CS 422 HW4

Abdullah Moizuddin

Recitation Exercises

Excercise 5.4

4. Given *K* equally sized clusters, the probability that a randomly chosen initial centroid will come from any given cluster is $1/K$, but the probability that each cluster will have exactly one initial centroid is much lower. (It should be clear that having one initial centroid in each cluster is a good starting situation for K-means.) In general, if there are *K* clusters and each cluster has *n* points, then the probability, *p*, of selecting in a sample of size *K* one initial centroid from each cluster is given by **Equation 7.20**. (This assumes sampling with replacement.) From this formula we can calculate, for example, that the chance of having one initial centroid from each of four clusters is $4!/4^4 = 0.0938.$

(7.20)

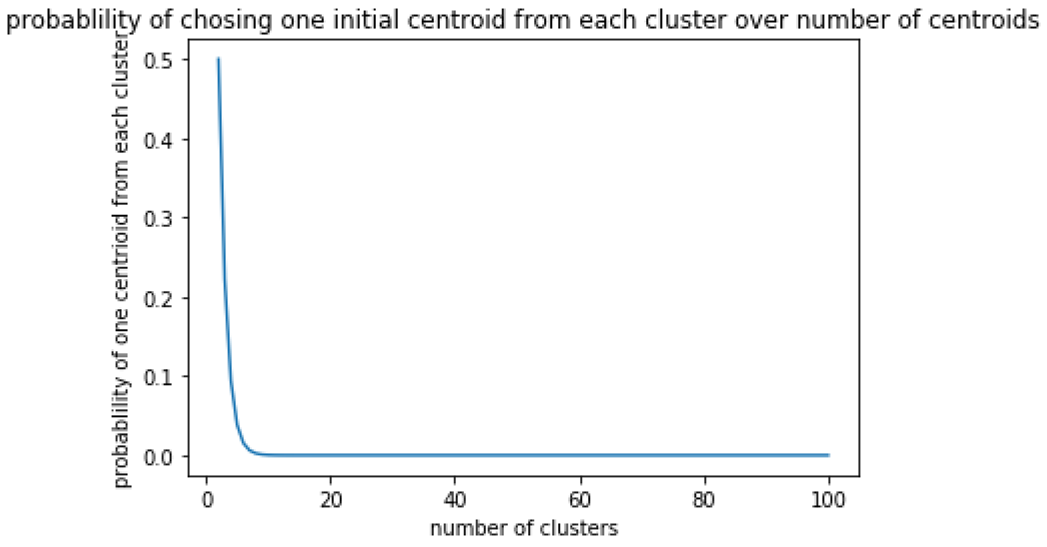
$$p = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

a. Plot the probability of obtaining one point from each cluster in a sample of size *K* for values of *K* between 2 and 100.

b. For *K* clusters, $K = 10, 100,$ and 1000, find the probability that a sample of size 2*K* contains at least one point from each cluster. You can use either mathematical methods or statistical simulation to determine the answer.

A.

In [19]:
```
import matplotlib.pyplot as plt
import math
def prob(x):
    return math.factorial(x)/x**x
x = [z for z in range(2,101)]
y = [prob(clusters) for clusters in x]

plt.plot(x,y)
plt.xlabel('number of clusters')
plt.ylabel('probablility of one centrioid from each cluster')
plt.title('probablility of chosing one initial centroid from each clus
ter over number of centroids')
plt.show()
```



B.

In [22]:
```
x = [10,100,1000]
y = [prob(cluster*2) for cluster in x]
y
for z in range(len(x)):
    print('for k =', x[z],'probablility of having one initial centroid
from each cluster is:', y[z])
```

```
for k = 10 probablility of having one initial centroid from each clust
er is: 2.32019615953125e-08
for k = 100 probablility of having one initial centroid from each clus
ter is: 4.907829957616477e-86
for k = 1000 probablility of having one initial centroid from each clu
ster is: 0.0
```

Excercise 5.7

7. Suppose that for a data set
   - there are *m* points and *K* clusters,
   - half the points and clusters are in “more dense” regions,
   - half the points and clusters are in “less dense” regions, and
   - the two regions are well-separated from each other.

For the given data set, which of the following should occur in order to minimize the squared error when finding *K* clusters:

a. Centroids should be equally distributed between more dense and less dense regions.
b. More centroids should be allocated to the less dense region.
c. More centroids should be allocated to the denser region.

Note: Do not get distracted by special cases or bring in factors other than density. However, if you feel the true answer is different from any given above, justify your response.

When trying to minimize squared error for this dataset, centroids should be allocated to the less dense region. this is because more dense reigons have more points concentrated in a small area and so a centroid in that region would have less distance from neighboring points. so even if the total amount of points is low the calculation is accurate. for the less dense area, becuase we allocate more points to it. it increases the accuracy by increasing the concentration and lessening the distance between points.

Excercise 5.11

11. Total SSE is the sum of the SSE for each separate attribute. What does it mean if the SSE for one variable is low for all clusters? Low for just one cluster? High for all clusters? High for just one cluster? How could you use the per variable SSE information to improve your clustering?

If SSE of one variable is low for every cluster, then the variable is constant and isnt good to use when partitioning the data into groups. If SSE of one variable is low for only one cluster, then that variable can be used to help define what that cluster can be. If SSE of one variable is high for all clusters then that variable could probably be noise. If SSE is high for one cluster that means the other variables have low SSE and so can just be an edge-case and so it cant help in defining the cluster. We can use the per variable SSE info to eliminate variables which do nothing to distinguish clusters. So variables with high or low SSE are not usefull for clustering.

Excercise 5.16

16. Use the similarity matrix in **Table 7.13** to perform single and complete link hierarchical clustering. Show your results by drawing a dendrogram. The dendrogram should clearly show the order in which the points are merged.

Table 7.13. Similarity matrix for Exercise 16.

	p1	p2	p3	p4	p5
p1	1.00	0.10	0.41	0.55	0.35
p2	0.10	1.00	0.64	0.47	0.98
p3	0.41	0.64	1.00	0.44	0.85
p4	0.55	0.47	0.44	1.00	0.76
p5	0.35	0.98	0.85	0.76	1.00

Agglomerative Heirarchial Clustering:

Step 1. Find most corrolated cluster:
Most corrolated clusters are p2 and p5 with a simmilarity score of .98
Merge the clusters to make cluster 'p:25'
Step 2.
Calculate simmilarity of each point to the new cluster 'p:25':
Simmilarity of 'p1' and 'p:25':
min(.1,.35) = .1
Simmilarity of 'p3' and 'p:25':
min(.64,.85) = .64
Simmilarity of 'p4' and 'p:25':
min(.47,..76) = .47

	P:25	**P1**	**P3**	**P4**
P:25	1	.1	.64	.47
P1	.1	1	.41	.55
P3	.64	.41	1	.44
P4	.47	.55	.44	1

max value is .64 so we can cluster that as 'P:235'
simmilarity of 'p1' and 'p235':
min(sim(p1,p3), sim(p1,p25))

= min(.41,.1) = .1

simmilarity of 'p4' with 'p:235'
min(sim(p4,p3), sim(p4,p25))

=min(0.44,.47) = .44

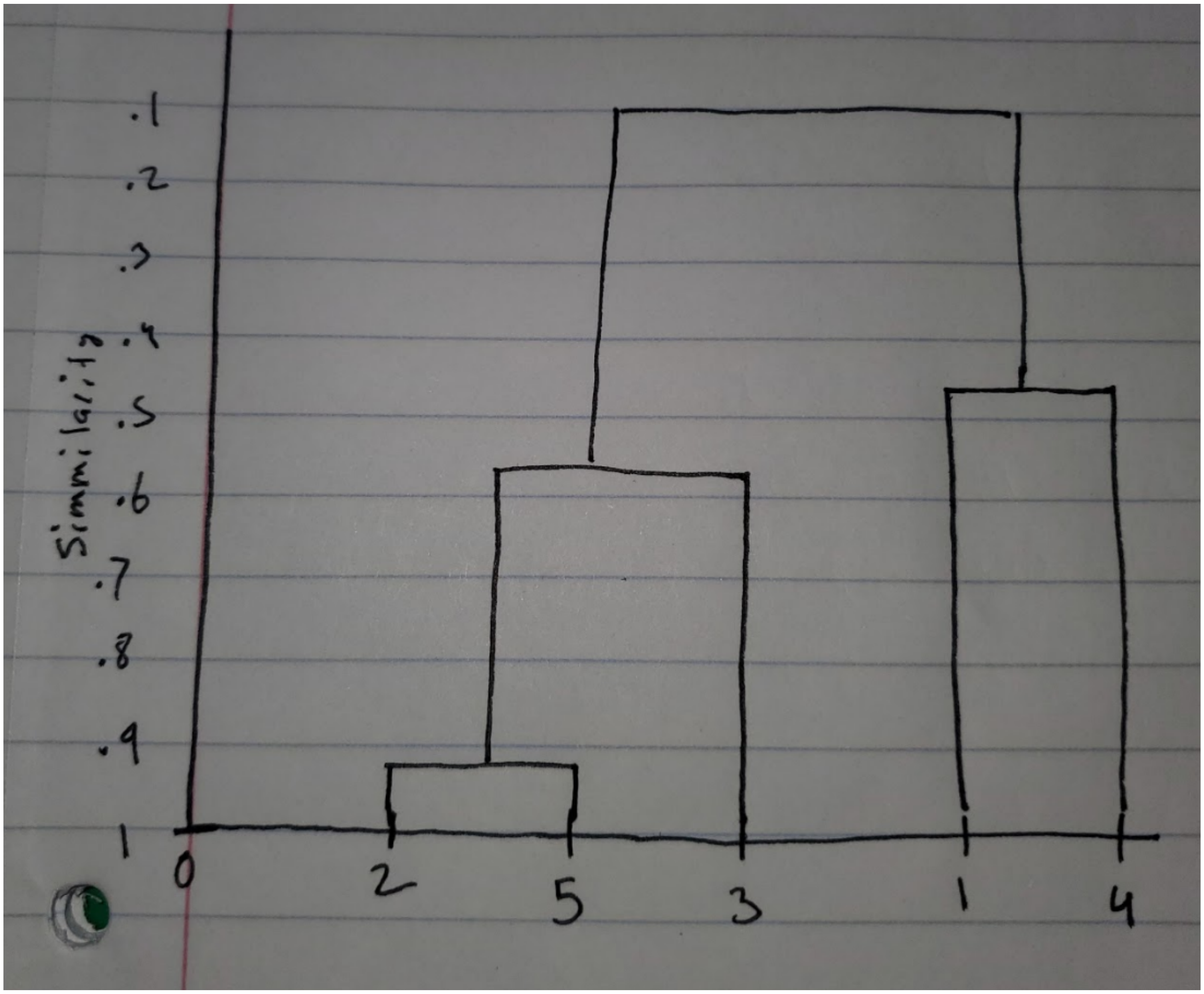
	P:235	**P1**	**P4**
P:235	1	.1	.44
P1	.1	1	.55
P4	.44	.55	1

Max = .55 at p1 and p4 1,4 become P:14

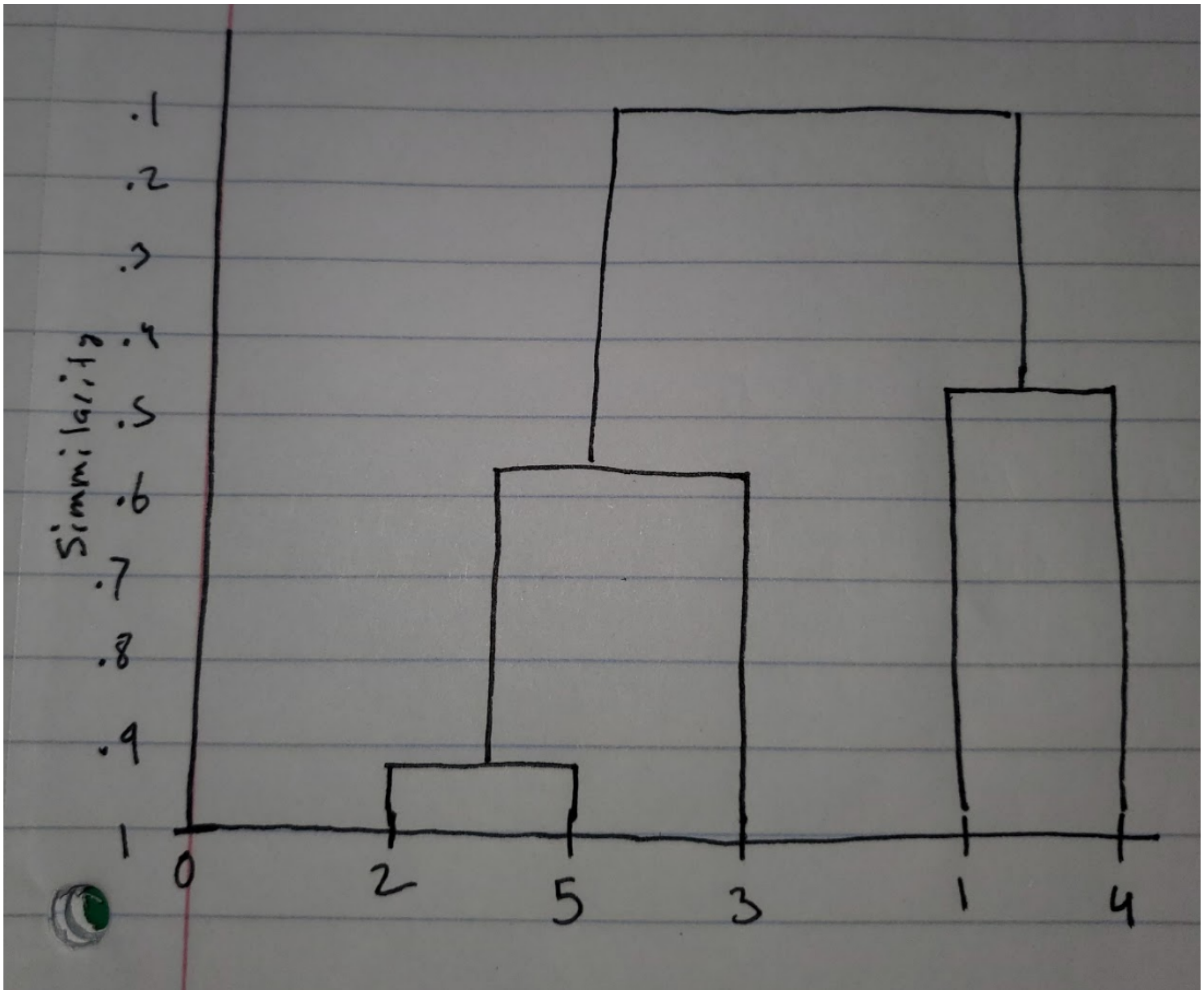
	P:235	**P:14**
P:235	1	.1
P:14	.1	1

Dendogram

single link:



complete link:


Simmilarity
.1
.2
.3
.4
.5
.6
.7
.8
.9
1
0
2
5
3
1
4

Excercise 7.17

17. Hierarchical clustering is sometimes used to generate *K* clusters, $K > 1$ by taking the clusters at the K^{th} level of the dendrogram. (Root is at level 1.) By looking at the clusters produced in this way, we can evaluate the behavior of hierarchical clustering on different types of data and clusters, and also compare hierarchical approaches to K-means.

The following is a set of one-dimensional points: {6, 12, 18, 24, 30, 42, 48}.

a. For each of the following sets of initial centroids, create two clusters by assigning each point to the nearest centroid, and then calculate the total squared error for each set of two clusters. Show both the clusters and the total squared error for each set of centroids.

   i. {18, 45}

   ii. {15, 40}

b. Do both sets of centroids represent stable solutions; i.e., if the K-means algorithm was run on this set of points using the given centroids as the starting centroids, would there be any change in the clusters generated?

c. What are the two clusters produced by single link?

d. Which technique, K-means or single link, seems to produce the “most natural” clustering in this situation? (For K-means, take the clustering with the lowest squared error.)

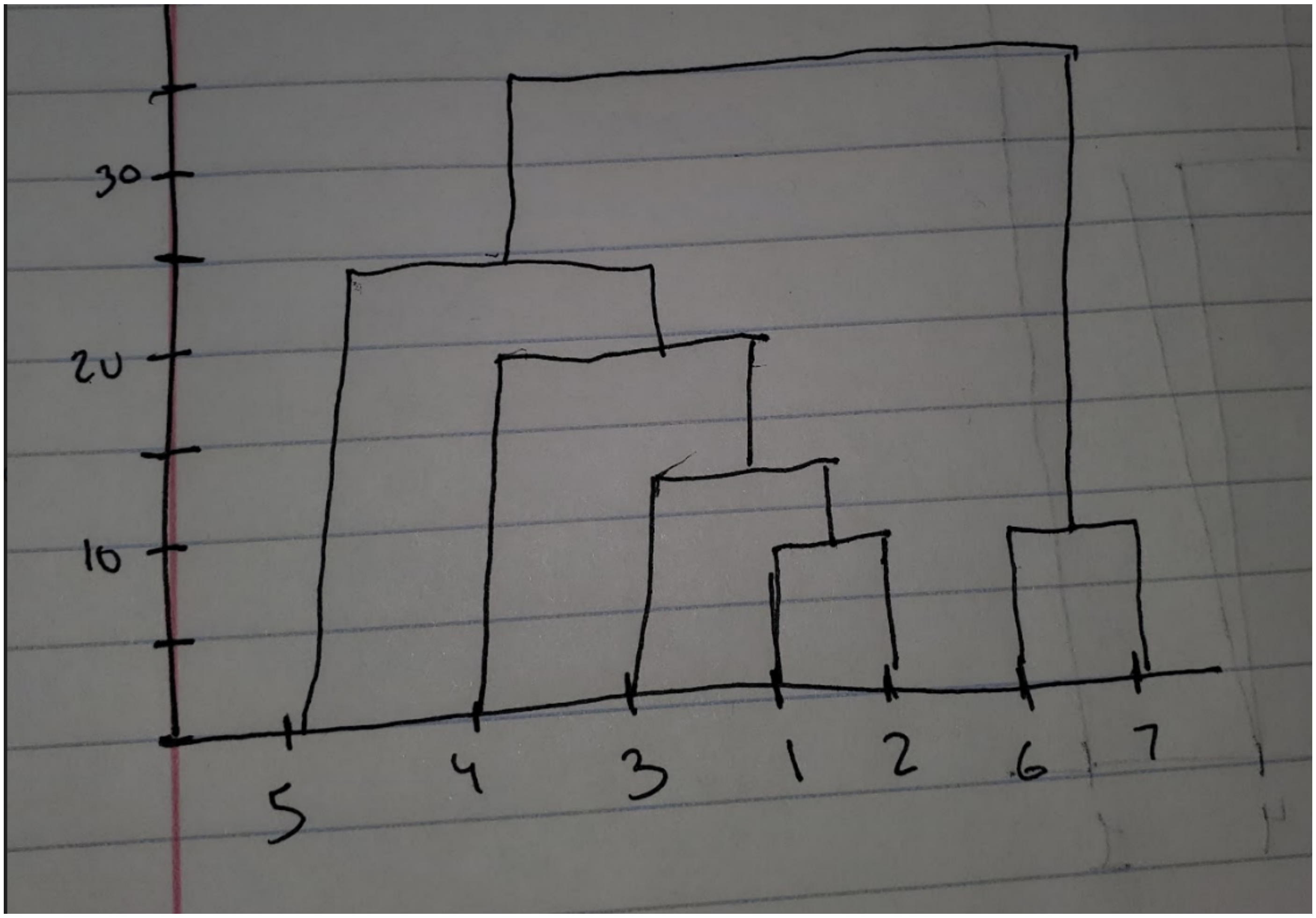
e. What definition(s) of clustering does this natural clustering correspond to? (Well-separated, center-based, contiguous, or density.)

f. What well-known characteristic of the K-means algorithm explains the previous behavior?

A.
i.
clusters:(6,12,18,24,30), (42,48)
square error: $(12^2+6^2+0^2+6^2+12^2)+(3^2+3^2)$
$=360+18$
$=378$
ii clusters:(6,12,18,24), (30,42,48)
square error: $(9^2+3^2+3^2+9^2)+(10^2+2^2+82)$
= $180+168$
= 348

B. The centroids are unchanged after clustering so both clusters are stable in both cases.

C. Clusters produced by single linking: $(6,12,18,24,30)$ and $(42,48)$
Dendogram:



D.
K-means chooses the second clustering from the first part. Single link, however, chooses the first clustering. out of these two clusters, Single-link chooses more natural clusters.
E.
single link is more biased towards dense clusters which are generally contiguous.

F.

The clusters all have different sizes so k-means struggles there. so it breaks up large clusters and moves centroids together.

Excercise 7.21

21. Compute the entropy and purity for the confusion matrix in **Table 7.14**.

Table 7.14. Confusion matrix for Exercise 21.

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Total
#1	1	1	0	11	4	676	693
#2	27	89	333	827	253	33	1562
#3	326	465	8	105	16	29	949
Total	354	555	341	943	273	738	3204

$$Entropy = P(X)Log_2(P(X))$$

entropy(#1):

$$-\frac{1}{693}Log_2(\frac{1}{693}) - \frac{1}{693}Log_2(\frac{1}{693}) - \frac{0}{693}Log_2(\frac{0}{693}) - \frac{11}{693}Log_2(\frac{11}{693}) - \frac{4}{693}Log_2(\frac{4}{693}) - \frac{676}{693}Log_2(\frac{676}{693}) = .2$$

entropy(#2):

$$-\frac{27}{1562}Log_2(\frac{27}{1562}) - \frac{89}{1562}Log_2(\frac{89}{1562}) - \frac{333}{1562}Log_2(\frac{333}{1562}) - \frac{872}{1562}Log_2(\frac{872}{1562}) - \frac{253}{1562}Log_2(\frac{253}{1562}) - \frac{33}{1562}L$$
$$= 1.841$$

entropy(#3):

$$-\frac{326}{949}Log_2(\frac{326}{949}) - \frac{465}{949}Log_2(\frac{465}{949}) - \frac{8}{949}Log_2(\frac{8}{949}) - \frac{105}{949}Log_2(\frac{105}{949}) - \frac{16}{949}Log_2(\frac{16}{949}) - \frac{29}{949}Log_2(\frac{29}{949}) = 1.696$$

Purity:

purity(#1) $= \frac{676}{693} = .975$
purity(#2) $= \frac{872}{1562} = .592$
purity(#3) $= \frac{465}{949} = .490$

Total Entropy and Purity:

$$Entropy = \frac{1}{3204}(693 * 0.200 + 1562 * 1.841 + 949 * 1.696) = 1.443$$
$$Purity = \frac{1}{3204}(693 * 0.975 + 1562 * .529 + 949 * .490) = .614$$

Excercise 7.22

22. You are given two sets of 100 points that fall within the unit square. One set of points is arranged so that the points are uniformly spaced. The other set of points is generated from a uniform distribution over the unit square.

a. Is there a difference between the two sets of points?

b. If so, which set of points will typically have a smaller SSE for $K = 10$ clusters?

c. What will be the behavior of DBSCAN on the uniform data set? The random data set?

A.
The set of points with uniform spacing will have the same space between all points in the square. The one with a uniform distribution however might have some areas which vary in density.
B.
The points with uniform distibution.
C.
uniformly spaced: DBSCAN will make everything into one cluster or it will all be noise depending on what the distance threshold value is. uniformly distributed: it will just find clusters.

Practicum Problems

In [31]:
```
import numpy as np
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.datasets import load_boston, load_wine
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score, homogeneity_score, compl
eteness_score
```

Problem 1

Statement of the Problem: Load the auto-mpg sample dataset from the UCI Machine Learning Repository (auto-mpg.data) into Python using a Pandas dataframe. Using only the continuous fields as features, impute any missing values with the mean, and perform a Hierarchical Clustering (Use sklearn.cluster.AgglomerativeClustering) with linkage set to average and the default affinity set to a euclidean. Set the remaining parameters to obtain a shallow tree with 3 clusters as the target. Obtain the mean and variance values for each cluster, and compare these values to the values obtained for each class if we used origin as a class label. Is there a clear relationship between cluster assignment and class label?

In [45]:
```
#Cleaning and Formatting Dataset


p = r'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mp
g/auto-mpg.data'
ds = pd.read_csv(p,delim_whitespace = True)
ds = ds.replace('?',np.NaN)
ds.columns = ["mpg", "cylinders", "displacement", "horsepower", "weigh
t",
              "acceleration", "model_year", "origin", "car_name"]

car_ds = ds[["mpg", "displacement", "horsepower", "weight", "accelerat
ion"]]

#imputing missing vals
my_imp = SimpleImputer(missing_values = np.NaN, strategy = 'mean')
my_imp = my_imp.fit(car_ds)
car_ds = my_imp.transform(car_ds)

car_ds= pd.DataFrame(car_ds)
car_ds.columns = ["mpg", "displacement", "horsepower", "weight", "acce
leration"] #continuous vals from dataset

cols = ds[['origin']]

#Preforming Heirarchial Clustering
model = AgglomerativeClustering(linkage='average', affinity = 'euclide
an', n_clusters=3)
result = model.fit(car_ds)
lables = result.labels_

#array for origins
o=[]
for index,row in ds.iterrows():
    o.append(row["origin"])
o = np.array(o)
car_ds["origin"] = o
car_ds["cluster_id"] = lables
lno = zip(lables, o) #lables and origins

#being grouped by origin
gbo = car_ds[["mpg", "displacement", "horsepower", "weight", "accelera
tion", "origin"]].groupby(by=["origin"])
#being grouped by cluster
gbc = car_ds[["mpg", "displacement", "horsepower", "weight", "accelera
tion", "cluster_id"]].groupby(by=["cluster_id"])

#printing mean and variance values resulting from grouping on cluster
 id and origin
print("mean when grouped by origin")
print(gbo.mean())
print("\n")
print("mean when grouped by cluster")
print(gbc.mean())

print("\n \n \n \n")
print("variance when grouped by origin")
print(gbo.var())
print("\n")
print("variance when grouped by cluster")
print(gbc.var())

print("\n \n \n \n")

#making a matrix to see the relationship between class and origin
categories = {
    0:[],
    1:[],
    2:[]
}
for l, o in lno:
    categories[l].append(o)

print(" Origin:", "1".ljust(3), "2".ljust(3), "3".ljust(3))
print("-------------------")
print("Class 0:", str(categories[0].count(1)).ljust(3), str(categories
[0].count(2)).ljust(3), str(categories[0].count(3)).ljust(3))
print("Class 1:", str(categories[1].count(1)).ljust(3), str(categories
[1].count(2)).ljust(3), str(categories[1].count(3)).ljust(3))
print("Class 2:", str(categories[2].count(1)).ljust(3), str(categories
[2].count(2)).ljust(3), str(categories[2].count(3)).ljust(3))
```

```
mean when grouped by origin
              mpg  displacement  horsepower       weight  acceleration
origin
1       20.091935    245.655242  118.768614  3361.358871     15.045968
2       27.891429    109.142857   81.240117  2423.300000     16.787143
3       30.450633    102.708861   79.835443  2221.227848     16.172152


mean when grouped by cluster
                  mpg  displacement  horsepower       weight  accelera
tion
cluster_id
0           27.365414    131.934211   84.298589  2459.511278     16.29
8120
1           13.889062    358.093750  167.046875  4398.593750     13.02
5000
2           17.502985    278.567164  124.388060  3626.641791     15.15
2239
```

```
variance when grouped by origin
              mpg  displacement   horsepower         weight  accelerat
ion
origin
1       41.145360   9726.719729  1575.408439  634170.554901      7.561
846
2       45.211230    509.950311   410.571987  240142.328986      9.276
209
3       37.088685    535.465433   317.523856  102718.485881      3.821
779


variance when grouped by cluster
                  mpg  displacement  horsepower         weight  accele
ration
cluster_id
0           41.976309   2828.083391  369.083949  182632.099872      5.
718298
1            3.359085   2138.213294  756.521577   74312.340278      3.
591429
2            8.959991   2914.097693  723.422886   38123.627318     10.
568593
```

```
 Origin: 1   2   3
-------------------
Class 0: 120 67  79
Class 1: 64  0   0
Class 2: 64  3   0
```

Analysis:

Mean
we dont get too much infromation when we look at the mean when grouped by origin and cluster. So we can disregard it.

Variance
When looking at variance,cluster 1 and two in mpg have reletivly low variance. this means that it probably contributed alot when splitting between cluster 0 and cluster 1,2. the other features that probably contributed was weight. The variance from cluster 1,2 is reletivly low but in cluster 0 it is significantly higher.

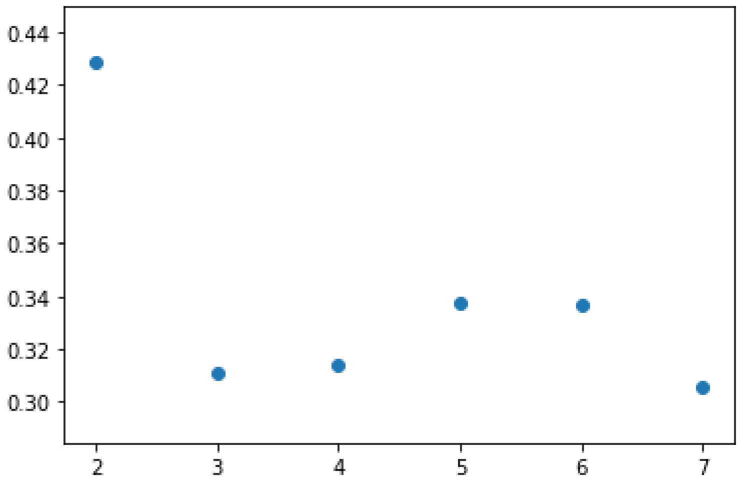
Relationship between Class label and origin
As seen from the table, there isnt a 1:1 mapping with origin and cluster. But, nearly everything from origin 2,3 belonged to class 0. So, cluster 1 and 2 most likely didnt have features which generally could not be attributed to origin 2 or 3.

Problem 2

Statement of the Problem: Load the Boston dataset (sklearn.datasets.load boston()) into Python using a Pandas dataframe. Perform a K-Means analysis on scaled data, with the number of clusters ranging from 2 to 6. Provide the Silhouette score to justify which value of k is optimal. Calculate the mean values for all features in each cluster for the optimal clustering - how do these values differ from the centroid coordinates?

In [61]:
```
#loading and scaling boston data set
data = load_boston()
boston_ds = pd.DataFrame(data.data, columns=data.feature_names)
s = MinMaxScaler()
s.fit(boston_ds)
boston_ds = pd.DataFrame(s.transform(boston_ds), columns=data.feature_
names)

#function that returns score for diffrent cluster sizes
def silhlo_score(x):
    model = KMeans(n_clusters=x)
    result = model.fit_predict(boston_ds)
    return silhouette_score(boston_ds,result)

x = [z for z in range(2,8)]
y = [silhlo_score(count) for count in x]

#printing scores
for things in zip(x,y):
    print(things[0], "cluster score:", things[1])
plt.scatter(x,y)
plt.show()

#3 clusters is optimal

boston2 = boston_ds.copy()
opt_clust = KMeans(n_clusters = 3)
result = opt_clust.fit_predict(boston2)
boston2["cluster_id"] = result

print("\n\n\n\ncoordinates:\n\n",opt_clust.cluster_centers_)
print("\nfeature mean:")
boston2.groupby(by="cluster_id").mean()
```

```
2 cluster score: 0.4283539950397875
3 cluster score: 0.3110514683967143
4 cluster score: 0.3139570298922441
5 cluster score: 0.3374165698096861
6 cluster score: 0.33644652277307147
7 cluster score: 0.305814392237243
```



```
coordinates:

 [[1.07999159e-03 2.99000000e-01 1.72794477e-01 3.33333333e-02
  1.25442387e-01 5.67418194e-01 3.46527063e-01 4.38343937e-01
  1.41062802e-01 2.00879983e-01 5.39184397e-01 9.81088865e-01
  1.56553532e-01]
 [1.39175844e-01 2.35922393e-16 6.57020873e-01 5.88235294e-02
  5.87554466e-01 4.68553814e-01 8.97172412e-01 8.40657367e-02
  9.74424552e-01 9.16647957e-01 8.08197747e-01 7.31102581e-01
  4.68644494e-01]
 [7.33158115e-03 1.93684211e-02 4.08311468e-01 1.10526316e-01
  3.90480832e-01 5.16879620e-01 8.30787577e-01 1.70053044e-01
  1.58810069e-01 2.77973082e-01 5.69652856e-01 9.40259880e-01
  3.18935169e-01]]

feature mean:
```

Out[61]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
cluster_id									
0	0.001080	0.299000	0.172794	0.033333	0.125442	0.567418	0.346527	0.438344	0.14
1	0.139176	0.000000	0.657021	0.058824	0.587554	0.468554	0.897172	0.084066	0.97
2	0.007332	0.019368	0.408311	0.110526	0.390481	0.516880	0.830788	0.170053	0.15

Analysis:

Plot and Silhouette score:
as seen from the silhouette score and plot, 3 clusters is optimal because it is really close to the best score without diminishing returns.

Difference between mean values of features and centroids coordinates:
They are the same.

Problem 3 ¶

Statement of the Problem: Load the wine dataset (sklearn.datasets.load wine()) into Python using a Pandas dataframe. Perform a K-Means analysis on scaled data, with the number of clusters set to 3. Given the actual class labels, calculate the Homogeneity/Completeness for the optimal k - what information do each of these metrics provide?

In [63]:
```
#loading and scaling wine dataset
data = load_wine()
wine_ds = pd.DataFrame(data.data, columns= data.feature_names)
s = MinMaxScaler()
s.fit(wine_ds)
wine_ds = pd.DataFrame(s.transform(wine_ds), columns =data.feature_nam
es)
rows = wine_ds.shape[0]
model = KMeans(n_clusters=3)
result = model.fit_predict(wine_ds)
print("Homogeneity score:",homogeneity_score(result, data.target))
print("Completeness score:",completeness_score(result, data.target))
```

```
Homogeneity score: 0.8488717548840309
Completeness score: 0.8570247637781875
```

Analysis:

Homogeneity score is if the cluster only contains points that belong to a single class.
Completeness score is how accurately the algorithm assigned points to their true label.

In [ ]:
```
```