09.00 — UNSUPERVISED LEARNING —

- SET OF STATISTICAL TOOLS FOR SCENARIO WHERE THERE ARE ONLY FEATURES AND NO TARGET.
- WE CANNOT MAKE PREDICTIONS.
- FIND INTERESTING WAY TO VISUALIZE DATA OR GROUP SIMILAR OBSERVATIONS.
- MORE CHALLENGING BECAUSE THE ANALYSIS IS SUBJECTIVE
- HARD TO ASSESS, SINCE WE DON'T KNOW THE TRUE ANSWER.
- PRINCIPLE COMPONENT ANALYSIS (PCA)
- CLUSTERING

PRINCIPLE COMPONENT ANALYSIS (PCA)

- PROCESS BY WHICH PRINCIPLE COMPONENTS ARE COMPUTED AND USED TO BETTER UNDERSTAND DATA.
- CAN BE USED FOR VISUALIZATION.
- VISUALIZE 'n' OBSERVATIONS WITH MEASUREMENT ON SET OF p FEATURES.
- CAN EXAMPLE 2D PLOTS OF 2 FEATURES AT A TIME. BUT THAT'S NOT EFFICIENT (ESPECIALLY IF 'p' IS VERY LARGE)

WITH PCA —

- FIND LOW DIMENSIONAL REPRESENTATION OF DATASET THAT CONTAINS AS MUCH AS POSSIBLE OF THE VARIANCE.
- ONLY CONSIDER THE MOST 'INTERESTING' FEATURES, BECAUSE THEY ACCOUNT FOR THE MAJORITY OF THE VARIANCE.

24 '22 June

Week 26 Day (175-190) | Friday

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00

A PRINCIPLE COMPONENT IS NORMALIZED LINEAR COMBINATION

10.00 OF THAT FEATURES THAT HAVE LARGEST VARIANCE.

11.00

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \_\_\_$$

12.00

$$\_\_\_ + \phi_{p1} X_p$$

13.00

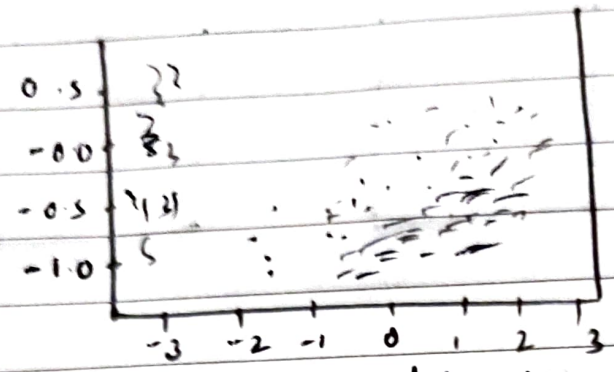
14.00

PCA In Python

15.00

```
Iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)
print(pca.explained_variance_ratio)
```

09.00 PCA In Python —



10.00 PCA of Iris dataset

11.00

12.00

13.00

CLUSTERING METHODS

14.00 - SET OF TECHNIQUE FOR FINDING SUBGROUPS OR CLUSTERS IN DATASET.

- PARTITION DATA INTO OBSERVATIONS THAT ARE SIMILAR TO EACH OTHER.

15.00 - EX- MARKET SEGMENTATION IN CONTEXT OF MARKETING.

- K-MEANS CLUSTERING - PARTITION DATA INTO SPECIFIED NUMBER

16.00 OF K CLUSTERS

- HIERARCHIAL CLUSTERING - UNSPECIFIED NUMBER OF CLUSTERS.

17.00 GET DENDROGRAM TO VIEW ALL CLUSTERS OBTAINED FOR Sunday 26

EACH POSSIBLE NUMBER OF CLUSTERS.

18.00

K-MEANS CLUSTERING -

- SEPERATE THE OBSERVATIONS INTO K CLUSTERS. IT ASSUME THAT

→ EACH OBSERVATION BELONGS TO ATLEAST ONE OF K CLUSTERS

→ THE CLUSTER DO NOT OVERLAP

- VARIATION IS MINIMIZED WITHIN A CLUSTER.

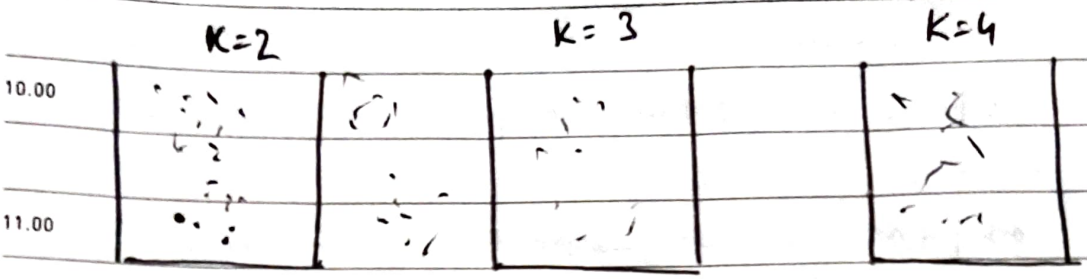
27 '22 June

Week 27 Day (178-187) | Monday

JUNE 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00 K-MEANS CLUSTERING —



10.00

11.00

12.00

- MINIMIZING THE SUM OF SQUARED EUCLIDEAN DISTANCE BETWEEN

13.00 EACH OBSERVATION WITHIN A CLUSTER.

14.00

$$\min\left\{\sum_{K=1}^{K}\frac{1}{|C_K|}\sum_{i,i' \in C_K}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2\right\}$$

15.00

16.00

17.00 To minimize —

- Randomly assign a number from 1 to K, to each observation

18.00 (initial cluster assignment).

- For each cluster, compute the centroid; a vector representing the mean of the features for the observations in the cluster.
- Assign each observation to cluster whose centroid is the closest (shortest Euclean distance)
- Iterate until cluster assignment stops changing.

09.00 IMPORTANT NOTE -

- ALGORITHM FINDS LOCAL MINIMUM

10.00 - HIGHLY DEPENDS ON THE INITIAL RANDOM CLUSTER ASSIGNMENT.

- MOST RUN ALGORITHM MULTIPLE TIMES.

11.00

- K-Means clustering in Python -

12.00

13.00

```
n_colors = 64
kmeans = kMeans(n_clusters = n_colors,
random_state=42).fit(image_sample)
labels = kmeans.predict(image_array)
```

14.00

15.00

16.00

17.00

18.00

29 June

Week 27 Day (180-185) | Wednesday

MON	TUE	WED	THU	FRI	SAT	SUN						
		1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24	25	26
27	28	29	30										

09.00 HIERARCHICAL CLUSTERING —

- K-Means - require a human input to specify the number of clusters.

10.00 Hierarchical clustering - no initial number of clusters required.

11.00 AGGLOMERATIVE CLUSTERING - COMMON TYPE OF HIERARCHICAL CLUSTERING.

12.00 DENDOGRAM GENERATED FROM LEAVES AND CLUSTERS, ARE COMBINED UP TO THE TRUNK.

13.00

UNDERSTANDING THE ALGORITHM -

14.00 DEFINE A DISSIMILARITY MEASURE BETWEEN EACH PAIR OF OBSERVATION AND ASSUME THAT EACH OBSERVATION PERTAINS

15.00 TO ITS OWN CLUSTER.

- THE 2 MOST SIMILAR CLUSTERS ARE FUSED - n-1 CLUSTERS.

16.00 OTHER 2 SIMILAR CLUSTERS ARE FUSED - n-2 CLUSTERS.

- REPEATED UNTILL ALL OBSERVATIONS ARE PART OF 1 CLUSTER.

17.00

- DISSIMILARITY DEPEND ON TYPE OF LINKAGE -

18.00 4 TYPE OF LINKAGE

- COMPLETE
- SINGLE
- AVERAGE
- CENTROID

09.00

1) COMPLETE - MAXIMAL INTERCLUSTER DISSIMILARITY

10.00 COMPOTE ALL PAIRWISE DISSIMILARITIES IN CLUSTER A and B. RECORD THE LARGEST OF THESE DISSIMILARITIES.

11.00

2) SINGLE - MINIMAL INTERCLUSTER DISSIMILARITY

12.00 COMPOTE ALL PAIRWISE DISSIMILARITIES IN CLUSTER A and B. RECORD THE SMALLEST OF THESE DISSIMILARITIES.

13.00 SINGLE OBSERVATIONS ARE FUSED ONE AT A TIME.

14.00 3) AVERAGE - MEAN INTERCLUSTER DISSIMILARITY

COMPUTE ALL PAIRWISE DISSIMILARITIES IN CLUSTER A and B

15.00 RECORD THE AVERAGE OF THESE DISSIMILARITIES.

16.00 4) CENTROID - DISSIMILARITY BETWEEN THE CLUSTER A and B

SMALLER CLUSTERS ARE NO MORE SIMILAR TO THE NEW

17.00 LARGER CLUSTER THAN TO THEIR INDIVIDUAL CLUSTERS.

CAN RESULT IN INVERSIONS.

18.00

FINAL DENDOGRAM DEPENDS ON TYPE OF LINKAGE.

UNSUPERVISED LEARNING #CodeDaniel

```
import numpy as np
import matplotlib.pyplot as
plt
from sklearn.utils import
shuffle.
```

Numpy is powerful library for numerical computing in Python used for working with arrays and matrices.

Matplotlib is powerful library for visualization options.

Shuffle function from ScikitLearn is utility function for randomly shuffling rows of dataset.

```
from sklearn.datasets import
load_sample_image
from sklearn.cluster
import KMeans
```

Code import 'load_sample_image' function from 'datasets' module in Scikitlearn library, KMeans class from the 'cluster' module in same library.
'load_sample_image' is utility function that loads sample image from Scikit Learn.

AUGUST 2022

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

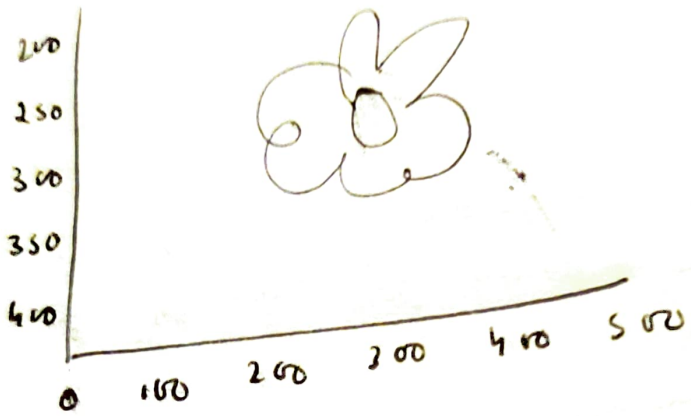
July '22

Saturday 02

09.00
```
flower = load_sample_image('flower.jpg')
flower = np.array(flower, dtype=np.float64)/255
plt.imshow(flower);
```

10.00

11.00



12.00

13.00

14.00

15.00

This code loads sample image called 'flower.jpg' using the 'load_sample_image' function and convert it to a Numpy array of type float64. The pixel value are also normalized to the range [0,1] by dividing 255.

16.00

17.00

Sunday 03

18.00

Finally, image is displayed using the 'imshow' function from the matplotlib library.

04 '22 July

Monday

Week 28 Day (185-180)

JULY 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
				1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31							

09.00

10.00

11.00

12.00

```
w, h, d = original_shape
= tuple (flower.shape)
assert d == 3
image_array = np.reshape
(flower, (w * h, d))
```

The Code extracts original dimension of loaded image (number of rows, colums, color channels) and stores them in variable `original_shape'

13.00

14.00 IT THEN CHECKS THAT THE IMAGE HAS 3 COLORS CHANNELS, WHICH IS EXPECTED, FOR A TYPICAL RGB IMAGE.

15.00 FINALLY, IT RESHAPES 3D NUMPY_ARRAY 'FLOWER' INTO 2D NUMPY ARRAY. WITH SHAPE (w*h, d)

16.00 w - width, h - height, d - no of colors channels

THE RESHAPED ARRAY 'IMAGE_ARRAY' CONTAINS ALL

17.00 PIXELS VALUE OF ORIGINAL IMAGE IN LONG, 2D FORMAT. WHERE EACH ROW REPRESENT SINGLE PIXEL AND COLUMNS

18.00 REPRESENT R, G, B COLORS.

```
image_sample = shuffle(image_array,
random_state=42)[:1000]
# fit kMeans
n_colors = 4
kmeans = KMeans(n_clusters=n_colors,
  random_state=42).fit(image_sample)
# Get color indices for full image
labels = kmeans.predict(image_array)
```

This code randomly selects 1000 pixels from the reshaped image array `image_array' using shuffle function from scikit learn library. Random state is set to 42 to ensure reproducibility.

It creates a KMeans clustering object with n_colors clusters (64 in the case), fit it to randomly 1000 selected pixels using fit. method of KMeans object.

The code is part of common image processing pipeline that uses KMeans clustering to reduce number of colors in image.

06 '22 July
Wednesday
Week 28 Day (187-178)

JULY 2022

MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT	SUN
				1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31							

09.00

```
def reconstruct_image (clustey_centers, labels, w, h):
    d= clusley_center. shape [1]
    image = np. zeros ((w, h, d))
    label_index = 0
    for i in range (w):
        for j in range (h):
            image [i][j] = clusley_centers [labels [label_index]]
            label_index += 1
    return image
```

→ This code defines a function called 'reconstruct_image' that take 'clusley_centers', 'labels', 'w', 'h'.

```
plt. figure (1)
plt. clf()
plt. axis ('off')
plt. title ('Original image with 96 615 colors')
plt. imshow (flower);

plt. figure (2)
plt. clf ()
plt. axis ('off')
plt. title (f "Reconstructed in {n_colors}')
plt. imshow (reconstruct_image (
    Kmeans. clusley_centers_, labels,
    w, h ));
```

PRINCIPLE COMPONENT ANALYSIS (PCA) FOR DIMENTIONALITY REDUCTION

```
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
pca = PCA(n_components = 2)
X_r = pca.fit(X).transform(X)
print('Explained variance ratio from PCA'
.format(pca.explained_variance_ratio_))
colors = ['#003f5c', '#bc5090', '#ffa600']
lw = 2
```

→ EACH LINE OF CODE EXPLAINED IN SOURCE FILE.

```
plt.figure()
for color, i, target_name in zip(colors,
[0, 1, 2], target_names):
plt.scatter(X_r[y==i, 0], X_r[y==i, 1],
color=color, alpha = 0.8, lw = lw,
label = target_name)
plt.legend(loc='best', shadow = false,
scatterpoints = 1)
plt.title('PCA of Iris dataset')
```