Unsupervised Learning: Word embedding(2)

Parcours Progis

Etudes, Medias, communication, Marketing

Bahareh Afshinpour

02.06.2025

Université Grenoble Alpes

Sciences Po Grenoble UGA

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Réduction dimensionnelle

La réduction de la dimension est un processus qui consiste à prendre des données dans un espace de grande dimension et à les remplacer par des données dans un espace de plus petite dimension

Dimension Reduction identifies the combination of attributes (principal components, or directions in the feature space) that account for the most variance in the data.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Réduction dimensionnelle

- We can say some of the features tell us the same information as other features.
- In this case we do not need all features. Maybe we need a little bit of one feature and a lit a bit of anothers.
- Or we can say that is a combination of this features that captures the informations.
- Linear methods:
  - Principal Component Analysis (PCA)
  - Singualr Value Decomposition (SVD)
- Non-Linear Methods
  - T-SNE

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

PCA (Principal Component Analysis)

- With PCA we figure out, if we have a bunch of features, how many component in the space do we need to explain the data.
- We are not deciding which feature to keep and drop, but we are **creating** new features that it is kind of a little bit of multiple features.

- Covariance metrix : when one variable changes how much does the other variable change
- Eigenvalue decomposition: what direction is the change

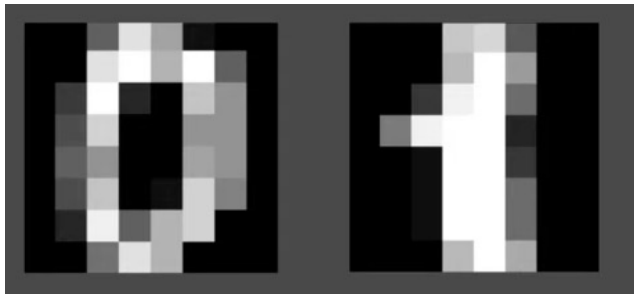
- The principal component are those with the largest eigenvalues

PCA is a process of figuring out the most important features, or principal components, that have the most impact on the target variable.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

PCA

- Think that each pixel is one feature.
- Some of the pixels do not play any role at all in finding what digit it is.



- For example, these pixels, no matter what number we have, they are always black.

https://www.youtube.com/watch?v=8klqIM9UvAc

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

PCA

What if we eliminate features that aren't essential?

- Faster training and inference
- Data visualization gets simpler.

Visualization helps us. Regarding the final decision-making process.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

PCA

Before using PCA, there are a few things to consider.

- Scale features before applying PCA.
- Accuracy might drop (imagine 100D to 2D; we lose lots of information).
- We should split our data if we have a label column.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

PCA in Python

```python
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(sessions_embedding)

y=[0] * 61
# Plot the result
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y,cmap='viridis', edgecolor='k')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - 2D Projection')
plt.legend(*scatter.legend_elements(), title="Classes")
plt.grid(True)
plt.show()
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Range() function

- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Loop

```
for i in range(6):
  print(i)
```

→ {0,1,2,3,4,5}

```
0
1
2
3
4
5
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Len() function

- The len() function in Python is used to **get the number of items:**

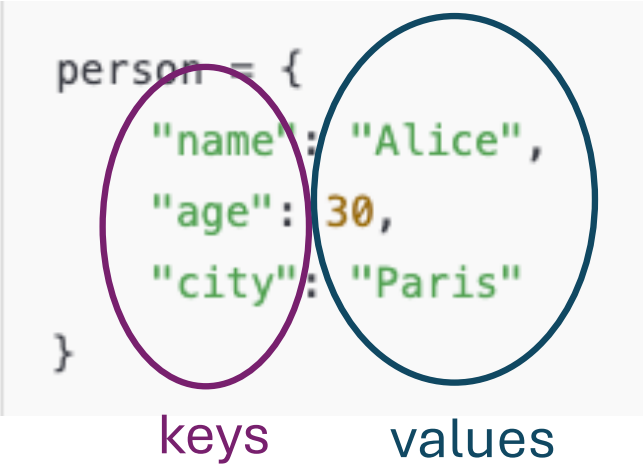
```
text = "hello"
print(len(text))  # Output: 5
```

```
numbers = [1, 2, 3, 4]
print(len(numbers))  # Output: 4
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Dictionary

- is a built-in data structure that **stores data in key-value pairs**.



Here, "**person**" refers to the name of the dictionary.

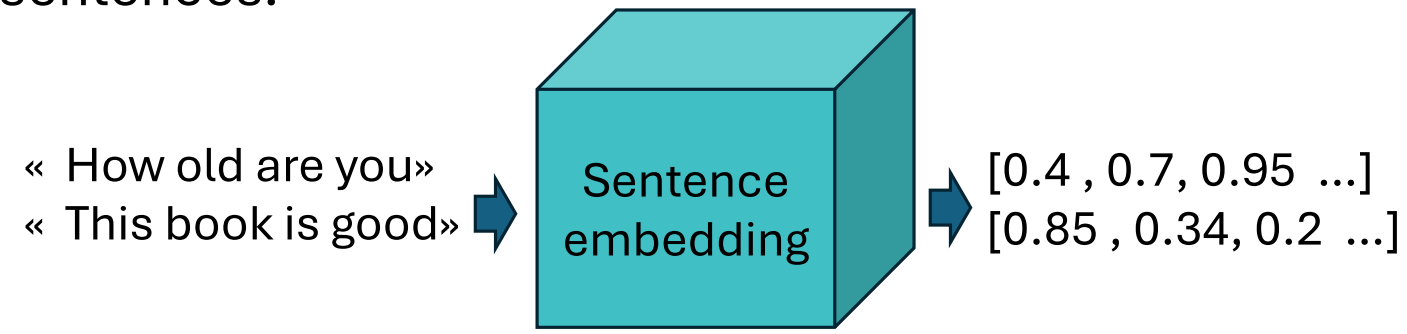
Scanner


Scan as you shop
1ST CAPE SHIRZ 5.50
BANANAS 5 PACK 1.35
LARGE STIR FRY 1.70
2 SOUP 2.30
2 CARROT SOUP 2.30
2 SALMON 4.60
3 W 400G W/M MED 2.55
MUESLI
Items 14 26.30
Savings 0.00
Item
Remove

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Sentence embeddings

- The idea behind sentence embeddings is to convert sentences into numerical vectors.
- These numerical vectors represent the sentences in a structured and machine-friendly format.
- The beauty of this vector representation lies in its ability to capture semantic relationships and contextual similarities between sentences.



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Scanner case study

- Logs
  - Are records of events that occurred during the running of a software system.
  - Are generated by log statements in software source code.
  - Logs are a primary source for problem diagnosis.
  - Event : Units of information in a log are often called events.

Index	Time	Session ID	Object	Action	Input	Output
51,	1585070116817,	client6,	scan12,	unlock,	[],	0
52,	1585070116819,	client0,	scan0,	scan,	[3270190022534],	0
53,	1585070116820,	client1,	cashier1,	CloseSession,	[],	0
54,	1585070116820,	client2,	cashier2,	add,	[3570590109324],	0
55,	1585070116824,	client5,	scan5,	scan,	[8718309259938],	0
56,	1585070116825,	client6,	scan12,	scan,	[3560070139675],	0
57,	1585070116837,	client0,	scan0,	scan,	[3560070048786],	0
58,	1585070117030,	client6,	scan12,	scan,	[7640164630021],	-2
59,	1585070117073,	client6,	scan12,	delete,	[7640164630021],	-2
60,	1585070116838,	client1,	cashier1,	pay,	[353.06],	0

event

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Scanner case study

Client6 **session** from `Unlock` to `Pay`

Index	Time	Session ID	Object	Action	Input	Output
51,	1585070116817,	client6,	scan12,	unlock,	[],	0
52,	1585070116819,	client0,	scan0,	scan,	[3270190022534],	0
53,	1585070116820,	client1,	cashier1,	CloseSession,	[],	0
54,	1585070116820,	client2,	cashier2,	add,	[3570590109324],	0
55,	1585070116824,	client5,	scan5,	scan,	[8718309259938],	0
56,	1585070116825,	client6,	scan12,	scan,	[3560070139675],	0
57,	1585070116837,	client0,	scan0,	scan,	[3560070048786],	0
58,	1585070117030,	client6,	scan12,	scan,	[7640164630021],	-2
59,	1585070117073,	client6,	scan12,	delete,	[7640164630021],	-2
60,	1585070116838,	client1,	cashier1,	pay,	[353.06],	0
61,	1585070116839,	client2,	cashier2,	CloseSession,	[],	0
62,	1585070116840,	client3,	cashier3,	add,	[3570590109324],	0
64,	1585070117687,	client6,	scan12,	transmission,	[caisse6],	0
65,	1585070117687,	client6,	scan12,	abandon,	[],	?
66,	1585070117701,	client6,	cashier4,	OpenSession,	[],	0
67,	1585070116855,	client0,	scan0,	transmission,	[cashier0],	0
68,	1585070116855,	client0,	scan0,	abandon,	[],	?
69,	1585070117716,	client6,	cashier4,	add,	[7640164630021],	0
70,	1585070117731,	client6,	cashier4,	CloseSession,	[],	0
71,	1585070117747,	client6,	cashier4,	Pay,	[260],	9.11

We treat each client as a sentence (session)

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

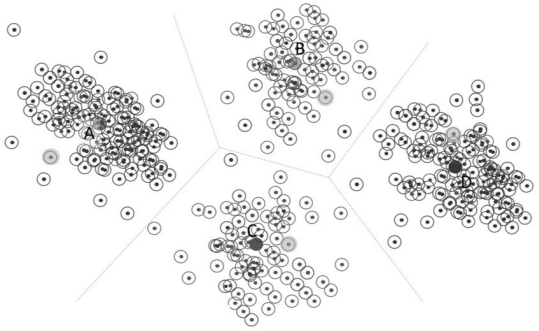
The propose model

Log partitioning And abstraction

Event embedding (word embedding)

Session embedding

K-means clustering



Each point is a vec tor and represents one client.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Log partitioning based on ClienTID

1026-clients-U.csv → Log abstraction → Sessions-1026.npy

Log partitioning And abstraction

	1	1584454655792	client0	scan0_0	debloquer	[]	0
0	2	1584454655801	client0	scan0_0	scanner	[8718309259938]	0
1	3	1584454656089	client1	scan1_1	debloquer	[]	0
2	4	1584454656095	client0	scan0_0	scanner	[3560070976478]	0
3	5	1584454656105	client1	scan1_1	scanner	[3560070048786]	0
4	6	1584454656127	client2	scan2_2	debloquer	[]	0

client0': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-float', '0']],
' **client1**': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-integer', 'Float Number']],
' **client2**': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing', '0'], ['ajouter', 'Barcode', '0'], ['fermerSession', 'Nothing', '0'], ['payer', 'Price-float', '0']],

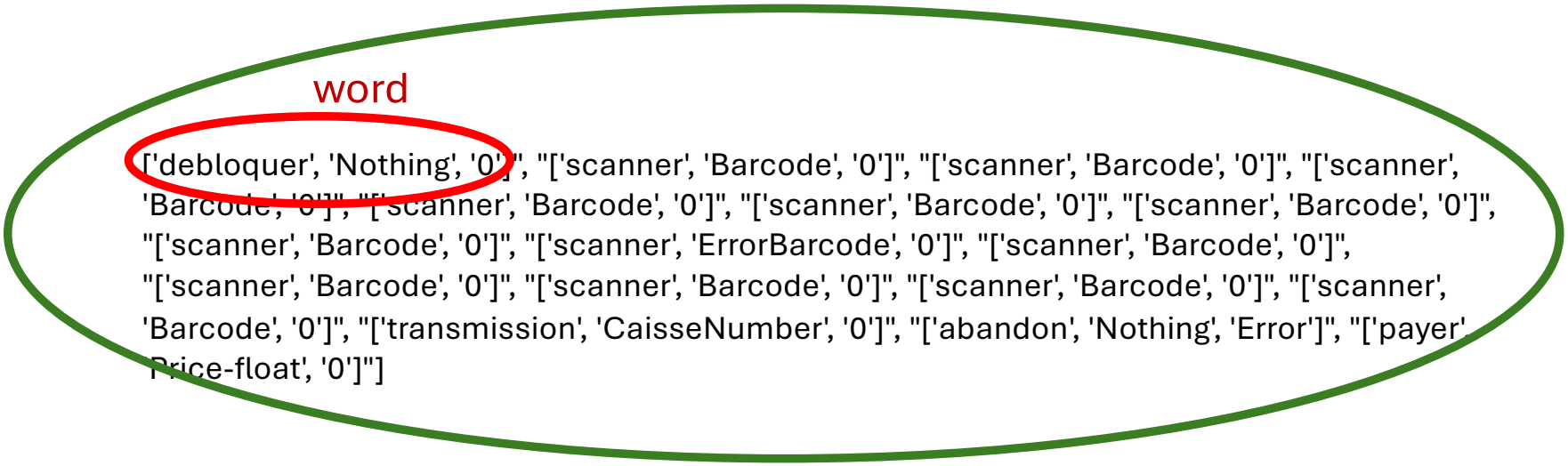
PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Word2Vec on all the Sessions

Event embedding
(word embedding)

The Word2Vec package from the Gensim libray

The dimension of the W2Vvectors are equal to the number of vocabs



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Session embedding

Averaging Word Embeddings (Averaging a session)

- **client60:**

['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['abandon', 'Nothing', 'Error']", "['payer', 'Price-float', '0']]

([0.15698704 -0.79322034 -0.3945347 -0.82248145 0.21587323 -0.4298337 0.11635129 0.06533043 -0.01920184 0.09027459 -0.78751177 -0.46802115 -0.5892027 -0.9514586 -0.21800822] + [1.13778 -2.4118261 -1.158108 -2.5120077 0.8757284 -0.8175933 0.31543404 0.43288264 0.16992638 -0.16874686 -2.5441096 -1.5867229 -2.180273 -1.9017196 -0.28566715] + ...) = [7.398182 -13.992377 -6.3737016 -14.275244 5.5125384 -3.9403927 1.6540279 2.750914 1.7304657 -1.40134 -14.432299 -9.090877 -12.594746 -9.036384 -1.28486] / 7 = **[0.7877907 -0.47995558 -0.06433037 -1.0645988 -0.9146954 0.08257212 -0.16243137 0.6493701 -1.9037664 0.31899315 1.2860477 1.3689787 0.51519257 -0.04324638 -0.37049222]**

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Session embedding

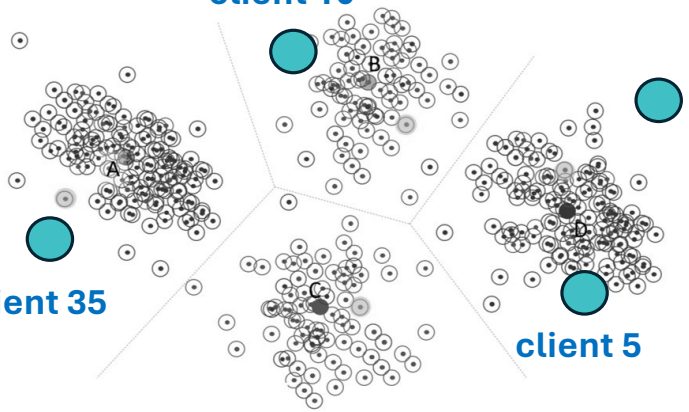
We are going to cluster these 61 clients

[1.0956031 -0.26459935 -0.99176484 -1.8269157 - 0.2858453 0.22351813 0.62547946 -0.40456355 -0.01479345 0.02950979 - 1.4409974 0.97218895 -0.90637374 1.5324388 0.19554672]

client 10

[1.0375887 -0.24739444 - 0.92388165 -1.7032957 - 0.2800482 0.19364282 0.5891389 -0.38360417 - 0.02879436 0.03572838 - 1.3417056 0.9103135 -0.83895904 1.4372357 0.19943428]

client 60



client 35

client 5

[1.1439486 -0.2789368 -1.0483342 -1.9299325 - 0.29067624 0.24841422 0.6557632 -0.4220297 - 0.00312602 0.02432763 - 1.5237406 1.0237519 -0.9625526 1.6117748 0.19230707]

[1.1439486 -0.2789368 -1.0483342 -1.9299325 - 0.29067624 0.24841422 0.6557632 -0.4220297 - 0.00312602 0.02432763 - 1.5237406 1.0237519 -0.9625526 1.6117748 0.19230707]

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

What is the dimension of our vectors?

```
print(len(sessions_embedding[0]))  #client0
```

```
15
```

```
print(len(sessions_embedding[1]))  #client1
```

```
15
```

All vectors have dimension 15. Since the dimension is equal to the dimension of each word(W2V vector).

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

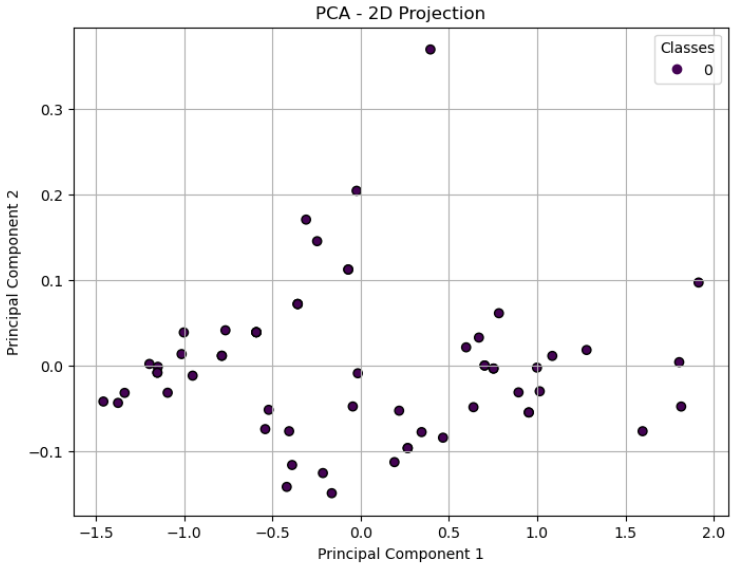
Visualisation de l'espace vectoriel (PCA)

15D → 2D

```python
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(sessions_embedding)

y=[0] * 61
# Plot the result
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y,cmap='viridis', edgecolor='k')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - 2D Projection')
plt.legend(*scatter.legend_elements(), title="Classes")
plt.grid(True)
plt.show()
```

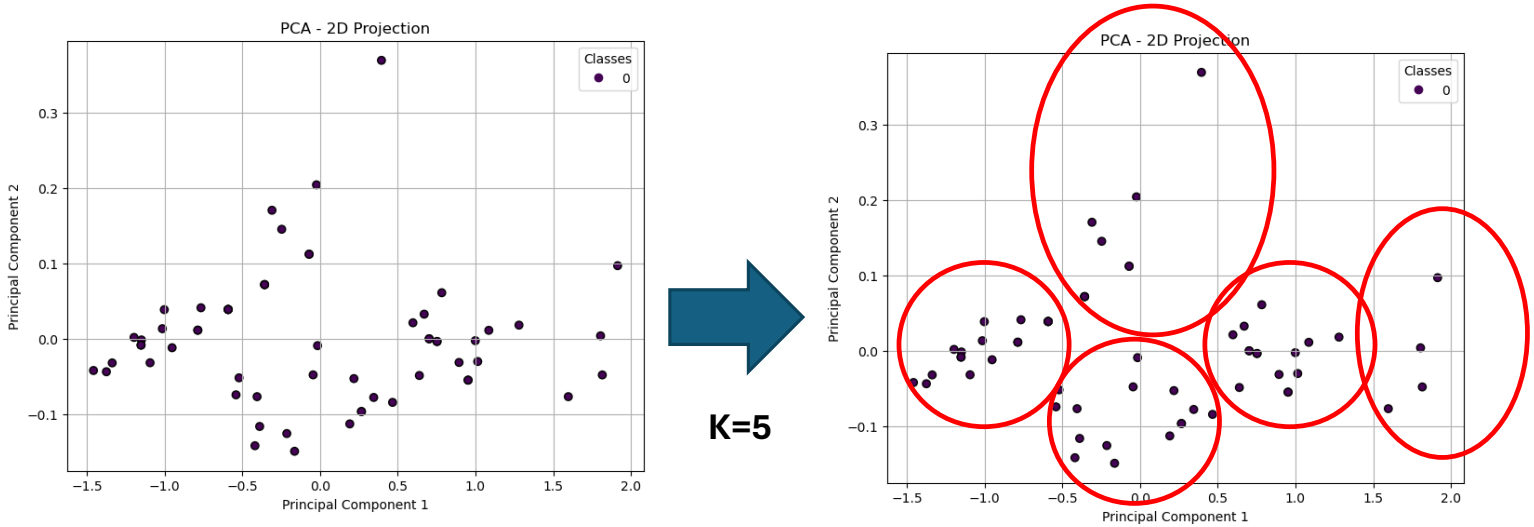


PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Clustering

K-means clustering

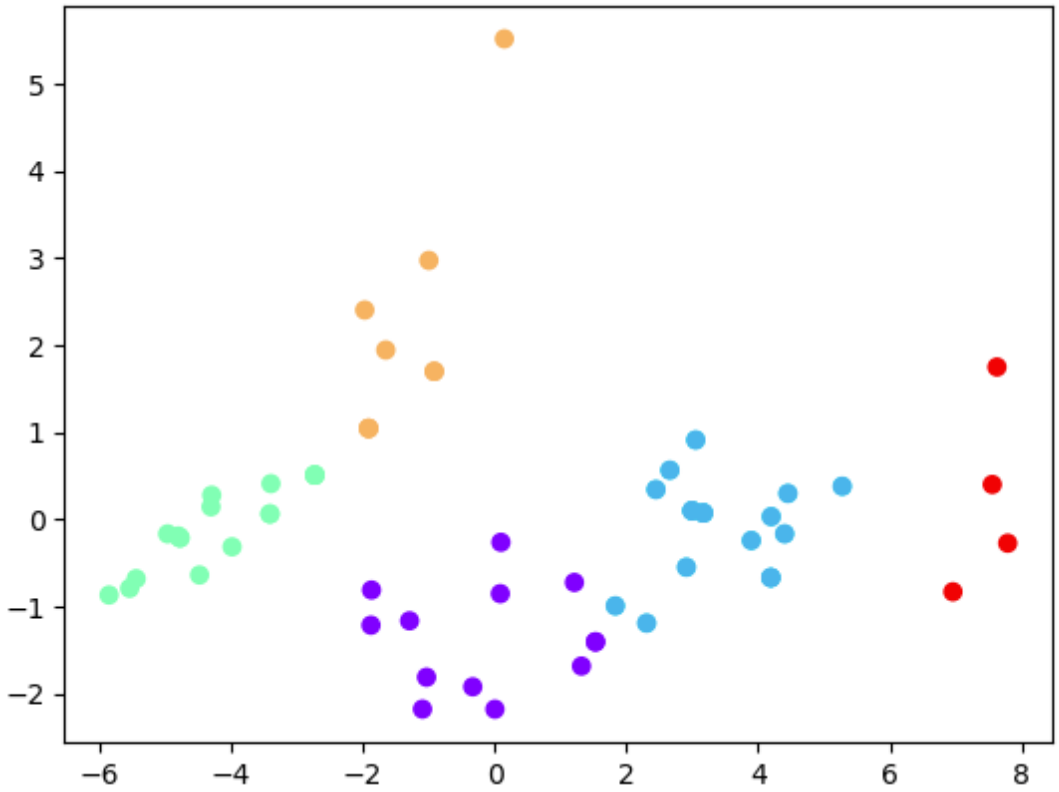
- K-Means & Elbow method
  - K-means needs to know number of cluster K
  - Elbow method finds the optimal K



1026-event

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

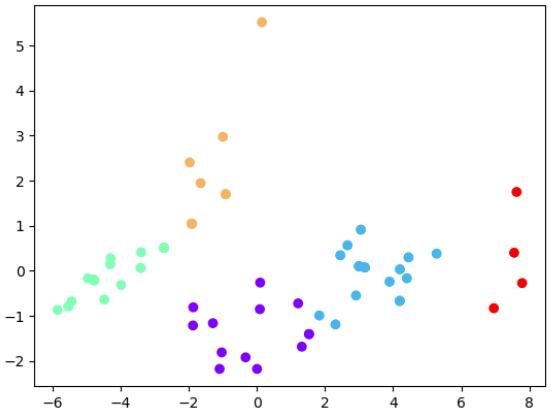
Clustering



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Clustering

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5)
kmeans.fit(X_pca)
plt.scatter(X_pca[:,0],X_pca[:,1], c=kmeans.labels_, cmap='rainbow')
plt.show()
```



TP

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

TP

```
import pandas as pd
data='./1026-clients-U.csv'
df=pd.read_csv(data)
df.head()
```

	1	1584454655792	client0	scan0_0	debloquer	[]	0
0	2	1584454655801	client0	scan0_0	scanner	[8718309259938]	0
1	3	1584454656089	client1	scan1_1	debloquer	[]	0
2	4	1584454656095	client0	scan0_0	scanner	[3560070976478]	0
3	5	1584454656105	client1	scan1_1	scanner	[3560070048786]	0
4	6	1584454656127	client2	scan2_2	debloquer	[]	0

```python
import numpy as np
Sessions = np.load('Sessions-1026.npy',allow_pickle='TRUE').item()
print(Sessions)
```

```
{' client0': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barc
code', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Ba
arcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abando
' client1': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barco
ode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing', 'Error'], ['payer', 'Price-integer', 'Float Nu
anner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner', 'Barcode', '0'], ['scann
r', '0'], ['abandon', 'Nothing', 'Error'], ['ouvrirSession', 'Nothing', '0'], ['ajouter', 'Barcode', '0'], ['fermerSessi
lient3': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode'
e', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['abandon', '
ber']], ' client4': [['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner'
r', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['supprimer', 'Barcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner
r', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['scanner',
'Barcode', '0'], ['supprimer', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['transmission', 'CaisseNumber', '0'], ['ab
'0'], ['ajouter', 'Barcode', '0'], ['fermerSession', 'Nothing', '0'], ['payer', 'Price-float', '0']], ' client5': [['deb
['scanner', 'Barcode', '0'], ['scanner', 'ErrorBarcode', '0'], ['scanner', 'Barcode', '-2'], ['scanner', 'Barcode', '0']
```

Here, Sessions is a dictionay.

Iterate over dictionary keys: keys()

```
for sent in Sessions:
    print(sent)
```

```
client0
client1
client2
client3
client4
client5
client6
client7
client8
client9
client10
client11
client12
client13
client14
client15
client16
client17
client18
client19
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Iterate over dictionary values: values()

```
for sent in Sessions.values():
    print(sent)
```

```
[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],
['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'], ['s
'0'], ['scanner', 'Barcode', '0'], ['scanner', 'Barcode', '0'
[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],
['transmission', 'CaisseNumber', '0'], ['abandon', 'Nothing',
[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0'],
'0'], ['transmission', 'CaisseNumber', '0'], ['abandon', 'Not
['payer', 'Price-float', '0']]
```

Iterate over dictionary key-value pairs: items()

```
for c,sent in Sessions.items():
    print(c,sent)
```

```
client0 [['debloquer', 'Nothing', '0'], ['scanner',
e', '0'], ['scanner', 'Barcode', '0'], ['scanner',
ode', '0'], ['scanner', 'Barcode', '0'], ['scanner',
client1 [['debloquer', 'Nothing', '0'], ['scanner',
e', '0'], ['transmission', 'CaisseNumber', '0'], ['a
 client2 [['debloquer', 'Nothing', '0'], ['scanner',
arcode', '0'], ['transmission', 'CaisseNumber', '0'
g', '0'], ['payer', 'Price-float', '0']]
client3 [['debloquer', 'Nothing', '0'], ['scanner',
e', '0'], ['scanner', 'Barcode', '0'], ['scanner',
ber']]
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Change event to string

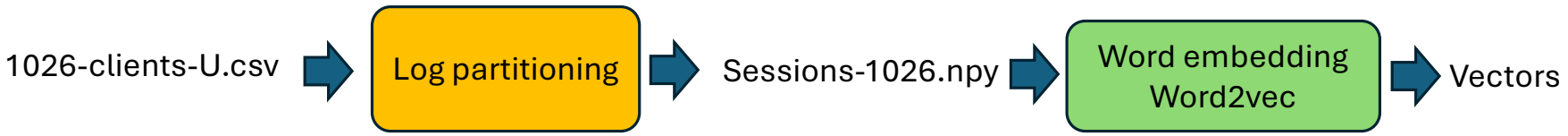
```
Sentences=[]
all_clients_string=[]
for c,sent in Sessions.items():
    Sentences.append(sent)
    SentencesString=[]
    for i in sent:
        SentencesString.append(str(i))
    all_clients_string.append(SentencesString)
print(all_clients_string)
```

[['debloquer', 'Nothing', '0'], ['scanner', 'Barcode', '0’] ….

["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']"

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Log partitioning based on ClienTID



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Word2Vec

```
import multiprocessing
from gensim.models import Word2Vec
cores = multiprocessing.cpu_count()
w2v_model = Word2Vec(min_count=1,window=3,vector_size=15,sample=0,alpha=0.03,min_alpha=0.0007,negative=2,workers=cores-1)
w2v_model.build_vocab(all_clients_string)
w2v_model.train(all_clients_string, total_examples=w2v_model.corpus_count, epochs=10, report_delay=1)
```

```
index2word_set = set(w2v_model.wv.index_to_key)  # Our Vocab
print("the word that we have are:  ",index2word_set)
```

The word that we have are: 15

{"['supprimer', 'Barcode', '0']", "['payer', 'Price-float', '0']", "['ajouter', 'Barcode', '0']", "['payer', 'Price-integer', 'Float Number']", "['scanner', 'ErrorBarcode', '0']", "['scanner', 'Barcode', '-2']", "['fermerSession', 'Nothing', '0']", "['abandon', 'Nothing', 'Error']", "['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']", "['ouvrirSession', 'Nothing', '0']", "['transmission', 'CaisseNumber', '0']", "['transmission', 'CaisseNumber', 'Integer Number']", "['supprimer', 'ErrorBarcode', '0']", "['supprimer', 'Barcode', '-2']"}

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Each word is represented as a vector

```
w2v_model.wv["['payer', 'Price-float', '0']"]
```

```
array([-0.208438  , -0.17140363,  0.58019   , -0.8334066 , -0.4235734 ,
        0.43202496,  0.16815467,  0.51377785, -0.1497903 ,  0.54200566,
        0.1650343 ,  0.39274028,  0.00320614, -0.7347753 ,  0.13456789],
      dtype=float32)
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

All words and their associated vectors

```
#************************** Create a dictionary of words and associated vectors.  ******************
word_vector=dict()
for w in index2word_set:
    if w in w2v_model.wv:
        print("word is ",w)
        print("w2v_vector is",w2v_model.wv[w])
        word_vector[w]=w2v_model.wv[w]
    print("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^")
#*****************************************************************************************************
```

Dictionary (word_vector)

Word 1…….. vector

Word 2…….. vector

Word 15……. vector

word1

```
['ouvrirSession', 'Nothing', '0']                    vector

w2v_vector is [-0.43123573 -0.24409345  1.1653835  -1.482715   -0.5842829   0.602372
  0.2035547   0.751147   -0.2633315   0.82852894  0.3240775   0.5277217
 -0.04162983 -1.1414808   0.19814692]
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

A session concerning the client0

```
print(all_clients_string[0])
```

```
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']", "['scanner', '
'Barcode', '0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']",
'0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['scanner
r']", "['payer', 'Price-float', '0']"]
```

Sessions	
client0	[scan, scan, ...]
client1	[scan, delete,...]
client2	[scan, delete,...]
⋮	⋮

All_client_string	
0	[''scan'','' scan'', ...]
1	
2	

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

A vector concerning the client0

```
print(all_clients_string[0])

feature_vec = np.zeros((15, ), dtype='float32')
number_words = 0
for word in all_clients_string[0]:
    number_words = number_words+1
    feature_vec = np.add(feature_vec, w2v_model.wv[word])

feature_vec = np.divide(feature_vec, number_words)
print(feature_vec)
```

List of Actions of client0=session of client0

Counting the number of actions that client0 did

```
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode', '0']", "['scanner
'Barcode', '0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '
'0']", "['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['sca
r']", "['payer', 'Price-float', '0']"]
[ 0.20394726  0.1303705   0.4773702   2.5052588  -0.05649199 -1.7890764
  0.95154786  0.04148404 -0.29469082 -1.2973394   0.07733324 -0.4541708
 -1.4051955   1.7533631  -0.36733612]
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Range() function

- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Loop

```
for i in range(6):
  print(i)
```

→ {0,1,2,3,4,5}

```
0
1
2
3
4
5
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

61 vectors concerning all cilents

```
sessions_embedding=[]
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words)  #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec)
print(sessions_embedding)
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

61 vectors concerning all cilents

```
sessions_embedding=[]
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words)  #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec)
print(sessions_embedding)
```

By using a loop (for), we can calculate the average for all 61 clients

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

61 vectors concerning all cilents

```
sessions_embedding=[]
for i in range(61):
    feature_vec = np.zeros((15, ), dtype='float32')
    number_words = 0
    for word in all_clients_string[i]:
        number_words = number_words+1
        feature_vec = np.add(feature_vec, w2v_model.wv[word])

    feature_vec = np.divide(feature_vec, number_words)  #vector for one client
    print(i,":",feature_vec)
    sessions_embedding.append(feature_vec)
print(sessions_embedding)
```

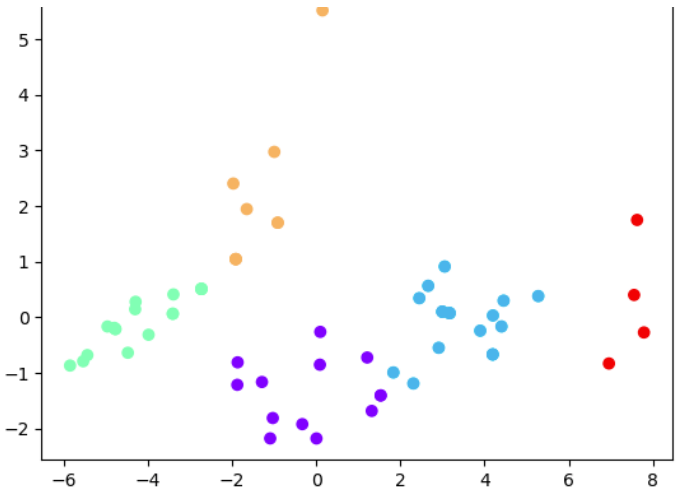
To keep all the vectors

.append can add new vector

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Clustering

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5)
kmeans.fit(X_pca)
plt.scatter(X_pca[:,0],X_pca[:,1], c=kmeans.labels_, cmap='rainbow')
plt.show()
```



PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Clustering

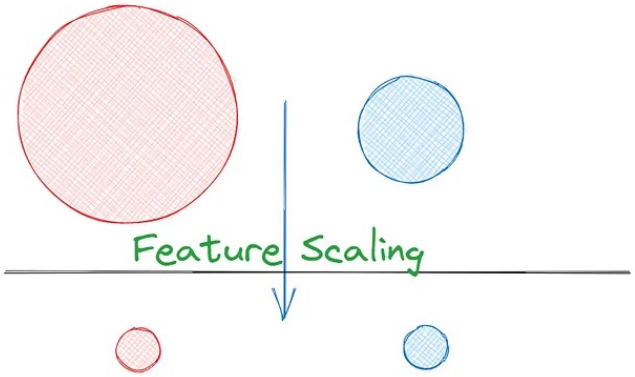
```
clusters=dict()
for l in range(5):
    vocab_elem=[]
    OneCluster = np.where(kmeans.labels_ == l)
    print("Cluster ",l,":")
    for j in OneCluster[0]:
        print("client",j,":")
        print(all_clients_string[j])
        vocab_elem.append(j)
    clusters[l]=vocab_elem
    print("**********************************************
```

```
Cluster  0 :
client 4 :
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
r', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "['
de', '0']", "['scanner', 'Barcode', '0']", "['scanner',
['scanner', 'Barcode', '0']", "['scanner', 'Barcode', '0
['abandon', 'Nothing', 'Error']", "['ouvrirSession', 'No
'0']"]
client 5 :
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
canner', 'Barcode', '0']", "['scanner', 'Barcode', '0']"
arcode', '0']", "['scanner', 'Barcode', '-2']", "['scann
'0']", "['scanner', 'Barcode', '0']", "['transmission',
code', '0']", "['ajouter', 'Barcode', '0']", "['fermerSe
client 11 :
["['debloquer', 'Nothing', '0']", "['scanner', 'Barcode'
er', 'Barcode', '0']", "['scanner', 'Barcode', '0']", "[
```

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Feature Scaling

- Imagine you’re comparing apples and oranges; it's nonsensical.
- Features measured in vastly different units (like income and age) can skew machine learning algorithms if not addressed.
- Feature scaling addresses this by transforming data into a standard scale, enabling fair comparison between different features.



https://medium.com/@yashbaravaliya206/feature-scaling-and-normalization-ca484a16882a

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Feature Scaling

- Feature scaling is a fundamental preprocessing step in machine learning aimed at ensuring that numerical features have a similar scale.
- Apply feature scaling before feeding the data into the machine learning model (K-NN), **except** for algorithms that are scale-invariant, such as **decision trees**.

PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

Common Techniques for Feature Scaling

- **Normalization :**
  - This method scales each feature so that all values are within the range of 0 and 1.
  - x is the value you want to normalize
  - min(X) is the minimum value in your data set
  - max(X) is the maximum value in your data set

$$(x - \min(X)) / (\max(X) - \min(X))$$

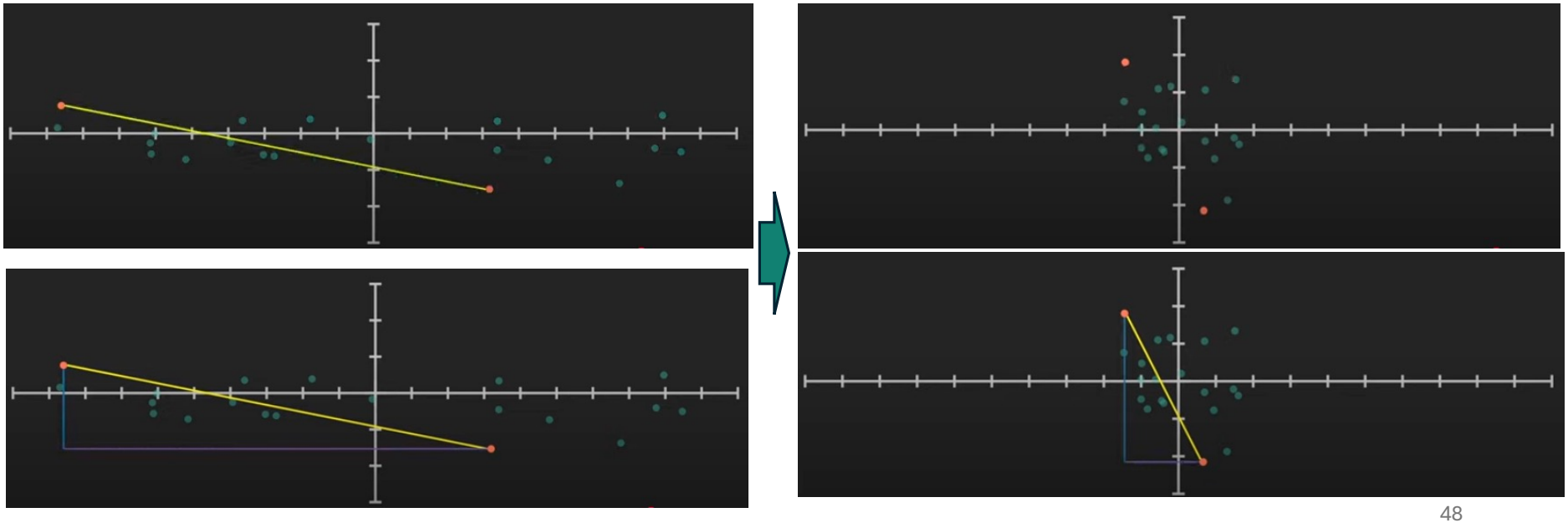
- **Standardization (Z-score Scaling):**
  - Here, each feature is transformed to have a mean of 0 and a standard deviation of 1.
  - This is achieved by subtracting the mean value and dividing by the standard deviation of the feature.
  - x is the original value you want to standardize, μ (mu) is the mean of the data set, σ (sigma) is the standard deviation of the data set

$$Z = (x - \mu) / \sigma$$

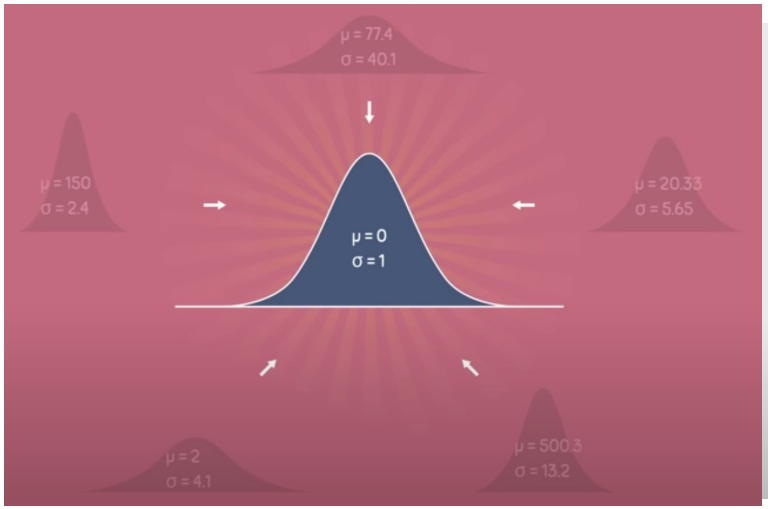
PROGIS
ETUDES, MEDIAS, COMMUNICATION, MARKETING

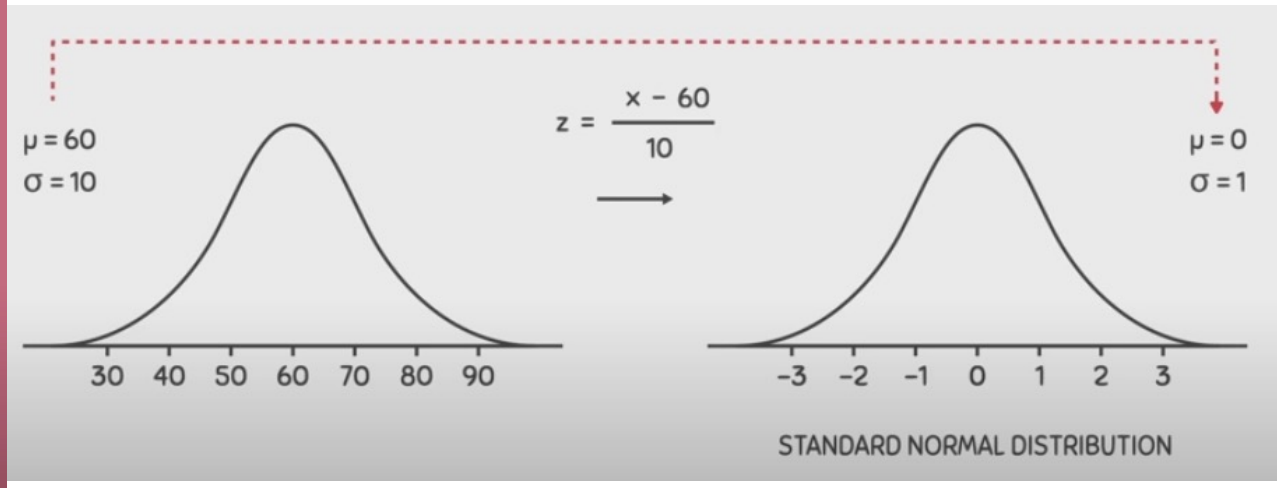
Feature Scaling

- In the last figure, both feature contribute equally to the distance
- After that, your algorithm won't be affected by the feature with a higher scale.



```
from sklearn.preprocessing import StandardScaler
stdsc=StandardScaler()
X_std=pd.DataFrame(stdsc.fit_transform(X_all), index=X_all.index, columns=X_all.columns)
```





Any normal distribution with any value of mean and standard deviation(sigma) can be transformed into the standard normal distribution where we have a mean of zero and a standars deviation of one.

End

bahareh.afshinpour@univ-grenoble-alpes.fr