In [22]:
```
import pandas as pd
import numpy as np
import random
import operator
from collections import Counter
import copy
```

In [23]:
```
data=pd.read_csv('agaricus-lepiota.data',header=None)
```

In [24]:
```
data.columns=['class','cap-shape','cap-surface','cap-color','bruises?'
              'gill-spacing','gill-size','gill-color','stalk-shape','stal
              'stalk-surface-above-ring','stalk-surface-below-ring','stal
              'stalk-color-below-ring','veil-type','veil-color','ring-num
              'spore-print-color','population','habitat']
```

In [25]:
```
print(np.unique(data))
data2=data[data.columns[1:23]]
data2
```

```
['?' 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'k' 'l' 'm' 'n' 'o' 'p' 'r' 's'
't'
 'u' 'v' 'w' 'x' 'y']
```

Out[25]:

	cap-shape	cap-surface	cap-color	bruises?	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	s surf be
0	x	s	n	t	p	f	c	n	k	e	...	
1	x	s	y	t	a	f	c	b	k	e	...	
2	b	s	w	t	l	f	c	b	n	e	...	
3	x	y	w	t	p	f	c	n	n	e	...	
4	x	s	g	f	n	f	w	b	k	t	...	
...	...	...	...	...	...	...	...	...	...	...	...	
8119	k	s	n	f	n	a	c	b	y	e	...	
8120	x	s	n	f	n	a	c	b	y	e	...	
8121	f	s	n	f	n	a	c	b	n	e	...	
8122	k	y	n	f	y	f	c	n	b	t	...	
8123	x	s	n	f	n	a	c	b	y	e	...	

8124 rows × 22 columns

In [26]:
```
rslt_df1 = data.loc[data['class'] =='?']
rslt_df2 = data.loc[data['cap-shape'] =='?']
rslt_df3 = data.loc[data['cap-surface'] =='?']
rslt_df4 = data.loc[data['cap-color'] =='?']
rslt_df5 = data.loc[data['bruises?'] =='?']
rslt_df6 = data.loc[data['odor'] =='?']
rslt_df7 = data.loc[data['gill-attachment'] =='?']
rslt_df8 = data.loc[data['gill-spacing'] =='?']
rslt_df9 = data.loc[data['gill-size'] =='?']
rslt_df10 = data.loc[data['gill-color'] =='?']
rslt_df11= data.loc[data['stalk-shape'] =='?']
rslt_df12 = data.loc[data['stalk-root'] =='?']
rslt_df13 = data.loc[data['stalk-surface-above-ring'] =='?']
rslt_df14 = data.loc[data['stalk-surface-below-ring'] =='?']
rslt_df15 = data.loc[data['stalk-color-above-ring'] =='?']
rslt_df16 = data.loc[data['stalk-color-below-ring'] =='?']
rslt_df17 = data.loc[data['veil-type'] =='?']
rslt_df18 = data.loc[data['veil-color'] =='?']
rslt_df19 = data.loc[data['ring-number'] =='?']
rslt_df20 = data.loc[data['ring-type'] =='?']
rslt_df21= data.loc[data['spore-print-color'] =='?']
rslt_df22 = data.loc[data['population'] =='?']
rslt_df23 = data.loc[data['habitat'] =='?']
```

In [27]:
```
rslt_df12
```

Out[27]:

	class	cap-shape	cap-surface	cap-color	bruises?	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	st surfa bel
3984	e	x	y	b	t	n	f	c	b	e	...	
4023	p	x	y	e	f	y	f	c	n	b	...	
4076	e	f	y	u	f	n	f	c	n	h	...	
4100	p	x	y	e	f	y	f	c	n	b	...	
4104	p	x	y	n	f	f	f	c	n	b	...	
...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	...	
8120	e	x	s	n	f	n	a	c	b	y	...	
8121	e	f	s	n	f	n	a	c	b	n	...	
8122	p	k	y	n	f	y	f	c	n	b	...	
8123	e	x	s	n	f	n	a	c	b	y	...	

2480 rows × 23 columns

In [28]:
```
#Initializing centroids
#data.sample(n = 3)
df=np.array(data2)
df[0][1]
```

Out[28]:
```
's'
```

In [29]:
```
#Initialization of centroids
def initialize_centroids(X, n_clusters):
    return np.array(random.sample(X, n_clusters))
```

In [30]:
```
noOfClusters=5
x=list(df)
centroids=initialize_centroids(x,noOfClusters)
```

In [31]:
```
centroids
#range(len(data))
#range(0,22)
```

Out[31]:
```
array([['k', 'f', 'w', 'f', 'n', 'f', 'w', 'b', 'g', 'e', '?', 's', '
k',
        'w', 'w', 'p', 'w', 't', 'p', 'w', 's', 'g'],
       ['x', 'f', 'y', 'f', 'f', 'f', 'c', 'b', 'g', 'e', 'b', 'k', '
k',
        'p', 'b', 'p', 'w', 'o', 'l', 'h', 'y', 'd'],
       ['f', 'y', 'n', 't', 'p', 'f', 'c', 'n', 'p', 'e', 'e', 's', '
s',
        'w', 'w', 'p', 'w', 'o', 'p', 'n', 'v', 'g'],
       ['f', 's', 'w', 't', 'f', 'f', 'c', 'b', 'p', 't', 'b', 's', '
f',
        'w', 'w', 'p', 'w', 'o', 'p', 'h', 's', 'g'],
       ['b', 'y', 'w', 't', 'n', 'f', 'c', 'b', 'w', 'e', 'b', 's', '
s',
        'w', 'w', 'p', 'w', 't', 'p', 'r', 'v', 'm']], dtype=object)
```

In [ ]:
```
```

Type *Markdown* and LaTeX: α^2

In [32]:
```
#Assignment of observations to a cluster
(centroids.shape[1])
(data2.shape[0])
d=np.zeros((8124,2))
d2=pd.DataFrame(d)
(centroids.shape[1])
dist = np.zeros((8124,2))
dist2=pd.DataFrame(dist)
dist2
df[2,:]
```

Out[32]:
```
array(['b', 's', 'w', 't', 'l', 'f', 'c', 'b', 'n', 'e', 'c', 's', 's
',
       'w', 'w', 'p', 'w', 'o', 'p', 'n', 'n', 'm'], dtype=object)
```

In [ ]:
```
```

In [68]:
```
#Assignment of observations to a cluster
def distances(X, centroids):
    #x=np.array(X)
    #print(x)
    dist = np.zeros(centroids.shape[0])
    #print(dist)
    for i in range(centroids.shape[0]):
        for j in range(centroids.shape[1]):
            if X[j] != centroids[i, j]:
                dist[i] += 1;
   # print(dist)
    #y=min(enumerate(dist), key=operator.itemgetter(1))[0]
    #print('The cluster is:',y )
    #return print('Least distance is',min((dist)))
    return dist

def assignment(X, centroids):
    dist=distances(X,centroids)
    y=min(enumerate(dist), key=operator.itemgetter(1))[0]
```

In [69]:
```
d=distances(df[1000,:],centroids)
d
```

Out[69]:
```
array([14., 10., 13., 11., 11.])
```

In [73]:
```
d=np.zeros((len(data2),noOfClusters))
for i in range(len(data2)):
    for j in range(noOfClusters):
        d[i][j]=distances(df[i,:],centroids)[j]
        #print(d[i])
        #for i in range (data2.shape[0]):
        # d[i]
#print(d)
dis=pd.DataFrame(d)
data2['c1']=dis[0]
data2['c2']=dis[1]
data2['c3']=dis[2]
data2['c4']=dis[3]
data2['c5']=dis[4]
data2
```

Out[73]:

s?	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	ring-number	ring-type	spore-print-color	population	habita
t	p	f	c	n	k	e	...	o	p	k	s	
t	a	f	c	b	k	e	...	o	p	n	n	
t	l	f	c	b	n	e	...	o	p	n	n	r
t	p	f	c	n	n	e	...	o	p	k	s	
f	n	f	w	b	k	t	...	o	e	n	a	
...	...	...	...	...	...	...	...	...	...	...	...	.
f	n	a	c	b	y	e	...	o	p	b	c	
f	n	a	c	b	y	e	...	o	p	b	v	
f	n	a	c	b	n	e	...	o	p	b	c	
f	y	f	c	n	b	t	...	o	e	w	v	
f	n	a	c	b	y	e	...	o	p	o	c	

In [ ]:
```
```

In [16]:
```
def update_centroids(X, belongs, centroid):
    n_centroids = centroids.shape[0]
    print (n_centroids)
    for i in range(n_centroids):
        points = np.array([X[j, :] for j in range(X.shape[0]) if belor
        print(points)
        for k in range(points.shape[1]):
            temp_points = [points[j, k] for j in range(points.shape[0]
            print(temp_points)
            count = Counter(temp_points)
            print (count)
            centroids[i, k] = max(count.iteritems(), key=operator.item
    return centroids
```

In [17]:
```
#clusterCentroidsDf = pd.DataFrame(centroids)
#clusterCentroidsDf.columns = data.columns
```

In [18]:
```
#clusterCentroidsDf
```

In [ ]:
```

```

In [19]:
```
def kmodes(data, n_clusters):
    x=list(data)
    cluster_centers = initialize_centroids(x, n_clusters)
    print ("Initial centroids:")
    print (cluster_centers)
    print ("|-----------------------------------|")

    n_points = len(data)
    belongs_to = np.full(n_points, 0, dtype='int')
    has_changed = False

    # Calculates the belonging of each point among the cluster centers
    for ii in range(100):
        for jj in range(n_points):
            belongs = distances(data[jj, :], cluster_centers)
            if belongs_to[jj] != belongs:
                belongs_to[jj] = belongs
                has_changed = True
        if not has_changed:
            break
        else:
            cluster_centers = update_centroids(data, belongs_to, clust
            has_changed = False
    return cluster_centers, belongs_to
```

```
if __name__ == "__main__":
    # Importing data from dataset and reformatting into attributes and
    # x = np.genfromtxt('soybean.csv', dtype=str, delimiter=',')[:, :-
    # y = np.genfromtxt('soybean.csv', dtype=str, delimiter=',', usecc

    centroids, y_test = kmodes(df, 7)

    print ("|-----------------------------------------------|")
    print ("Centroids:")
    print (centroids)
    print ("|-----------------------------------------------|")
    print ("Y train:Y test")
    combo = [(ii,jj) for ii,jj in zip(y,y_test)]
    for x in combo:
        print (x)
    print ("|-----------------------------------------------|")
```

```
Initial centroids:
[['x' 's' 'n' 'f' 'n' 'a' 'c' 'b' 'o' 'e' '?' 's' 's' 'o' 'o' 'p' 'o'
'o'
  'p' 'n' 'c' 'l']
 ['f' 'y' 'y' 't' 'a' 'f' 'c' 'b' 'w' 'e' 'r' 's' 'y' 'w' 'w' 'p' 'w'
'o'
  'p' 'n' 's' 'g']
 ['f' 'f' 'g' 'f' 'f' 'f' 'c' 'b' 'h' 'e' 'b' 'k' 'k' 'b' 'b' 'p' 'w'
'o'
  'l' 'h' 'v' 'd']
 ['f' 's' 'n' 'f' 'n' 'f' 'w' 'b' 'h' 't' 'e' 's' 's' 'w' 'w' 'p' 'w'
'o'
  'e' 'k' 'a' 'g']
 ['f' 'f' 'g' 't' 'n' 'f' 'c' 'b' 'p' 't' 'b' 's' 's' 'p' 'p' 'p' 'w'
'o'
  'p' 'k' 'v' 'd']
 ['k' 'y' 'e' 'f' 'f' 'f' 'c' 'n' 'b' 't' '?' 'k' 's' 'p' 'w' 'p' 'w'
'o'
  'e' 'w' 'v' 'l']
 ['x' 'y' 'n' 'f' 's' 'f' 'c' 'n' 'b' 't' '?' 'k' 'k' 'w' 'p' 'p' 'w'
'o'
  'e' 'w' 'v' 'd']]
|---------------------------------|

---------------------------------------------------------------------
------
ValueError                                Traceback (most recent call
last)
<ipython-input-19-3e2cebe2125b> in <module>
     30     # y = np.genfromtxt('soybean.csv', dtype=str, delimiter='
,', usecols=(21,))
     31
```

```
---> 32     centroids, y_test = kmodes(df, 7)
     33
     34     print ("|-----------------------------------------------
-|")

<ipython-input-19-3e2cebe2125b> in kmodes(data, n_clusters)
     14         for jj in range(n_points):
     15             belongs = distances(data[jj, :], cluster_centers)
---> 16             if belongs_to[jj] != belongs:
     17                 belongs_to[jj] = belongs
     18                 has_changed = True

ValueError: The truth value of an array with more than one element is
ambiguous. Use a.any() or a.all()
```

In [ ]:
```
n_points = data.shape[0]
belongs_to = np.full(n_points, 0, dtype='int')
```

In [ ]:
```
belongs_to
```

In [ ]:
```
n_points = data.shape[0]
```

In [ ]:
```
len(data)
```

In [ ]:
```
len(data)
```

In [ ]:
```

```