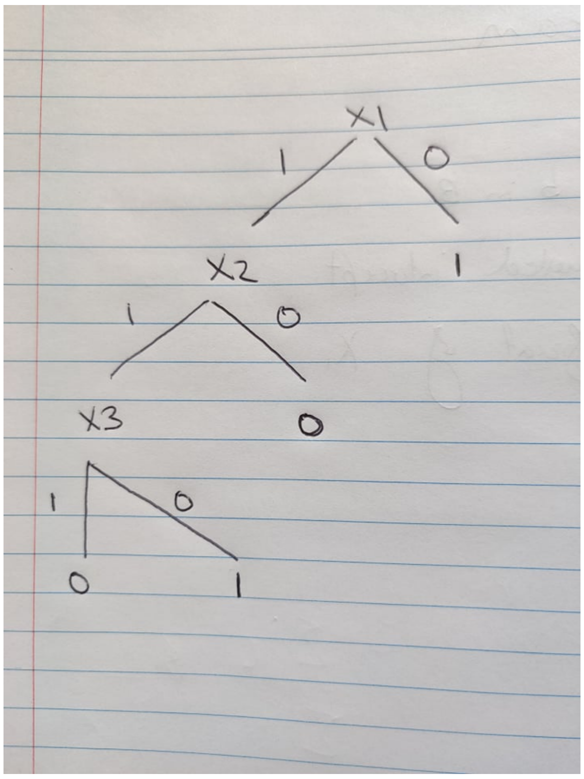
Question 1

a



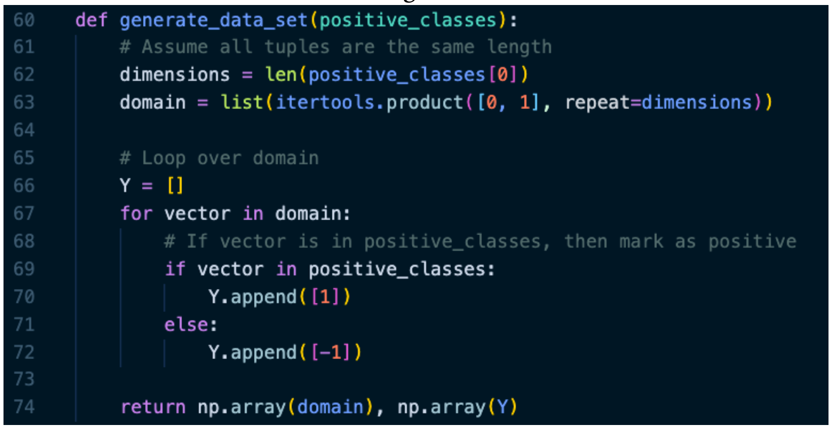
i.From the working here 0.1, we can see that X_1 has the largest gain when splitting the data and therefore should be the first branch. X_2 and X_3 both have 0 gain. For the next split at $X_{1,1}$, both X_2 and X_3 produce the same gain which is $gain(X_{1,1}, X_2) = gain(X_{1,1}, X_3) = 0.39317$ which is why it doesn't matter which one is selected. I chose X_2 to be the next split and then finished the tree.

The training error for this tree is 0 since our tree uses every feature and there are no observations where the exact same values for each feature is given but it maps to a separate output.

ii. There is no depth 2 decision tree that has a lower training error than the one found with ID3. The training error is already the lowest it can be at 0. This tells us that ID3 can fit any dataset with no conflicting classifications and therefore has very high complexity. Without specifying the ID3 algorithm to stop early and with a non-conflicting dataset, ID3 can always achieve 0 training error.

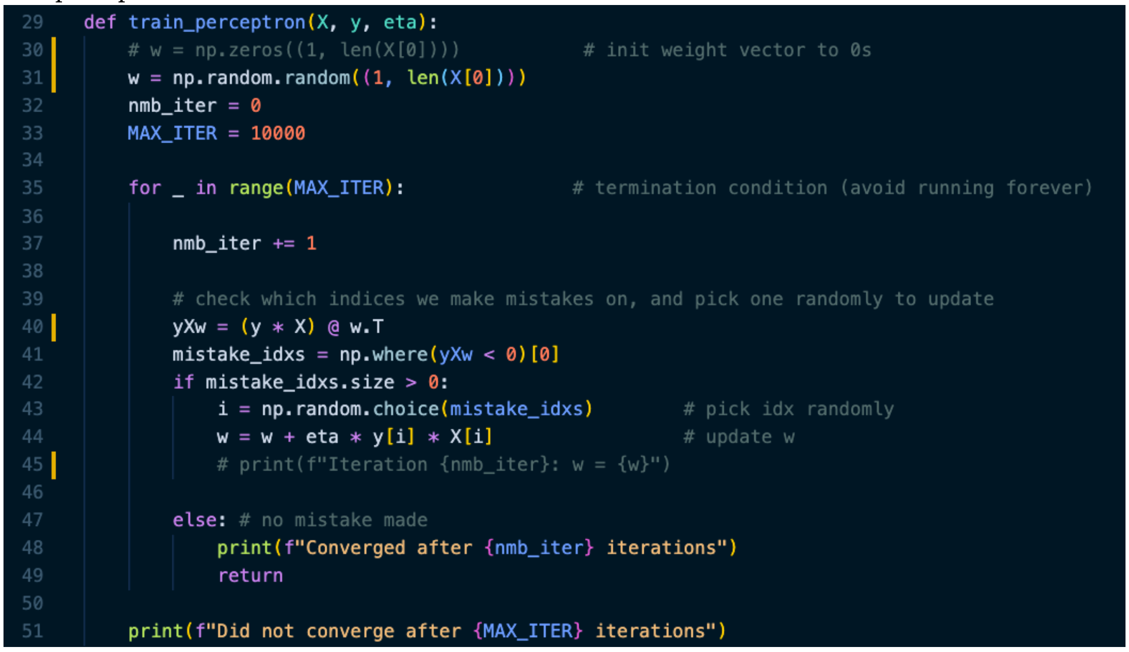
b

I generated the data with the following code:



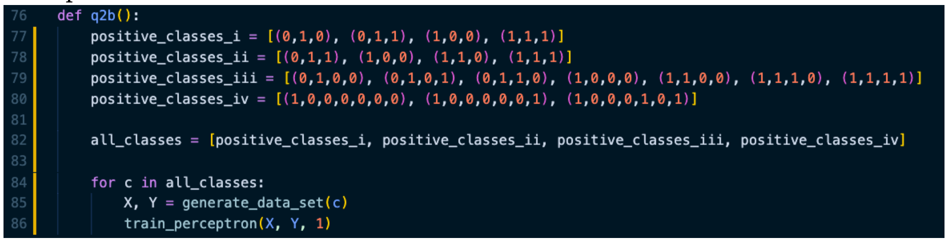
```python
def generate_data_set(positive_classes):
    # Assume all tuples are the same length
    dimensions = len(positive_classes[0])
    domain = list(itertools.product([0, 1], repeat=dimensions))

    # Loop over domain
    Y = []
    for vector in domain:
        # If vector is in positive_classes, then mark as positive
        if vector in positive_classes:
            Y.append([1])
        else:
            Y.append([-1])

    return np.array(domain), np.array(Y)
```

The perceptron is then trained with:



```python
def train_perceptron(X, y, eta):
    # w = np.zeros((1, len(X[0])))          # init weight vector to 0s
    w = np.random.random((1, len(X[0])))
    nmb_iter = 0
    MAX_ITER = 10000

    for _ in range(MAX_ITER):               # termination condition (avoid running forever)

        nmb_iter += 1

        # check which indices we make mistakes on, and pick one randomly to update
        yXw = (y * X) @ w.T
        mistake_idxs = np.where(yXw < 0)[0]
        if mistake_idxs.size > 0:
            i = np.random.choice(mistake_idxs)          # pick idx randomly
            w = w + eta * y[i] * X[i]                   # update w
            # print(f"Iteration {nmb_iter}: w = {w}")

        else: # no mistake made
            print(f"Converged after {nmb_iter} iterations")
            return

    print(f"Did not converge after {MAX_ITER} iterations")
```

The spaces were then tested with:

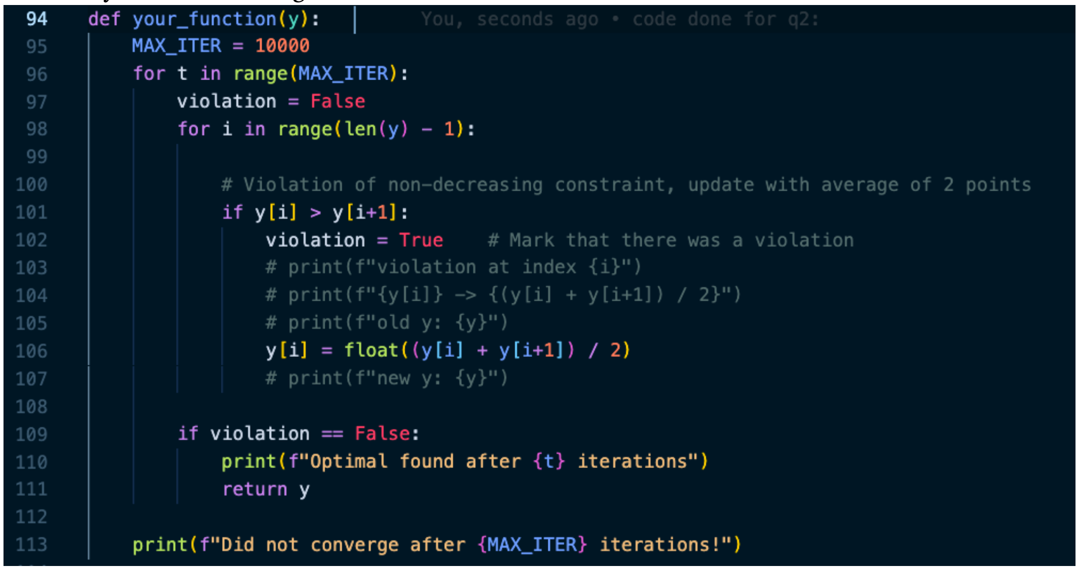


```python
def q2b():
    positive_classes_i = [(0,1,0), (0,1,1), (1,0,0), (1,1,1)]
    positive_classes_ii = [(0,1,1), (1,0,0), (1,1,0), (1,1,1)]
    positive_classes_iii = [(0,1,0,0), (0,1,0,1), (0,1,1,0), (1,0,0,0), (1,1,0,0), (1,1,1,0), (1,1,1,1)]
    positive_classes_iv = [(1,0,0,0,0,0,0), (1,0,0,0,0,0,1), (1,0,0,0,1,0,1)]

    all_classes = [positive_classes_i, positive_classes_ii, positive_classes_iii, positive_classes_iv]

    for c in all_classes:
        X, Y = generate_data_set(c)
        train_perceptron(X, Y, 1)
```

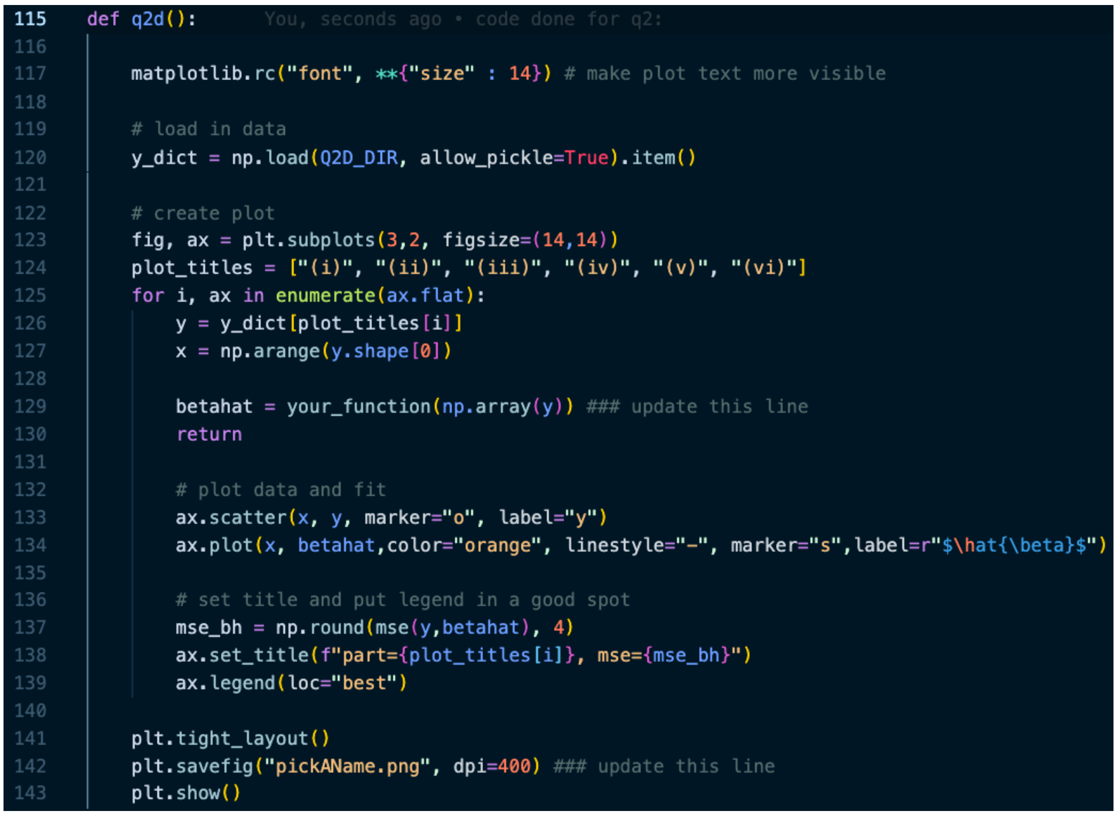
Note that this produces **non-linearly separable** for all spaces which I'm assuming is not correct. The issues is that the vector for **w** doesn't iterate in a direction that separates the data better and keeps moving further and further away from the classifications.

d

I had already started working on this before its removal. This is what I had:



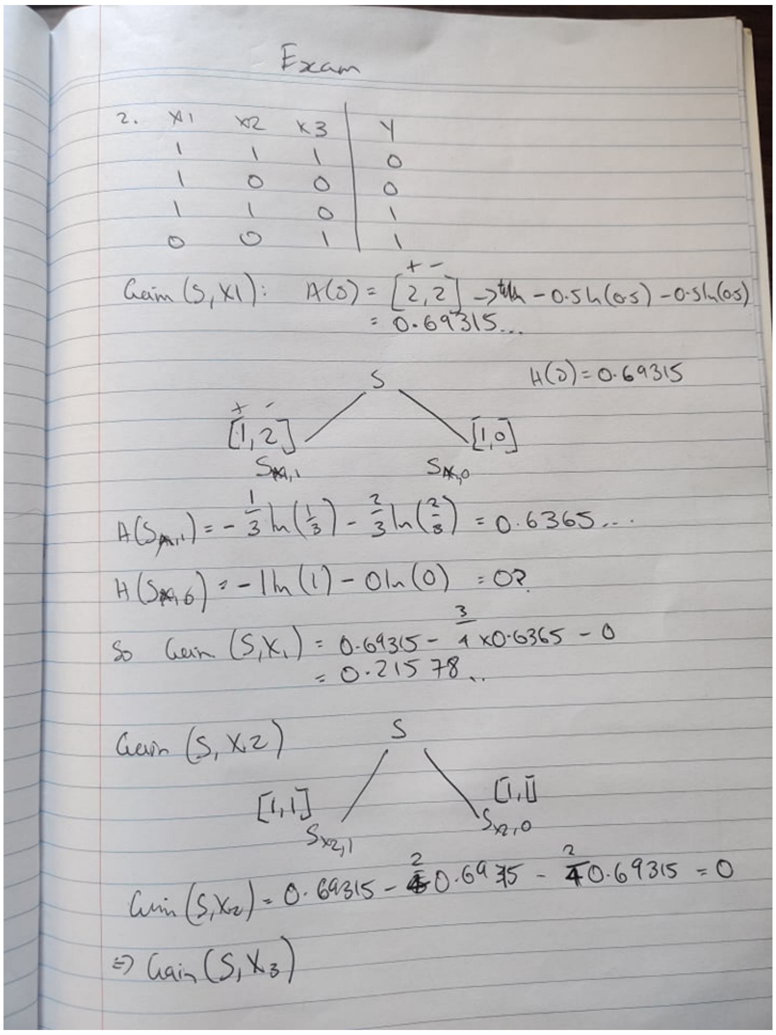
```python
def your_function(y):       You, seconds ago • code done for q2:
    MAX_ITER = 10000
    for t in range(MAX_ITER):
        violation = False
        for i in range(len(y) - 1):

            # Violation of non-decreasing constraint, update with average of 2 points
            if y[i] > y[i+1]:
                violation = True    # Mark that there was a violation
                # print(f"violation at index {i}")
                # print(f"{y[i]} -> {(y[i] + y[i+1]) / 2}")
                # print(f"old y: {y}")
                y[i] = float((y[i] + y[i+1]) / 2)
                # print(f"new y: {y}")

        if violation == False:
            print(f"Optimal found after {t} iterations")
            return y

    print(f"Did not converge after {MAX_ITER} iterations!")
```



```python
def q2d():      You, seconds ago • code done for q2:

    matplotlib.rc("font", **{"size" : 14}) # make plot text more visible

    # load in data
    y_dict = np.load(Q2D_DIR, allow_pickle=True).item()

    # create plot
    fig, ax = plt.subplots(3,2, figsize=(14,14))
    plot_titles = ["(i)", "(ii)", "(iii)", "(iv)", "(v)", "(vi)"]
    for i, ax in enumerate(ax.flat):
        y = y_dict[plot_titles[i]]
        x = np.arange(y.shape[0])

        betahat = your_function(np.array(y)) ### update this line
        return

        # plot data and fit
        ax.scatter(x, y, marker="o", label="y")
        ax.plot(x, betahat,color="orange", linestyle="-", marker="s",label=r"$\hat{\beta}$")

        # set title and put legend in a good spot
        mse_bh = np.round(mse(y,betahat), 4)
        ax.set_title(f"part={plot_titles[i]}, mse={mse_bh}")
        ax.legend(loc="best")

    plt.tight_layout()
    plt.savefig("pickAName.png", dpi=400) ### update this line
    plt.show()
```

Appendix

0.1 q2a



Exam

2. X1	X2	X3	Y
1	1	1	0
1	0	0	0
1	1	0	1
0	0	1	1

Gain $(S, X1)$: $H(S) = \overset{+\ -}{[2,2]} \rightarrow -0.5\ln(0.5) - 0.5\ln(0.5)$
$= 0.69315...$

$H(S) = 0.69315$

$$\begin{tikzcd} & S \arrow[dl, dash] \arrow[dr, dash] & \\ \overset{+\ -}{[1,2]} & & {[1,0]} \end{tikzcd}$$

$S_{x1,1}$ $S_{x1,0}$

$H(S_{x1,1}) = -\frac{1}{3}\ln\left(\frac{1}{3}\right) - \frac{2}{3}\ln\left(\frac{2}{3}\right) = 0.6365...$

$H(S_{x1,0}) = -1\ln(1) - 0\ln(0) = 0?$

So Gain $(S, X_1) = 0.69315 - \frac{3}{4} \times 0.6365 - 0$
$= 0.21578..$

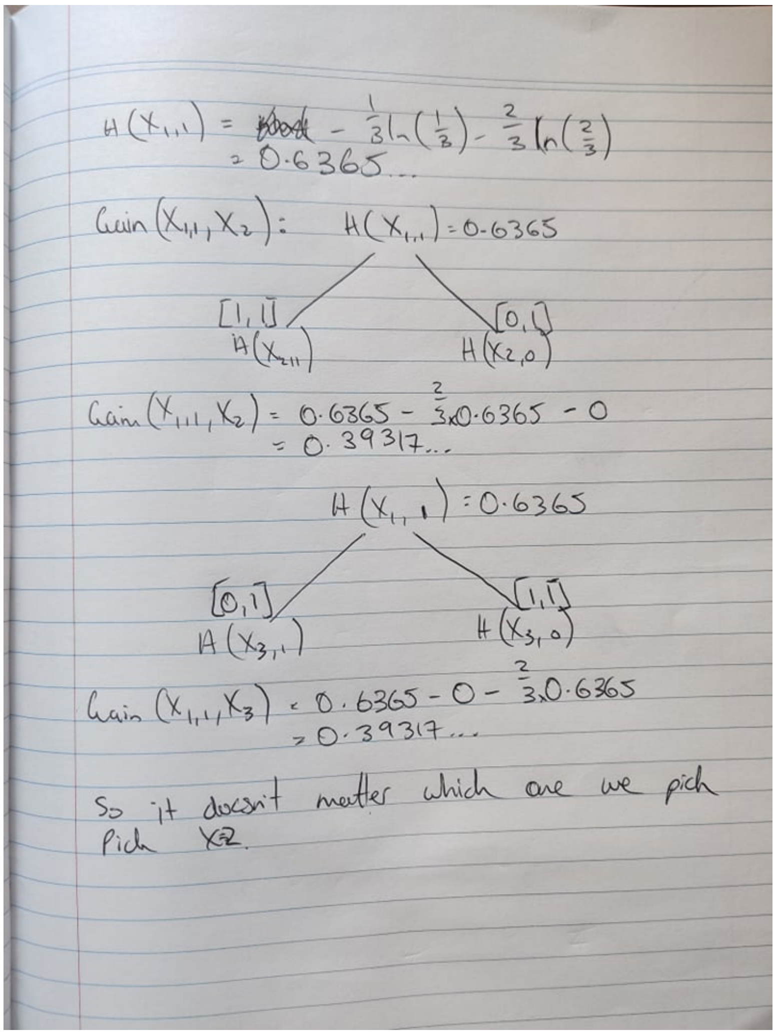
Gain $(S, X2)$

$$\begin{tikzcd} & S \arrow[dl, dash] \arrow[dr, dash] & \\ {[1,1]} & & {[1,1]} \end{tikzcd}$$

$S_{x2,1}$ $S_{x2,0}$

Gain $(S, X_2) = 0.69315 - \frac{2}{4}0.69315 - \frac{2}{4}0.69315 = 0$

$\Rightarrow$ Gain (S, X_3)



$H(X_{1,1}) = \cancel{[illegible]} - \frac{1}{3}\ln\left(\frac{1}{3}\right) - \frac{2}{3}\ln\left(\frac{2}{3}\right)$
$= 0.6365\ldots$

$\mathrm{Gain}(X_{1,1}, X_2)$:

$$\begin{tikzcd}
 & H(X_{1,1}) = 0.6365 \arrow[dl, no head] \arrow[dr, no head] & \\
{[1,1]} \; H(X_{2,1}) & & {[0,1]} \; H(X_{2,0})
\end{tikzcd}$$

$\mathrm{Gain}(X_{1,1}, X_2) = 0.6365 - \frac{2}{3} \times 0.6365 - 0$
$= 0.39317\ldots$

$$\begin{tikzcd}
 & H(X_{1,1}) = 0.6365 \arrow[dl, no head] \arrow[dr, no head] & \\
{[0,1]} \; H(X_{3,1}) & & {[1,1]} \; H(X_{3,0})
\end{tikzcd}$$

$\mathrm{Gain}(X_{1,1}, X_3) = 0.6365 - 0 - \frac{2}{3} \times 0.6365$
$= 0.39317\ldots$

So it doesn't matter which one we pick
Pick X2.