```
# EMILIO BERBER MALDONADO - A01640603
# ACT 4: KMEANS
# SEMANA TEC
```

K-means clustering

The notebook aims to study and implement a k-means clustering using "sklearn". A synthetic dataset will be used to identify clusters automatically using the K-means method.

Acknowledgments

- Inquiries: mauricio.antelis@tec.mx

Importing libraries

```
# Define where you are running the code: colab or local
RunInColab          = True     # (False: no  | True: yes)

# If running in colab:
if RunInColab:
    # Mount your google drive in google colab
    from google.colab import drive
    drive.mount('/content/drive')

    # Find location
    #!pwd
    #!ls
    #!ls "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPython/"

    # Define path del proyecto
    Ruta            = "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPython/"

else:
    # Define path del proyecto
    Ruta            = ""
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/
```

```
# Import the packages that we will be using
import numpy as np                  # For array
import pandas as pd                 # For data handling
import seaborn as sns               # For advanced plotting
import matplotlib.pyplot as plt     # For showing plots
```

```
# Note: specific functions of the "sklearn" package will be imported when needed to show co
```

Importing data

```
# Dataset url
url = "/content/drive/MyDrive/SyntheticData4Clustering_X.csv"

# Load the dataset
df  = pd.read_csv(url)
```

Undertanding and preprocessing the data

1. Get a general 'feel' of the data

```
# Print the dataframe
df
```

	x1	x2	x3	x4	x5	x6
0	1.914825	-1.380503	-3.609674	4.236011	-5.158681	5.712978
1	1.356415	9.767893	7.263659	8.750819	5.568930	-6.039122
2	1.185186	11.528344	9.999419	7.890027	7.308210	-8.899397
3	-1.739155	12.648965	7.965588	7.850296	10.235743	-10.175542
4	7.890985	-3.210880	-7.672016	2.438106	3.310904	-3.308334
...	...	...	...	...	...	...
1019	3.685106	-1.715503	-5.674443	6.510551	-0.121862	-6.166649
1020	-7.014173	-9.697874	4.093272	-0.590262	-9.882245	2.339336
1021	-2.993762	7.528182	7.877165	8.895835	9.318544	-7.445100
1022	4.576644	-1.720788	-6.581909	4.745839	1.497980	-4.828975
1023	2.616634	0.274593	-5.521864	9.582110	0.878266	-8.274990

1024 rows × 6 columns

```
xRows = df.shape[0]
xRows
```

```
1024
```

```
xCols = df.shape[1]
```

```
yCols = df.shape[1]
yCols
```

```
6
```

```
# get the number of observations and variables
df.shape
```

```
(1024, 6)
```

2. Drop rows with any missing values

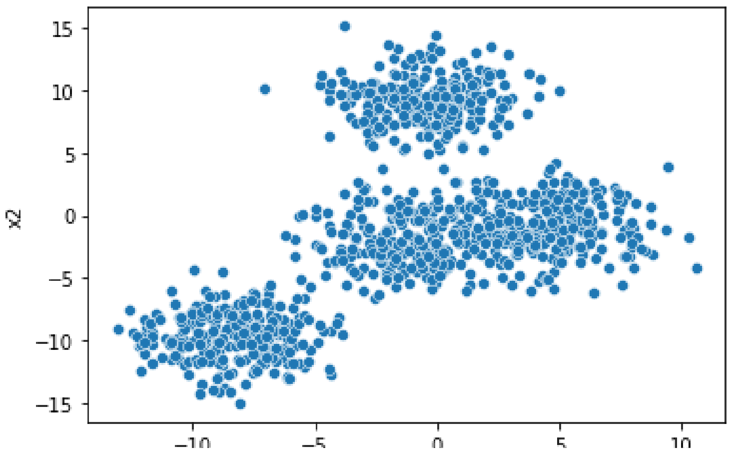
```
# Drop rows with NaN values if existing
df = df.dropna()

# Print the new shape
df
```

	x1	x2	x3	x4	x5	x6
0	1.914825	-1.380503	-3.609674	4.236011	-5.158681	5.712978
1	1.356415	9.767893	7.263659	8.750819	5.568930	-6.039122
2	1.185186	11.528344	9.999419	7.890027	7.308210	-8.899397
3	-1.739155	12.648965	7.965588	7.850296	10.235743	-10.175542
4	7.890985	-3.210880	-7.672016	2.438106	3.310904	-3.308334
...	...	...	...	...	...	...
1019	3.685106	-1.715503	-5.674443	6.510551	-0.121862	-6.166649
1020	-7.014173	-9.697874	4.093272	-0.590262	-9.882245	2.339336
1021	-2.993762	7.528182	7.877165	8.895835	9.318544	-7.445100
1022	4.576644	-1.720788	-6.581909	4.745839	1.497980	-4.828975
1023	2.616634	0.274593	-5.521864	9.582110	0.878266	-8.274990

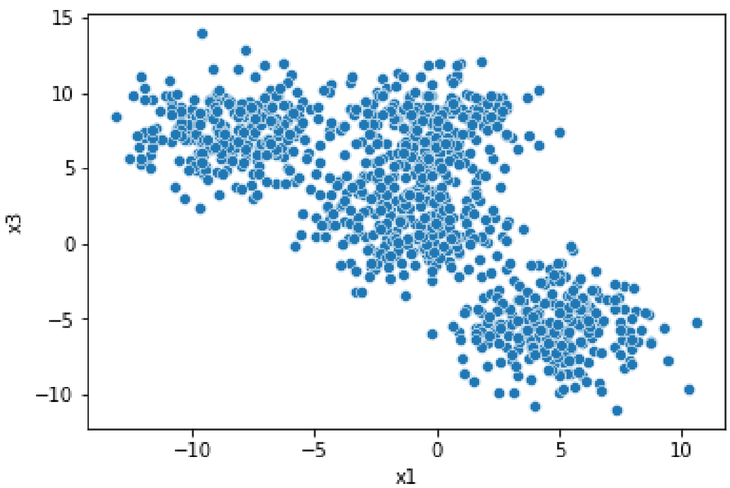
1024 rows × 6 columns

3. Scatterplot

```
# Scatterplot of x1 and x2
sns.scatterplot(data = df, x= "x1", y = "x2")
plt.show()
```

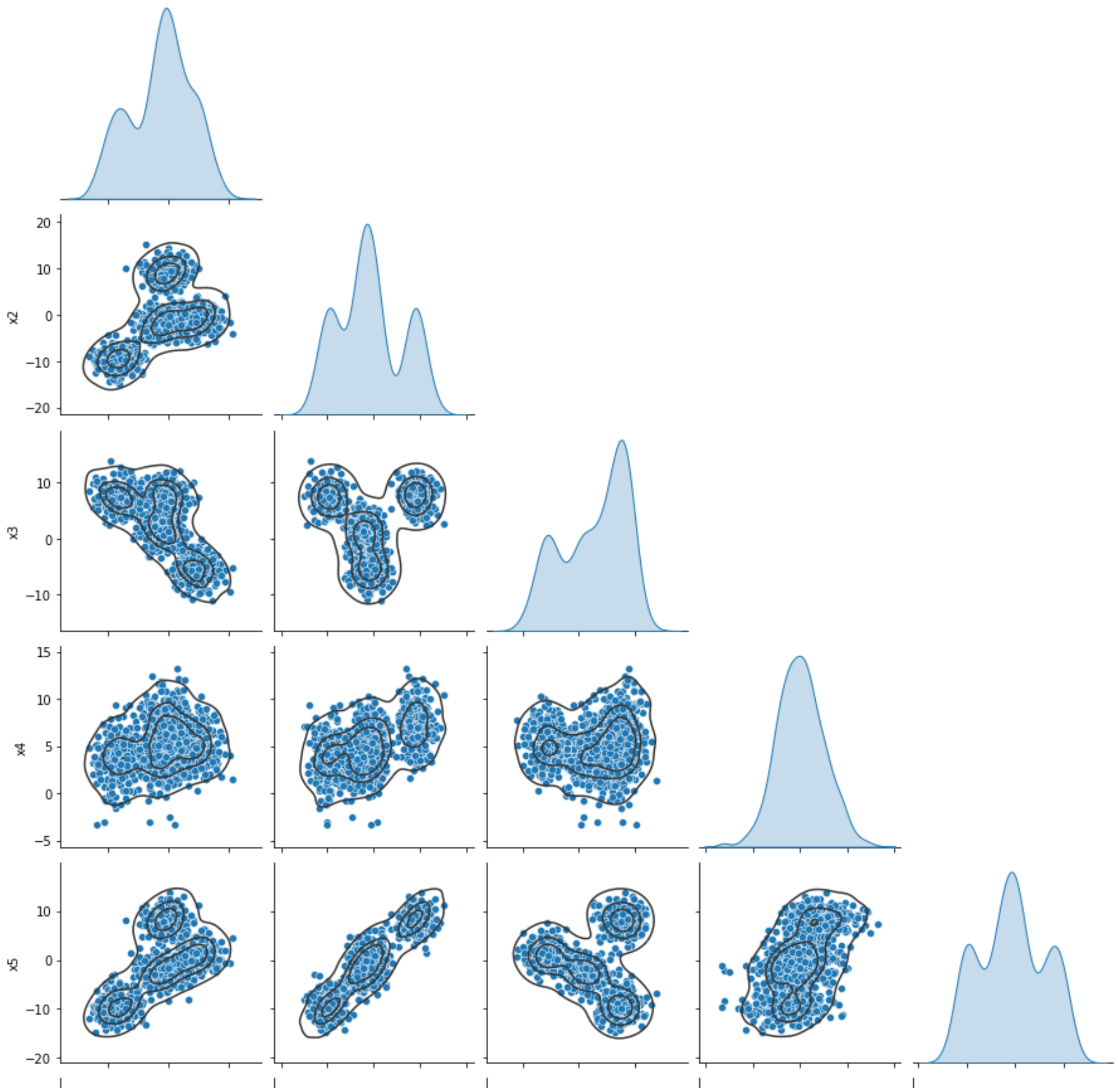



```
# Scatterplot of x1 and x3
sns.scatterplot(data = df, x= "x1", y = "x3")
plt.show()
```



Difficult to plot independetly all combinations, let's use pairplot

```
# Pairplot: Scatterplot of all variables
''' sns.pairplot(df)
plt.show() '''
g = sns.pairplot(df, corner = True, diag_kind="kde")
g.map_lower(sns.kdeplot, levels=4, color=".2")
plt.show()
```



It looks like there are 3 or 4 clusters/groups

Note that we do not know in advance the class/cluster/group to which each point belongs to: we need to apply unsupervised learning ¡

Kmeans clustering

Kmeans clustering

```
# Import sklearn KMeans
from sklearn.cluster import KMeans

# Define number of clusters
K  =  3 # Let's assume there are 2,3,4,5...? clusters/groups

# Create/initialize the Kmeans box/object
km = KMeans(n_clusters=K, n_init="auto")

# Do K-means clustering (assing each point in the dataset to a cluster)
```

```
yestimated = km.fit_predict(df)
```

```
# Print estimated cluster of each point in the dataset
yestimated
```

```
array([0, 2, 2, ..., 2, 0, 0], dtype=int32)
```

```
# Add a new column to the dataset with the cluster information
df['yestimated'] = yestimated
df
```

	x1	x2	x3	x4	x5	x6	yestimated
0	1.914825	-1.380503	-3.609674	4.236011	-5.158681	5.712978	0
1	1.356415	9.767893	7.263659	8.750819	5.568930	-6.039122	2
2	1.185186	11.528344	9.999419	7.890027	7.308210	-8.899397	2
3	-1.739155	12.648965	7.965588	7.850296	10.235743	-10.175542	2
4	7.890985	-3.210880	-7.672016	2.438106	3.310904	-3.308334	0
...	...	...	...	...	...	...	...
1019	3.685106	-1.715503	-5.674443	6.510551	-0.121862	-6.166649	0
1020	-7.014173	-9.697874	4.093272	-0.590262	-9.882245	2.339336	1
1021	-2.993762	7.528182	7.877165	8.895835	9.318544	-7.445100	2
1022	4.576644	-1.720788	-6.581909	4.745839	1.497980	-4.828975	0
1023	2.616634	0.274593	-5.521864	9.582110	0.878266	-8.274990	0

1024 rows × 7 columns

```
# Label of the estimated clusters
df.yestimated.unique()
```

```
array([0, 2, 1], dtype=int32)
```

```
# Cluster centroides
km.cluster_centers_
```

```
array([[ 1.85043266, -1.34592151, -2.11883656,  4.5718429 , -0.79519547,
        -0.55114018],
       [-8.3650671 , -9.59550917,  7.40711607,  3.77249056, -9.44226128,
         2.67666451],
       [-0.44229417,  9.13121533,  7.61409814,  7.22984721,  8.13001382,
        -7.6264221 ]])
```

```
# Sum of squared error (sse) of the final model
km.inertia_
```

```
44295.1263266536
```

```
# The number of iterations required to converge
km.n_iter_
```

```
18
```

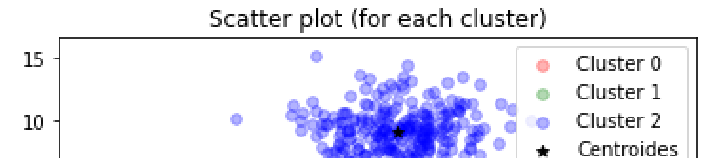
Important remarks

- The number of each cluster is randomly assigned
- The order of the number in each cluster is random

Plot estimated clusters

Plot estimated clusters

```
# Get a dataframe with the data of each clsuter
df1 = df[df.yestimated == 0]
df2 = df[df.yestimated == 1]
df3 = df[df.yestimated == 2]

# Scatter plot of each cluster
plt.scatter(df1.x1, df1.x2, label = "Cluster 0", c="r", marker = "o", s=32, alpha = 0.3)
plt.scatter(df2.x1, df2.x2, label = "Cluster 1", c="g", marker = "o", s=32, alpha = 0.3)
plt.scatter(df3.x1, df3.x2, label = "Cluster 2", c="b", marker = "o", s=32, alpha = 0.3)

# Plot centroides
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1], color = "black", marker = '*

plt.title("Scatter plot (for each cluster)")
plt.xlabel("x1")
plt.ylabel("x2")
plt.legend()
plt.show()
```

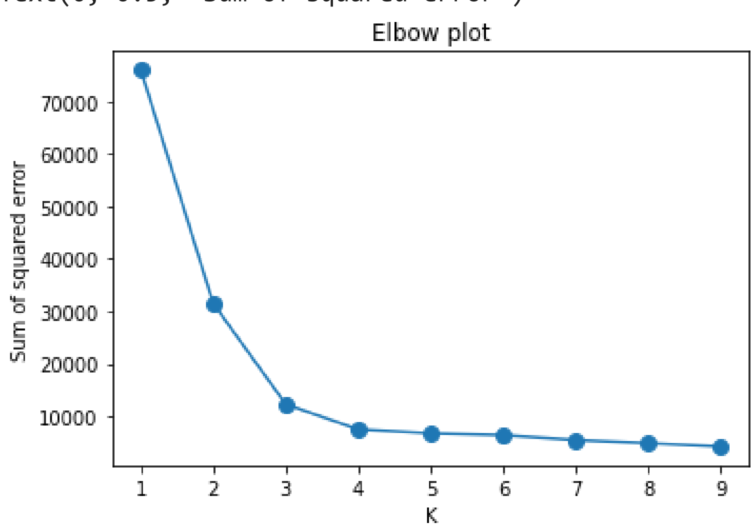


Selecting K: elbow plot

Check the acurracy of the model using k-fold cross-validation


```
# Intialize a list to hold sum of squared error (sse)
sse = []

# Define values of k
k_rng = range(1,10)

# For each k
for k in k_rng:
  # create model
  km = KMeans(n_clusters=k, n_init="auto")
  # Do K-means clustering
  km.fit_predict(df[["x1", "x2"]])
  # Save see for each k
  sse.append(km.inertia_)


#·Plot·sse·versus·k
plt.plot(k_rng,·sse,·'o-',·markersize·=·8)

plt.title("Elbow·plot")
plt.xlabel('K')
plt.ylabel("Sum·of·squared·error")
```

```
Text(0, 0.5, 'Sum of squared error')
```



Choose the k after which the sse is minimally reduced

Important remarks

- Observations?

Final remarks

- K-Means clustering algorithm is perhaps the simplest and most popular unsupervised learning algorithm
- The number of clusters have to be defined by the user (i.e., by you ¡¡)
- The number assigned to each cluster is randomly assigned from set 0, 1, 2
- If there is no information about the number of clusters k, then use the elbow plot method to choose the best number of clusters k
- The order of the number in each cluster is random
- The **sklearn** package provides the tools for data processing suchs as k-means

Activity:

1. Repeat this analysis using other pair of features, e.g., x3 and x6
2. Repeat this analysis using all six features, e.g., x1, x2,..., x6
3. Provide conclusions

Activity: work with the iris dataset

1. Do clustering with the iris flower dataset to form clusters using as features the four features
2. Do clustering with the iris flower dataset to form clusters using as features the two petal measurements: Drop out the other two features
3. Do clustering with the iris flower dataset to form clusters using as features the two sepal measurements: Drop out the other two features
4. Which one provides the better grouping? Solve this using programming skills, e.g., compute performance metrics

```
url = "/content/drive/MyDrive/iris.csv"

dfI = pd.read_csv(url, header = None)
dfI = dfI.rename(columns={0: "Largo_Sepalo"})
dfI = dfI.rename(columns={1: "Ancho_Sepalo"})
```

```
dfI = dfI.rename(columns={1: "Ancho_Sepalo"})
dfI = dfI.rename(columns={2: "Largo_Petalo"})
dfI = dfI.rename(columns={3: "Ancho_Petalo"})
dfI = dfI.rename(columns={4: "Especie"})
dfI
```

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

```
dfI.columns
```

```
Index(['Largo_Sepalo', 'Ancho_Sepalo', 'Largo_Petalo', 'Ancho_Petalo',
       'Especie'],
      dtype='object')
```

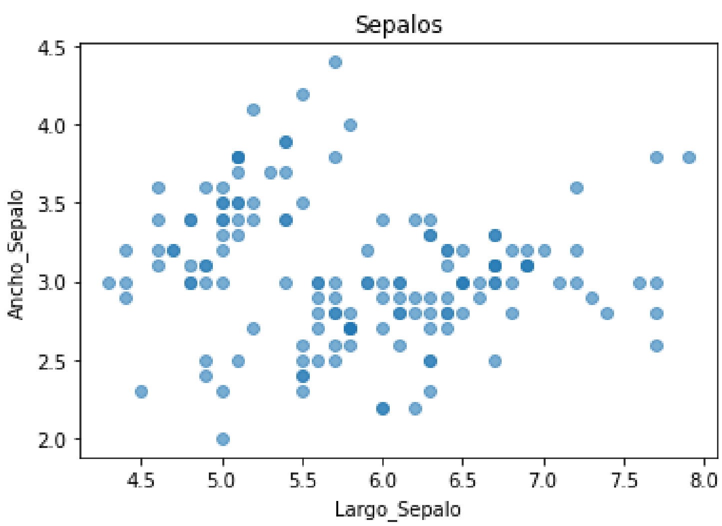
```
# get the number of observations and variables
dfI.shape
```

```
(150, 5)
```

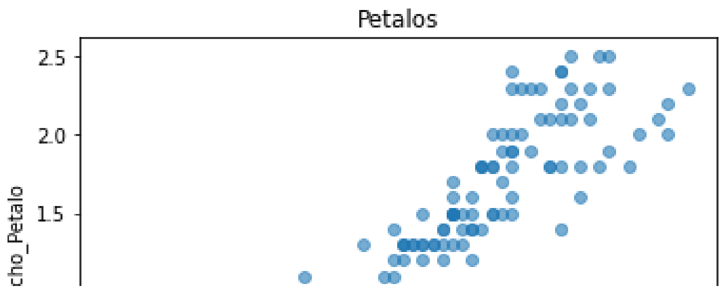
```
# Drop rows with NaN values if existing
dfI = dfI.dropna()

# Print the new shape
dfI
```

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica

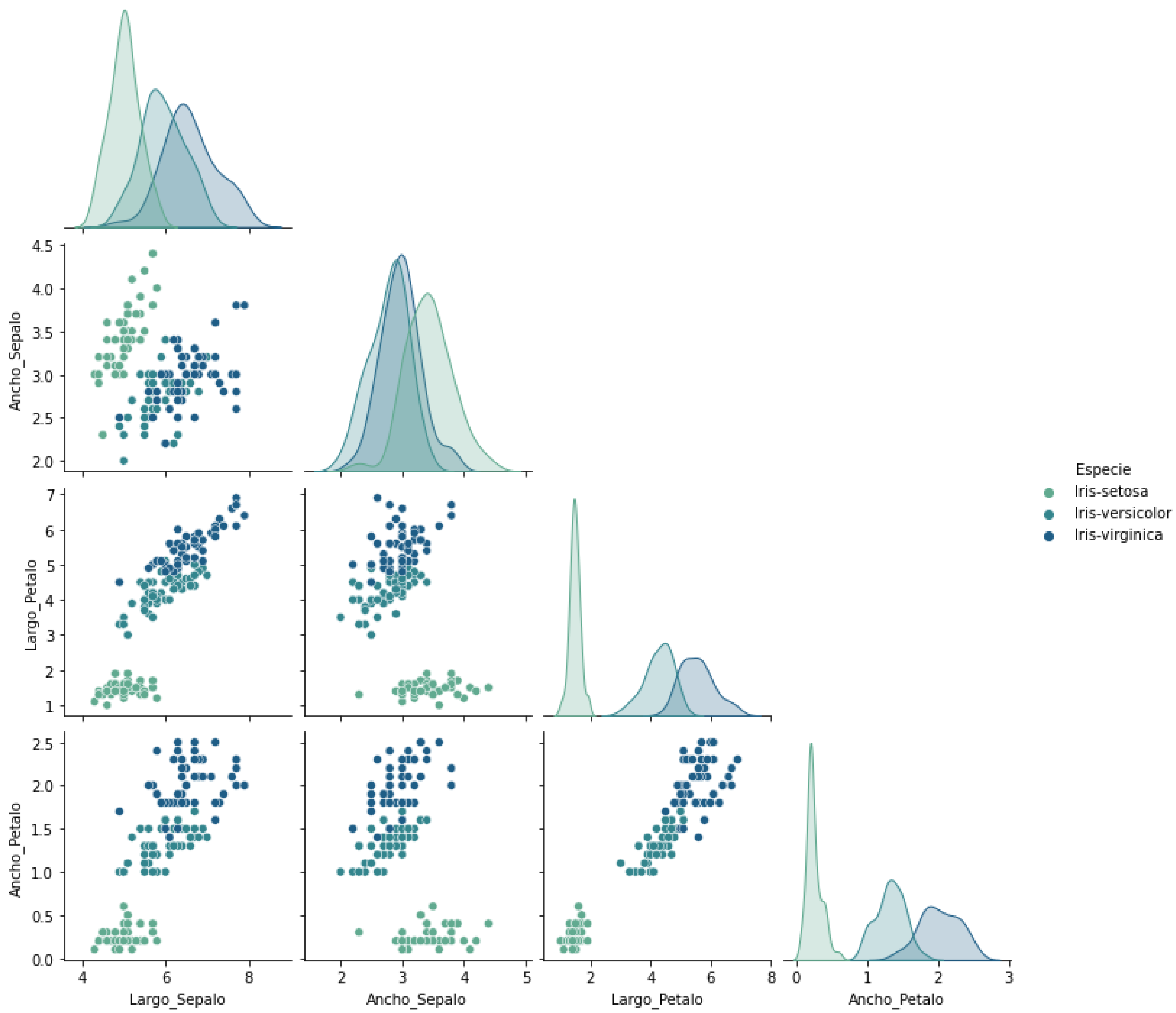
```
# Scatterplot of 0 and 1
plt.scatter(dfI.Largo_Sepalo, dfI.Ancho_Sepalo, alpha = .6)
plt.xlabel("Largo_Sepalo")
plt.ylabel("Ancho_Sepalo")
plt.title("Sepalos")
plt.show()
```




```
# Scatterplot of 0 and 2
plt.scatter(dfI.Largo_Petalo, dfI.Ancho_Petalo, alpha = .6)
plt.xlabel("Largo Petalo")
plt.ylabel("Ancho_Petalo")
plt.title("Petalos")
plt.show()
```




```
# Pairplot: Scatterplot of all variables
sns.pairplot(dfI, corner = True, diag_kind = "kde", hue = "Especie", palette="crest")
plt.show()
```




```
# Import sklearn KMeans
K = 4
km = KMeans(n_clusters = K, n_init="auto")
```

```
#Clustering
yestimated = km.fit_predict(dfI.iloc[:,0:4])
yestimated
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 2, 2, 2, 0, 2, 0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2,
       0, 0, 2, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 2, 2, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 3, 2, 3, 2, 3, 3, 0, 3, 3, 3,
       2, 2, 3, 2, 2, 2, 2, 3, 3, 2, 3, 2, 3, 2, 3, 3, 2, 2, 2, 3, 3, 3,
       2, 2, 2, 3, 3, 2, 2, 3, 3, 2, 2, 3, 3, 2, 2, 2, 2, 2], dtype=int32)
```

```
dfI.insert(5, "Clusters", yestimated, True)
#df.drop("Clusters", axis=1, inplace = True)
```

```
dfI.Clusters.unique()
```

```
array([1, 2, 0, 3], dtype=int32)
```

```
# Cluster centroides
kmc = km.cluster_centers_
kmc
```

```
array([[5.58      , 2.63333333, 3.98666667, 1.23333333],
       [5.006     , 3.428     , 1.462     , 0.246     ],
       [6.29361702, 2.9       , 4.95106383, 1.72978723],
       [7.08695652, 3.12608696, 6.01304348, 2.14347826]])
```

```
# Sum of squared error (sse) of the final model
km.inertia_
```

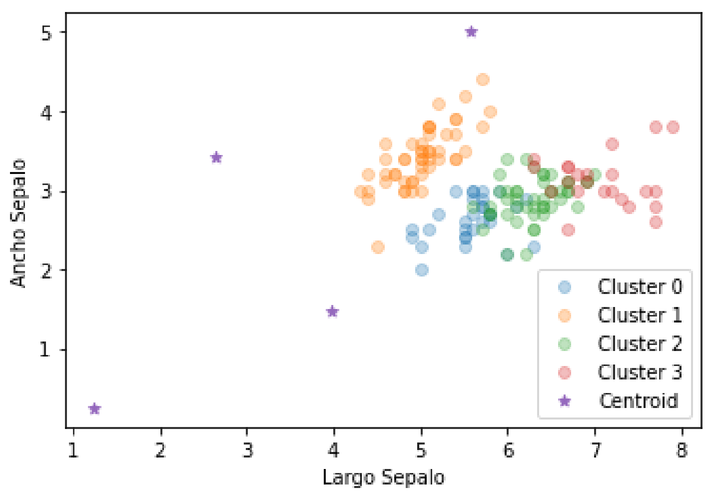
```
57.38387326549494
```

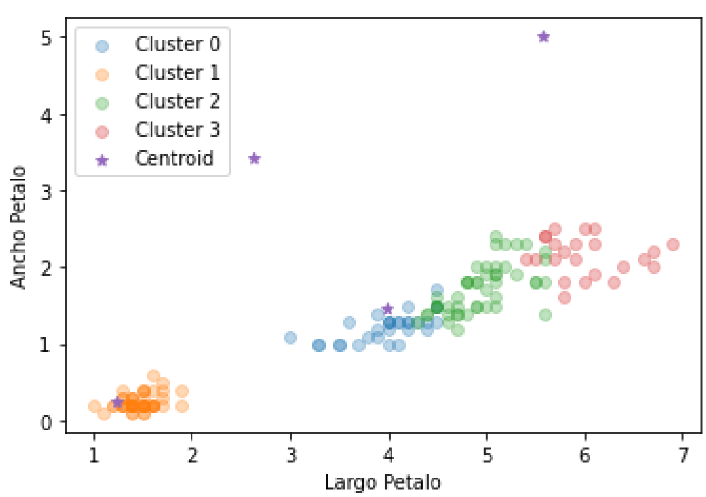
```
# The number of iterations required to converge
km.n_iter_
```

```
12
```

```
# Get a dataframe with the data of each clsuter
df1 = dfI[dfI.Clusters==0]
df2 = dfI[dfI.Clusters==1]
df3 = dfI[dfI.Clusters==2]
df4 = dfI[dfI.Clusters==3]

# Scatter plot of each cluster

kmc = km.cluster_centers_

plt.scatter(df1.Largo_Sepalo, df1.Ancho_Sepalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Sepalo, df2.Ancho_Sepalo, label="Cluster 1", alpha = .3)
```

```
plt.scatter(df3.Largo_Sepalo, df3.Ancho_Sepalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Sepalo, df4.Ancho_Sepalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Sepalo")
plt.ylabel("Ancho Sepalo")
plt.legend()
plt.show()

plt.scatter(df1.Largo_Petalo, df1.Ancho_Petalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Petalo, df2.Ancho_Petalo, label="Cluster 1", alpha = .3)
plt.scatter(df3.Largo_Petalo, df3.Ancho_Petalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Petalo, df4.Ancho_Petalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Petalo")
plt.ylabel("Ancho Petalo")

plt.legend()
plt.show()
```






```
dfI.drop("Clusters", axis=1, inplace = True)
dfI
```

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

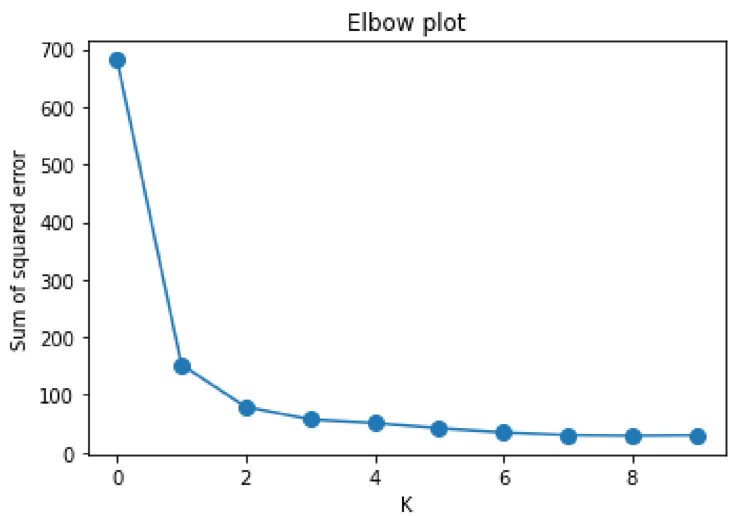
```
dfI
```

	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

```
# Intialize a list to hold sum of squared error (sse)
SSE = []
#Define K
K = [1,2,3,4,5,6,7,8,9,10]
#For each K
for x in K:
    tempkm = KMeans(n_clusters = x, n_init="auto")
    tempkm.fit_predict(dfI.iloc[:, 0:4])
```

```
    SSE.append(tempkm.inertia_)


# Plot sse vs K
plt.plot(range(0,10),SSE,'o-', markersize = 8)
plt.title("Elbow plot")
plt.xlabel('K')
plt.ylabel("Sum of squared error")
plt.show()
```




```
########### ACTIVITY
# 1
SSE = []
#Define K
K = [1,2,3,4,5,6,7,8,9,10]
#For each K
for x in K:
    tempkm = KMeans(n_clusters = x, n_init="auto")
    tempkm.fit_predict(dfI.iloc[:, 0:4])
    SSE.append(tempkm.inertia_)

plt.plot(range(0,10),SSE,'o-', markersize = 8)
plt.show()
```

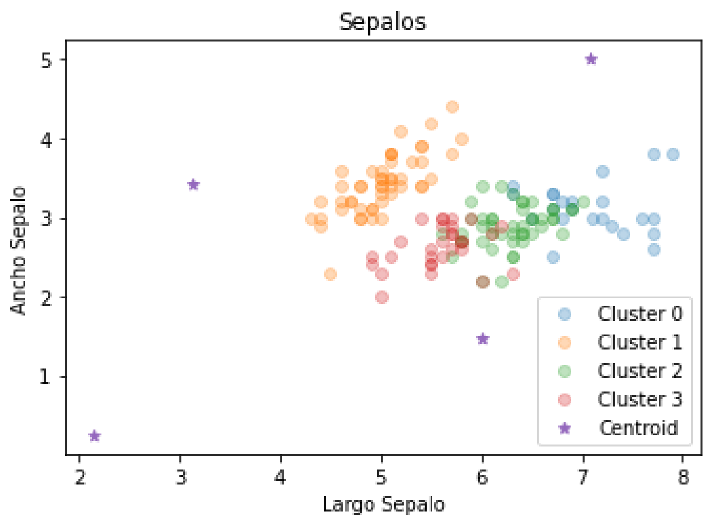

```
K = 4
km = KMeans(n_clusters = K, n_init="auto")

#Clustering
yestimated = km.fit_predict(dfI.iloc[:, 0:4])

#Add Column
dfI.insert(5, "Clusters", yestimated, True)
dfI
```

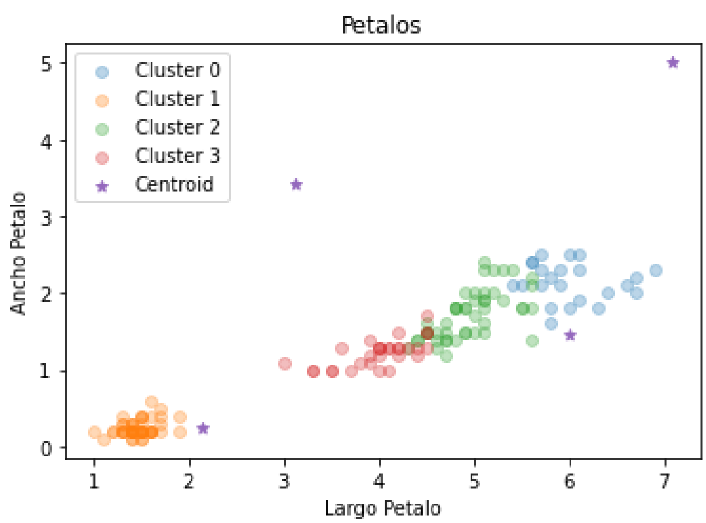
	Largo_Sepalo	Ancho_Sepalo	Largo_Petalo	Ancho_Petalo	Especie	Clusters
0	5.1	3.5	1.4	0.2	Iris-setosa	1
1	4.9	3.0	1.4	0.2	Iris-setosa	1
2	4.7	3.2	1.3	0.2	Iris-setosa	1
3	4.6	3.1	1.5	0.2	Iris-setosa	1
4	5.0	3.6	1.4	0.2	Iris-setosa	1
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	2
146	6.3	2.5	5.0	1.9	Iris-virginica	2
147	6.5	3.0	5.2	2.0	Iris-virginica	2
148	6.2	3.4	5.4	2.3	Iris-virginica	2
149	5.9	3.0	5.1	1.8	Iris-virginica	2

150 rows × 6 columns

```
#Dataframe
df1 = dfI[dfI.Clusters==0]
df2 = dfI[dfI.Clusters==1]
df3 = dfI[dfI.Clusters==2]
df4 = dfI[dfI.Clusters==3]

#Scatter clusters

kmc = km.cluster_centers_

plt.scatter(df1.Largo_Sepalo, df1.Ancho_Sepalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Sepalo, df2.Ancho_Sepalo, label="Cluster 1", alpha = .3)
plt.scatter(df3.Largo_Sepalo, df3.Ancho_Sepalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Sepalo, df4.Ancho_Sepalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Sepalo")
plt.ylabel("Ancho Sepalo")
```

```
plt.title("Sepalos")
plt.legend()
plt.show()

plt.scatter(df1.Largo_Petalo, df1.Ancho_Petalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Petalo, df2.Ancho_Petalo, label="Cluster 1", alpha = .3)
plt.scatter(df3.Largo_Petalo, df3.Ancho_Petalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Petalo, df4.Ancho_Petalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Petalo")
plt.ylabel("Ancho Petalo")
plt.title("Petalos")

plt.legend()
plt.show()
dfI.drop("Clusters", axis=1, inplace = True)
```





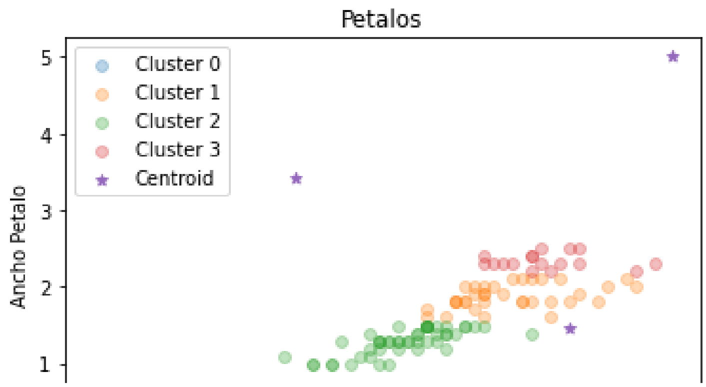

```
#2
K = 4
km = KMeans(n_clusters = K, n_init="auto")

#Clustering
```

```
yestimated = km.fit_predict(dfI.iloc[:, 3:4])

#Add Column
dfI.insert(5, "Clusters", yestimated, True)


df
```

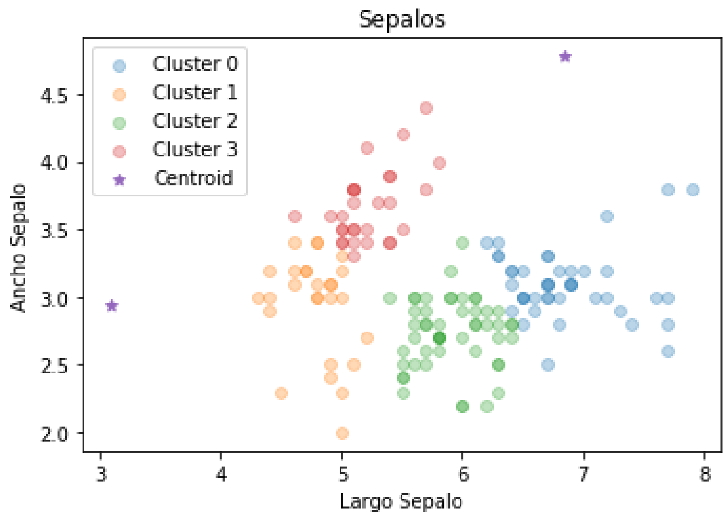
	x1	x2	x3	x4	x5	x6	yestimated
0	1.914825	-1.380503	-3.609674	4.236011	-5.158681	5.712978	0
1	1.356415	9.767893	7.263659	8.750819	5.568930	-6.039122	2
2	1.185186	11.528344	9.999419	7.890027	7.308210	-8.899397	2
3	-1.739155	12.648965	7.965588	7.850296	10.235743	-10.175542	2
4	7.890985	-3.210880	-7.672016	2.438106	3.310904	-3.308334	0
...	...	...	...	...	...	...	...
1019	3.685106	-1.715503	-5.674443	6.510551	-0.121862	-6.166649	0
1020	-7.014173	-9.697874	4.093272	-0.590262	-9.882245	2.339336	1
1021	-2.993762	7.528182	7.877165	8.895835	9.318544	-7.445100	2
1022	4.576644	-1.720788	-6.581909	4.745839	1.497980	-4.828975	0
1023	2.616634	0.274593	-5.521864	9.582110	0.878266	-8.274990	0

1024 rows × 7 columns

```
df1 = dfI[dfI.Clusters==0]
df2 = dfI[dfI.Clusters==1]
df3 = dfI[dfI.Clusters==2]
df4 = dfI[dfI.Clusters==3]

plt.scatter(df1.Largo_Petalo, df1.Ancho_Petalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Petalo, df2.Ancho_Petalo, label="Cluster 1", alpha = .3)
plt.scatter(df3.Largo_Petalo, df3.Ancho_Petalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Petalo, df4.Ancho_Petalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Petalo")
plt.ylabel("Ancho Petalo")
plt.title("Petalos")

plt.legend()
plt.show()
dfI.drop("Clusters", axis=1, inplace = True)
```




```
# 3
K = 4
km = KMeans(n_clusters = K, n_init="auto")

#Clustering
yestimated = km.fit_predict(dfI.iloc[:, 0:2])

#Add Column
dfI.insert(5, "Clusters", yestimated, True)


kmc = km.cluster_centers_

df1 = dfI[dfI.Clusters==0]
df2 = dfI[dfI.Clusters==1]
df3 = dfI[dfI.Clusters==2]
df4 = dfI[dfI.Clusters==3]

plt.scatter(df1.Largo_Sepalo, df1.Ancho_Sepalo, label="Cluster 0", alpha = .3)
plt.scatter(df2.Largo_Sepalo, df2.Ancho_Sepalo, label="Cluster 1", alpha = .3)
plt.scatter(df3.Largo_Sepalo, df3.Ancho_Sepalo, label="Cluster 2", alpha = .3)
plt.scatter(df4.Largo_Sepalo, df4.Ancho_Sepalo, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "*")
plt.xlabel("Largo Sepalo")
plt.ylabel("Ancho Sepalo")
plt.title("Sepalos")
plt.legend()
plt.show()
dfI.drop("Clusters", axis=1, inplace = True)
```



0 s se ejecutó 19:45