ملاحظة:

من الممكن أن يظهر جزء كود بايثون من اليمين إلى اليسار، لذا يفضل نسخة ولصقة في برنامجك المفضل لكتابة الكود أو عرضة بشكل أفضل في صفحات الملخصات في موقي الشخصي:

https://alioh.github.io/DSND-Notes-11/

https://alioh.github.io/DSND-Notes-12/

https://alioh.github.io/DSND-Notes-13/

https://alioh.github.io/DSND-Notes-14/

https://alioh.github.io/DSND-Notes-15/

https://www.alioh.com

Clustering - الدرس الأول - التجميع

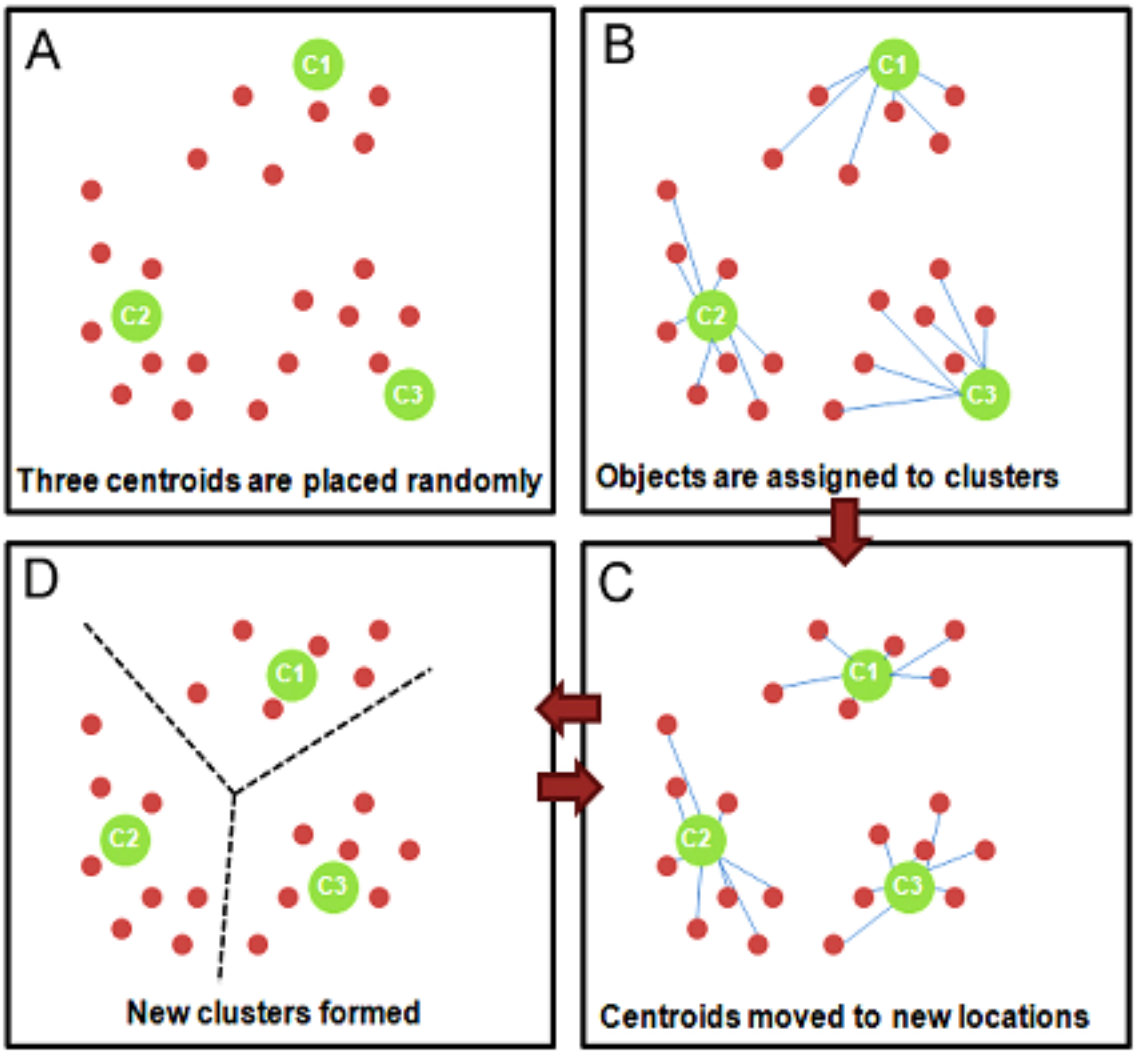
عن التعلم غير الموجَّه بشكل عام، تعريفة وأمثله عليه واشهر خوارزمياته كتبت في **منشور سابق**.

أنواع التعلم غير الموجَّه

- **Clustering التجميع:** جمع البيانات على شكل فئات حسب خصائص معينه تتاشبه فيها.
- **Dimensionality reduction تقليص الأبعاد:** عرض البيانات بأقل عدد من الخصائص.
- ويضاف لها أحياناً:
- **Association Rules قواعد / قوانين الربط:** ربط البيانات فيما بينها عن طريق محاولة أكتشاف علاقات بينها.

خوارزميات التعلم غير الموجَّه

تجميع بالمتوسطات K-Means



تستخدم هذه الخوارزمية كثيراً في برامج تقديم الإقتراحات مثلاً في Spotify و Netflix. تم شرحها سابقاً بشكل مفصل [هنا](#).

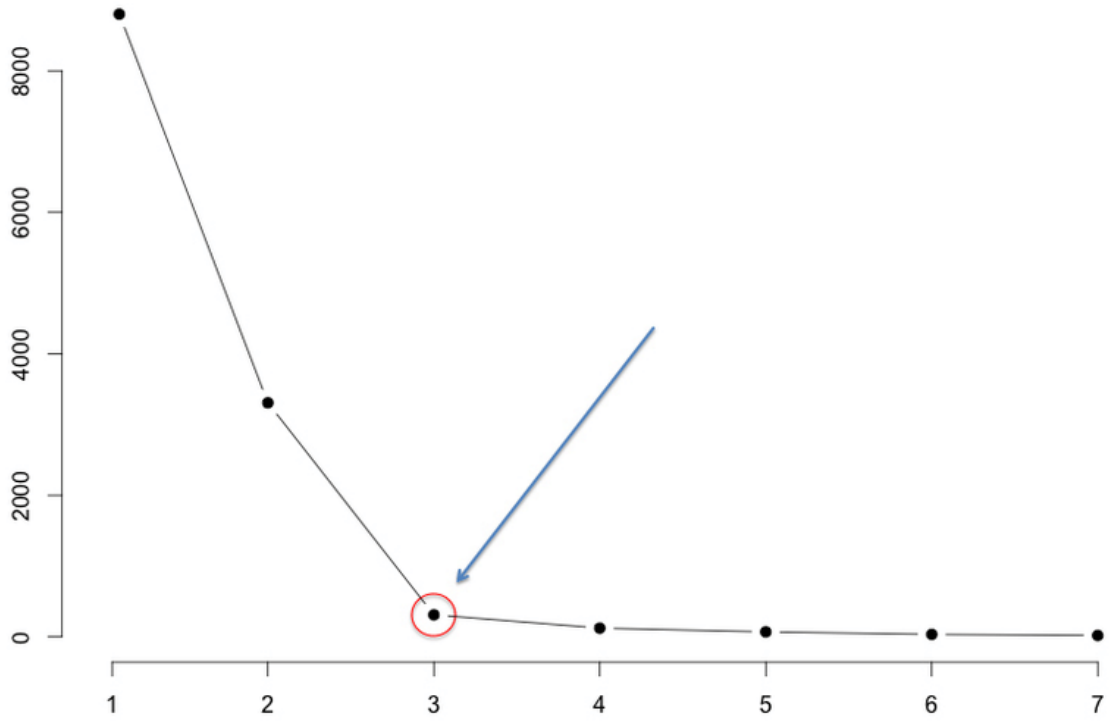
طريقة عمل الخوارزمية K-Means

- يتم إعطاء الخوارزمية رقم K إلى البيانات (تجميع) ويعني عدد مرات تقسيم Clusters.
- يتم رسم نقاط تعرف بالنقاط المركزية K بنفس الرقم Centroids.

- تربط كل بيانات بما يقارب لها من نقاط مركزية.
- تقوم الخوارزمية بحساب المتوسط لجميع النقاط داخل كل Cluster.
- تكرار الخطوتين الأخيرتين مرة أخرى حتى تصل لنقاط مركزية وتجميعة مناسبة لجميع النقاط.

Elbow Method

في البيانات لدينا. تستخدم هذه الطريقة Clustersتعني هنا عدد الـ K ، و K-Means في خوارزمية Kطريقة لإيجاد عدد الـ عندما لا نعرف عدد التصنيفات لدينا. طريقة عملها كالتالي:



- نتبع الخطوات السابقة لعمل الخوارزمية
- يحتوي المودل على دالة Score عند انشاء الخوارزمية بواسطة Scikit-learn
- نظهر النتائج في رسم بياني يكون شكله كالتالي
- في الصورة 3 هو العدد المناسب الذي يتوقف فيه التغير العالي في متوسط المسافة بين النقاط و الـ Cluster.

X = Number of K - Y = Average Distance to Cluster

مثال بايثون

طريقة كتابة خوارزمية K-means وإستخدام الـ Elbow Method

```
from sklearn.cluster import KMeans
```

```
import numpy as np
import matplotlib.pyplot as plt

# لنفرض لدينا بيانات معرفة في data
data = data

# نعرف المودل ونضيف له عدد ال Clusters
kmeans_4 = KMeans(n_clusters=4)

# نقوم بربط البيانات مع المودل
model_4 = kmeans_4.fit(data)

# للتوقع نقوم بالتالي
labels_4 = model_4.predict(data)


# الجزء التالي لعرض كرسم بياني بإستخدام مكتبة matplotlib Elbow method

# هنا نجرب من 1 إلى 10 ونظهر نتائجها Clusters
No_of_clusters = range(1, 10)

# ننشأ قائمة تحتوي على 10 مودلز من 1 إلى 10 Clusters
kmeans = [KMeans(n_clusters=i) for i in No_of_clusters]

# ننشأ قائمة ثانية تحتوي على نتائج كل مودل من التي أنشئناها بالخطوة السابقة
score = [np.abs(kmeans[i].fit(data).score(data)) for i in range(len(kmeans))]

# عرض النتائج في رسم بياني
plt.plot(No_of_clusters, score, linestyle='--', marker='o', color='b');
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

Feature Scaling

طريقة لإعادة تعين قيمة النتائج حتى نحصل على بيانات ونتائج صحيحة، توجد طريقتين لذلك:

Values	Normalized	Standardized
47	0.9302	1.1560
7	0.0000	-1.9267
21	0.3256	-0.8478
28	0.4884	-0.3083
41	0.7907	0.6936
49	0.9767	1.3102
50	1.0000	1.3872
25	0.4186	-0.5395
25	0.4186	-0.5395
35	0.6512	0.2312
24	0.3953	-0.6165

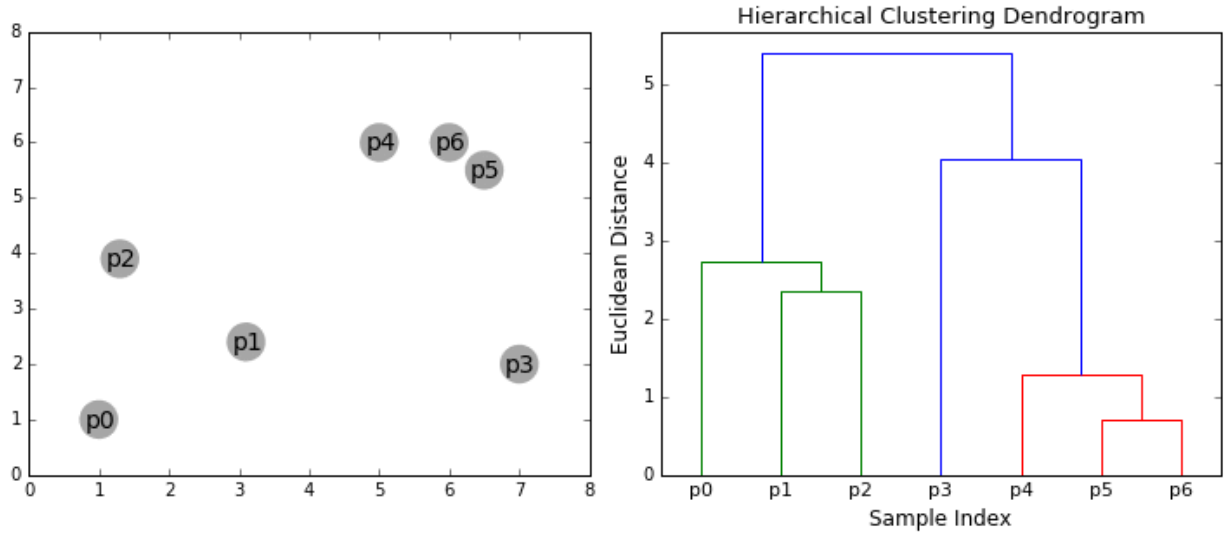
- Standardizing: تحويل القيم إلى رقم جديد يكون متوسط Mean وتباين Variance كل النتائج يساوي 0 يساوي 1. تكون النتائج عادة متقاربة من -3 إلى 3 مثلاً.
- Normalizing: تحويل القيم إلى رقم جديد من 0 إلى 1.

مثال لنتائج قبل وبعد التحويل لإحدى الطريقتين:

الدرس الثاني - التجميع القائم على الكثافة والهرمية - Hierarchical and Density Based Clustering

التجميع الهرمي Hierarchical Clustering

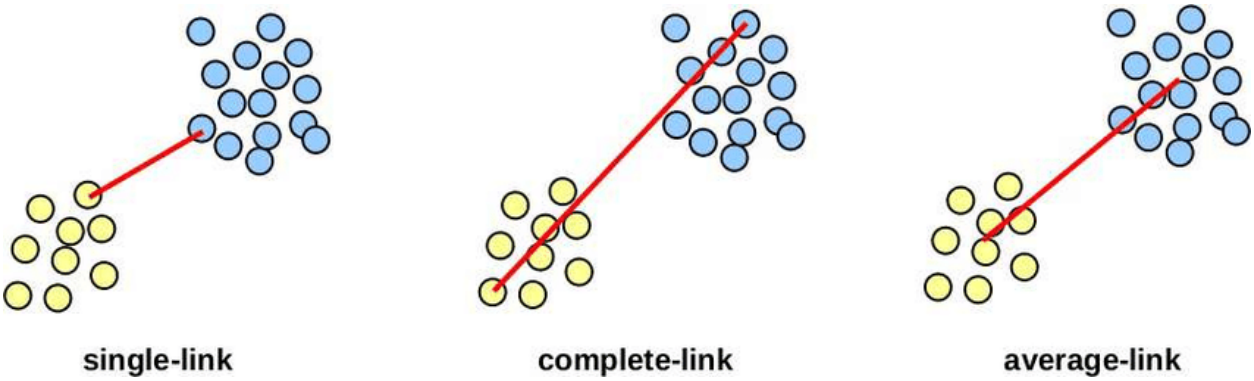
كتبت **سابقاً** عن التجميع الهرمي. وشرحت طريقتها كالتالي:



- لكل نقطة لدينا Cluster التجميع الهرمي يُكَوّن مجموعة أو.
- يتم جمع أقرب نقطتين مع بعضهم البعض وتُكَون مجموعة جديدة.
- تبحث الخوارزمية عن نقطة أخرى قريبة لها وتضمها إلى نفس المجموعة Cluster.
- تُجمع فيها كل النقاط Cluster تستمر الخوارزمية بعمل الخطوتين السابقتين إلى أن تبقى مجموعة واحده.

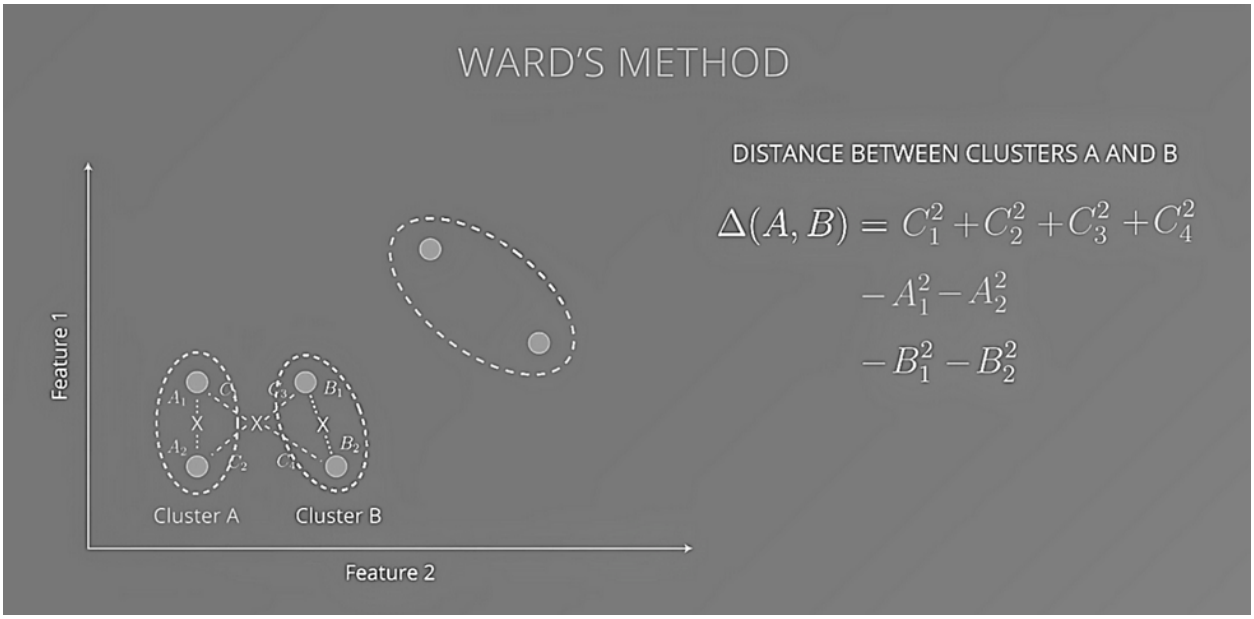
أنواع التجميع الهرمي:

توجد أربع أنواع لتحديد المسافات بين الـClusters في التجميع الهرمي:



- **Single Link**: نقطتين لبعضها البعض في كل **أقرب** وأخرى عن طريق حساب Cluster تحدد المسافه بين كل Cluster.
- **Complete Link**: نقطتين لبعضها البعض في **أبعد** وأخرى عن طريق حساب Cluster تحدد المسافه بين كل Cluster كل.
- **Average Link**: في **متوسط المسافه بين كل النقاط** وأخرى عن طريق حساب Cluster تحدد المسافه بين كل Cluster كل.
- **Ward's Method**: بين Centroid تقوم بتكوين نقطة مركزية Ward's على عكس الطرق السابقة، طريقة ، بعد ذلك تحسب المسافة من النقاط إلى النقطة الجديدة التي أوجدتها سابقاً، تربع المسافه Clusters مجموعتين

السابقة لكل نقطة ثم تُجمع جميعها. بعد ذلك نقوم بتربيع ثم طرح المسافة بين كل نقطة والنقطة المركزية في الـCluster الخاص بها.



كود بايثون

تطبيق عملي في بايثون للتجميع الهرمي

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import adjusted_rand_score
from sklearn import preprocessing
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
import matplotlib.pyplot as plt
from sklearn import datasets


# تحميل البيانات، في هذه المثال استخدمنا بيانات مرفقة من Scikit-learn
# بأسم Iris
iris = datasets.load_iris()

# ننشأ المودل ونعطية عدد الـ Clusters
# النوع التلقائي المستخدم من المكتبة هي Ward's Method
# إذا اردنا إختيار نوع مختلف من الأربعة التي ذكرت سابقاً
# نضيف الخيار linkage
# ونعطية أحد الخيارات التاليه:
# “ward”, “complete”, “average”, “single”
# مثال:
# AgglomerativeClustering(n_clusters=3, linkage='complete')
ward = AgglomerativeClustering(n_clusters=3)

# الربط والتوقع بدالة واحدة fit_predict
ward_pred = ward.fit_predict(iris.data)
```

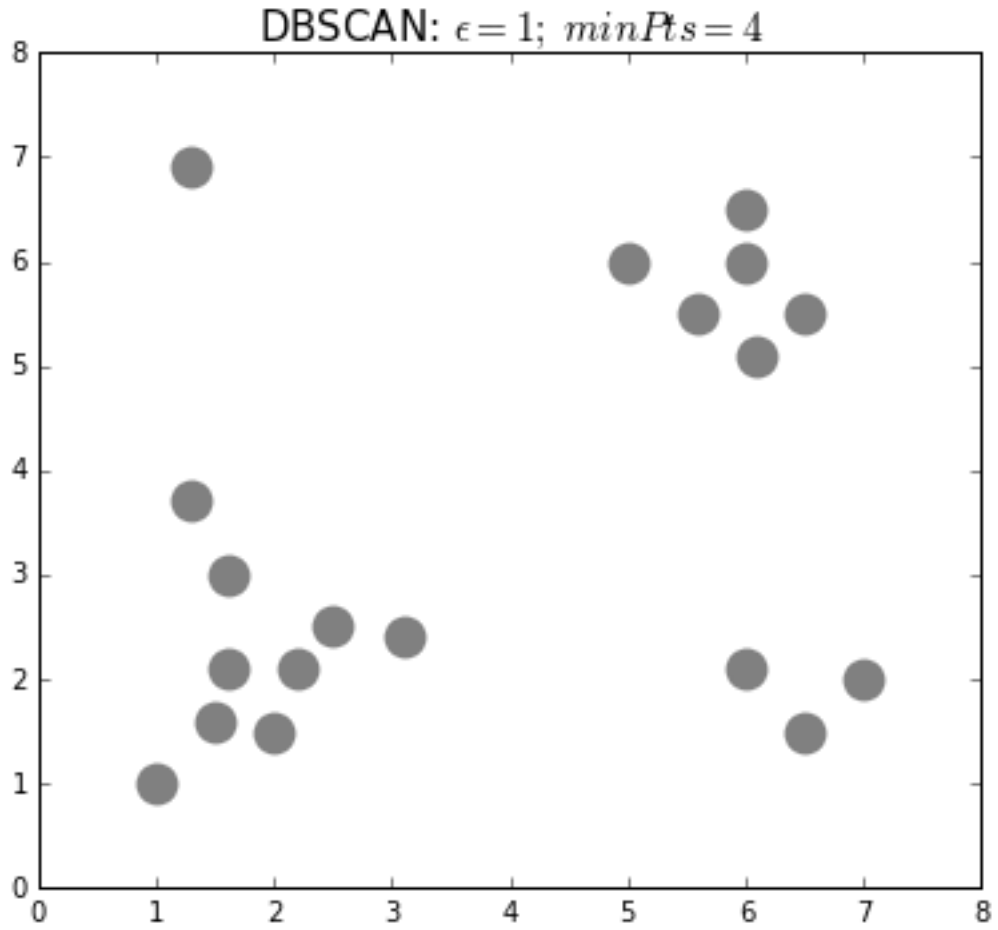
```
# لتقييم المودل ومعرفة دقته
ward_ar_score = adjusted_rand_score(iris.target, ward_pred)
# النتيجة: 0.73119

# لتحسين النتائج أو إذا كانت لدينا فروقات كثيره بين البيانات من
# الممكن أن بالـ Normalizing
# راجع الجزء 11 لشرح عن Normalization
normalized_X = preprocessing.normalize(iris.data)
# النتيجة: 0.88569

# لعرضها بشكل رسم بياني نقوم بالتالي
linkage_type = 'ward'
linkage_matrix = linkage(normalized_X, linkage_type)
plt.figure(figsize=(22,18))
dendrogram(linkage_matrix)
plt.show()
```

Density-Based Clustering التجميع القائم على الكثافة

قبل انشاء المودل نعطيه بعض المتغيرات:

1. min_samples أو Cluster عدد النقاط في كل
2. Epsilon ε المسافة بين كل نقطة وأخرى

ε نبدأ من نقطة معينة ونرى إذا كان هناك نقاط اخرى في المسافة التي حددناها:



1. اذا لم يكن هناك نقاط أخرى، تحسب هذه النقطه كنقطة مزعجة ويتم تجاهلها.
2. اذا كان هناك نقاط أخرى ومجموعها أقل من الحد الادنى لعدد النقاط الذي حددناه، نتجاهل هذه النقاط وننتقل لأي نقطة قريبة أخرى
3. هنا، في حال وجدت Cluster أذا وجد نقاط أخرى ومجموع النقاط أعلى أو يساوي العدد الذي حددناه، نقوم بإنشاء Cluster لهذه النقطة، نجمعها جميعاً في Cluster نقاط أخرى توافي نفس الشروط بالحد الادنى والمسافة وفي نفس الـ Cluster واحد.

مثال بايثون

مثال عملي لطريقة إنشاء مودل للتجميع القائم على الكثافة

```
from sklearn.cluster import DBSCAN

# لتكن لدينا بيانات معرفة في المتغير data
data = data

# ننشأ المودل ونحدد المسافة eps
# والحد الادنى من النقاط min_samples
```

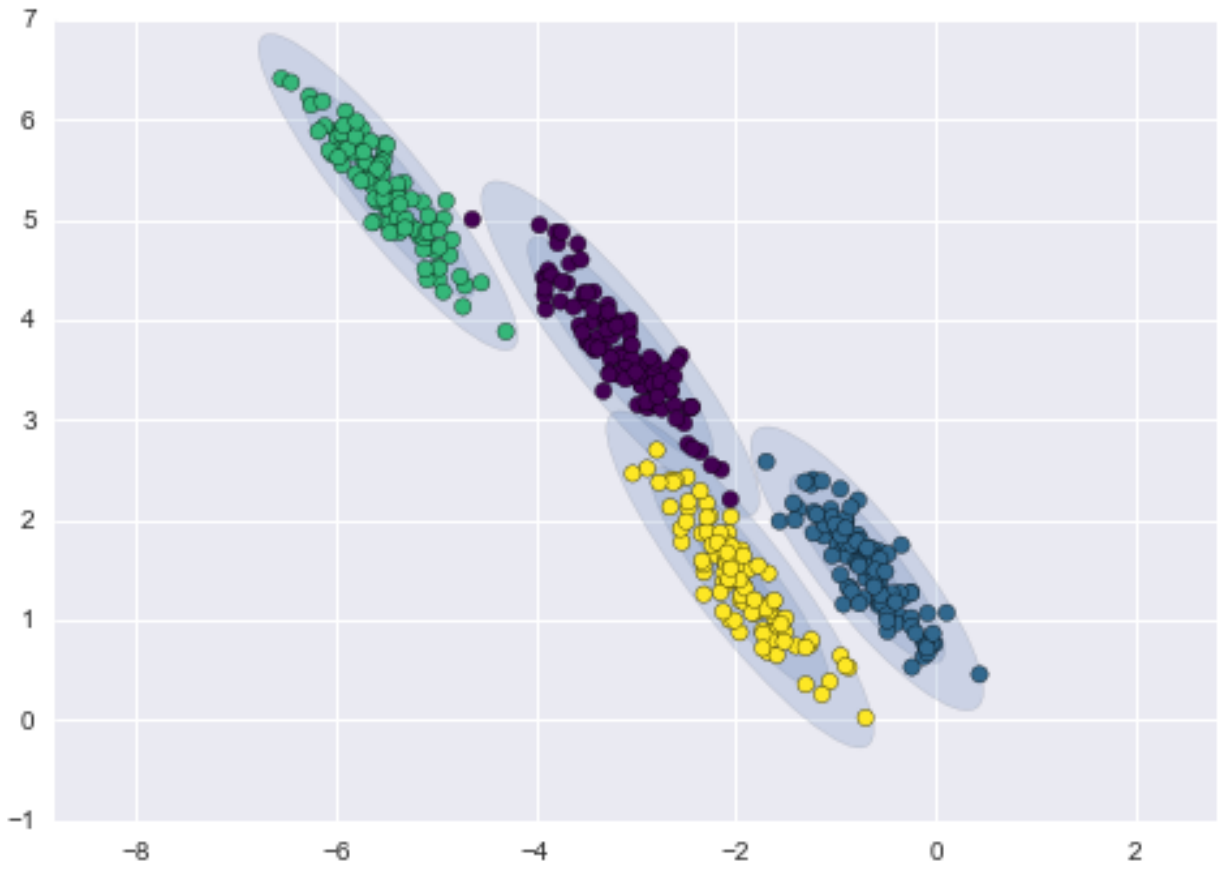
```
dbscan = DBSCAN(eps=3, min_samples=2)

# الربط والتوقع بدالة واحدة fit_predict
# النتيجة -1 تعني ان القيمة مزعجه وتم تجاهلها
clustering_labels_1 = dbscan.fit_predict(data)
```

الدرس الثالث - Gaussian Mixture Models and Cluster Validation

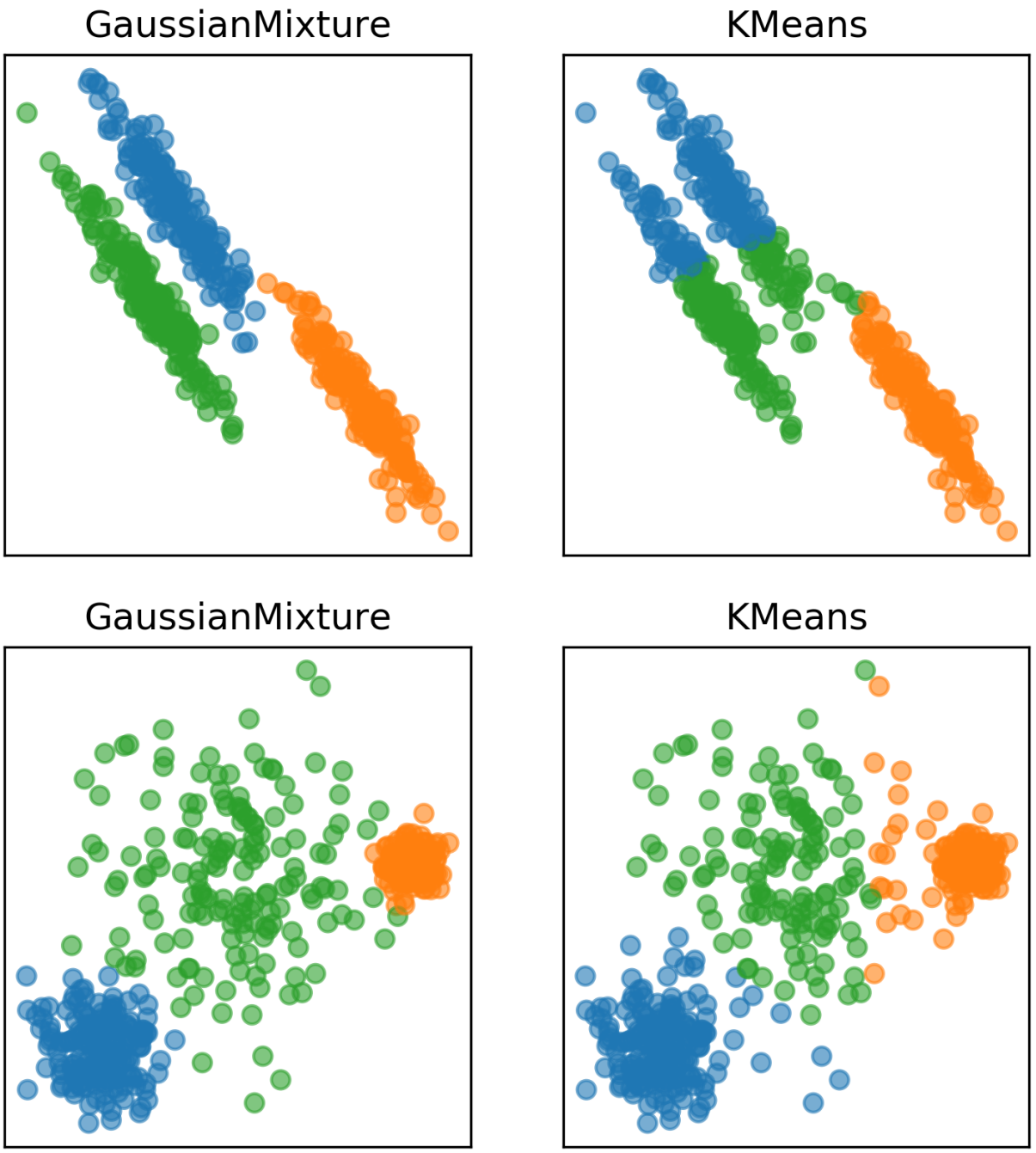
Gaussian Mixture Models (GMM)

وهي طريقة لرسم Gaussianولكن تأخذ بعين الإعتبار التباين. وسنستخدم فيها الـ K-meanخوارزمية تجميع، مشابهه لـ البيانات وفهم الخوارزمية لكيفية تشكيل البيانات لدينا.

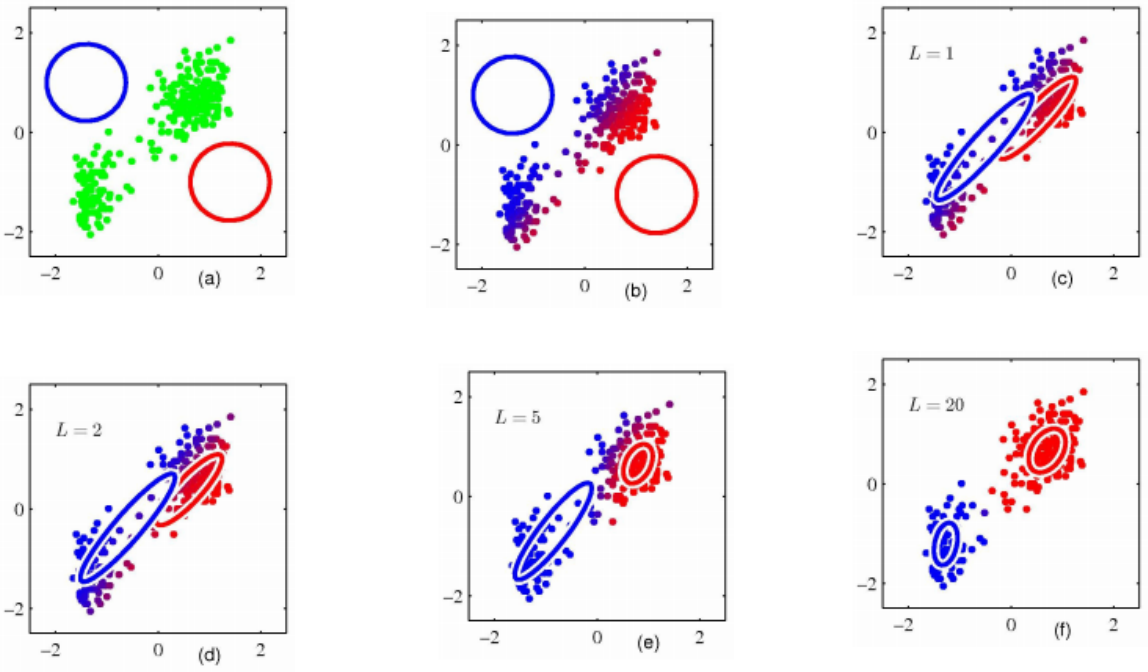
- **Mean المتوسط:** وهو دائماً يكون في وسط الـ Cluster.
- **Variance التبيان:** على سبيل المثال، الصورة التالية كونت 4 Cluster. ويشكل طريقة توسع البيانات في الـ Clusters، وكما يظهر لنا طريقة توسعها بسبب عمل الخوارزمية على المتوسط والتبيان لتشكيل الـ Clusters التي تأدي عملها افضل على البيانات ذات الشكل الدائري K-mean على عكس ما كانت عليه خوارزمية.



مثال أخر للفرق في تكوين الـClusters في K-mean و GMM



Expectation Maximization (EM)



وطريقتها كالتالي GMM طريقة تتبع في خوارزمية:

- وأختيار نقاط بداية عشوائية Clusters كعدد للـ K اختيار.
- خاص بها Cluster نحدد لكل نقطة في البيانات.
- نحسب المسافة والتبيان بين كل نقطة والنقطة المركزية، ثم نقوم بتغير موقع النقطة المركزية، ثم إعادة الحساب للمتوسط والتباين.
- اذا توقف التغير في المتوسط يتم إختيار المنطقة كنقطة مركزية جديدة.

طريقة تحليل الـ Clusters

- **Feature Selecting:** اختيار المتغيرات او الخصائص المناسبة وتوجد طرق لإختيار الأفضل.
- **Clustering Algorithm:** اختيار الخوارزمية المناسبة في التجميع وإعادة ضبطها حتى نصل لنتائج مرضية.
- **Clustering Validation:** التحقق وعرض النتائج وتقيمها.

Cluster Validation

في التعلم غير الموجّه توجد ثلاث Precision والـ Accuracy في التعليم الموجَّه كانت لدينا خيارات تقييم للمودل كـ ، وهي Clusterخيارات لتقييم الـ:

- **External Index:** يتم فيه مقارنة النتائج في الـ Clusterمع النتائج الصحيحة (Labels) المعروفة مسبقاً.
- **Relative Index:** تقارن بين اثنين من الـ Clusters.
- **Internal Index:** يتحقق من النتائج للـ Cluster بدون الإعتماد على النتائج الصحيحه (ان لم تكن موجودة مثلاً).

$$Si = \frac{bi - ai}{Max(bi - ai)}$$

Silhouette Coefficient: والمعادلة الخاصة بها (Labels) اذا لم تكن هناك نتائج Clusterطريقة لحساب اداء الـ كالتالي:

1. a = معدل المسافة من نقطة إلى النقاط الاخرى في نفس الـCluster.
2. b = للتي تحتوي على نقطتنا الحالية. نكرر هذه Cluster معدل المسافة من نقطة إلى النقاط الأخرى في أقرب العملية على كل نقطة لدينا، ثم نجمع النتائج ونأخذ المعدل. وهو بالغالب رقم من -1 إلى 1.

مثال بايثون

مثال لطريقة إنشاء مودل بخوارزمية GMM

```
from sklearn.mixture import GaussianMixture

data = data
# ننشأ المودل ونحدد عدد الـ Cluster 3
gmm = GaussianMixture(n_components=3)

# الربط والتوقع
gmm = gmm.fit(X)
pred_gmm = gmm.predict(X)
```

الدرس الرابع - Dimensionality Reduction and PCA

تقوم الخوارزمية بجمع البيانات ومحاولة إيجاد انماط فيما Patterns. في التعلم غير الموجَّه لإكتشاف الأنماط PCA تستخدم خاصة بها بجانب ما يشابهها. الفيديو التالي يشرح طريقة عملها Cluster بينها، ثم جمع كل واحده في

Latent Features - Feature Selection

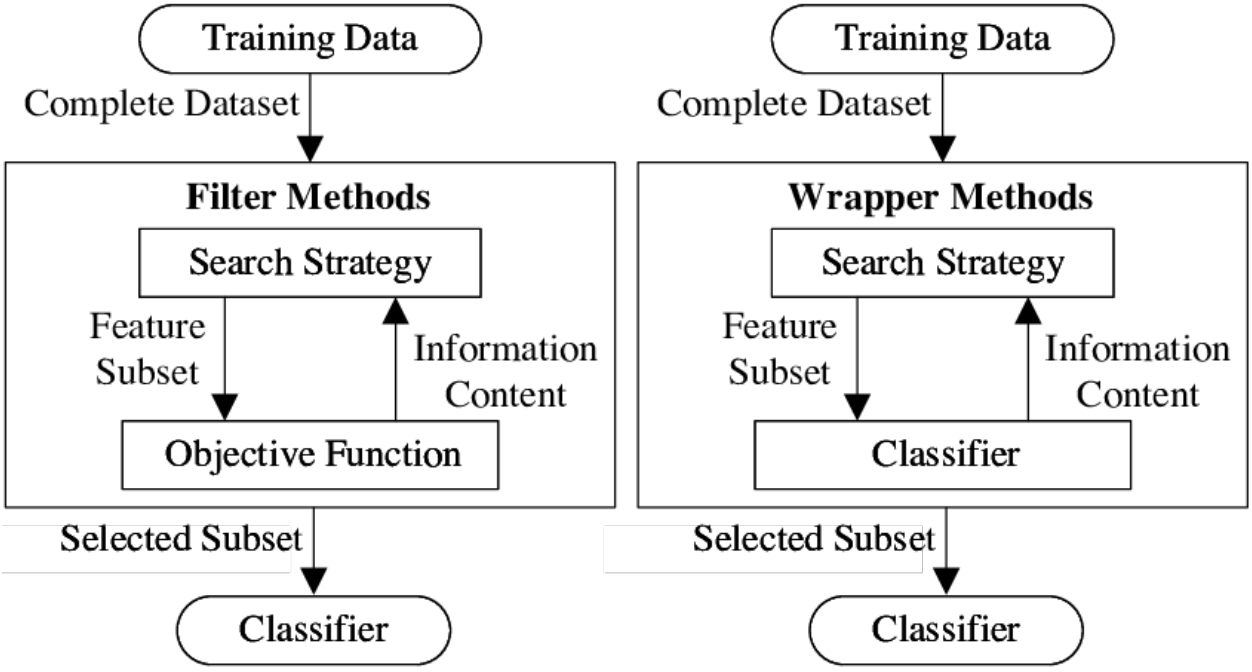
متغيرات ليست موجودة في البيانات الأصلية لدينا، لنقل أن لدينا بيانات منازل وفيها المتغيرات التالية:

- حجم الأرض
- عدد الغرف

- مساحة المنزل
- مساحة المخزن
- عدد غرف النوم
- عدد دورات المياة
- معدل الجرائم في الحي
- عدد المدارس في الحي
- الضرائب في الحي
- متوسط مصادر الدخل للجيران
- متوسط جودة الهواء
- المسافة إلى الخط السريع

في المتغيرات السابقة ممكن ان نستخرج متغيرين جدد غير الموجودة لدينا وهي لأول 6 متغيرات وآخر 6 متغيرات، بذلك قللنا عدد المتغيرات لدينا في البيانات. المتغيرات الجدد هي **معلومات الحي** والثاني **حجم المنزل**.

توجد عدة طرق لإختيار المتغيرات وهي:



- **Filter methods:** ومن مسماها ممكن ان نتوقع أنها تقوم بفلترة المتغيرات الموجودة وترتيبها حسب اهميتها ثم إعتمادها (هنا لدينا، يتم بعد ذلك تقيمها بعدة طرق (تفاصيل أكثر عن هذه الطرق.
- **Wrapper Methods:** هذة الطريقة تقوم بالتحقق من اهمية المتغير مباشرة مع المودل، اذا انتجت المودل افضل نتيجة له فسيتم إختيار تلك المتغيرات.

مثال بايثون

مثال عملي طويل ومفصل لطريقة عمل خوارزمية PCA في الرابط التالي: هنا

PCA متى نستخدم

لدينا. في بعض الأحيان في الصور مثلاً، قللنا Features تستخدم الخوارزمية متى ما أردنا التقليل من عدد المتغيرات المتغيرات من 700 متغير إلى 30 متغير وحصلنا على نفس النتائج.

الدرس الخامس - Random Projection & ICA

أو المتغيرات لدينا. لنفترض أن Dimensionsلتقليل المتغيرات. هدف الطريقة الرئيسي هو تغليل الـ PCAهي حل بديل لـ Random Projection وتشغيل دالة Scikit-learn لدينا قاعدة بيانات فيها 12000 عامود أو متغير، عند إدخالها في عليها ستعود لنا البيانات بعدد عواميد حوالي 6000.

مثال بايثون

طريقة عمل Random Projection في بايثون و Scikit-learn

```
from sklearn import random_projection

data = data

# إنشاء المودل الخاص ب Random Projection
# عدد المتغيرات المطلوب n_componentsمن الممكن تحديد عدد
# وتحديد قيمة eps ε
# اذا تركت بدون تحديد فتكون قيمة eps = 0.1 و n_components = 'auto'
# وهي القيم التلقائية
rp = random_projection.SparseRandomProjection()

# تشغيل المودل على البيانات لدينا
new_data = rp.fit_transform(data)
```

Independent Component Analysis - ICA

مثال عليها في ملفات الصوت، اذا كان لدينا ثلاث ادوات صوتية Random Projection و PCA طريقة أخرى مشابهه لـ تعمل بنفس الوقت، تشغيل هذه الخوارزمية عليها ستحاول فصل كل اداة لوحدها.

مثال بايثون

طريقة عمل Random Projection في بايثون و Scikit-learn

```
from sklearn.decomposition import FastICA

# البيانات لدينا متكونة من قائمة تحتوي على 3 ملفات صوتية
data = list(zip(audio_1, audio_2, audio_3))

# إنشاء المودل الخاص
# عدد المتغيرات المطلوب n_componentsتحديد عدد
```

```
ica = FastICA(n_components=3)

# تشغيل المودل على البيانات لدينا
new_data = ica.fit_transform(data)
```

أنتهى المخلص.

الملخصات منشورة بشكل مفصل وبترتيب أفضل في صفحتي الشخصية هنا:

https://alioh.github.io/DSND-Notes-11/

https://alioh.github.io/DSND-Notes-12/

https://alioh.github.io/DSND-Notes-13/

https://alioh.github.io/DSND-Notes-14/

https://alioh.github.io/DSND-Notes-15/

https://www.alioh.com

مصادر:

1. **mubaris** - K-Means Clustering
2. **Quora** - Feature Scaling
3. **saedsayad** - Hierarchical Clustering
4. **dashee87** - Hierarchical Clustering 2
5. **jakevdp** - Gaussian Mixture
6. **amueller** - Clustering and Mixture Models
7. **columbia.edu** - Gaussian Mixture Models
8. **kent.edu** - Cluster Validation
9. **towardsdatascience** - Clustering Analysis
10. **analyticsvidhya** - Feature Selection methods