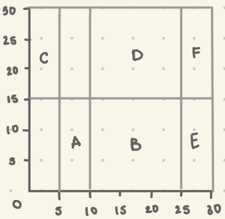
Homework 3 → Decision Trees and K-Means Clustering

Q 1 (a)



A → $x_1 \in [0, 10)$
$x_2 \in [0, 15)$

B → $x_1 \in [10, 25)$
$x_2 \in [0, 15)$

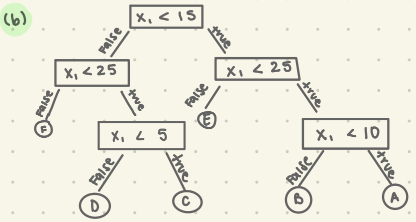
C → $x_1 \in [0, 5)$
$x_2 \in [15, 30]$

D → $x_1 \in [5, 25)$
$x_2 \in [15, 30]$

E → $x_1 \in [25, 30]$
$x_2 \in [0, 15]$

F → $x_1 \in [25, 30]$
$x_2 \in [15, 30]$

(b)



(c) It is difficult to find an accurrate tree with this data set. To improve the accuracy we need to re duce the overfitting of the graph by pruning. Pruning is when you remove sections that do not hold much power over classification.

Q 2 Information gain (ig) = E (target) - E (target, ___)

Information gain A :

A	T	0	1
0	3	2	1
1	3	1	2

$$E(0) = -\left[\tfrac{2}{3}\log\tfrac{2}{3} + \tfrac{1}{3}\log\tfrac{1}{3}\right]$$
$$= -\left[\tfrac{2}{3}(\log(2) - \log(3)) + \tfrac{1}{3}(\log(1) - \log(3))\right]$$
$$= -\left[\tfrac{2}{3}(-0.58) + \tfrac{1}{3}(-1.58)\right]$$
$$= -[-0.9]$$
$$= 0.9$$

$$E(1) = -\left(\tfrac{1}{3}\log\tfrac{1}{3} + \tfrac{2}{3}\log\tfrac{2}{3}\right)$$
$$= -\left[\tfrac{1}{3}(\log(1) - \log(3)) + \tfrac{2}{3}(\log(2) - \log(3))\right]$$
$$= -\left[\tfrac{1}{3}(-1.58) + \tfrac{2}{3}(-0.58)\right]$$
$$= -[-0.9]$$
$$= 0.9$$

$$E(Y, A) = \Sigma\, P(\ )\cdot \log P(x)$$
$$= P(0)\cdot E(0) + P(1)\cdot E(1)$$
$$= \tfrac{3}{6}(0.9) + \tfrac{3}{6}(0.9)$$
$$= 0.45 + 0.45 \rightarrow 0.9$$

Information gain $= 1 - 0.9$
$= 0.1$

Information gain B :

B	T	0	1
0	2	1	1
1	4	2	2

$$E(0) = -\left(\tfrac{1}{2}\log\tfrac{1}{2} + \tfrac{1}{2}\log\tfrac{1}{2}\right)$$
$$= -\left[\tfrac{1}{2}(\log(1) - \log(2)) + \tfrac{1}{2}(\log(1) - \log(2))\right]$$
$$= -[-1/2 - 1/2]$$
$$= 1$$

$$E(1) = \left(\tfrac{2}{4}\log\tfrac{2}{4} + \tfrac{2}{4}\log\tfrac{2}{4}\right)$$
$$= -\left[\tfrac{2}{4}(\log(2) - \log(4)) + \tfrac{2}{4}(\log(2) - \log(4))\right]$$
$$= -\left[2/4(-1) + \tfrac{2}{4}(-1)\right]$$
$$= 1$$

$$E(y, B) = \tfrac{2}{3}(1) + \tfrac{4}{3}(1)$$
$$= \tfrac{2}{6} + \tfrac{4}{6}$$
$$= 0.9$$

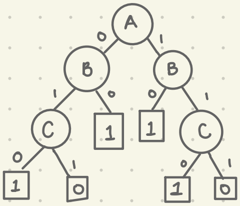
Information gain $= 1 - 0.9$
$= 0.1$

Information gain C :

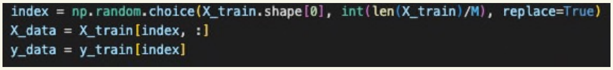
C	T	0	1
0	3	1	2
1	3	2	1

$E(0) = 0.9$ (same as A for $E(1)$)
$E(1) = 0.9$ (same as A fore $E(0)$)

All the atributes have the same Information gain so the tree can be made In any order. On example could look like the following so every possible out come can be predicted...

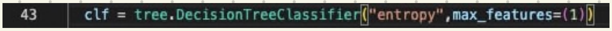


Q3 In decison.py this is what I modified :



```
index = np.random.choice(X_train.shape[0], int(len(X_train)/M), replace=True)
X_data = X_train[index, :]
y_data = y_train[index]
```

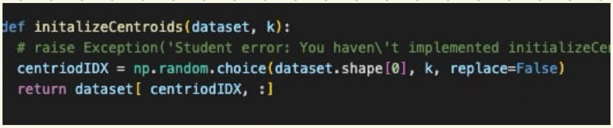
This is under the "modify in here to decrease the correlation"



```
43      clf = tree.DecisionTreeClassifier("entropy",max_features=(1))
```

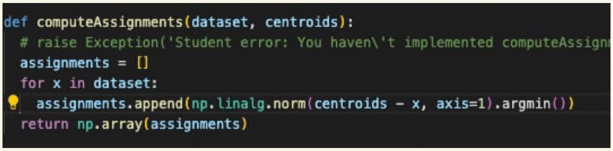
→ this is line 43 so it now includes max_ Features

Q4 (a)



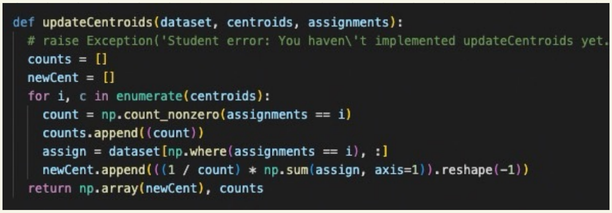
```
def initalizeCentroids(dataset, k):
  # raise Exception('Student error: You haven\'t implemented initializeCe
  centriodIDX = np.random.choice(dataset.shape[0], k, replace=False)
  return dataset[ centriodIDX, :]
```

(B)



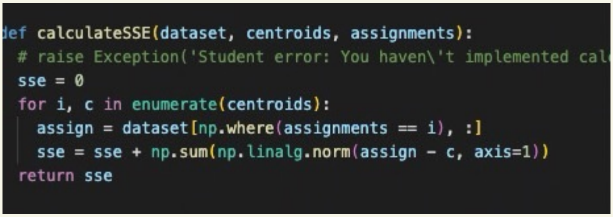
```
def computeAssignments(dataset, centroids):
  # raise Exception('Student error: You haven\'t implemented computeAssign
  assignments = []
  for x in dataset:
    assignments.append(np.linalg.norm(centroids - x, axis=1).argmin())
  return np.array(assignments)
```

(C)



```
def updateCentroids(dataset, centroids, assignments):
  # raise Exception('Student error: You haven\'t implemented updateCentroids yet.
  counts = []
  newCent = []
  for i, c in enumerate(centroids):
    count = np.count_nonzero(assignments == i)
    counts.append((count))
    assign = dataset[np.where(assignments == i), :]
    newCent.append(((1 / count) * np.sum(assign, axis=1)).reshape(-1))
  return np.array(newCent), counts
```

(d)

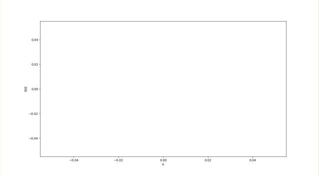


```
def calculateSSE(dataset, centroids, assignments):
  # raise Exception('Student error: You haven\'t implemented cal
  sse = 0
  for i, c in enumerate(centroids):
    assign = dataset[np.where(assignments == i), :]
    sse = sse + np.sum(np.linalg.norm(assign - c, axis=1))
  return sse
```

* I have a folder with all the plots I got for this problem.
↳ folder called Kmeans-#4

Q5 When running k=5 I noticed that each plot changed a lot more often in comparison with k=3. There is also more data. A large k can effect the data we are messuring because it is taking more into account.
I have included a folder with the plots I got from this problem called k-mean #5

Q6 For this problem I have only include the resulting plot because 0-150 plots was too many plots. The starting plot for the range looked normal but for the final one I kept getting an empty plot. So running Error vs k does not really make sense.



Q7 (a) the images seem to all be buildings, trees and empty roads. There is one random panda that appears each time I run it. The k of 10 seems to do okay but only a few figures match well.

(b) I lowered the k to 5 and the images were grouped pretty well but when I changed k to 15 the photos were grouped much better. There was even a cluster of all trees. Also I noticed the higher the k, the more selective so I found clusters with blank boxes. Having a higher k produces spesific results (I included k_Mean #7 with clusters) despite the results being more specific, I still noticed random photos that did not fit and clusters that would repeat so I stuck with 5.

(c) I think k of 15 produced better clusters than k = 10 because they were more generic a lower k allows you to generically classify like all of these are roads. The higher k makes the clusters more specific but alot of the clusters could have been grouped together

Q 8 (a) Figue 1 → roads

Figue 2 → Trees

Figue 3 → random Panda with city things :(

Figue 4 → buildings

Figue 5 → buildings

(b)

Figure	Purity
1	50 / 50
2	46/ 50
3	45 / 50 ← this 50 is both buildings and roads
4	50 / 50
5	48/50

Not including #3 all of them were very pure. (its out of 50 because thats the # of photos)

Debriefing:

① 3 days

② difficult

③ Mostly alone

④ Okay I think

⑤ N/A