HW10

1. **Kaggle public LB rank & score**: 4th place, 0.52522
2. **Kaggle team name**: DF-UCK-Xu-Sokolov-Yangirov
   A. Use format: `[ProGroup ID]-[Your fancy Kaggle group name]-[Team member names]`.
   B. Eg. `DA-Avengers-Lazareva,Iazykova,Ovyan` (see Rules in Kaggle for ProGroup ID)
3. Our Colab uses [CPU]

(Your private LB score must be reproducible exactly. Seed all RNG (https://en.wikipedia.org/wiki/Random_number_generation). Don't exceed runtime.)

Theory Task (DSBA/ICEF only)

Task 1

Suppose, we want to apply k-means for the following dataset with the five observations of two variables:

$$\{(0, 11), (1, 8), (0.1, 12), (1.1, 4), (1.2, 3)\}$$

We seek $k := 2$ clusters with a Euclidean distance (https://en.wikipedia.org/wiki/Euclidean_distance). Our random initialization ended up with the following cluster centers $C_1 = (1.0, 4.5)$ and $C_2 = (0.6, 11)$

Questions

---

1) What are the initial clusters (i.e. which observations they contain)?

SOLUTION 1.1:

According to the algorithm of K-means clustering, the distances are calculated by using the Euclidean distance $d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$ (in two dimensions). By comparing the distances between the observation and randomly generated centers.

$P_1 = (0, 11)$

- $d(P_1, C_1) = \sqrt{(1 - 0)^2 + (4.5 - 11)^2} = 6.58$
- $d(P_1, C_2) = \sqrt{(0.6 - 0)^2 + (11 - 11)^2} = 0.6$

$\because d(P_1, C_1) > d(P_1, C_2) \therefore P_1 \in C_2$

$P_2 = (1, 8)$

- $d(P_2, C_1) = \sqrt{(1 - 1)^2 + (4.5 - 8)^2} = 3.5$
- $d(P_2, C_2) = \sqrt{(0.6 - 1)^2 + (11 - 8)^2} = 3.02$

$\because d(P_1, C_1) > d(P_1, C_2) \therefore P_2 \in C_2$

$P_3 = (0.1, 12)$

- $d(P_3, C_1) = \sqrt{(1 - 0.1)^2 + (4.5 - 12)^2} = 7.55$

- $d(P_3, C_2) = \sqrt{(0.6 - 0.1)^2 + (11 - 12)^2} = 1.12$

$\because d(P_3, C_1) > d(P_3, C_2) \therefore P_3 \in C_2$

$P_4 = (1.1, 4)$

- $d(P_4, C_1) = \sqrt{(1 - 1.1)^2 + (4.5 - 4)^2} = 0.51$
- $d(P_4, C_2) = \sqrt{(0.6 - 1.1)^2 + (11 - 4)^2} = 7.01$

$\because d(P_4, C_1) < d(P_4, C_2) \therefore P_4 \in C_1$

$P_5 = (1.2, 3)$

- $d(P_5, C_1) = \sqrt{(1 - 1.2)^2 + (4.5 - 3)^2} = 1.51$
- $d(P_5, C_2) = \sqrt{(0.6 - 1.2)^2 + (11 - 3)^2} = 8.02$

$\because d(P_5, C_1) < d(P_5, C_2) \therefore P_5 \in C_1$

Hence, our initial clusters are:

- Cluster with C_1 = $\{P_4, P_5\}$ = {(1.1, 4),(1.2, 3)}
- Cluster with C_2 = $\{P_1, P_2, P_3\}$ = {(0,11), (1,8), (0.1, 12)}

---

2) How the cluster centers change on each step of the algorithm? Provide all necessary calculations and justify all steps

3) How cluster assignments change? Provide all necessary calculations and justify all steps

SOLUTION 1.2 & 1.3:

We combined the answers for 1.2 and 1.3, because on each iteration it requires all the final clusters of the previous step.

Now, we start the interation step:

- Compute new cluster centroids.
- Assign each observations to the closest centroid .

Iteration-1

- The new centroids:
  - C_1 = ((1.1 + 1.2)/2 , (4 + 3)/2) = (1.15, 3.5)
  - C_2 = ((0+1+0.1)/3 , (11+8+12)/3) = (0.37, 10.34)
- Assigning each observations to the closest centroid based on Euclidean distance.
  - $P_1 = (0, 11)$
    - $d(P_1, C_1) = \sqrt{(1.15 - 0)^2 + (3.5 - 11)^2} = 7.59$
    - $d(P_1, C_2) = \sqrt{(0.37 - 0)^2 + (10.34 - 11)^2} = 0.76$
    - $\because d(P_1, C_1) > d(P_1, C_2) \therefore P_1 \in C_2$
  - $P_2 = (1, 8)$
    - $d(P_2, C_1) = \sqrt{(1.15 - 1)^2 + (3.5 - 8)^2} = 4.5$
    - $d(P_2, C_2) = \sqrt{(0.37 - 1)^2 + (10.34 - 8)^2} = 2.42$
    - $\because d(P_1, C_1) > d(P_1, C_2) \therefore P_2 \in C_2$

- $P_3 = (0.1, 12)$
  - $d(P_3, C_1) = \sqrt{(1.15 - 0.1)^2 + (3.5 - 12)^2} = 8.56$
  - $d(P_3, C_2) = \sqrt{(0.37 - 0.1)^2 + (10.34 - 12)^2} = 1.68$
  - $\because d(P_3, C_1) > d(P_3, C_2) \therefore P_3 \in C_2$
- $P_4 = (1.1, 4)$
  - $d(P_4, C_1) = \sqrt{(1.15 - 1.1)^2 + (3.5 - 4)^2} = 0.5$
  - $d(P_4, C_2) = \sqrt{(0.37 - 1.1)^2 + (10.34 - 4)^2} = 6.38$
  - $\because d(P_4, C_1) < d(P_4, C_2) \therefore P_4 \in C_1$
- $P_5 = (1.2, 3)$
  - $d(P_5, C_1) = \sqrt{(1.15 - 1.2)^2 + (3.5 - 3)^2} = 0.5$
  - $d(P_5, C_2) = \sqrt{(0.37 - 1.2)^2 + (10.34 - 3)^2} = 7.39$
  - $\because d(P_5, C_1) < d(P_5, C_2) \therefore P_5 \in C_1$

- For the iteration-1, we end up with :
  - Cluster with $C_1 = \{P_4, P_5\}$ = {(1.1, 4),(1.2, 3)}
  - Cluster with $C_2 = \{P_1, P_2, P_3\}$ = {(0,11), (1,8), (0.1, 12)}
- The cluster assignments remained the same as the previous step in 1.1. Therefore, we obtained the final clusters.

Show all of the changes in a table:

step	Previous Assignment	C_1	C_2	Cluster Assignment
0	NULL	$(1, 4.5)$	$(0.6, 11)$	$c_2^{(0)} = \{P_1, P_2, P_3\}; c_1^{(0)} = \{P_4, P_5\}$
1	$c_2^{(0)} = \{P_1, P_2, P_3\}; c_1^{(0)} = \{P_4, P_5\}$	$(1.15, 3.5)$	$(0.37, 10.34)$	$c_2^{(1)} = \{P_1, P_2, P_3\}; c_1^{(1)} = \{P_4, P_5\}$
Stop	$\because$ Assignment unchanged	-	-	-

Task 2

On the internet you may find many recomendations to scale your data before clustering. Let's follow this advice, scale you data by substracting the sample mean and dividing by the sample standard deviation of the variable.

Suppose that we rerun our algorithm and have two initial clusters:

- C_1: the first two observations
- C_2: the latter three

Questions

---

1) What are the initial clusters? What are their centers?

SOLUTION 2.1:

Z-Score Normalization: $P_{new} = \frac{P_{old} - mean}{std}$, the scaled points are:

$$\{(-1.31, 0.94), (0.62, 0.11), (-1.12, 1.22), (0.81, -1), (1, -1.27)\}$$

Given that:

- Cluster with C_1 = $\{P_1, P_2\}$ = {(-1.31,0.94), (0.62, 0.11)}
- Cluster with C_2 = $\{P_3, P_4, P_5\}$ = {(-1.12,1.22), (0.81,-1), (1,-1.27)}

The corresponding centroids of the initial clusters are:

- C_1 = ((-1.31 + 0.62)/2 , (0.94 + 0.11)/2) = (-0.35, 0.53)
- C_2 = ((-1.12+0.81+1)/3 , (1.22-1-1.27)/3) = (0.23, -0.35)

---

2) How the cluster centers change on each step? Provide all necessary calculations and justify all steps

3) How cluster assignments change? Provide all necessary calculations and justify all steps

SOLUTION 2.2 & 2.3:

Again, we combined the answers for 2.2 and 2.3, because on each iteration it requires all the final clusters of the previous step.

$$\{(-1.31, 0.94), (0.62, 0.11), (-1.12, 1.22), (0.81, -1), (1, -1.27)\}$$

Now, we start the interation step:

- Compute new cluster centroids.
- Assign each observations to the closest centroid .

Iteration-0

- The new centroids:
  - C_1 = ((-1.31 + 0.62)/2 , (0.94 + 0.11)/2) = (-0.35, 0.53)
  - C_2 = ((-1.12+0.81+1)/3 , (1.22-1-1.27)/3) = (0.23, -0.35)
- Assigning each observations to the closest centroid based on Euclidean distance.
  - $P_1 = (-1.31, 0.94)$
    - $d(P_1, C_1) = \sqrt{(-0.35 - (-1.31))^2 + (0.53 - 0.94)^2} = 1.05$
    - $d(P_1, C_2) = \sqrt{(0.23 - (-1.31))^2 + (-0.35 - 0.94)^2} = 2.01$
    - $\because d(P_1, C_1) < d(P_1, C_2) \therefore P_1 \in C_1$
  - $P_2 = (0.62, 0.11)$
    - $d(P_1, C_1) = \sqrt{(-0.35 - 0.62)^2 + (0.53 - 0.11)^2} = 1.05$
    - $d(P_1, C_2) = \sqrt{(0.23 - 0.62)^2 + (-0.35 - 0.11)^2} = 0.6$
    - $\because d(P_1, C_1) > d(P_1, C_2) \therefore P_2 \in C_2$
  - $P_3 = (-1.12, 1.22)$
    - $d(P_1, C_1) = \sqrt{(-0.35 - (-1.12))^2 + (0.53 - 1.22)^2} = 1.04$
    - $d(P_1, C_2) = \sqrt{(0.23 - (-1.12))^2 + (-0.35 - 1.22)^2} = 2.07$
    - $\because d(P_3, C_1) < d(P_3, C_2) \therefore P_3 \in C_1$
  - $P_4 = (0.81, -1)$
    - $d(P_1, C_1) = \sqrt{(-0.35 - 0.81)^2 + (0.53 - (-1))^2} = 1.91$
    - $d(P_1, C_2) = \sqrt{(0.23 - 0.81)^2 + (-0.35 - (-1))^2} = 0.87$
    - $\because d(P_4, C_1) > d(P_4, C_2) \therefore P_4 \in C_2$
  - $P_5 = (1, -1.27)$
    - $d(P_1, C_1) = \sqrt{(-0.35 - 1)^2 + (0.53 - (-1.27))^2} = 2.25$
    - $d(P_1, C_2) = \sqrt{(0.23 - 1)^2 + (-0.35 - (-1.27))^2} = 1.2$
    - $\because d(P_5, C_1) > d(P_5, C_2) \therefore P_5 \in C_2$

- For the iteration-0, we end up with :
  - Cluster with C_1 = $\{P_1, P_3\}$ = {(-1.31,0.94), (-1.12,1.22)}
  - Cluster with C_2= $\{P_2, P_4, P_5\}$ = {(0.62, 0.11), (0.81,-1), (1,-1.27)}

Iteration-1

- The new centroids:
  - C_1 = ((-1.31 - 1.12)/2 , (0.94 + 1.22)/2) = (-1.21, 1.08)
  - C_2 = ((0.62+0.81+1)/3 , (0.11-1-1.27)/3) = (0.81, -0.72)
- Assigning each observations to the closest centroid based on Euclidean distance.
  - $P_1 = (-1.31, 0.94)$
    - $d(P_1, C_1) = \sqrt{(-1.21 - (-1.31))^2 + (1.08 - 0.94)^2} = 0.17$
    - $d(P_1, C_2) = \sqrt{(0.81 - (-1.31))^2 + (-0.72 - 0.94)^2} = 2.69$
    - $\because d(P_1, C_1) < d(P_1, C_2) \therefore P_1 \in C_1$
  - $P_2 = (0.62, 0.11)$
    - $d(P_1, C_1) = \sqrt{(-1.21 - 0.62)^2 + (1.08 - 0.11)^2} = 2.07$
    - $d(P_1, C_2) = \sqrt{(0.81 - 0.62)^2 + (-0.72 - 0.11)^2} = 0.85$
    - $\because d(P_1, C_1) > d(P_1, C_2) \therefore P_2 \in C_2$
  - $P_3 = (-1.12, 1.22)$
    - $d(P_1, C_1) = \sqrt{(-1.21 - (-1.12))^2 + (1.08 - 1.22)^2} = 0.17$
    - $d(P_1, C_2) = \sqrt{(0.81 - (-1.12))^2 + (-0.72 - 1.22)^2} = 2.73$
    - $\because d(P_3, C_1) < d(P_3, C_2) \therefore P_3 \in C_1$
  - $P_4 = (0.81, -1)$
    - $d(P_1, C_1) = \sqrt{(-1.21 - 0.81)^2 + (1.08 - (-1))^2} = 2.9$
    - $d(P_1, C_2) = \sqrt{(0.81 - 0.81)^2 + (-0.72 - (-1))^2} = 0.28$
    - $\because d(P_4, C_1) > d(P_4, C_2) \therefore P_4 \in C_2$
  - $P_5 = (1, -1.27)$
    - $d(P_1, C_1) = \sqrt{(-1.21 - 1)^2 + (1.08 - (-1.27))^2} = 3.23$
    - $d(P_1, C_2) = \sqrt{(0.81 - 1)^2 + (-0.72 - (-1.27))^2} = 0.59$
    - $\because d(P_5, C_1) > d(P_5, C_2) \therefore P_5 \in C_2$

- For the iteration-1, we end up with :
  - Cluster with C_1 = $\{P_1, P_3\}$ = {(-1.31,0.94), (-1.12,1.22)}
  - Cluster with C_2= $\{P_2, P_4, P_5\}$ = {(0.62, 0.11), (0.81,-1), (1,-1.27)}

Show all of the changes in a table:

step	Previous Assignment	C_1	C_2	Cluster Assignment
0	$\{P_1, P_2, P_3\} and \{P_4, P_5\}$	$(-0.35, 0.53)$	$(0.23, -0.35)$	$c_1^{(0)} = \{P_1, P_3\}; c_2^{(0)} = \{P_2, P_4, P_5\}$
1	$c_1^{(0)} = \{P_1, P_3\}; c_2^{(0)} = \{P_2, P_4, P_5\}$	$(-1.21, 1.08)$	$(0.81, -0.72)$	$c_1^{(1)} = \{P_1, P_3\}; c_2^{(1)} = \{P_2, P_4, P_5\}$

---

4) Are cluster assignments different from the original data? Why it might happen?

SOLUTION 2.4:

Yes, the final assignments in task 2 has changed. The main reason is that standardization prevents the feature variables with larger scales from dominaing how clusters are defined. Moreover the metric used in our

algorithm is Euclidean Distance which is sensitive to the scale of the direction of space

---

Task 2: Kaggle-Crystals (for DSBA/ICEF/OOC)

Private URL for students (allows submission) is in Moodle's HW assignment. **Public URL** with read-only access is here (https://www.kaggle.com/c/hse-ml-hw10-nov-1521-Crystals/rules). See competition rules, submission, grading, dataset, and performance metric. The **starter code** below produces a baseline model, which you should beat, while respecting the competition rules. Your code starts after the timer. This is your baseline model. Remember to seed RNG (https://en.wikipedia.org/wiki/Random_number_generation) in all experiments for reproducibility.

Instructions (to enable Kaggle API in Colab):

1. Accept competition rules before running Kaggle API (https://github.com/Kaggle/kaggle-api#api-credentials). Loading Kaggle dataset example (https://www.analyticsvidhya.com/blog/2021/06/how-to-load-kaggle-datasets-directly-into-google-colab)
2. In your Kaggle Account, Create API Token (https://github.com/Kaggle/kaggle-api#api-credentials) and save the resulting **kaggle.json** file to the root of your Google Drive (https://drive.google.com/drive/u/0/my-drive)
3. In Colab, open **Files** panel (on the left) and click gray folder icon to mount your Google drive

Your Kaggle/Google Drive credentials are secure; and Colab's kaggle.json only lasts a Colab session.

In [336]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [ ]:
```
!pip -q install --upgrade --force-reinstall --no-deps kaggle > log  # upgrade kaggle package
!mkdir -p ~/.kaggle                                                 # .kaggle folder must contain kaggle.json for kaggle executable to properly authenticate you to
!cp /content/drive/MyDrive/kaggle.json ~/.kaggle/kaggle.json  # First, download kaggle.json from kaggle.com (in Account page) and place it in the root of mou
!chmod 600 ~/.kaggle/kaggle.json                                    # give only the owner full read/write access to kaggle.json
!kaggle config set -n competition -v hse-ml-hw10-nov-1521-crystals  # set the competition context for the next few kaggle API calls. !kaggle config view - sh
!kaggle competitions download >> log                                # download competition dataset as a zip file
!unzip -o *.zip >> log                                              # unzip dataset
!kaggle competitions leaderboard --show                             # print public leaderboard
```

In [338]:
```
%%time
%%capture
%reset -f
!pip -q install scikit-learn-extra > log
from IPython.core.interactiveshell import InteractiveShell as IS; IS.ast_node_interactivity = "all"
from sklearn_extra.cluster import KMedoids
from sklearn.cluster import KMeans, DBSCAN
import numpy as np, pandas as pd, time, matplotlib.pyplot as plt, os

class Timer():
  def __init__(self, lim:'RunTimeLimit'=60): self.t0, self.lim, _ = time.time(), lim, print(f'⏳ started. You have {lim} sec. Good luck!')
  def ShowTime(self):
    msg = f'Runtime is {time.time()-self.t0:.0f} sec'
    print(f'\033[91m\033[1m' + msg + f' > {self.lim} sec limit!!!\033[0m' if (time.time()-self.t0-1) > self.lim else msg)

np.set_printoptions(linewidth=10000, precision=2, edgeitems=20, suppress=True)
pd.set_option('max_colwidth', 1000, 'max_columns', 100, 'display.width', 1000, 'max_rows', 4)
```

```
CPU times: user 168 ms, sys: 20.2 ms, total: 189 ms
Wall time: 3.18 s
```

`IsTest=1` are the observations of interest, but you can use others to create more reliable clusters. `Y` contains only 3 labels. You can deduce the fourth, `'orthorhombic'`.

In [339]:
```
dfRaw = pd.read_csv('./crystals_perovskites.csv')
dfRaw
```

Out[339]:

	IsTest	Y	v(A)	v(B)	r(A_XII)(Å)	r(A_VI)(Å)	r(B_VI)(Å)	EN(A)	EN(B)	l(A-O)(Å)	l(B-O)(Å)	ΔENR	tG	τ	μ
0	1	cubic	1	5	1.28	0.95	0.61	1.93	2.16	2.761453	1.789300	-1.663536	0.942809	4.126280	0.435714
1	1	rhombohedral	1	5	1.28	0.95	0.76	1.93	2.02	2.761453	2.215655	-1.992714	0.877336	4.072914	0.542857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6002	0	NaN	0	0	0.89	0.72	0.74	1.33	1.65	2.383420	2.096141	-2.035750	0.756670	NaN	0.528571
6003	0	NaN	0	0	0.89	0.72	0.72	1.33	1.33	2.383420	2.043778	-2.097821	0.763809	NaN	0.514286

6004 rows × 15 columns

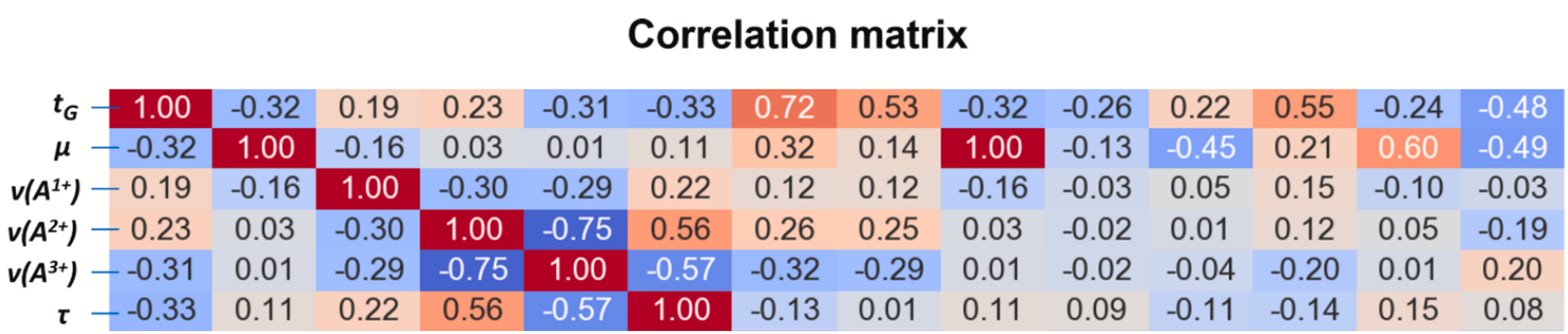
In [340]:
```
tmr = Timer() # runtime limit (in seconds). Add all of your code after the timer
```

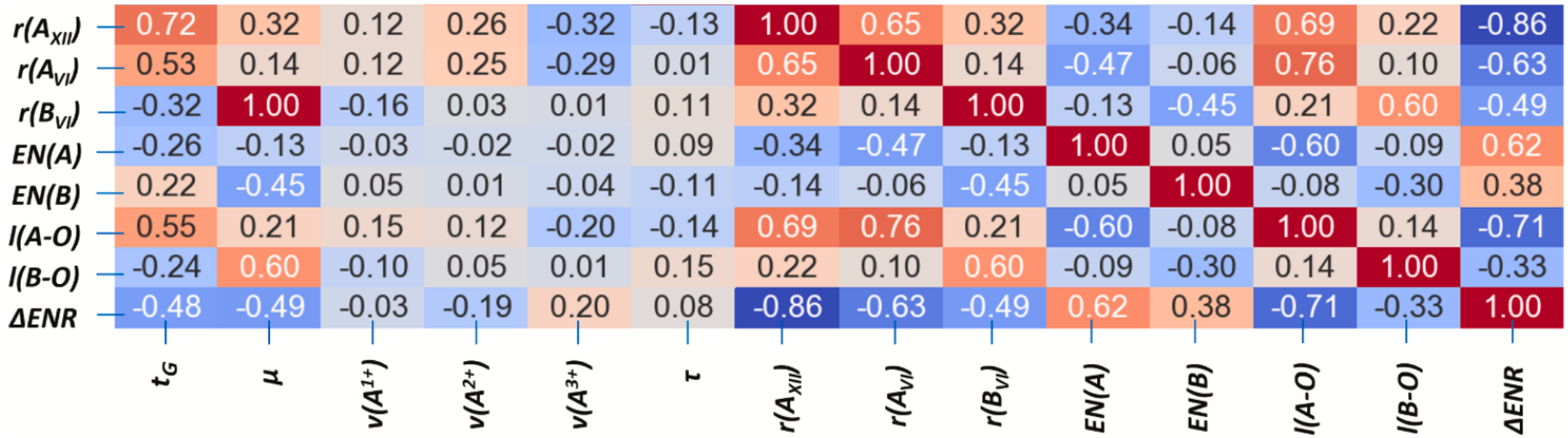
```
⏳ started. You have 60 sec. Good luck!
```

⏳ Your Code, Documentation, Ideas and Timer Start Here

According to the paper [Crystal structure classification in ABO3 perovskites via machine learning], the description of the given features:

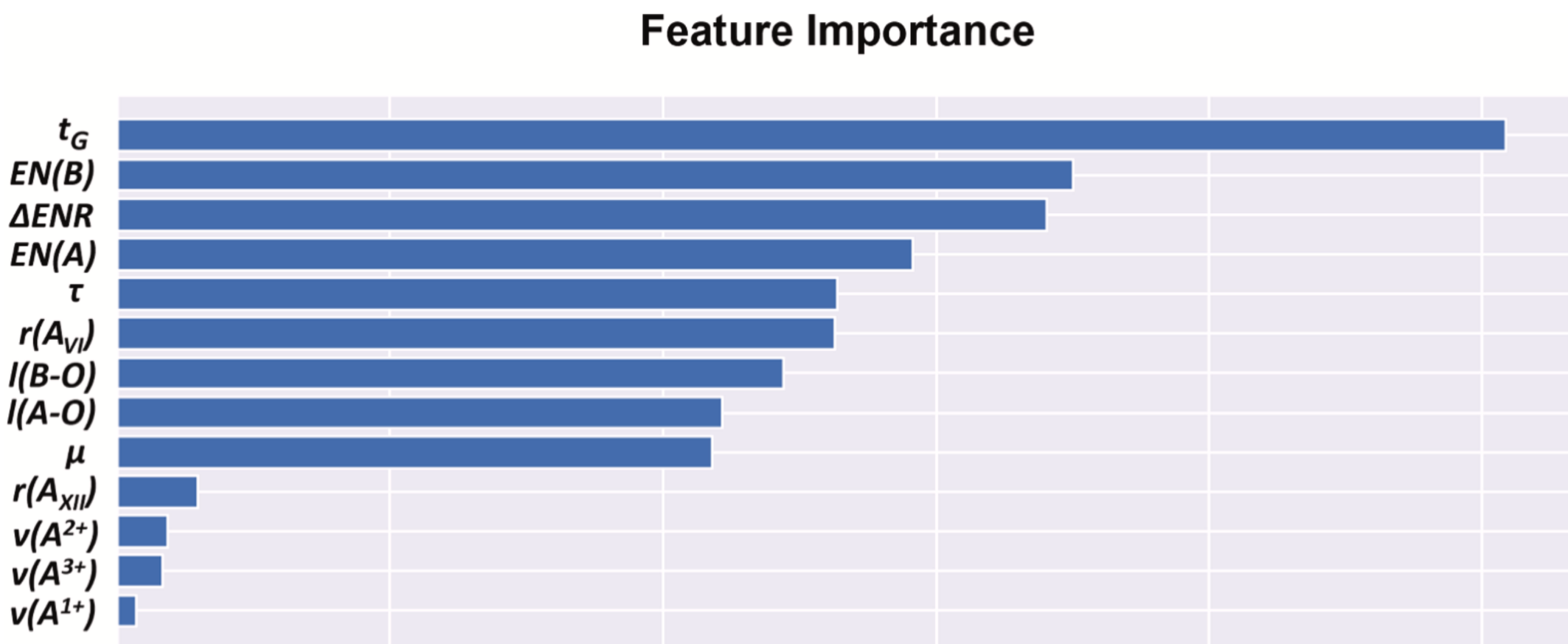
1. Valence A (v(A))
2. Valence B (v(B))
3. Radius A at 12 coordination (r(AXII))
4. Radius of A at 6 coordination (r(AVI))
5. Radius of B at 6 coordination (r(BVI))
6. Electronegativity of A (EN(A))
7. Electronegativity of B (EN(B))
8. Bond length of A-O pair (l(A-O))
9. Bond length of B-O pair (l(B-O))
10. Electronegativity difference with radius (ΔENR)
11. Goldschmidt tolerance factor (tG)
12. New tolerance factor (τ)
13. Octahedral factor (μ)





See [Santosh Behara's paper (https://www.sciencedirect.com/science/article/pii/S0927025620306820)](https://www.sciencedirect.com/science/article/pii/S0927025620306820) Fig.2.

Note that we can remove feature `r(B_VI)(Å)` since it is highly correlated with μ



In [341]:
```
df = dfRaw.query('IsTest==1').iloc[:,2:].fillna(0) # drop irrelevant features: IsTest and Y
```

In [342]:
```
print(f"The range of tG: {round(df['tG'].min(),5)} to {round(df['tG'].max(), 5)}")
```

```
The range of tG: 0.82001 to 1.0976
```

This plot illustrates the importance of each feature. The most important feature is `tG` , but the range of its value is not wide. We can apply exp-transformation on this feature.

In [343]:
```
df['log-tG'] = np.exp(df['tG'])
df.drop(['v(B)'], axis='columns', inplace=True)

df['log-tG'] = np.exp(df['tG'])
```

According to the Fig.7. [Santosh Behara's paper (https://www.sciencedirect.com/science/article/pii/S0927025620306820)](https://www.sciencedirect.com/science/article/pii/S0927025620306820) , we now know the SHAPLEY additive explanations for ABO_3 perovskites.

In [344]:
```
# Add up key factors of each structure.
df['rhom'] = df['ΔENR'] + df['l(A-O)(Å)']
df['orth'] = df['l(B-O)(Å)'] + (df['EN(A)']**(-1))
df['cubi'] = df['τ'] + df['EN(B)']**(-1)
df['tetr'] = df['r(A_VI)(Å)'] + df['EN(A)']
df
```

Out[344]:

	v(A)	r(A_XII)(Å)	r(A_VI)(Å)	r(B_VI)(Å)	EN(A)	EN(B)	l(A-O)(Å)	l(B-O)(Å)	ΔENR	tG	τ	μ	log-tG	rhom	orth	cubi	tetr
0	1	1.28	0.95	0.61	1.93	2.16	2.761453	1.789300	-1.663536	0.942809	4.126280	0.435714	2.567183	1.097918	2.307435	4.589243	2.88
1	1	1.28	0.95	0.76	1.93	2.02	2.761453	2.215655	-1.992714	0.877336	4.072914	0.542857	2.404486	0.768739	2.733790	4.567963	2.88
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
673	2	1.14	0.95	0.72	1.10	1.33	2.519261	2.043778	-2.345357	0.847194	4.835508	0.514286	2.333091	0.173904	2.952869	5.587388	2.05
674	2	0.90	0.74	0.58	1.65	1.63	2.375479	1.758039	-1.665143	0.821387	5.477252	0.414286	2.273650	0.710336	2.364100	6.090749	2.39

675 rows × 17 columns

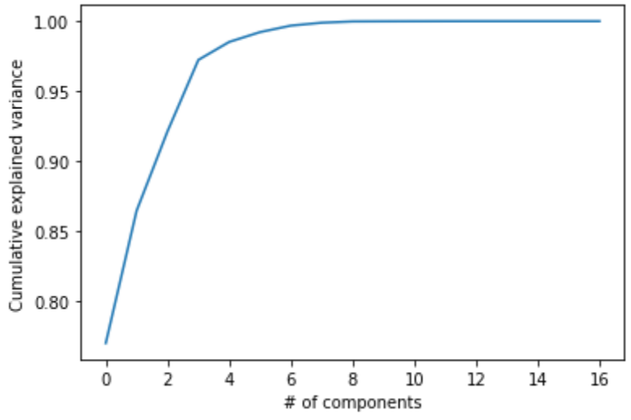
In [345]:
```
import warnings
warnings.filterwarnings('ignore')
from sklearn.decomposition import PCA
```

In [346]:
```
pca = PCA(17)
pca_full = pca.fit(np.array(df))
plt.plot(np.cumsum(pca_full.explained_variance_ratio_))
plt.xlabel('# of components')
plt.ylabel('Cumulative explained variance')
```

Out[346]: [<matplotlib.lines.Line2D at 0x7f4b1b73f4d0>]

Out[346]: Text(0.5, 0, '# of components')

Out[346]: Text(0, 0.5, 'Cumulative explained variance')



According to the elbow method, we can choose 3 as our n_components.

In [347]:
```
pca = PCA(n_components=3, whiten=False)
pca.fit(np.array(df))
pca_train_data = pd.DataFrame(pca.transform(np.array(df)))
```

Out[347]: PCA(n_components=3)

In [348]:
```
pca_train_data
```

Out[348]:

	0	1	2
0	-1.174042	-1.607286	0.304110
1	-1.156609	-1.048518	0.483392
...	...	...	...
673	-2.093628	0.617730	-0.010784
674	-2.835009	-0.423725	-0.906558

675 rows × 3 columns

In [349]:
```
from sklearn.cluster import SpectralClustering
from sklearn import metrics
```

In [350]:
```
K = 4
Cols = pca_train_data.columns
for index, gamma in enumerate((0.01,0.1,0.5, 1, 3, 5)):
    y_pred = SpectralClustering(n_clusters=K, affinity='rbf', gamma=gamma, random_state=122).fit_predict(pca_train_data[Cols])
```

```
    print("Silhouette Score with gamma=", gamma, "n_clusters=", K,"score:", metrics.silhouette_score(pca_train_data[Cols], y_pred))
```

```
Silhouette Score with gamma= 0.01 n_clusters= 4 score: 0.3797788527097577
Silhouette Score with gamma= 0.1 n_clusters= 4 score: 0.4935869880291518
Silhouette Score with gamma= 0.5 n_clusters= 4 score: 0.56541112853447
Silhouette Score with gamma= 1 n_clusters= 4 score: 0.5499678021699392
Silhouette Score with gamma= 3 n_clusters= 4 score: 0.5499678021699392
Silhouette Score with gamma= 5 n_clusters= 4 score: 0.3789760986942848
```

In [351]:

```
K = 4
spectral = SpectralClustering(n_clusters=K, affinity='rbf', gamma=0.5, random_state=122)
spectral_predict = spectral.fit_predict(pca_train_data)
pca_train_data['Cl'] = spectral.labels_
print("Silhouette Score", metrics.silhouette_score(pca_train_data, spectral_predict))
```

```
Silhouette Score 0.6115375732918016
```

In [352]:

```
(unique, counts) = np.unique(spectral.labels_, return_counts=True)
frequencies = np.asarray((unique, counts)).T
print(f"Frequencies: \n{frequencies}")
```

```
Frequencies:
[[  0 650]
 [  1   7]
 [  2   9]
 [  3   9]]
```

Type *Markdown* and LaTeX: α^2

In [353]:

```
df.loc[pca_train_data.Cl==0, 'structure'] = 'orthorhombic'
df.loc[pca_train_data.Cl==1, 'structure'] = 'tetragonal'
df.loc[pca_train_data.Cl==2, 'structure'] = 'rhombohedral'
df.loc[pca_train_data.Cl==3, 'structure'] = 'cubic'
```

The dataset contains 675 data points out of which 19 are tetragonal, 261 are cubic, 362 are orthorhombic, and 33 are rhombohedral. The dataset is highly imbal- anced as 53.63% of the data in the dataset is orthorhombic. --- PAPER

In [354]:

```
# df.loc[df['structure'].isnull(), 'structure'] = 'orthorhombic'  # rename observations in the remaining cluster
df['structure'].value_counts()   # show distribution of labels
df['structure'].to_csv('crystals_goes2moon.csv', index_label='id')
```

Out[354]:

```
orthorhombic    650
rhombohedral      9
cubic             9
tetragonal        7
Name: structure, dtype: int64
```

⏳Do not exceed competition's runtime limit!

In [355]:

```
tmr.ShowTime()    # measure Colab's runtime. Do not remove. Keep as the last cell in your notebook.
```

```
Runtime is 2 sec
```

💡Starter Ideas

1. Try different hyperparameters and models
2. Try engineering new features, scaling, interactions, ...
   A. Look for linear and non-linear relations between features and target
3. Learn about perovskite (https://en.wikipedia.org/wiki/Perovskite) crystals to build new features based on exact equations (https://www.youtube.com/results?search_query=perovskite+crystals). See Santosh Behara's paper (https://www.sciencedirect.com/science/article/pii/S0927025620306820) for their feature engineering; they used supervised ML. The paper is saved to our shared Google Drive folder. Paper references can also help.
4. You don't have to use the provided labels.
5. Silhoutte score (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html) (and other metrics) can help identify stable clusters.