INFO 4604 - 5604: Exam 1 - Part 2

Julia Troni

Student ID: 109280095

In [1]:
```
#Instructions:

#1. Write your name and student ID below:
#2. Each question is worth 1 point
#3. Remember to save your work and upload the file back into Canvas
#4. Deadline for submission on Canvas is 11:59PM Boulder Time, February 16, 2023
```

In [2]:
```
#Student Name: Julia Troni

#Student ID:109280095
```

In [3]:
```
#RUN THIS CELL FIRST TO ANSWER QUESTIONS 1 - 2

# load numpy
import numpy as np
# load pandas
import pandas as pd

#Create a numpy array with 10 integers
# We use Numpy's random module to generate random numbers
# between 25 and 200

# set a random seed to reproduce
np.random.seed(123)
# create 10 random integers
x = np.random.randint(low=55, high=650, size=10)
```

In [4]:
```
#Question 1

#Write Python code to discretize x into the following 3 categories:

#values less than 55
#values greater than 55 but less than 300
#values greater than 300

#Do not assign the code to an object

#Write your answer below:

np.digitize(x,bins=[55,300])
```

Out[4]:
```
array([2, 2, 2, 2, 1, 1, 1, 1, 2, 1], dtype=int64)
```

In [5]:
```

```

```
#Question 2

#Write Python code to discretize x into the following 4 categories:

#values less than 55
#values greater than 55 but less than 80
#values greater than 80 but less than 150
#values greater than 150

#Do not assign the code to an object - so running this cell displays the result without

#Write your answer below:

np.digitize(x,bins=[55,80,150])
```

Out[5]:
```
array([3, 3, 3, 3, 3, 1, 3, 3, 3, 3], dtype=int64)
```

In [6]:
```
#RUN THIS CELL FIRST TO ANSWER QUESTIONS 3 - 4

#Create a Pandas dataframe with the data we stored in object
# x above

df = pd.DataFrame({"height":x})
df.head() # lists the first few observations in dataframe df
```

Out[6]:

	height
0	565
1	420
2	437
3	377
4	153

In [7]:
```
#Question 3

#Create a column called binned in the dataframe df
#Use the cut() function to discretize values in the dataframe df into the column you cr

#height values between 100 and 150 are in one category
#height values between 150 and 300 are in a second category
#height values between 300 and 500 are in a third category
#height values between 500 and 600 are in a fourth category


#Write your answer below:


df['binned']=pd.cut(x=df['height'], bins=[100,150,300,500,600])
```

In [8]:
```
#Question 4SS
```

```
#Write code to display the first 5 rows of the resulting dataframe df in Question 3 - s
df.head()
```

Out[8]:

	height	binned
0	565	(500, 600]
1	420	(300, 500]
2	437	(300, 500]
3	377	(300, 500]
4	153	(150, 300]

In [9]:

```
#RUN THIS CELL FIRST TO ANSWER QUESTIONS 4 - 5

# dictionary of lists
dict = {'First Score':[100, 90, np.nan, 95],
        'Second Score': [30, 45, 56, np.nan],
        'Third Score':[np.nan, 40, 80, 98]}
```

In [10]:

```
#Question 4

#Write Python code to convert the dictionary above into a dataframe. Store your result

#Create a dataframe from dictionary above
df2= pd.DataFrame(dict)


#Write Python code to fill missing values in dataframe df2 with zeros

# filling missing value using fillna()  - replace/hard code missing
# values with a 0
df2.fillna(0)
```

Out[10]:

	First Score	Second Score	Third Score
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

In [11]:

```
#Question 5

#Write Python code to fill missing values in dataframe df2 in Question 4 above with the

# fill missing value using fillna()  - replace/hard code missing
# values with the mean value of each column

#Write your code below:

df2.fillna(df2.mean())
```

Out[11]:

	First Score	Second Score	Third Score
0	100.0	30.000000	72.666667
1	90.0	45.000000	40.000000
2	95.0	56.000000	80.000000
3	95.0	43.666667	98.000000

In [12]:

```
#RUN THIS CELL FIRST TO ANSWER QUESTIONS 6 - 15

from numpy import array
from numpy.linalg import norm
from math import inf

# define first vector
a = array([4, 12, 13, -65])

# define second vector
b = array([0.5, 0.5, 0.5,0.4])

#define third vector
c = array([4, -2, 7])

#define fourth vector
f = array([5, -22, 17,17])

#define first matrix
A = array([
[4, 25, 3],
[2, -5, 6],
[0.2, 0.4,-0.9]])

# define second matrix
B = array([
[1, 2, 3],
[4, 5, 6]])

#define third matrix
V = array([
[36, 69, 55],
[41, 15, 26],
[-0.7, 6,-0.89]])

#define fourth matrix
G = array([
[5, 55, 6],
[4, 52, 13]])

#define fifth matrix
Q = array([
[0.1, 34, 22],
[47, 13, 26],
[-0.9, 16,-0.11]])
```

In [13]:

```
#Question 6
```

```
#Write Python code below to implement the Hadamard product of matrix V and matrix A

V*A
```

Out[13]:
```
array([[ 1.440e+02,  1.725e+03,  1.650e+02],
       [ 8.200e+01, -7.500e+01,  1.560e+02],
       [-1.400e-01,  2.400e+00,  8.010e-01]])
```

In [14]:
```
#Question 7

#Write Python code below to implement the Hadamard product of matrix A and the transpos

A*V.T
```

Out[14]:
```
array([[ 1.440e+02,  1.025e+03, -2.100e+00],
       [ 1.380e+02, -7.500e+01,  3.600e+01],
       [ 1.100e+01,  1.040e+01,  8.010e-01]])
```

In [15]:
```
#Question 8

#Write Python code below to implement element-wise division of matrix G by matrix B

G/B
```

Out[15]:
```
array([[ 5.        , 27.5       ,  2.        ],
       [ 1.        , 10.4       ,  2.16666667]])
```

In [16]:
```
#Question 9

#Write Python code below to (1) compute the Euclidean norm of vector f, and (2) to norm

eu= norm(f,2)
print(eu)
normalized= f/eu
print(normalized)
```

```
32.96968304366907
[ 0.15165448 -0.66727969  0.51562522  0.51562522]
```

In [17]:
```
#Question 10

#Write Python code below to:

#(1) compute the max norm of vector a
maxnorm = norm(a, inf)
print(maxnorm)

#(2) normalize vector a with its max norm
normalize= a/maxnorm
print(normalize)

#(3) add the result of (2) to vector b
add=b+normalize
print(add)
```

```
65.0
```

```
[ 0.06153846  0.18461538  0.2        -1.        ]
[ 0.56153846  0.68461538  0.7        -0.6       ]
```

In [18]:
```
#Question 11

#Find the matrix-matrix dot product of A, V and Q in that order. Write your Python code
AV=A.dot(V)
AVQ= AV.dot(Q)
print(AVQ)
```

```
[[30779.093   62248.88    42970.3937 ]
 [ 4662.086   -3783.24     -441.6126 ]
 [  659.2421   1366.236     905.01789]]
```

In [19]:
```
#Question 12

#Find the matrix-vector product of matrix Q and vector c. Write your Python code below:
Q.dot(c)
```

Out[19]:
```
array([ 86.4 , 344.  , -36.37])
```

In [20]:
```
#Question 13

#(1)Find the sum of A and V.

#(2)Divide your answer to (1) by Q
#Write your Python code lines below:

sumAV=A+V
print("sum of A and V \n", sumAV)
Q13=sumAV/Q
print("Divide by Q \n", Q13)
```

```
sum of A and V
 [[40.   94.   58.  ]
 [43.   10.   32.  ]
 [-0.5   6.4  -1.79]]
Divide by Q
 [[400.          2.76470588   2.63636364]
 [  0.91489362   0.76923077   1.23076923]
 [  0.55555556   0.4         16.27272727]]
```

In [21]:
```
#Question 14


#(1)Find the sum of A and the division of V by Q.

#Write your Python code line(s) below:

Q14=A+(V/Q)
print(Q14)
```

```
[[364.          27.02941176   5.5       ]
 [  2.87234043  -3.84615385   7.        ]
 [  0.97777778   0.775        7.19090909]]
```

In [22]:
```
#Question 15


#What is the sum of the top left value/entry of the results in Question 13 and Question

#Write your answer below

print("top left Q13 ", Q13[0][2])
print("top left Q14 ", Q14[0][2])
```

```
top left Q13  2.6363636363636362
top left Q14  5.5
```

In [23]:
```
#Question 16

#Given the setup and matrix below:

from numpy import array
from numpy.linalg import norm
from math import inf


# define matrix
K = array([[1, 2], [3, 4], [5, 6]])

# If we define scalar b as follows:
b = 0.9

#Write Python code below to implement scalar multiplication of K by b

K*b
```

Out[23]:
```
array([[0.9, 1.8],
       [2.7, 3.6],
       [4.5, 5.4]])
```

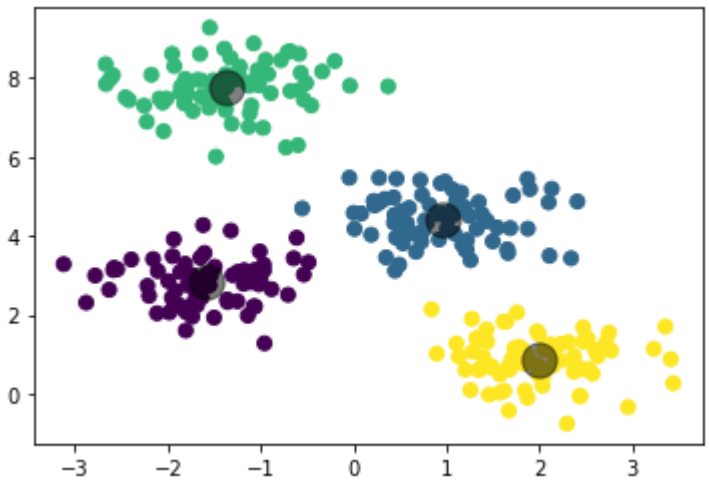
In [24]:
```
#Question 17

#Given the setup and matrix below:

from numpy import array
from numpy.linalg import norm
from math import inf

# define matrix
B = array([[13, -4], [-23, 4], [0.5, 61]])

# If we define vector d as follows:
d = [23, 4, 15, -5]

#Write Python code below to implement scalar multiplication of B by the Euclidean norm

B*norm(d,2)
```

Out[24]:
```
array([[ 366.54467668, -112.78297744],
       [-648.50212027,  112.78297744],
       [  14.09787218, 1719.94040594]])
```

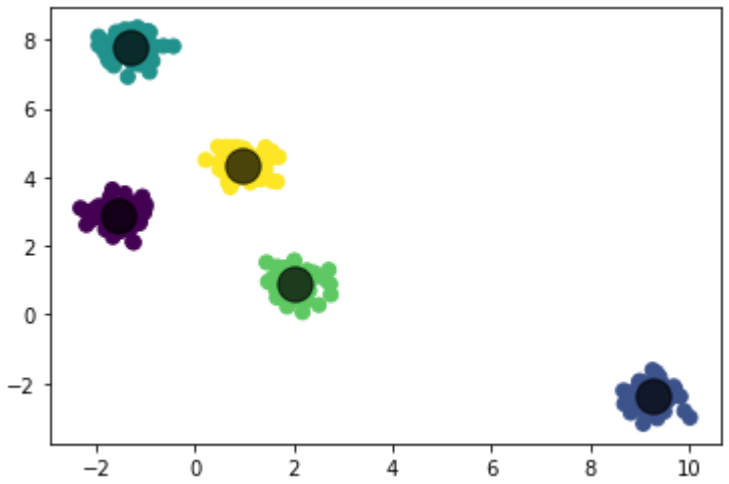
In [25]:
```
#Question 18

from numpy import array
from numpy.linalg import norm
from math import inf

# define matrix B as follows
B = array([[11, 24], [-21, 14], [0.5, 0.61]])

# define matrix B as follows
T = array([[13, -4], [-23, 4], [0.5, 61]])

#Write Python code below to find the dot product of B with the transpose of T

B.dot(T.T)
```

Out[25]:
```
array([[  47.  , -157.  , 1469.5 ],
       [-329.  ,  539.  ,  843.5 ],
       [   4.06,   -9.06,   37.46]])
```

In [26]:
```
#RUN THIS CELL FIRST TO ANSWER QUESTIONS 19 - 20

from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

#generate dummy cluster datasets
K = 4
X, y_true = make_blobs(n_samples=300, centers=K,
                       cluster_std=0.60, random_state=0)

#Fit Model
k_means = KMeans(K)
k_means.fit(X)

cluster_centres = k_means.cluster_centers_

y_kmeans = k_means.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
for centroid in cluster_centres:
    plt.scatter(centroid[0],  centroid[1], s=300,  c='black', alpha=0.5)
```



In [27]:
```
#Question 19

#Perform the following tasks to modify the code above


#1. Copy and paste the entire code above into this cell
#2. Assign 5 to K to generate a dummy cluster dataset
#3. increase the number of samples n_samples to 400
#4. Change the cluster standard deviation (cluster_std) to 0.3
#5. Reduce the transparency of the centroids by changing alpha to 0.7
#6. Run this cell when you have performed (1) - (5) above

from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

#generate dummy cluster datasets
K = 5
X, y_true = make_blobs(n_samples=400, centers=K,
                       cluster_std=0.30, random_state=0)

#Fit Model
k_means = KMeans(K)
k_means.fit(X)

cluster_centres = k_means.cluster_centers_

y_kmeans = k_means.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
for centroid in cluster_centres:
    plt.scatter(centroid[0],  centroid[1], s=300,  c='black', alpha=0.7)
```



In [28]:
```
## Question 20

#Look at the cluster map in your answer in Question 19. Divide the number of clusters i
#corner of the map by 2. What result do you get?

#Write your answer (numeric value) rounded to 1 decimal place below:

1/2
```

Out[28]: 0.5

In [29]:
```
###END OF EXAM 1 - PART 2#######
```

In [ ]: