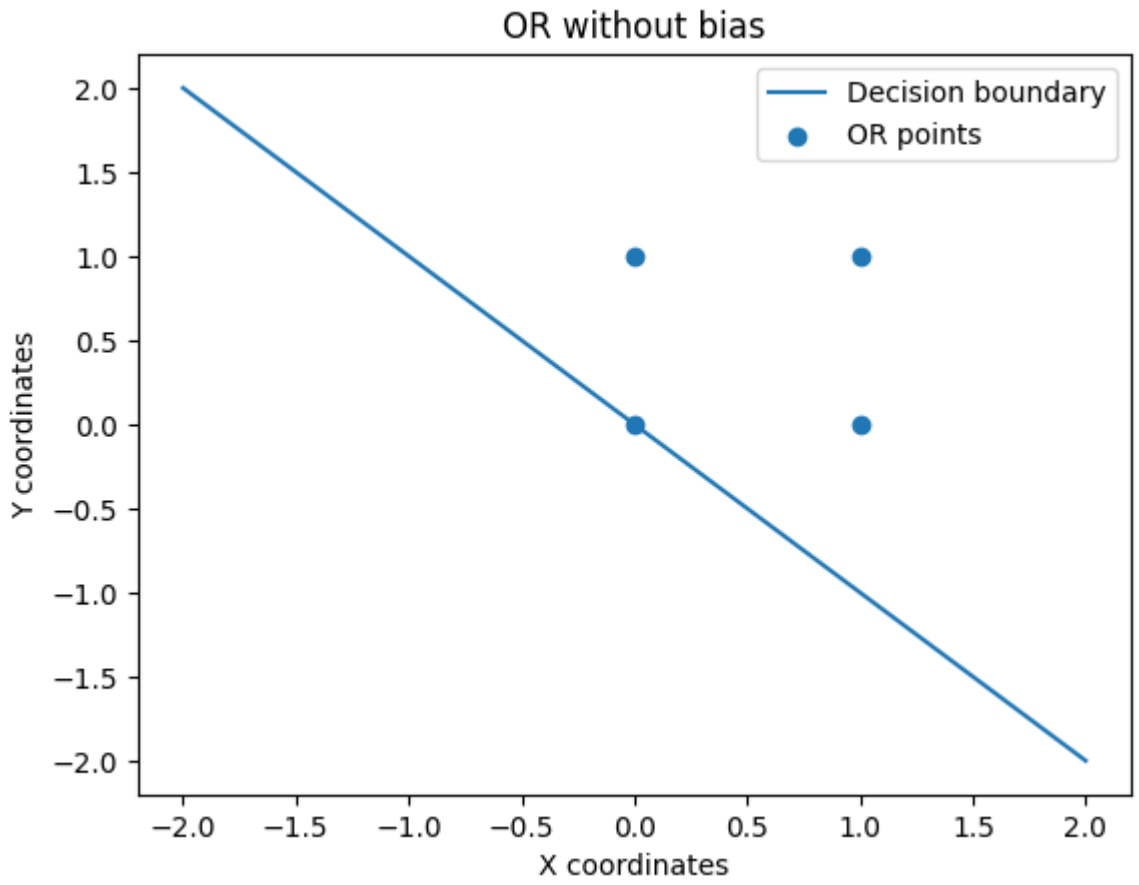
AND without bias:



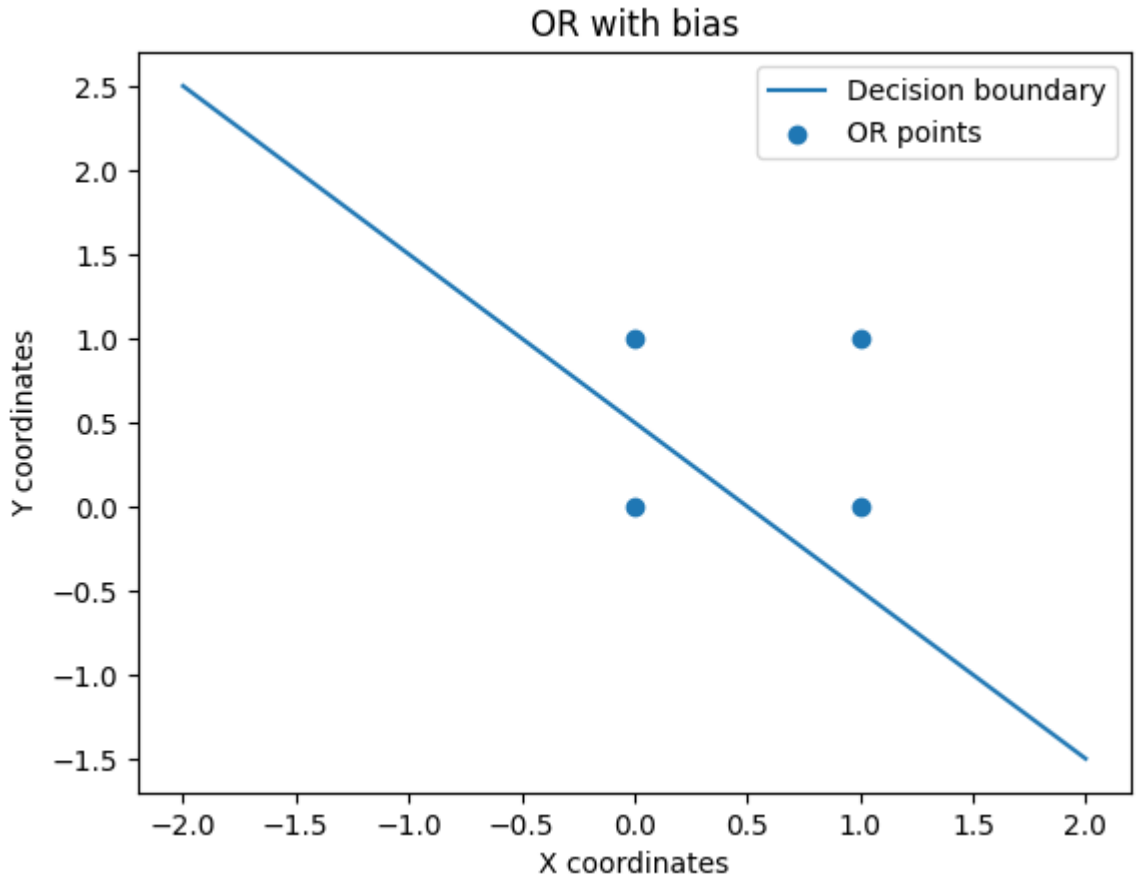
AND with bias:



OR without bias:


OR without bias
Decision boundary
OR points
Y coordinates
X coordinates
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

OR with bias:



Results:

- We can see that XOR doesn't have a decision boundary with or without bias.
- AND doesn't have decision boundary without bias, but has a decision boundary with bias. With bias, the decision boundary separates the point at (1,1) from the other three points and linearly separates them.
- OR has a decision boundary both with and without bias. With bias, the point at (0,0) is separated from the other three points and linearly separated. Without bias, the point at (0,0) lies on the line while the other three points lie on one side of the line. Here, we are considering the point on the line as lying on the other side of the line because of which the points are correctly classified.
- Thus, we can see that for the datasets where a decision boundary exists, the points are correctly classified.

6.

For an n dimensional plane, a hyperplane is an n-1 dimension subspace that divides it in two parts. We can classify the points on one side of the hyperplane as 0, and the points on the other side of the hyperplane as 1. The process is as follows:

- We can substitute the point in the equation of the hyperplane, and compute the value. Three cases will arise.
- If the value is 0, this means that the point lies on the hyperplane itself. We can assign the label for points lying on the hyperplane as 0 (we are making an assumption that the points on the hyperplane are being assigned a particular label).

- If the value is greater than 0, then the point lies on one side of the hyperplane. We can set the label for points on this side of the hyperplane as 1.
- If the value is smaller than, then the point lies on the other side of the hyperplane. We can se the label for points on this side f the hyperplane as 0.

To show this, we will take an example of three points and a hyperplane.

- Let the equation for hyperplane be x+y+z-3=0
- The point (1,1,1) has a value of 0, therefore it lies on the hyperplane itself. We can label this point as 0.
- The point (1,1,2) has a value of 1, therefore it lies on one side of the hyperplane. We can label this point as 1.
- The point (1,1,0) has a value of -1, therefore it lies on the other side of the hyperplane. We can label this point as 0.

Section C:

3.

a)

Explanation of Code:

- First, we import the necessary libraries, then we use read_csv to read the data from the dataset, and then convert it to a Dataframe. After that, we run some basic functions like describe(), info(), etc, to gain some basic information about our dataset.
- We handle the null values in the dataset by dropping the rows with null values from our dataframe. We can do this because our dataset is an extremely large dataset, with many instances, and losing a few of them wont cause any noticeable harm to the accuracy of the models we train on the dataset.
- After that, we use LabelEncoder to convert the address and label columns to numeric values. This is done to make it simpler to train the model on the dataset, as decision tree model in sklearn library does not work on columns with the type as strings.
- We separate the data into input and target columns, and then separate those into training, validation and testing sets manually.
- Now we train the decision tree models. We loop through the required values for maximum depth, and train decision trees with the required criterion (gini and entropy) and the required max_depth (4,8,10,15,20).
- Then, we make the predictions on the validation and the testing sets, and then report on the accuracy of our predictions.

Results:

```
(Testing set)Accuracy for max_depth = 4 and criterion gini is 0.9858630187083576
(Validation set)Accuracy for max_depth = 4 and criterion gini is 0.9858333047622313
(Testing set)Accuracy for max_depth = 8 and criterion gini is 0.9865692963508989
(Validation set)Accuracy for max_depth = 8 and criterion gini is 0.9865990102970251
(Testing set)Accuracy for max_depth = 10 and criterion gini is 0.9871338613272991
(Validation set)Accuracy for max_depth = 10 and criterion gini is 0.9869624347150319
(Testing set)Accuracy for max_depth = 15 and criterion gini is 0.9885167026662552
(Validation set)Accuracy for max_depth = 15 and criterion gini is 0.9883704186237872
(Testing set)Accuracy for max_depth = 20 and criterion gini is 0.9878744242922938
(Validation set)Accuracy for max_depth = 20 and criterion gini is 0.9880801362270145
(Testing set)Accuracy for max_depth = 4 and criterion entropy is 0.9857990194397779
(Validation set)Accuracy for max_depth = 4 and criterion entropy is 0.985771591181815
(Testing set)Accuracy for max_depth = 8 and criterion entropy is 0.9861624438577845
(Validation set)Accuracy for max_depth = 8 and criterion entropy is 0.9861304442234946
(Testing set)Accuracy for max_depth = 10 and criterion entropy is 0.9875041428097965
(Validation set)Accuracy for max_depth = 10 and criterion entropy is 0.9875178569387778
(Testing set)Accuracy for max_depth = 15 and criterion entropy is 0.9888458417618085
(Validation set)Accuracy for max_depth = 15 and criterion entropy is 0.9888024136867007
(Testing set)Accuracy for max_depth = 20 and criterion entropy is 0.9880001371412899
(Validation set)Accuracy for max_depth = 20 and criterion entropy is 0.987956709066182
```

We can see that we get the best accuracy score with the criterion entropy with max depth as 15 and the accuracy score is 0.9888458417618085 for the testing set, and for the validation set, we get the best accuracy score with the criterion entropy with max depth as 15 and the accuracy score is 0.9888024136867007

Therefore, we can see that we get the best results with the criterion set as entropy, and the max_depth set as 15.

Therefore, we will use these values in the following parts of the question.

b)

Explanation of Code:

- Ensembling is a technique, where we combine multiple weak classifiers, and then take the majority vote as our predicted output. In this way, we get a much better model than we would usually get. Ensembling greatly reduces variance, and helps increase the accuracy of our predictions.
- First, we create empty lists that will store the results of the predictions we make on the decision stump. Then, we run through a loop 100 times as we have 100 decision stumps.
- For each iteration, we randomly select 50% of the training data, and train the decision tree classifier on this data. For this classifier, we then train it on this data, and then append the prediction for validation and testing sets.
- Then, we convert these lists to numpy arrays as numpy arrays are faster than conventional lists.

- Then, we compare the majority vote of the prediction for each row of the set to the actual output, and then use that to predict the accuracy of our models.

Results:

```
(Testing set)Accuracy is 0.9857990194397779
(Validation set)Accuracy is 0.985771591181815
```

We can see that we are getting approximately 98% accuracy on both testing and validation set data. This is very high accuracy, and shows that our model is very good.

Comparison with part a:

- We can see that the accuracy using ensembling is also approximately 98% and is comparable to the best accuracy we got in part a, despite using multiple weaker decision tree classifiers.
- In part a, we used criterion entropy with the max depth as 15, whereas in part b, we used criterion entropy with the max depth as 3. Despite taking such a small maximum depth, we were able to get a comparable accuracy through the usage of ensembling techniques.
- Thus, we can see that ensembling by taking majority vote of a multiple number of weak classifiers is an effective method for improving the accuracy of our machine learning models.

c)

Explanation of Code:

- We are given the number of estimators we need to take for each of the classifiers.
- Adaboost is a popular boosting technique where multiple classifiers are used, and each classifier tries to learn from the mistakes of the previous classifier. Whenever an incorrect prediction is made, it is given more weight for next classifier so that it may be corrected.
- We iterate through the estimators, and train our classifiers on the training data, and then make predictions on the testing and validation data.
- Finally, we report our accuracy for the models.

Results:

```
(Testing set)Accuracy for n_estimators = 4 is 0.988669843773214
(Validation set)Accuracy for n_estimators = 4 is 0.9887361287299574
(Testing set)Accuracy for n_estimators = 8 is 0.9874995714334693
(Validation set)Accuracy for n_estimators = 8 is 0.9876481411641009
(Testing set)Accuracy for n_estimators = 10 is 0.987595570336339
(Validation set)Accuracy for n_estimators = 10 is 0.9877052833681901
(Testing set)Accuracy for n_estimators = 15 is 0.987558999325722
(Validation set)Accuracy for n_estimators = 15 is 0.98750642849796
(Testing set)Accuracy for n_estimators = 20 is 0.9879658518188363
(Validation set)Accuracy for n_estimators = 20 is 0.9878149964000411
(Testing set)Accuracy for n_estimators = 4 is 0.9876092844653204
(Validation set)Accuracy for n_estimators = 4 is 0.9876549982285917
(Testing set)Accuracy for n_estimators = 8 is 0.9873715728963097
(Validation set)Accuracy for n_estimators = 8 is 0.9874538576701981
(Testing set)Accuracy for n_estimators = 10 is 0.9872870024342579
(Validation set)Accuracy for n_estimators = 10 is 0.9874218580359082
(Testing set)Accuracy for n_estimators = 15 is 0.9873670015199826
(Validation set)Accuracy for n_estimators = 15 is 0.9874927143689787
(Testing set)Accuracy for n_estimators = 20 is 0.98739442977794454
(Validation set)Accuracy for n_estimators = 20 is 0.9875087141861236
```

- We can see that we have received approximately 99% accuracy for all the models. We can see that we have gotten amazing results, and the models have been well trained.
- The maximum accuracy for Adaboost is when the number of estimators is 4. The accuracy for testing set is 0.988669843773214, and the accuracy for validation set is 0.9887361287299574
- The maximum accuracy for Random Forest is when the number of estimators is 4. The accuracy for testing set is 0.9876092844653204, and the accuracy for validation set is 0.9876549982285917
- We can see that the maximum accuracy for adaboost is higher than the maximum accuracy for random forest.