Crea K grupos a partir de un conjunto de observaciones y caracteristicas , los elementos deben de tener similitudes.: Lo cque pueden llegar a compartir

**K-Means**

* Seleccionar el número de centroides o K
* Asignamos a cada uno de los elementos restantes al centro más cercano
* Asignamos cada punto a su centroide más cercano
* Repetimos paso 2 y 3 hasta que los centros no se modifiquen

**¿Como sabér cuántos centroides necesito?**

Utilizamos el **método del codo**

Calcula el agrupamiento para diferentes de K

El error al cuadrado para cada punto es el cuadrado de la distancia del punto a su centro

Todo lo va graficando en un eje “xy” y resulta un gráfico parecido a un codo

**Agrupando los datos**

Nuestro primer “Hola mundo” de machine learning es el conjunto de datos de “iris”

Tiene 3 grupos en particular: Virgínica, Versicolor y cetosa #50 imágenes de cada una

Estos grupos están clasificados mediante la medida de su largo y de sus pétalos y sépalos

**Pasamos a los pasos con código**

**Primero**

Importamos las librerías que vamos a usar para K-Means

import pandas as np

from sklearn.clusters import KMeans

from sklearn import datasets

import matplotlib.pyplot as plt

**Segundo**

Cargamos nuestro dataset

iris = datasets.load\_iris( )

**Tercero**

**Procedemos a separar los datos**

X\_iris = iris.data

Y\_iris = iris.data

Pero con K-Means es diferente: Aquí ya no lo entrenamos con X\_iris and Y\_iris.En **no supervisado** no entrenamos en base a una etiqueta, lo **entrenamos** **mediante similitudes** que tenemos.

Ahora sí haremos la separación de los datos

X= pd.DataFrame(iris.data, columns = ['Sepal Lenght', 'Sepal Width', 'Petal Lenght', 'Petal Width'])

Y = pd.DataFrame(iris.target, columns = ['Target'])

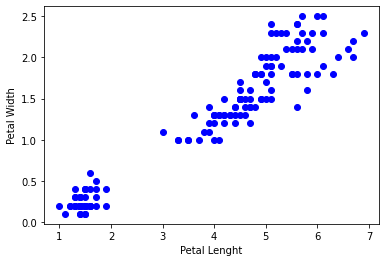
**Cuarto**

Procedemos a mostrar cómo se estarían relacionando los elementos. Para tomar un pequeño ejemplo, tomamos dos características y mostramos una gráfica para poder ver los elementos con sus similitudes.

plt.scatter(X['Petal Lenght'], X['Petal Width'], c = 'blue')

plt.xlabel('Petal Lenght', fontsize= 10)

plt.ylabel('Petal Width', fontsize = 10)



**Quinto**

Procedemos a construir nuestro modelo de K-means

model = KMeans(n\_clusters = 2, max\_iter = 1000)

#The number of clusters to form as well as the number of

centroids to generate

#max\_iter is how many times we are going to move every KMeans to find the less distance between each element and each KMeans

**Procedemos a entrenar nuestro modelo con “ .fit “**

model.fit(X)

# X son nuestros puntos en el gráfico y con ello lo estamos entrenando

Y\_labels = model.labels\_

**Ahora veremos cómo ha dividido nuestros datos**

y\_kmeans = model.predict(X)

print('Predicciones', y\_kmeans)

**Asimismo, veremos qué tan bueno es nuestro accuracy con 2 KMeans**

from sklearn import metrics

accuracy = metrics.adjusted\_rand\_score(Y\_iris, y\_labels)

print(accuracy)

0.5399218294207123

**Dado la deficiencia, aumentamos el número de KMeans a 3**

**0.7302382722834697** #Dado este resultado y visto que ,si usamos 4 KMeans, nos resulta también deficiente, nos quedamos con KMeans = 3

**Método del codo**

**from sklearn.cluster import KMeans**

**wcss = []**

**for i in range(1, 11):**

**kmeans = KMeans(n\_clusters=i, max\_iter=1000, random\_state=0)**

**kmeans.fit(x)**

**wcss.append(kmeans.inertia\_)**

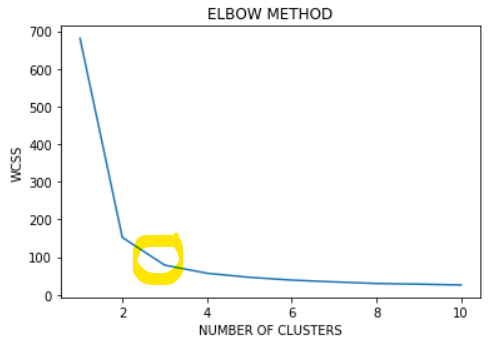
**plt.plot(range(1, 11), wcss)**

**plt.title('Elbow Method')**

**plt.xlabel('Number of clusters')**

**plt.ylabel('WCSS')**

**plt.show()**

****

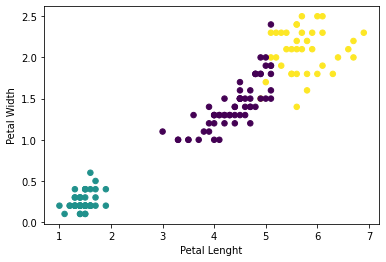
**Sexto**

Mostramos los puntos y los K\_means en un gráfico

**plt.scatter(X['Petal Lenght'], X['Petal Width'], c=y\_kmeans, s=30)**

**plt.xlabel('Petal Lenght', fontsize= 10)**

**plt.ylabel('Petal Width', fontsize= 10)**

****