Homework 4

Matthew Nickols

2024-03-29

Problem 2

```
from sklearn import svm
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

data=pd.read_csv("diabetes_train.csv")
m,n=data.shape
model_features = ['preg',   'plas', 'pres', 'skin', 'insu', 'mass', 'pedi', 'age']
ind = data[model_features]
dep = data['class']
train_ind, test_ind, train_dep, test_dep = train_test_split(ind, dep, test_size=0.01319, shuffle=False)

C=1e+03
gamma=1e-05
clf = svm.SVC(C=C, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=gamma, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

clf.fit(train_ind,train_dep)
```

```
## SVC(C=1000.0, gamma=1e-05)
```

```
pred=clf.predict(test_ind)
print("Prediction, True Class")
```

```
## Prediction, True Class
```

```
for i in range(10):
  print(pred[i], ", ", test_dep.iloc[i])
```

```
## tested_positive ,  tested_positive
## tested_positive ,  tested_positive
## tested_negative ,  tested_positive
## tested_negative ,  tested_negative
## tested_negative ,  tested_negative
## tested_positive ,  tested_positive
## tested_positive ,  tested_positive
## tested_negative ,  tested_positive
## tested_negative ,  tested_negative
## tested_negative ,  tested_positive
```

```
clf.support_vectors_.shape
```

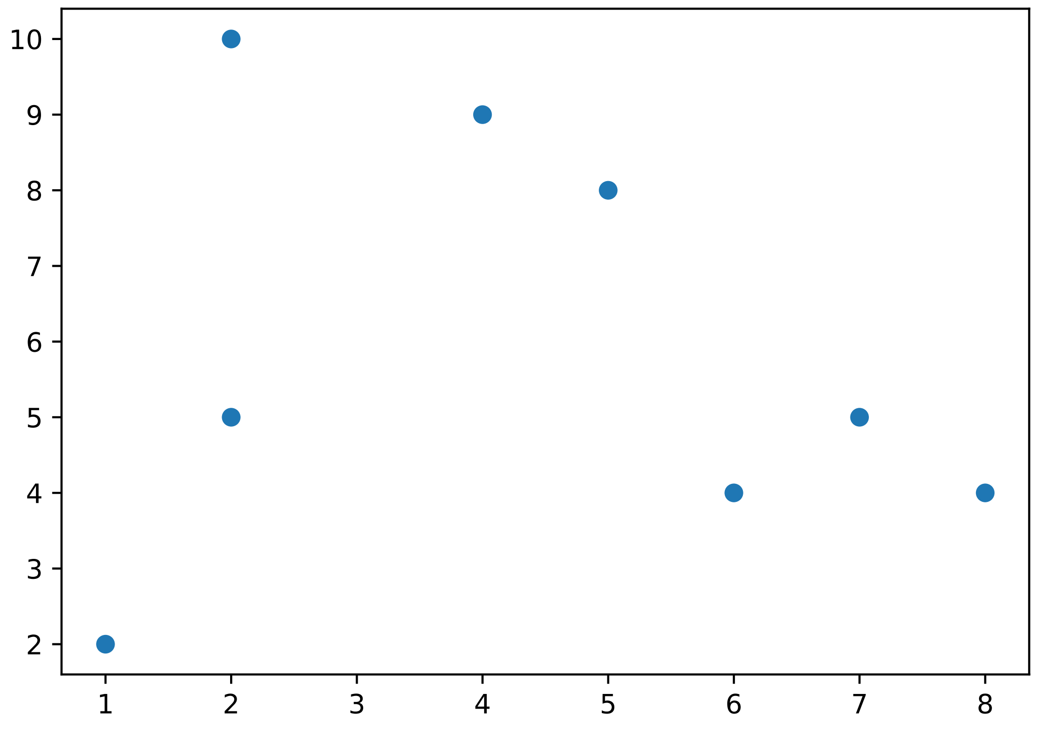
```
## (380, 8)
```

Problem 3

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


x = [2, 2, 8, 5, 7, 6, 1, 4]
y = [10, 5, 4, 8, 5, 4, 2, 9]

plt.scatter(x, y)
plt.show()
```



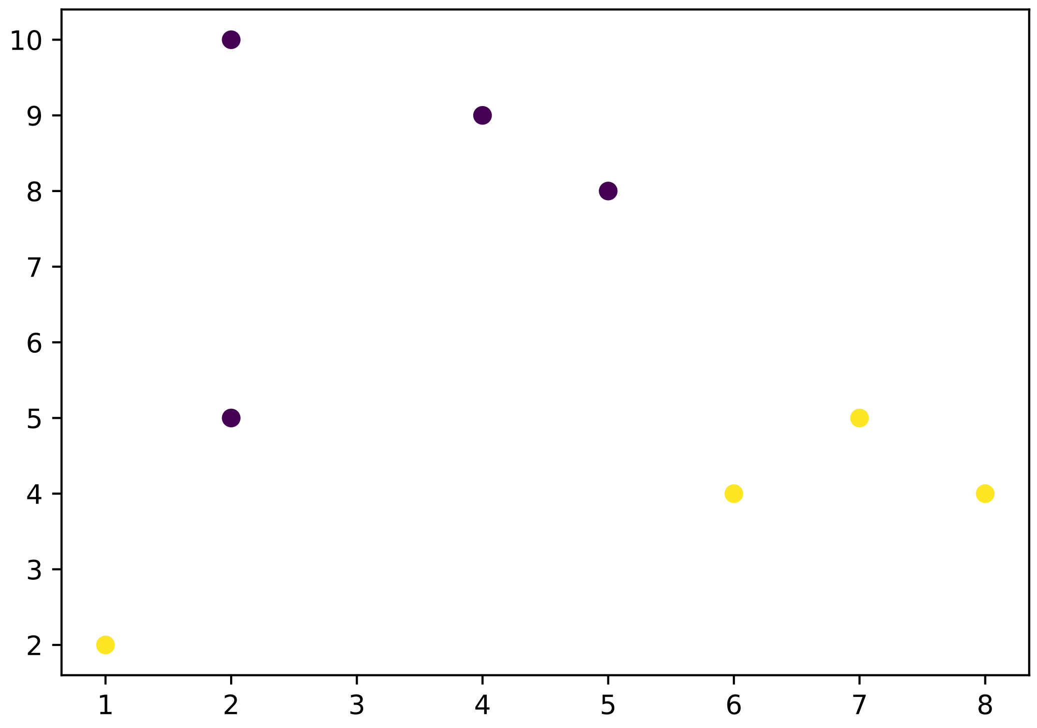
Part a

```
data = list(zip(x, y))
cluster_centers = np.array([[4, 9], [8, 4]])

kmeans = KMeans(n_clusters=2, init = cluster_centers)
kmeans.fit(data)
```

```
## KMeans(init=array([[4, 9],
##         [8, 4]]), n_clusters=2)
```

```
plt.scatter(x, y, c=kmeans.labels_)
plt.show()
```



```
kmeans.cluster_centers_
```

```
## array([[3.25, 8.  ],
##         [5.5 , 3.75]])
```

Part b

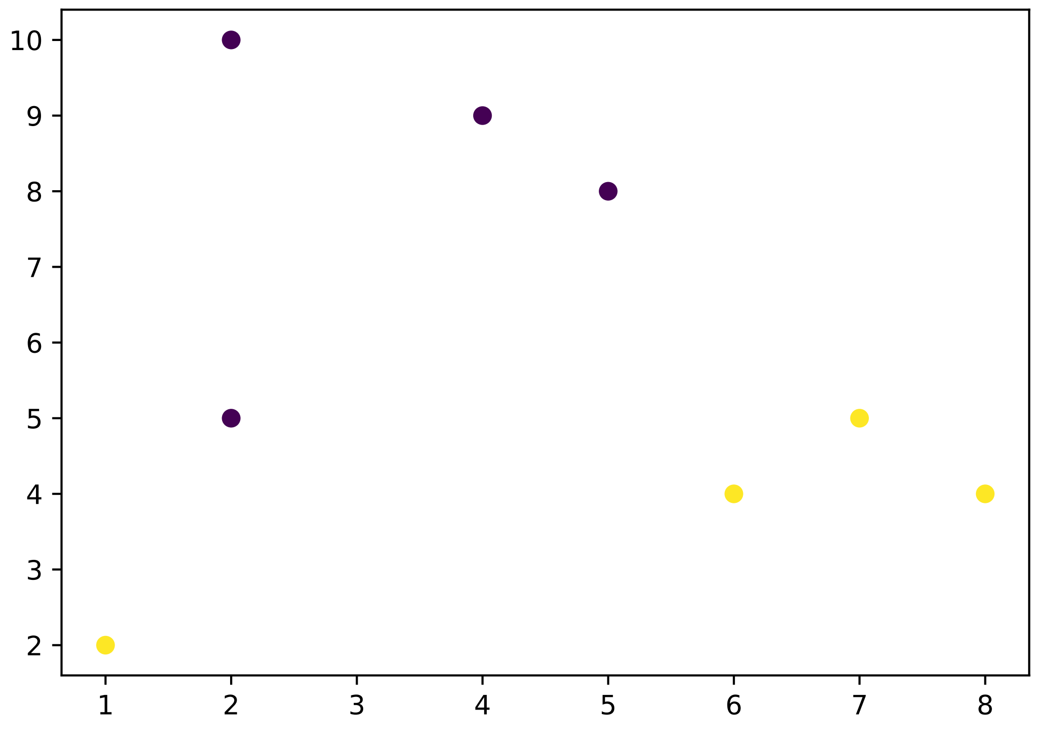
```
cluster_centers = kmeans.cluster_centers_
cluster_centers
```

```
## array([[3.25, 8.  ],
##         [5.5 , 3.75]])
```

```
kmeansb = KMeans(n_clusters=2, init = cluster_centers)
kmeansb.fit(data)
```

```
## KMeans(init=array([[3.25, 8.  ],
##         [5.5 , 3.75]]), n_clusters=2)
```

```
plt.scatter(x, y, c=kmeans.labels_)
plt.show()
```



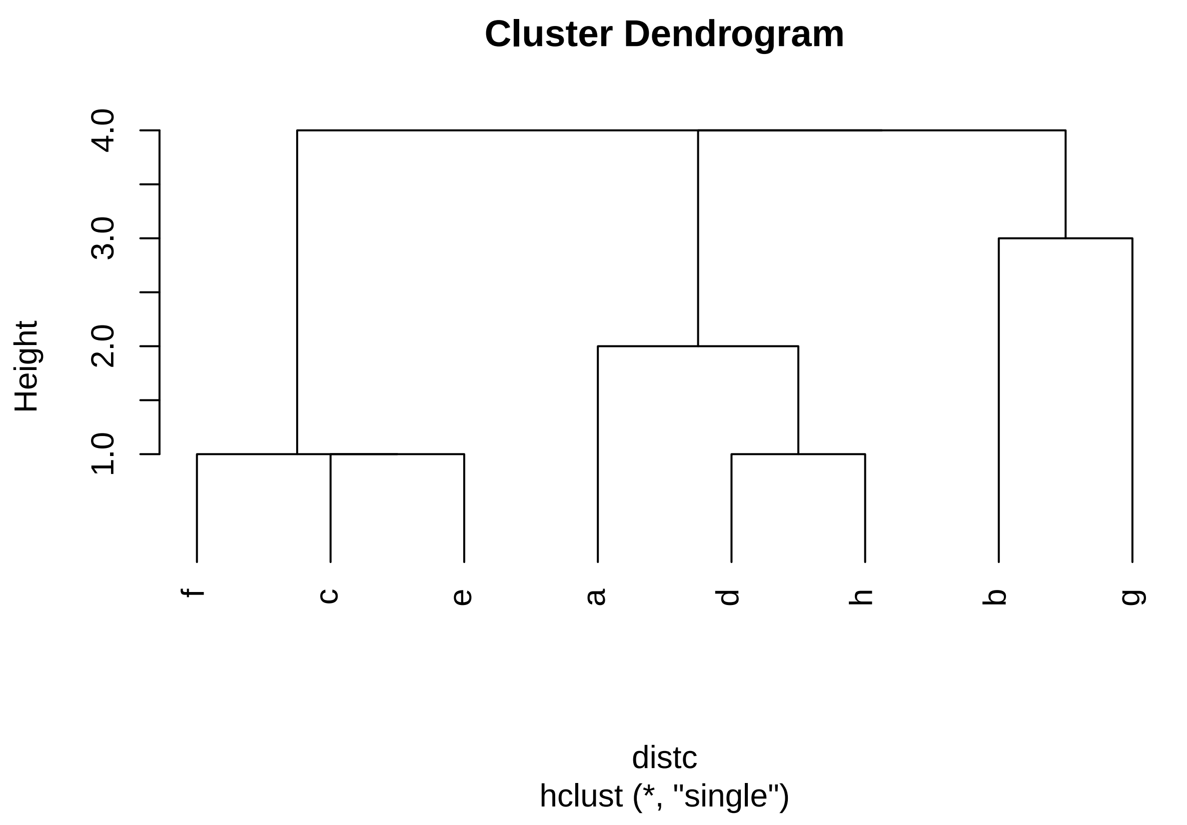
Problem 4

Part a

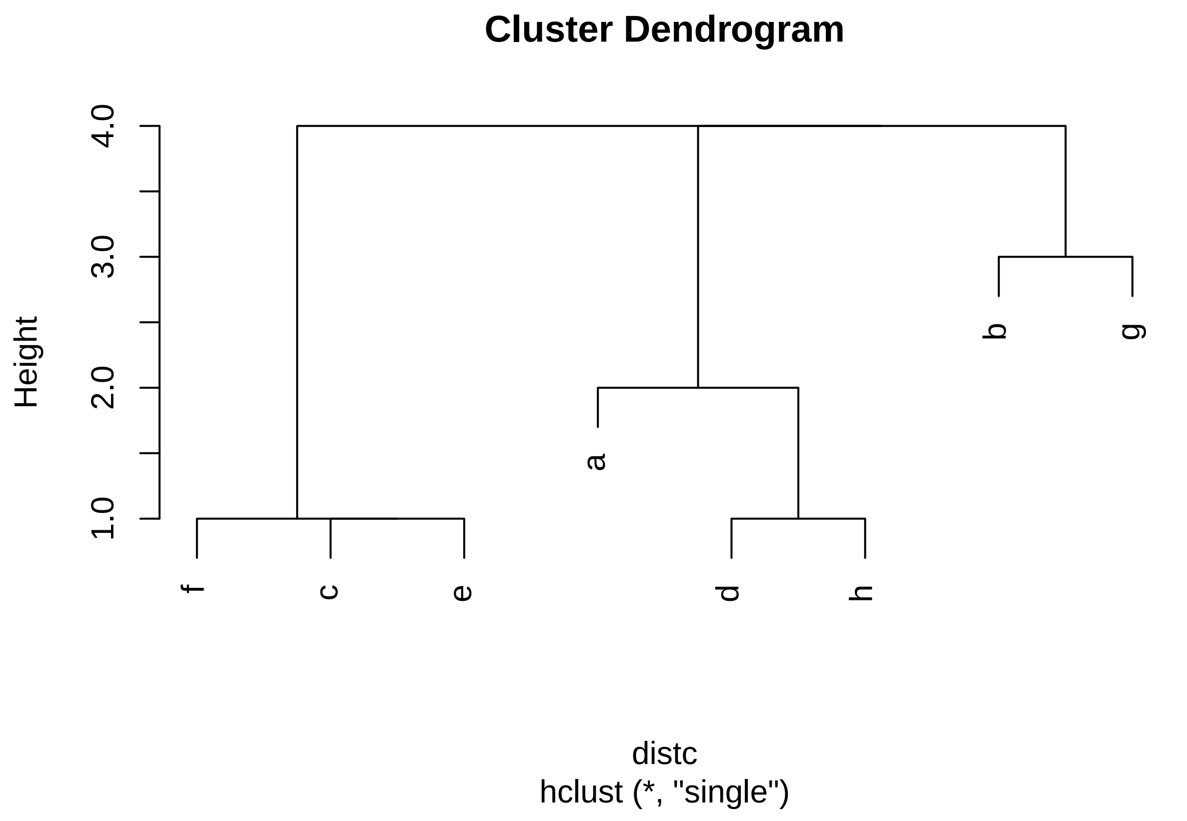
```
distc = read.delim('distances.csv', header=TRUE, sep = ',', row.names = 1)
distc = as.dist(distc)
print(distc)
```

```
##   a b c d e f g
## b 5
## c 8 6
## d 4 4 5
## e 7 5 1 4
## f 7 4 2 4 1
## g 8 3 7 7 7 5
## h 2 4 6 1 5 5 8
```

```
hc = hclust(distc, method='single')
plot(hc, hang = -1)
```


Cluster Dendrogram
Height
4.0
3.0
2.0
1.0
f
c
e
a
d
h
b
g
distc
hclust (*, "single")

```
plot(hc)
```



```r
clusters = cutree(hc, k=4)
```

Part b

```r
library(dbscan)
```

```
##
## Attaching package: 'dbscan'

## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```r
dbscan_result <- dbscan(distc, eps = 6, minPts = 2, borderPoints = FALSE)
dbscan_result
```

```
## DBSCAN clustering for 8 objects.
## Parameters: eps = 6, minPts = 2
## Using unknown distances and borderpoints = FALSE
## The clustering contains 1 cluster(s) and 0 noise points.
##
## 1
## 8
##
## Available fields: cluster, eps, minPts, dist, borderPoints
```

```r
dbscan_result <- dbscan(distc, eps = 6, minPts = 2, borderPoints = TRUE)
```