from sklearn.cluster import KMeans

from sklearn import datasets

import pandas as pd

import matplotlib.pyplot as plt

iris = datasets.load\_iris()

X\_iris = iris.data

Y\_iris = iris.target

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

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

**# EXPLORACION**

|  |  |
| --- | --- |
| x.head(5) |  |
| plt.scatter(x['Petal Length'],   x['Petal Width'], c = 'blue')  plt.xlabel('Petal Length', fontsize = 10)  plt.ylabel('Petal Width', fontsize = 10)  plt.Text(0, 0.5, 'Largo Petalo') |  |

Parece que son 2 grupos, pero y que tal si en el grupo grande hay mas grupos?

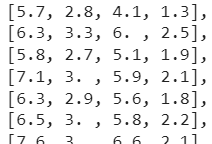
Apliquemos el metodo del codo

|  |  |
| --- | --- |
| 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() |  |

**# TARGET ???**

iris = datasets.load\_iris()

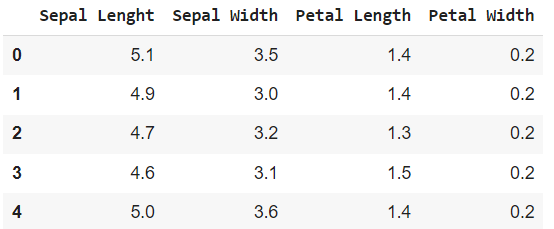
print(iris)



# DATA FOR TRAIN

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

x.head()



**Evaluación del modelo**

La evaluación será semi-supervisada puesto que usamos los datos del target del data set,

Comparamos nuestros resultados con los datos del target (accuracy).

El número de grupos (k) óptimo será el que mejor accuracy tenga.

**PARA ENCONTRAR EL K OPTIMO**

from sklearn.cluster import KMeans

from sklearn import datasets

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import metrics

iris = datasets.load\_iris()

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

DataToTrain = x #DATAFRAME

DataTarget = iris.target #ARRAY

K\_optimo = 0;

Mejor = 0

for K in range(2,8):

# Variamos K

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

#Se entrena modelo

model.fit(DataToTrain)

y\_labels = model.labels\_

#Creamos prediccion

y\_kmeans = model.predict(DataToTrain)

# Revisamos la precicion del modelo

accuracy = metrics.adjusted\_rand\_score(DataTarget, y\_kmeans)

print(K,' ',accuracy)

if accuracy > Mejor:

K\_optimo = K

Mejor = accuracy

print('')

print(f'Max accuracy: {round(Mejor,3)}% \nUsando K: {K\_optimo} ')