Question 1

[20 points] Download the dataset called 'country_information.xlsx' that can be found under the 'Data' tab on BlackBoard. Do the following:

```
a. [10 points] Provide a summary of what the dataset is about (around 100 words) by
checking the variable names.
b. [10 points] Excluding the 'country' column, apply 0-1 normalization on the numeric
columns. Save the resulting dataset as:
'country_information_normalized.xlsx'
[Note: Do not forget to add the 'country' column
to the normalized dataset. For normalization, you can use a package.]
```

In [261…

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
```

In [278…

```
# import the dataset
df_countryInfo = pd.ExcelFile(r"/Users/haileythanki/Downloads/country_information.xlsx").parse(0)
df_countryInfo.head()
```

Out[278…

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism	effecti
0	Afghanistan	40.09274	19.0	1	1.0	2.0	1.0	4.092874	
1	Albania	33.20000	36.0	2	3.0	3.0	2.0	4.940516	
2	Algeria	27.60000	36.0	1	3.0	2.0	2.0	4.850953	
3	Argentina	42.90000	42.0	3	4.0	3.0	3.0	8.000000	
4	Australia	34.40000	77.0	3	4.0	4.0	4.0	5.783460	

The dataset has information about 129 of 195 countries. Some of these attributes are as follows:

gini_index: gini index measures the extent to which the distribution of income among the population in a country deviates from a perfectly equal distribution. A gini index of 0 represents perfect equality and 100 would perfect inequality.

corruption_perceptions_index: the Corruption Perceptions Index (CPI) is an index that ranks countries by their perceived levels of corruption determined by expert assessments and opinion surveys.

freedom_house: a score of 0 represents the smallest degree of freedom and 4 the greatest degree of freedom.

hdi: The Human Development Index is a combined index that represents life expectancy, education, and per capita income indicators. It ranks countries into four tiers of human development.

The target attribute seems to be the colonized column. The descriptor variables are essentially useful in determining the contribution or consequence of factors like perceived level of corruption, trust in science, human development index which are a few of the descriptor variables for the colonization of a region.

In [280…

```
# scale the data except the "country" column

X = df_countryInfo.iloc[:, 1:13]
min_max_scaler = preprocessing.MinMaxScaler()
scaled_x = min_max_scaler.fit_transform(X)
scaled_x_df = pd.DataFrame(scaled_x)
scaled_x_df.head()
```

Out[280…

	0	1	2	3	4	5	6	7	8	9	10	11
0	0.415418	0.092105	0.0	0.000000	0.25	0.000000	0.441839	0.109375	0.501952	0.743412	0.153846	1.0
1	0.239518	0.315789	0.5	0.666667	0.50	0.333333	0.562931	0.593750	0.306081	0.519163	0.153846	1.0
2	0.096609	0.315789	0.0	0.666667	0.25	0.333333	0.550136	0.515625	0.257206	0.738070	0.057692	0.0
3	0.487058	0.394737	1.0	1.000000	0.50	0.666667	1.000000	0.453125	0.459459	0.153846	0.365385	1.0
4	0.270142	0.855263	1.0	1.000000	0.75	1.000000	0.683351	0.890625	0.594595	0.323077	0.596154	1.0

In [281…

```
# join the scaled data with the country column

country_information_normalized = pd.DataFrame(df_countryInfo.iloc[:,0]).join(scaled_x_df)
country_information_normalized.head()
```

Out[281…

	country	0	1	2	3	4	5	6	7	8	9	10	11
0	Afghanistan	0.415418	0.092105	0.0	0.000000	0.25	0.000000	0.441839	0.109375	0.501952	0.743412	0.153846	1.0

	country	0	1	2	3	4	5	6	7	8	9	10	11
1	Albania	0.239518	0.315789	0.5	0.666667	0.50	0.333333	0.562931	0.593750	0.306081	0.519163	0.153846	1.0
2	Algeria	0.096609	0.315789	0.0	0.666667	0.25	0.333333	0.550136	0.515625	0.257206	0.738070	0.057692	0.0
3	Argentina	0.487058	0.394737	1.0	1.000000	0.50	0.666667	1.000000	0.453125	0.459459	0.153846	0.365385	1.0
4	Australia	0.270142	0.855263	1.0	1.000000	0.75	1.000000	0.683351	0.890625	0.594595	0.323077	0.596154	1.0

In [282…

```
# rename the columns

col_names_old = country_information_normalized.columns.values.tolist()
col_names_new =  df_countryInfo.columns.values.tolist()[0:13]
new_columns = dict(zip(col_names_old, col_names_new))
country_information_normalized = country_information_normalized.rename(index=str, columns=new_columns)
country_information_normalized.head()
```

Out[282…

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism
0	Afghanistan	0.415418	0.092105	0.0	0.000000	0.25	0.000000	0.441839
1	Albania	0.239518	0.315789	0.5	0.666667	0.50	0.333333	0.562931
2	Algeria	0.096609	0.315789	0.0	0.666667	0.25	0.333333	0.550136
3	Argentina	0.487058	0.394737	1.0	1.000000	0.50	0.666667	1.000000
4	Australia	0.270142	0.855263	1.0	1.000000	0.75	1.000000	0.683351

In [266…

```
# export dataframe as excel file

country_information_normalized.to_excel(r'/Users/haileythanki/Downloads/country_information_normalized.xlsx', i
```

In [283…

```
# read exported and stored data frame

df_country_information_normalized = pd.ExcelFile(r"/Users/haileythanki/Downloads/country_information_normalized
df_country_information_normalized.head()
```

Out[283…

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism
0	Afghanistan	0.415418	0.092105	0.0	0.000000	0.25	0.000000	0.441839

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism
1	Albania	0.239518	0.315789	0.5	0.666667	0.50	0.333333	0.562931
2	Algeria	0.096609	0.315789	0.0	0.666667	0.25	0.333333	0.550136
3	Argentina	0.487058	0.394737	1.0	1.000000	0.50	0.666667	1.000000
4	Australia	0.270142	0.855263	1.0	1.000000	0.75	1.000000	0.683351

Question 2

[20 points] Code the kmeans++ algorithm from scratch. For more information about the individual steps of the algorithm, please check here: https://en.wikipedia.org/wiki/K-means%2B%2B.

As input, your algorithm should take a numpy matrix or a pandas dataframe and a k value that denotes the expected number of clusters. The output needs to be the labels associated with feature vectors coming from your dataset.

Note: You are welcome to use pre-packaged algorithms to calculate distances and means. If you need to pick a point randomly, please do the following:

```
i. Import the random package of Python.
ii. Set seed to 265 by running the following line: random.seed(265) [This should be
done at the very beginning of your code file, after importing the packages.]
iii. Run the following line: randrange(0,len(name_of_your_dataset),1).
```

Use the resulting the number as the index number for the data point that should be randomly picked in different stages of the kmeans++ algorithm.

In [268…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import math
from copy import deepcopy
from numpy import linalg
```

```
%matplotlib inline
```

In [269…

```
random.seed(265)
```

In [270…

```
from sklearn.preprocessing import MinMaxScaler

df_country_information_normalized = pd.ExcelFile(r"/Users/haileythanki/Downloads/country_information_normalized
X = df_country_information_normalized.drop(columns='country',axis=1).values
y = df_country_information_normalized.country
random.randrange(0,len(X),1)
```

Out[270… 100

In [271…

```
## initialisation algorithm
def KmeansPP(X, k, init='k-means++', max_iter=1, bfig=False):

    #initialization function

    def kmeansPP_init(X, k):

        # number of instances in the dataframe

        num_rows = X.shape[0]

        # The number of attributes in the dataframe

        num_cols = X.shape[1]

        # initialize the centroid list and randomly select a data point as the centroid

        all_centroids = np.zeros((k, num_cols))
        all_centroids[0, :] = X[np.random.randint(num_rows), :]

        # compute the remaining centroids

        for num_clusters in range(1, k):

            # initialize a list to store distances of points from all centroids

            all_distances = np.zeros((num_clusters, num_rows))
```

```
            # initialize a list to store shortest distance between points and centroids

            shortest_distance = np.zeros(num_rows)

            # calculate distance between point and centroids, store in a list

            for num_clusters_pre in range(num_clusters):
                all_distances[num_clusters_pre, :] = linalg.norm(X - all_centroids[num_clusters_pre, :], axis=1

            # store indices of all shortest distances between points and centroids

            shortest_distance_idx = np.argmin(all_distances, axis=0)

            for i in range(len(shortest_distance_idx)):
                shortest_distance[i] = all_distances[shortest_distance_idx[i], i]

            max_distance = np.argmax(shortest_distance) % num_rows
            all_centroids[num_clusters, :] = X[max_distance, :]

        return all_centroids

    # number of rows in training data

    num_rows = X.shape[0]

    # number of attributes in the data

    num_cols = X.shape[1]

    if init == 'random':
        mean = np.mean(X, axis=0)
        std = np.std(X, axis=0)
        centers_init = np.random.randn(k, num_cols)*std + mean
    else:
        centers_init = kmeansPP_init(X, k)

    # store new centers in list

    centers = deepcopy(centers_init)
    clusters = np.zeros(num_rows)
    all_distances = np.zeros((num_rows, k))

    # break loop if the values dont change
```

```
    for i in range(max_iter):

        # Measure the distance to every centroid

        for num_clusters in range(k):
            all_distances[:, num_clusters] = linalg.norm(X - centers[num_clusters], axis=1)

        # assign all points to closest centroid

        clusters = np.argmin(all_distances, axis=1)
        centers_pre = deepcopy(centers)

        # calculate mean of every cluster and update the centroid

        for num_clusters in range(k):
            if (clusters == num_clusters).any():
                centers[num_clusters, :] = np.mean(X[clusters == num_clusters], axis=0)

        error = linalg.norm(centers - centers_pre)

        if error == 0:
            break

    return clusters, centers
```

Question 3

[20 points] Now, we will test the code we have written in Q2 and apply dimension reduction: Specifically, do the following:

[10 points]. Set the random seed to 265 again (to (re-)guarantee the same initialization). Set k = 6. Run your kmeans++ code on the 'country_information_normalized.xlsx' dataset by excluding the 'country' column.* Record the labels. Attach the labels as a new column to your dataset by naming your new variable as kmeans_label.

[10 points] Excluding the 'country' and 'kmeans_label' columns, run dimension reduction (specifically PCA) on your dataset by using sklearn's PCA function: https://scikit- learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[Note: set n_components = 2 and random_state = 265. Other parameters should be left as 'default'.]. Add the new variables in your dataset as pca_dim_1 and pca_dim_2.

For the next question, use the attached 'visualization_code.py' file.

In [272…
```
from sklearn.decomposition import PCA
```

In [273…
```
clusters, centroids = KmeansPP(df_country_information_normalized.iloc[:,1:].values,6,bfig=True)
print(clusters)
print(centroids)
```

```
[4 4 0 1 1 1 5 5 1 4 4 4 4 0 0 0 1 1 5 4 0 1 1 5 1 1 4 0 4 0 4 1 4 0 1 1 0
 0 4 1 2 4 4 0 4 3 4 1 0 0 5 0 1 2 4 0 1 0 5 0 4 3 4 1 0 0 1 0 0 3 0 4 4 4
 4 0 0 4 1 1 0 5 4 0 0 0 4 4 0 4 1 3 4 5 0 5 0 4 3 1 2 1 0 1 3 0 1 1 0 1 5
 0 4 0 2 5 2 0 4 0 1 1 1 5 2 5 0 0 0]
[[0.3511815  0.25703109 0.24390244 0.30894309 0.29878049 0.21250661
  0.49113667 0.30754573 0.53950981 0.65146652 0.28757373 0.        ]
 [0.23218149 0.75933786 1.         1.         0.81451613 0.87096774
  0.73121597 0.80897177 0.55226837 0.40416537 0.4292804  1.        ]
 [0.47684995 0.43033056 1.         0.54571939 0.62269641 0.64927912
  0.53411077 0.36458333 0.64505449 0.52221581 0.30891387 0.        ]
 [0.25797794 0.62280702 0.41666667 0.94444444 0.33333333 0.44444444
  0.55852932 0.72886545 0.46297016 0.51775505 0.34192878 0.        ]
 [0.35227703 0.33059211 0.546875   0.66427863 0.4375     0.49143129
  0.52706674 0.46985106 0.41907789 0.54465664 0.15504808 1.        ]
 [0.29926338 0.28340081 0.         0.74855226 0.11538462 0.02564103
  0.42946884 0.45673077 0.53369751 0.58814551 0.38023795 1.        ]]
```

In [274…
```
df_country_information_normalized['kmeans_label'] = clusters
```

In [275…
```
pca = PCA(n_components=2, random_state=265)
prin_comps = pca.fit_transform(X)
df_prin_comp = pd.DataFrame(data = prin_comps, columns = ['pca_dim_1', 'pca_dim_2'])
df_country_information_normalized['pca_dim_1'] = df_prin_comp.iloc[:].pca_dim_1
df_country_information_normalized['pca_dim_2'] = df_prin_comp.iloc[:].pca_dim_2
df_country_information_normalized['kmeans_label']=clusters
df_country_information_normalized.head()
```

Out[275…

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism
0	Afghanistan	0.415418	0.092105	0.0	0.000000	0.25	0.000000	0.441839
1	Albania	0.239518	0.315789	0.5	0.666667	0.50	0.333333	0.562931
2	Algeria	0.096609	0.315789	0.0	0.666667	0.25	0.333333	0.550136
3	Argentina	0.487058	0.394737	1.0	1.000000	0.50	0.666667	1.000000

	country	gini_index	corruption_perceptions_index	freedom_house	hdi	press_freedom	democracy_economist	populism
4	Australia	0.270142	0.855263	1.0	1.000000	0.75	1.000000	0.683351

Question 4

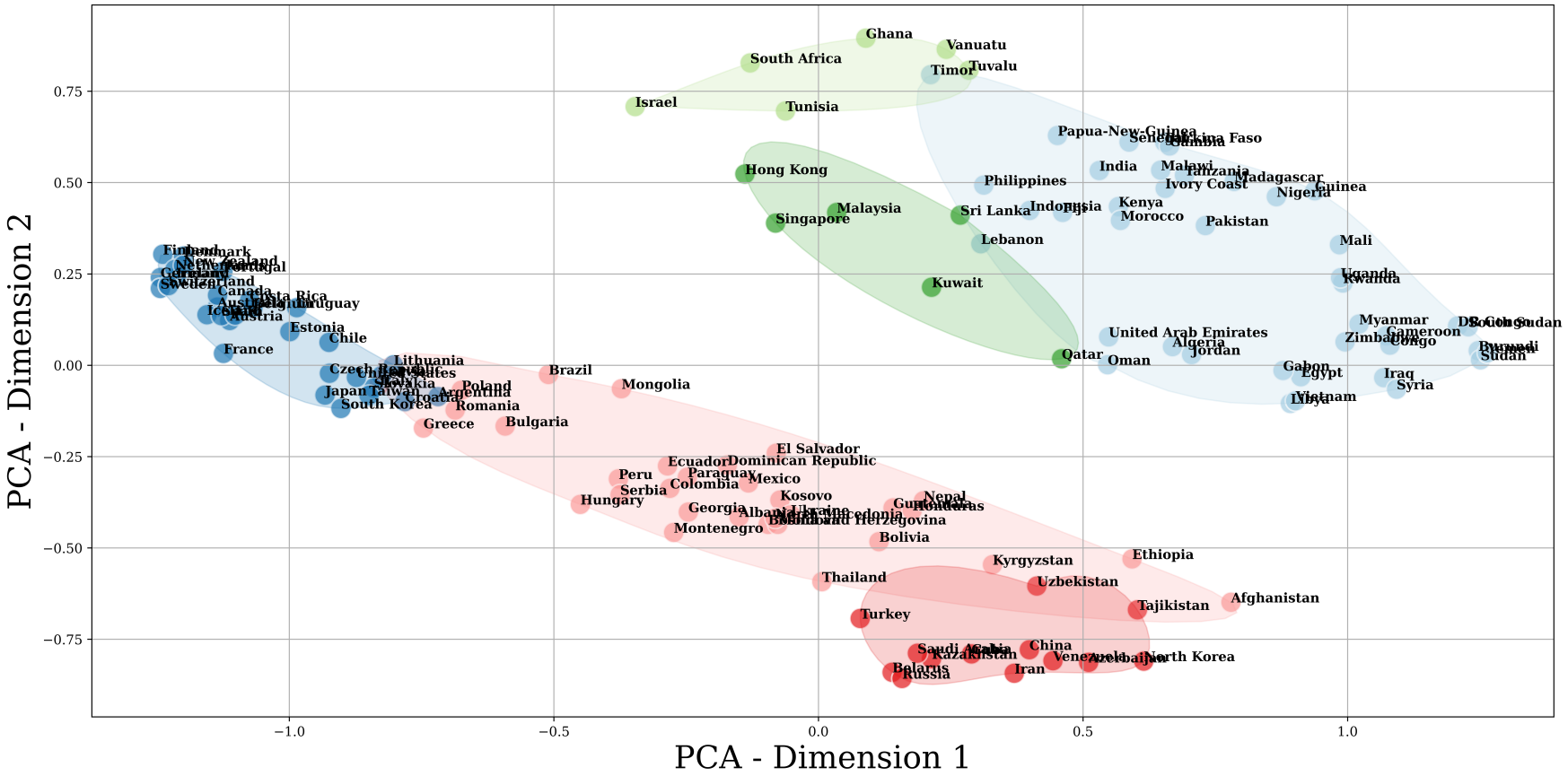
[20 points] Now, let's visualize the results, use the clustering labels to color our data points, and present them in convex hulls. Run the code provided to you in the 'visualization_code.py' file. Change the name of the dataset where it says [...]. Add the visual to your .pdf submission.

Note: For this exercise, you will need to find and explore the required packages that will need to be imported. The resulting plot should look (somewhat) similar to what is below (but, you will have k = 6).

In [276…

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns

mpl.rcParams['figure.dpi'] = 600
plt.rcParams['figure.figsize'] = (20.0, 10.0)
plt.rcParams['font.family'] = "serif"
plt.rcParams['font.size'] = 10

df = df_country_information_normalized
pal = sns.color_palette("Paired")[:len(set(df['kmeans_label']))]
p1 = sns.scatterplot(x="pca_dim_1", y='pca_dim_2', hue='kmeans_label', palette = pal, data=df, s=250, alpha=0.7

#For each point, we add a text inside the bubble
for line in range(0,df.shape[0]):
     p1.text(df.pca_dim_1[line], df.pca_dim_2[line], df.country[line], horizontalalignment='left', size='medium

plt.suptitle('Two-Dimensional Map of Countries (PCA)', fontsize=36)
plt.xlabel('PCA - Dimension 1', fontsize=24)
plt.ylabel('PCA - Dimension 2', fontsize=24)

from scipy import interpolate
from scipy.spatial import ConvexHull

for i in df.kmeans_label.unique():
    # get the convex hull
    points = df[df.kmeans_label == i][['pca_dim_1', 'pca_dim_2']].values
```

```
    hull = ConvexHull(points)
    x_hull = np.append(points[hull.vertices,0],
                       points[hull.vertices,0][0])
    y_hull = np.append(points[hull.vertices,1],
                       points[hull.vertices,1][0])

    # interpolate
    dist = np.sqrt((x_hull[:-1] - x_hull[1:])**2 + (y_hull[:-1] - y_hull[1:])**2)
    dist_along = np.concatenate(([0], dist.cumsum()))
    spline, u = interpolate.splprep([x_hull, y_hull],
                                    u=dist_along, s=0)
    interp_d = np.linspace(dist_along[0], dist_along[-1], 50)
    interp_x, interp_y = interpolate.splev(interp_d, spline)
    # plot shape
    plt.fill(interp_x, interp_y, '--', c=pal[i], alpha=0.2)


plt.grid()
plt.show()
```



Question 5

[20 points] Interpret the results (in around 300 words) by answering the following:

a. [5 points] Which countries seem to be similar? Why do you think these countries are clustered together?

b. [5 points] If you run the kmeans++ algorithm more than once, do you think the results will change?

c. [5 points] (Subjectively speaking) Do you think this is an accurate clustering of the countries? Would the results change greatly if we had different social/economic variables?

d. [5 points] Do you think PCA may have affected the results at all? In other words, if we had a different number of principle components, would our visual interpretation be different?

a. The countries clusteres very closely together in the far left margin seem to be the most similar out of all the others. This cluster of countries includes, France, Japan, South Korea, Finland, Denmark, Sweden etc. These countries have similar values for socio-economic varibles hence thet are more likely to be clustered together.

b. Yes, the results will change since the centroid is picked randomly during initialization.

c. Yes, given this data it is an appropriate clustering. If more variables are introduced then the clusterng may change but not very significantly since the data already has a large number of independent variables.

d. PCA likely gave us better results since it is known to reduce the noise and help clustering algorithms perform better. If there were more components, the visualizatio would not have been easy to interpret.

In [ ]: