Q1)

Initialization: $\alpha_i = 0$

Repeat until convergence:

for $i = 1, \dots N$

update $\alpha_i = \alpha_i - \eta \frac{\partial \ell(\vec{y}_i, y_i)}{\partial \alpha_i}$

Classification function of a test point $\vec{x}$:

$f(\vec{x}) = \sum_{j=1}^{\tilde{n}} \alpha_j k(x_j, \vec{x})$

Q2)

a) $P(yes) = \frac{4}{8} = 0.5$, $P(NO) = \frac{4}{8} = 0.5$, $H(Y) = -\sum_{c \in \{yes, no\}} P_c \log_2(P_c)$

$H(Y) = -(0.5 \log_2(0.5) + 0.5 \log_2(0.5)) = -(0.5 \cdot (-1) + 0.5 \cdot (-1)) = 1.0$, $\boxed{H(Y) = 1.0}$

b) $H(Y|X) = P(X=T)H(Y|X=T) + P(X=F)H(Y|X=F)$

3P% > 0.23

$P(X_1 = T) = \frac{2}{8} = 0.25$, $y = 2, N = 4$

$P(yes|X_1=F) = \frac{2}{6} = \frac{1}{3}$, $P(NO|X_1=F) = \frac{2}{3}$

$H(Y|X_1=F) = -(\frac{1}{3}\log_2(\frac{1}{3}) + \frac{2}{3}\log_2(\frac{2}{3})) = 0.918$

$P(X_1 = F) = 0.75$

$H(Y|X_1) = 0.25 \times 0 + 0.75 \times 0.918 = \boxed{0.6885}$

SPG > 7.8

$P(yes|X_2=T) = 3/4 = 0.75$, $y = 3, N = 1$

$P(NO|X_2=T) = 0.25$

$H(Y|X_2=T) = -(0.75 \log_2(0.75) + 0.25 \log_2(0.25)) = 0.8113$, $P(X_2=T) = \frac{4}{8} = 0.5$

$P(yes|X_2=F) = 1/4 = 0.25$, $P(NO|X_2=F) = 0.75$ $y = 1, N = 3$

$H(Y|X_2) = 0.5 \times 0.8113 + 0.5 \times 0.8113 = \boxed{0.8113}$

RPG > 44

$P(yes|X_3=T) = 0.75$, $P(NO|X_3=T) = 0.25$, $H(Y|X_3=T) = 0.8113$,

$P(X_3=T) = \frac{4}{8} = 0.5$,

Same as SPG > 7.8,

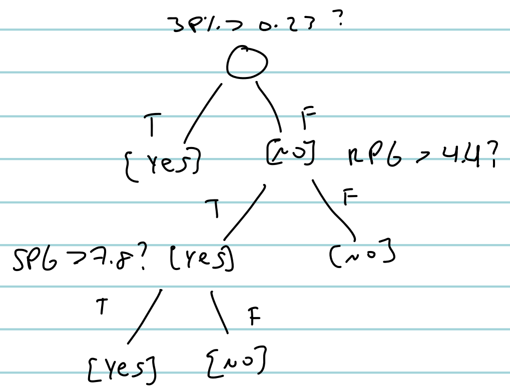
$H(Y|X_3) = \boxed{0.8113}$

c) 3P% > .23 : $1 - 0.6885 = 0.3115$

SPG > 7.8 : $1 - 0.8113 = 0.1887$

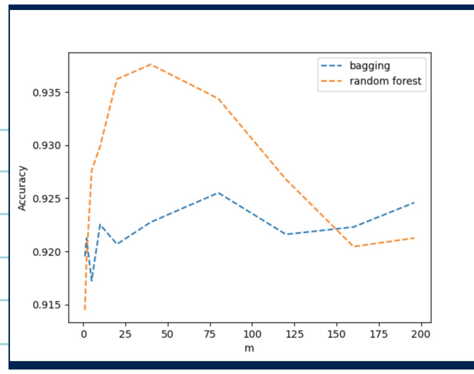
RPG > 44 : $1 - 0.8113 = 0.1887$

d)



Q3)

```
# TODO: Implement this function
#THIS HOMEWORK IS MY OWN WORK, WRITTEN WITHOUT COPYING FROM OTHER STUDENTS
#OR DIRECTLY FROM LARGE LANGUAGE MODELS SUCH AS CHATGPT.
#Any collaboration or external resources have been properly acknowledged.
#<add details>
#/* Jonatan Peguero  */
y_preds = []
n_train = X_train.shape[0]
for _ in range(n_clf):
    indicies = np.random.choice(n_train, size = n_train, replace = True)
    Xb = X_train[indicies]
    yb = y_train[indicies]
    tree = DecisionTreeClassifier(criterion = 'entropy', max_features = m)
    tree.fit(Xb, yb)
    y_pred = tree.predict(X_test)
    y_preds.append(y_pred)
y_pred_ensemble = majority_vote(y_preds)
accuracy = np.mean(y_pred_ensemble == y_test)
return accuracy
```



When m is small, Random Forest outperforms bagging because it encourages greater diversity among the trees. But as m gets large, the random forest curve starts declining. This is due to it losing it's randomness, and bagging starts outperforming random forest b/c it uses all of the features. We can also see that bagging remains pretty stable as m increases, while random forest does not.

Q4)

a)

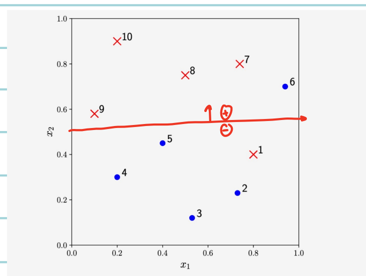


Figure 1: •-negative points and ×-positive points.

Horizontal line at $y = 0.55$

b)

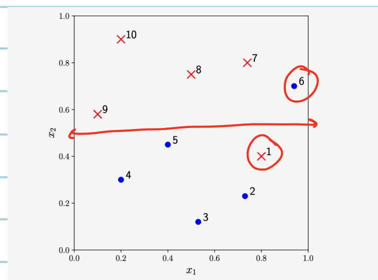


Figure 1: •-negative points and ×-positive points.

points 6 and 1 would have the largest weights

c) $\epsilon_1 = \frac{2}{10} = 0.2 = 20\%$

$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-0.2}{0.2}\right) = \frac{1}{2}\ln\left(\frac{0.8}{0.2}\right) = \frac{1}{2}\ln(4) \approx \boxed{0.693}$

d)

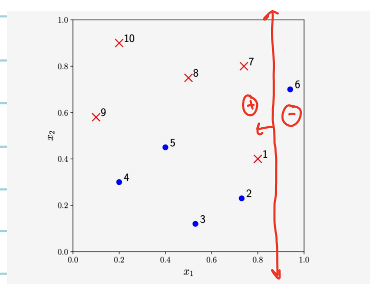
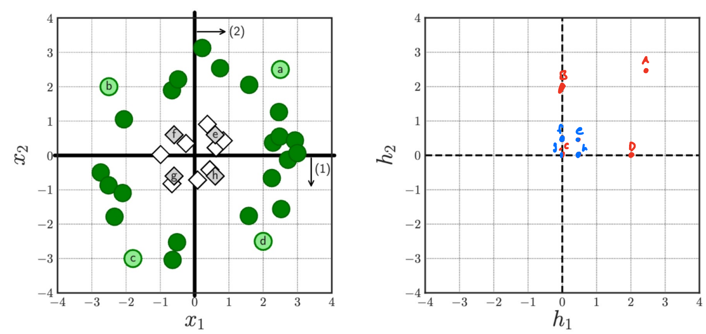


Figure 1: •-negative points and ×-positive points.

e) The final ensemble prediction is made by taking the sign of the weighted sum, so b/c the second stump corrects the errors from the first, the combination of both should correctly classify all of the points

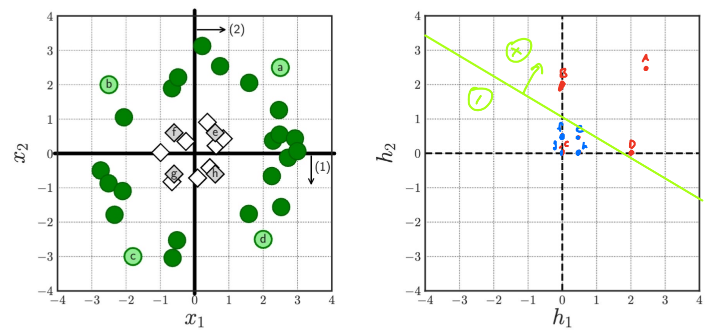
Q5

a)



* b and c are both at (0,0)

b)



The minimal misclassification error we can achieve would be $\frac{1}{8}$, or 12.5% b/c points c or b will always be misclassified.

c) No, because for points c and b, they map to (0,0) so every layer after that would see that as the input, and they won't be linearly seperable after that b/c they will have the same input.

D) Yes you can. For example, with points c and b, the 3 ReLU outputs make the output layer a linear classifier in $\mathbb{R}^3$, so with that, c can map to some $(0,0,h_3^c)$, while b can map to (0,0,0), which can allow us to linearly seperate them