Introducción al aprendizaje no supervisado

Alfonso Tobar Arancibia
Data Scientist
09-11-2020


github.com/Rcubes/clases-ml/Slides

Aprendizaje no Supervisado

A diferencia de los modelos vistos hasta ahora. El aprendizaje no supervisado no contiene un vector objetivo/variable de respuesta y, es decir no hay un conocimiento a priori de los datos. Por lo tanto estos modelos no tienen una guía a la cual converger.

Un ejemplo de aprendizaje no supervisado que ya hemos revisado es PCA, que es un procedimiento de Reducción de Dimensionalidad no supervisado que buscar maximizar la varianza explicada a través de las componentes principales.

Otros ejemplos:

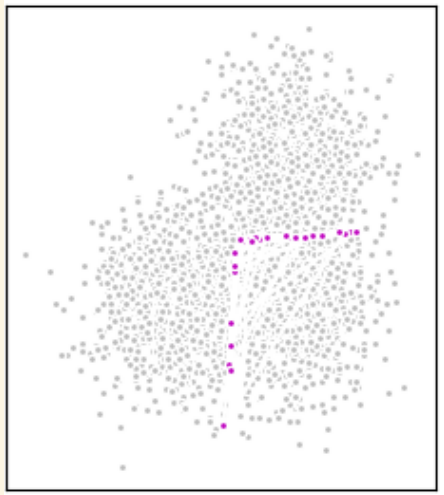
- KMeans
- Gaussian Mixtures
- LDA (Latent Dirichlet Allocation)

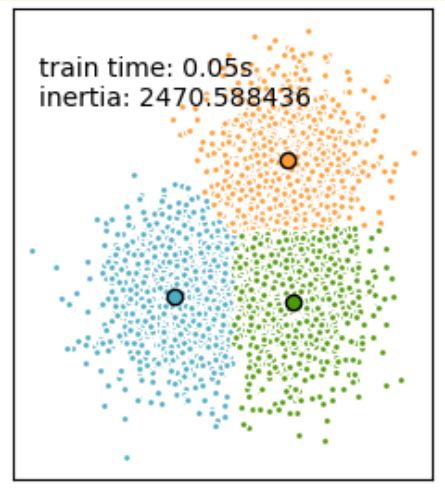
K-Means

Es un algoritmo muy sencillo que que encuentra K grupos de Datos minimizando la inercia entre un centroide dado y el resto de valores dentro del grupo. La inercia se define como:

$$\sum_{\mu_j \in C}^{n} ||x_i - \mu_j||^2$$

Estos modelos entonces buscaran segmentar y generar clusters de datos que contengan cierta similaridad entre ellos, esta similaridad será medida por qué tan cercanos están.





Implementaición en Scikit-learn

```
from sklearn.cluster import KMeans
km = KMeans(n_clusters = 8)
km.fit(X)
Km.predict(X)
km.pit_predict(X)
```

Hiperparámetros

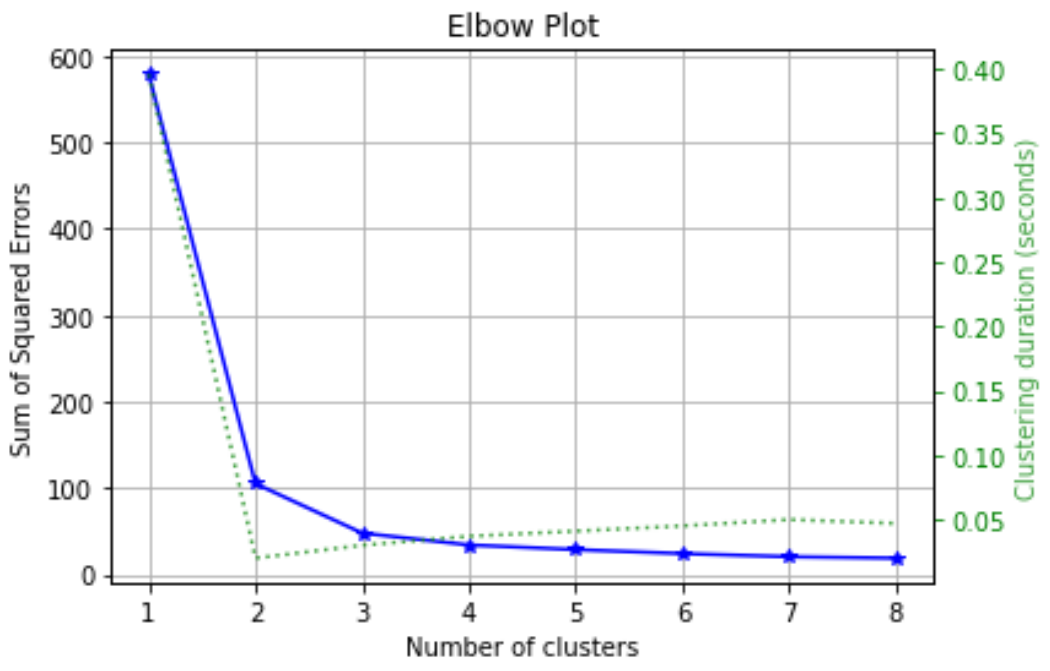
n_clusters: Número de Grupos a generar.

`.inertia`: Suma total de las distancias al cuadrado de cada punto con su centroide más cercano.

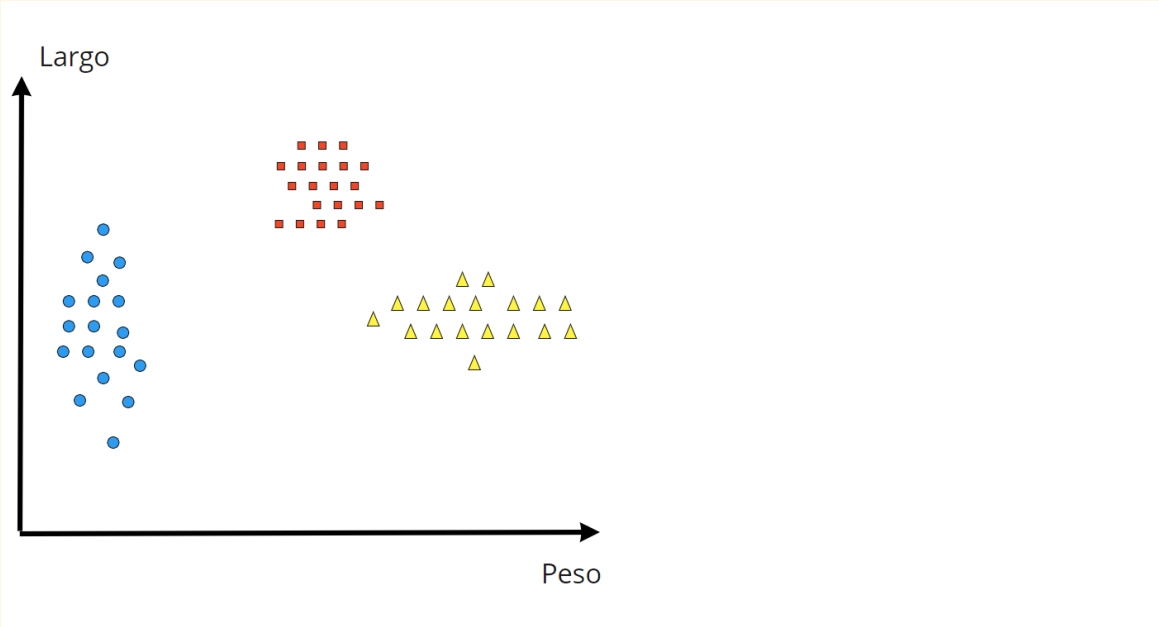
Elbow Plot

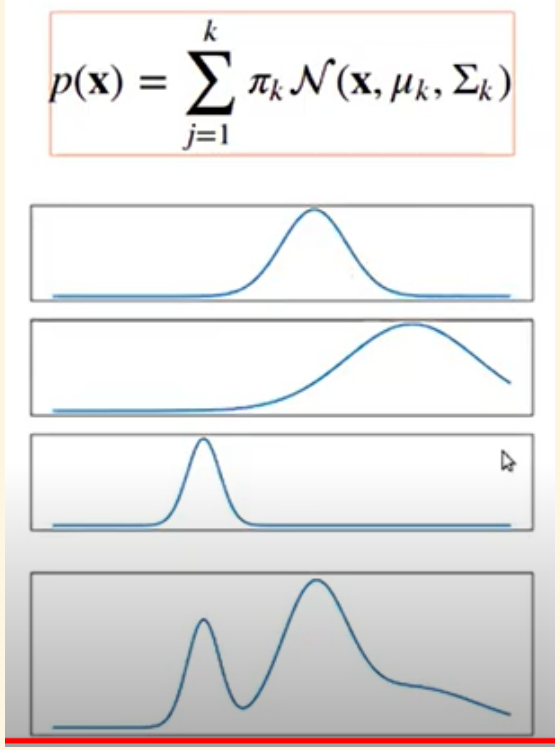
Permite encontrar el número óptimo de clusters.

```
import scikitplot as skplt
km = KMeans(random_state=123)
skplt.cluster.plot_elbow_curve(km, X, cluster_ranges=range(1, 9))
plt.show()
```



Gaussian Mixtures





- Ajusta una Distribución Normal multivariable a cada grupo y luego las combina para entregar una probabilidad de pertencer a cada grupo.

Implementación en Scikit-learn

```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components = 1, random_state = 123)
y_labels = gmm.fit_predict(X)
y_proba = gmm.predict_proba(X)
```

n_components: Número de Grupos a generar. **covariance_type**: Tipo de Matriz Covarianza.
`.bic(X)` permite calcular el bic para el Input X. Comparando BIC podría ser posible calcular qué modelo es óptimo.