UNSUPERVISED LEARNING

K-Means clustering

```
from sklearn.cluster import KMeans
mod = KMeans(n_clusters=N)
mod.fit(data)
```

→ Para ver el centro de los clusters ⇒ `mod.cluster_centers_`

Para evaluar un cluster → Usando inertia (como de dispersos están los datos en cada cluster) → `mod.inertia_` → Hay que minimizarla sin usar muchos clusters

Si trasformo los datos con StandardScaler funciona mejor para clusterizar →

```
from sklearn.preprocessing import StandardScaler
scaler = S...()
scaler.fit(samples)
StandardScaler(copy=True, with_mean=True, with_std=True)
samples_listos = scaler.transform(samples)
```

Visualizaciones para unsupervised learning

Hierarchical clustering →

```
from scipy.cluster.hierarchy import linkage, dendrogram
merging = linkage(samples, method='complete')
dendrogram(merging, labels= ..., leaf_rotation=90, ...)
plt.show()
```

Para sacar clusters a partir de una altura del dendrograma → (se corresponde con la distancia entre 2 clusters)

```
from scipy.cluster.hierarchy import fcluster
labels = fcluster(mergings, altura, criterion='distance')
```

t-SNE →

```
from sklearn.manifold import TSNE
mod = TSNE(learning_rate=100)
transformed = mod.fit_transform(samples)
xs = transformed[...
ys ...
plt.scatter(... , c=...)
```

Dimension Reduction

PCA → 'Principal Component Analysis' →

```
from sklearn.decomposition import PCA
mod = PCA()
mod.fit(samples)
transformed_samples = mod.transform(samples)
```

→ Los principal components → direcciones de la varianza

Para verlos → `mod.components_`

- Reduce la dimensión
- Elimina la correlación
- Centra los datos a los ejes

Para identificar dimensiones intrínsecas (mínimas para datos OK) → Se puede ver a partir de la varianza de las features de PCA ⇒

```
pca = PCA()
pca.fit(samples)
features = pca.n_components_
varianzas = pca.explained_variance_
```

→ Sin embargo, los resultados son ambiguos.

Para realizar la reducción → `PCA(n_components=N)` → Asume que a ↑ varianza ↑ importancia

Si es Sparse (Muchos ceros) → `TruncatedSVD(n_components=N)`

csr_matrix

UNSUPERVISED LEARNING

Como PCA
pero mas interpretable

Dimension Reduction

NMF -> 'Non-negative matrix factorization'

```
from sklearn.decomposition import NMF
mod = NMF (n_components = N)
mod.fit (samples)
transformados = mod.transform (samples)
```

Limitacion -> valores >= 0

Sistemas de recomendacion con NMF

PASO 1 -> Uso NMF par obtener features

```
nmf = NMF(n_components= .. )
nmf_features = nmf.fit_transform(samples)
```

PASO 2 -> Comparacion entre features

```
from sklearn.preprocessing import normalize
norm_fea = normalize(nmf_features)
df = pd.D...  (norm_fea , index = titulos)
valor_act = df.loc['.....']
similarities = df.dot ( valor_actual )
similarities.nlargest()
```

$\rightarrow$ Calcula la distancia coseno con el resto de valores del dataframe

$\rightarrow$ Articulos similares