

ML 無監督式學習

(Machine Learning, Unsupervised Learning)

利用 KMeans 將花(Iris)分成三種聚類
point fit 到模型裡面後，scatterplot 呈現出結果

```
script.py
1 # Import KMeans
2 from sklearn.cluster import KMeans
3
4 # Create a KMeans instance with 3 clusters: model
5 model = KMeans(n_clusters=3)
6
7 # Fit model to points
8 model.fit(points)
9
10 # Determine the cluster labels of new_points: labels
11 labels = model.predict(new_points)
12
13 # Print cluster labels of new_points
14 print(labels)
15
```

Run Code Submit Answer



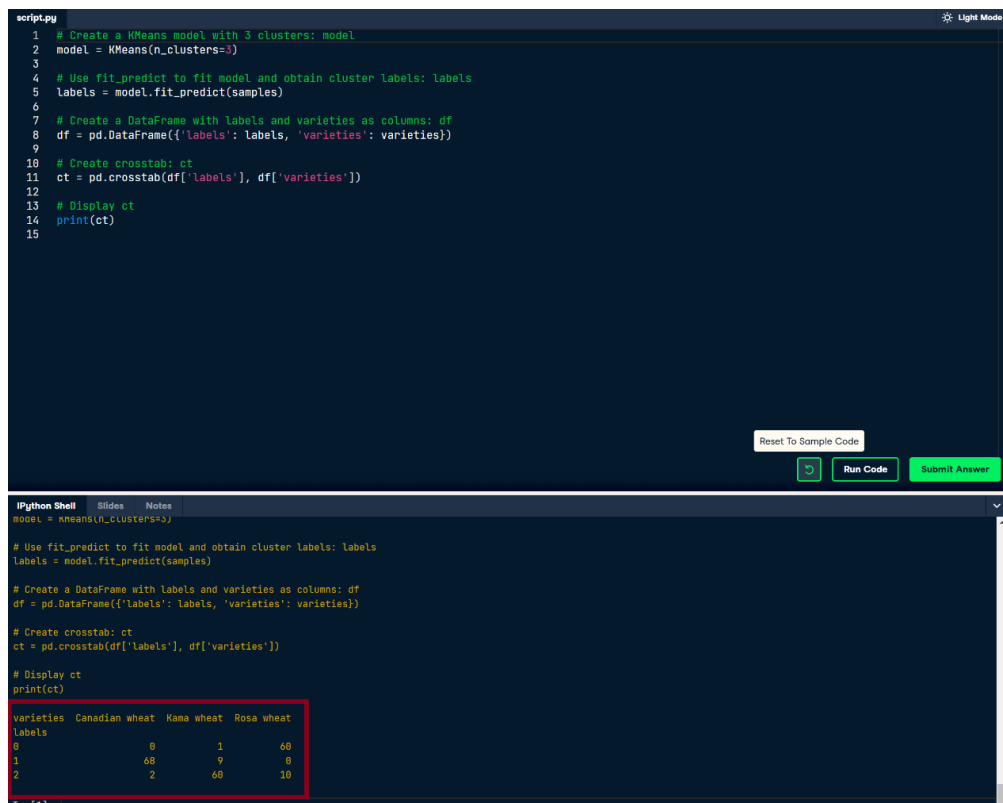
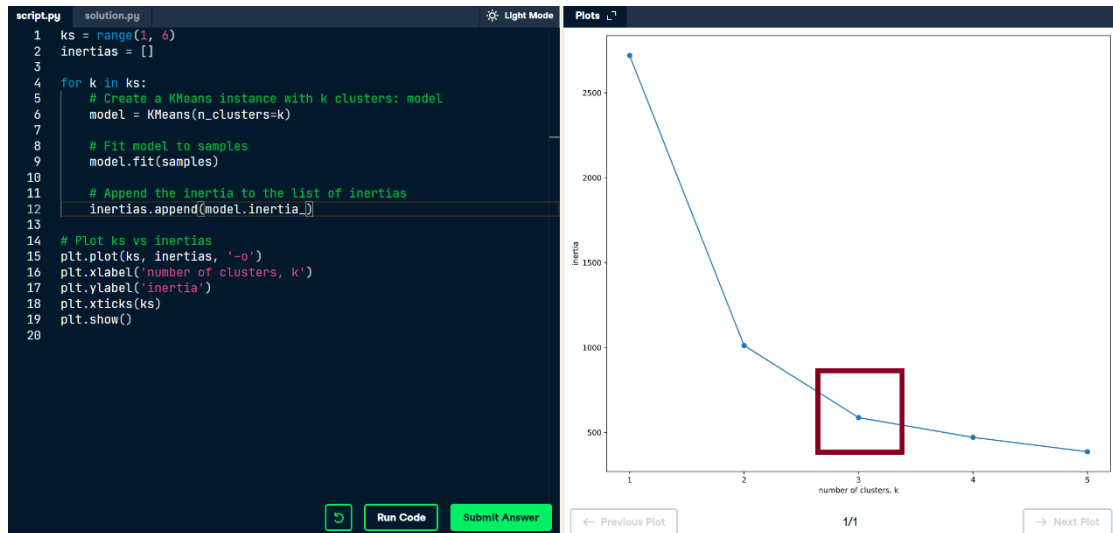
評估聚類 Evaluating a Clustering

慣性(Inertia)評估最佳聚類數量(KMeans)

慣性越小聚類越密集，而 KMeans 開始緩降為最佳解(紅框)

下圖，利用交叉表(cross-tabulation)評估聚類分布狀況(紅框)

評估花及品種是否有對應



優化調整類組 Transforming Features for Better Clusterings

假設交叉表分布沒有對應

先用 StandardScaler 或 Normalizer 與 KMeans 建構一個 pipeline

即可將資料標準化在分組，調整資料

下圖為優化後的聚類，交叉表(魚的品種)相互對應

```
script.py
1 # Perform the necessary imports
2 from sklearn.pipeline import make_pipeline
3 from sklearn.preprocessing import StandardScaler
4 from sklearn.cluster import KMeans
5
6 # Create scaler: scaler
7 scaler = StandardScaler()
8
9 # Create KMeans instance: kmeans
10 kmeans = KMeans(n_clusters=4)
11
12 # Create pipeline: pipeline
13 pipeline = make_pipeline(scaler, kmeans)
14
```

```
script.py evolution.py
1 # Import Normalizer
2 from sklearn.preprocessing import Normalizer
3
4 # Create a normalizer: normalizer
5 normalizer = Normalizer()
6
7 # Create a KMeans model with 10 clusters: kmeans
8 kmeans = KMeans(n_clusters=10)
9
10 # Make a pipeline chaining normalizer and kmeans: pipeline
11 pipeline = make_pipeline(normalizer, kmeans)
12
13 # Fit pipeline to the daily price movements
14 pipeline.fit(movements)
15
16
```

Run Code Submit Answer

```
script.py
1 # Import pandas
2 import pandas as pd
3
4 # Fit the pipeline to samples
5 pipeline.fit(samples)
6
7 # Calculate the cluster labels: labels
8 labels = pipeline.predict(samples)
9
10 # Create a DataFrame with labels and species as columns: df
11 df = pd.DataFrame({'labels' : labels, 'species' : species})
12
13 # Create crosstab: ct
14 ct = pd.crosstab(df['labels'], df['species'])
15
16 # Display ct
17 print(ct)
18
```

Run Code Submit Answer

IPython Shell Slides Notes

```
# Calculate the cluster labels: labels
labels = pipeline.predict(samples)

# Create a DataFrame with labels and species as columns: df
df = pd.DataFrame({'labels' : labels, 'species' : species})

# Create crosstab: ct
ct = pd.crosstab(df['labels'], df['species'])

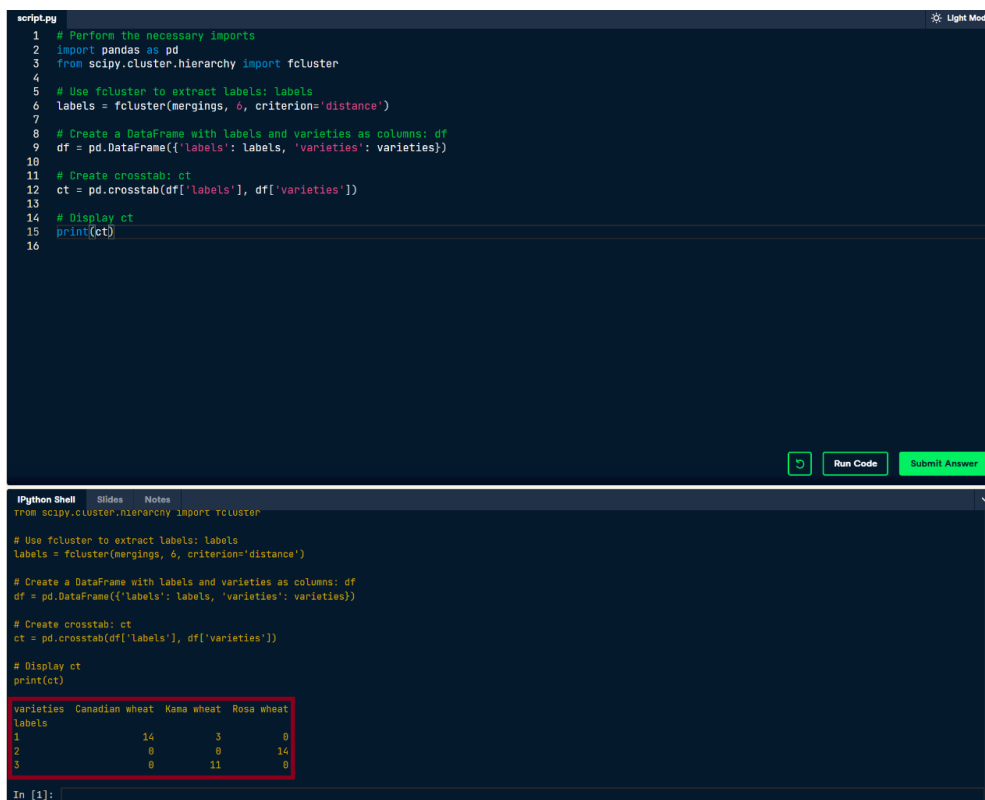
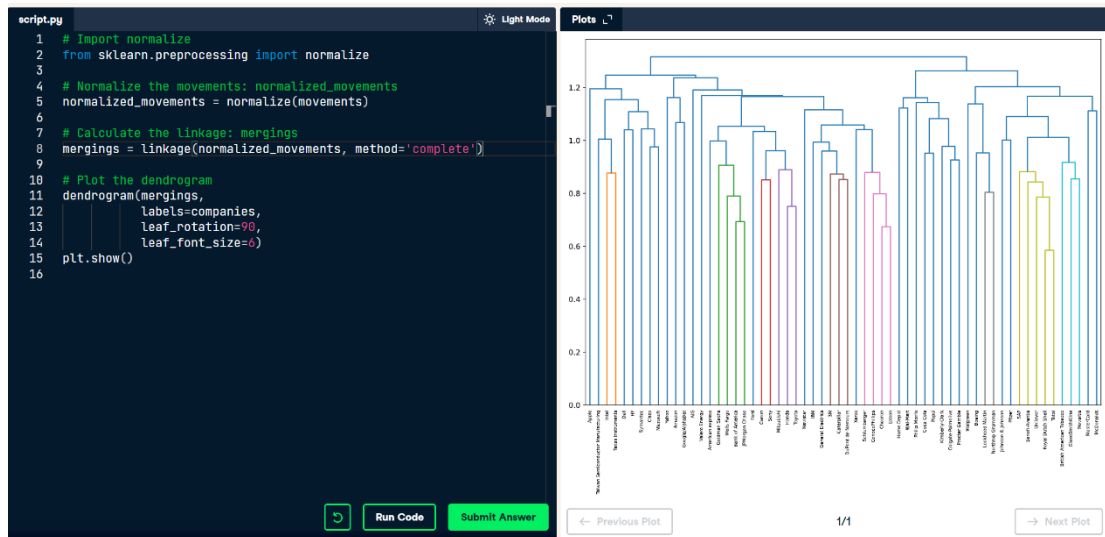
# Display ct
print(ct)
```

species	Bream	Pike	Roach	Smelt
0	0	0	0	13
1	33	0	1	0
2	0	17	0	0
3	1	0	19	1

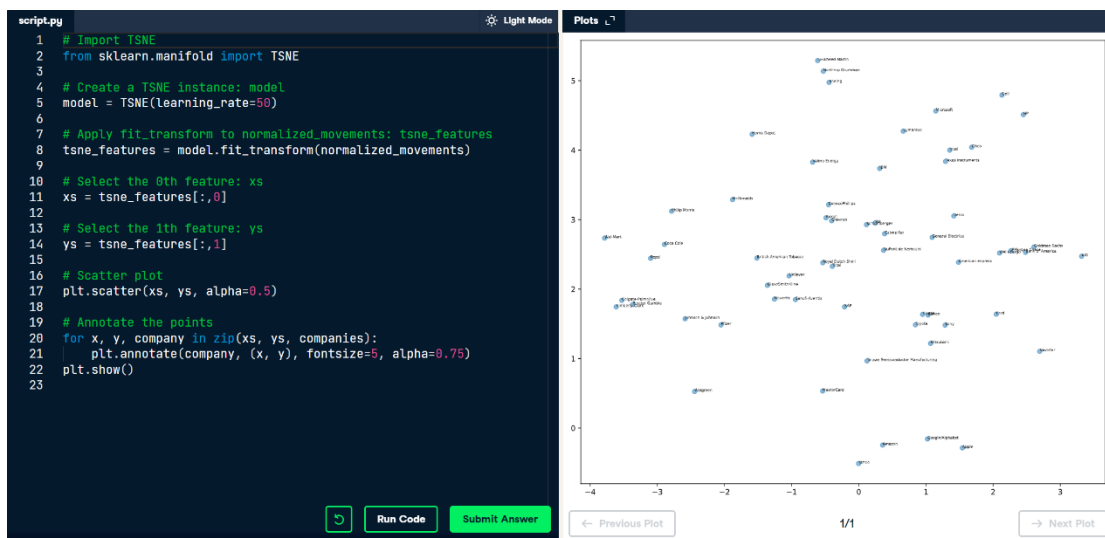
t-SNE 與層次聚類視覺化

Visualization with Hierarchical Clustering and t-SNE

Normalize 資料後用 dendrogram 視覺化
可用 fcluster 提取特定高度的聚類(下圖高度為 6)



t-SNE 視覚化



PCA 降維 Decorrelating Data and Dimension Reduction

先用皮爾森(Pearson Correlation)做 scatter plot

接著 PCA 降維(下圖)



內在維度 Intrinsic Dimension

資料先 PCA(箭頭為 Pearson 的值>0)

再做 StandardScaler 跟 PCA 的 pipeline

可看出 X=2 (下圖紅線右邊) PCA 特徵沒有顯著差異

表示內在維度為 2

X=2,3 沒有顯著差異，可用 PCA 降維至 2 (PCA(n_components=2))



PCA 降維，做 Pipeline (TruncatedSVD & KMeans)
再帶入得到結果(下圖)
PCA 維度降到 50，總共 6 組 (資料為維基百科熱門標題文章)

```
script.py solution.py
1 # Perform the necessary imports
2 from sklearn.decomposition import TruncatedSVD
3 from sklearn.cluster import KMeans
4 from sklearn.pipeline import make_pipeline
5
6 # Create a TruncatedSVD instance: svd
7 svd = TruncatedSVD(n_components=50)
8
9 # Create a KMeans instance: kmeans
10 kmeans = KMeans(n_clusters=6)
11
12 # Create a pipeline: pipeline
13 pipeline = make_pipeline(svd, kmeans)
14
```

Run Code Submit Answer

```
script.py solution.py
1 # Import pandas
2 import pandas as pd
3
4 # Fit the pipeline to articles
5 pipeline.fit(articles)
6
7 # Calculate the cluster labels: labels
8 labels = pipeline.predict(articles)
9
10 # Create a DataFrame aligning labels and titles: df
11 df = pd.DataFrame({'label': labels, 'article': titles})
12
13 # Display df sorted by cluster label
14 print(df.sort_values('label'))
15
```

Run Code Run Solution

Python Shell	Slides	Notes
1	4	Alexa Internet
2	4	Internet Explorer
3	4	HTTP cookie
4	4	Google Search
5	4	Tumblr
6	4	Hypertext Transfer Protocol
7	4	Social search
49	4	Lymphoma
42	4	Doxycycline
47	4	Fever
46	4	Prednisone
44	4	Gout
43	4	Leukemia
9	4	LinkedIn
48	4	Gabapentin
8	4	HTTP 404
45	5	Hepatitis C
41	5	Hepatitis B
40	5	Tonsillitis

非負矩陣降維 NMF

資料為調查安海瑟薇跟丹佐華盛頓的文章特性

下圖紅框可看出 3 值最高

也就是 3 為主要重建(construct)文章的部分

```
script.py
1 # Import NMF
2 from sklearn.decomposition import NMF
3
4 # Create an NMF instance: model
5 model = NMF(n_components=6)
6
7 # Fit the model to articles
8 model.fit(articles)
9
10 # Transform the articles: nmf_features
11 nmf_features = model.transform(articles)
12
13 # Print the NMF features
14 print(nmf_features.round(2))
15
```

```
script.py
1 # Import pandas
2 import pandas as pd
3
4 # Create a pandas DataFrame: df
5 df = pd.DataFrame(nmf_features, index=titles)
6
7 # Print the row for 'Anne Hathaway'
8 print(df.loc['Anne Hathaway'])
9
10 # Print the row for 'Denzel Washington'
11 print(df.loc['Denzel Washington'])
12
```

```
IPython Shell Slides Notes
print(df.loc['Anne Hathaway'])

# Print the row for 'Denzel Washington'
print(df.loc['Denzel Washington'])

0 0.004
1 0.000
2 0.000
3 0.576
4 0.000
5 0.000
Name: Anne Hathaway, dtype: float64
0 0.000
1 0.006
2 0.000
3 0.422
4 0.000
5 0.000
Name: Denzel Washington, dtype: float64
```

用 `iloc` 把 3 的資料提取出來
發現安海瑟威跟丹佐華盛頓的文章
主要都具有以下特徵 (紅框)

```
script.py
1 # Import pandas
2 import pandas as pd
3
4 # Create a DataFrame: components_df
5 components_df = pd.DataFrame(model.components_, columns=words)
6
7 # Print the shape of the DataFrame
8 print(components_df.shape)
9
10 # Select row 3: component
11 component = components_df.iloc[3]
12
13 # Print result of nlargest
14 print(component.nlargest())
15
```

Run Code Submit Answer

```
IPython Shell Slides Notes
# Create a DataFrame: components_df
components_df = pd.DataFrame(model.components_, columns=words)

# Print the shape of the DataFrame
print(components_df.shape)

# Select row 3: component
component = components_df.iloc[3]

# Print result of nlargest
print(component.nlargest())

(6, 13125)
film      0.628
award     0.253
starred   0.245
role      0.211
actress   0.186
Name: 3, dtype: float64
```

電影	0.628
獲獎	0.253
主角	0.245
角色	0.211
演員	0.186

利用 NMF 建置推薦系統

(資料為先前使用的熱門維基百科文章)

資料標準化後，利用.dot (餘弦定理)找出跟西羅有相關的文章(紅框)

下兩圖概念相同

根據喜愛的音樂家，利用.dot，推薦有相關聯的音樂家

```
script.py
1 # Perform the necessary imports
2 import pandas as pd
3 from sklearn.preprocessing import normalize
4
5 # Normalize the NMF features: norm_features
6 norm_features = normalize(nmf_features)
7
8 # Create a DataFrame: df
9 df = pd.DataFrame(norm_features, index= titles)
10
11 # Select the row corresponding to 'Cristiano Ronaldo': article
12 article = df.loc['Cristiano Ronaldo']
13
14 # Compute the dot products: similarities
15 similarities = df.dot(article)
16
17 # Display those with the largest cosine similarity
18 print(similarities.nlargest())
```

Python Shell Slides Notes

```
# Create a DataFrame: df
df = pd.DataFrame(norm_features, index= titles)

# Select the row corresponding to 'Cristiano Ronaldo': article
article = df.loc['Cristiano Ronaldo']

# Compute the dot products: similarities
similarities = df.dot(article)

# Display those with the largest cosine similarity
print(similarities.nlargest())
```

Cristiano Ronaldo	1.0
Franck Ribéry	1.0
Radamel Falcao	1.0
Zlatan Ibrahimović	1.0
France national football team	1.0

dtype: float64

```
script.py
1 # Perform the necessary imports
2 from sklearn.decomposition import NMF
3 from sklearn.preprocessing import Normalizer, MaxAbsScaler
4 from sklearn.pipeline import make_pipeline
5
6 # Create a MaxAbsScaler: scaler
7 scaler = MaxAbsScaler()
8
9 # Create an NMF model: nmf
10 nmf = NMF(n_components= 20)
11
12 # Create a Normalizer: normalizer
13 normalizer = Normalizer()
14
15 # Create a pipeline: pipeline
16 pipeline = make_pipeline(scaler, nmf, normalizer)
17
18 # Apply fit_transform to artists: norm_features
19 norm_features = pipeline.fit_transform(artists)
20
```

```
script.py
1 # Import pandas
2 import pandas as pd
3
4 # Create a DataFrame: df
5 df = pd.DataFrame(norm_features, index= artist_names)
6
7 # Select row of 'Bruce Springsteen': artist
8 artist = df.loc['Bruce Springsteen']
9
10 # Compute cosine similarities: similarities
11 similarities = df.dot(artist)
12
13 # Display those with highest cosine similarity
14 print(similarities.nlargest())
15
```

Python Shell Slides Notes

```
# Create a DataFrame: df
df = pd.DataFrame(norm_features, index= artist_names)

# Select row of 'Bruce Springsteen': artist
artist = df.loc['Bruce Springsteen']

# Compute cosine similarities: similarities
similarities = df.dot(artist)

# Display those with highest cosine similarity
print(similarities.nlargest())
```

Bruce Springsteen	1.000
Neil Young	0.954
Van Morrison	0.872
Leonard Cohen	0.865
Bob Dylan	0.859

dtype: float64