Lucas Wong Mang A01639032

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/Herramientas Computacionales/"

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

```
Mounted at /content/drive
```

```
# 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 concepts easily
```

Importing data

```
Ruta_General = "/content/drive/My Drive/Herramientas Computacionales/"
url = "SyntheticData4Clustering_X.csv"
url_SynClus=Ruta_General+url

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

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

```
# get the number of observations and variables
rows = df.shape[0]
print('# of Rows:', rows )
columns = df.shape[1]
print('# of Columns:', columns )
```

```
# of Rows: 1024
# of Columns: 6
```

2. Drop rows with any missing values

```
# Drop rows with NaN values if existing


# Print the new shape
```

3. Scatterplot

```
# Scatterplot of x1 and x2
```

```
# Scatterplot of x1 and x3
```

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

```
# Pairplot: Scatterplot of all variables
```

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  =  # Let's assume there are 2,3,4,5...? clusters/groups
km =

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


# Print estimated cluster of each point in the dataset


# Add a new column to the dataset with the cluster information
```

```
# Laber of the estimated clusters
```

```
# Cluster centroides
```

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

```
# The number of iterations required to converge
```

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
```

```
# Scatter plot of each cluster
```

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)
```

```
# Define values of k
```

```
# For each k
```

```
# Plot sse versus k
```

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

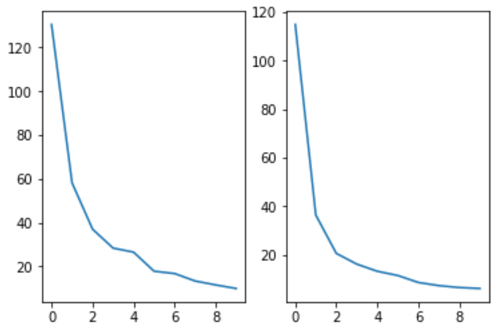
2. Repeat this analysis using all six features, e.g., x1, x2,..., x6
3. Provide conclusions

```
Ruta_General = "/content/drive/My Drive/Herramientas Computacionales/"
url = "iris.csv"
url_SynClus=Ruta_General+url

# Load the dataset
header = ['s_length', 's_width', 'p_length', 'p_width', 'Class']
df  = pd.read_csv(url_SynClus, names=header)
df
```

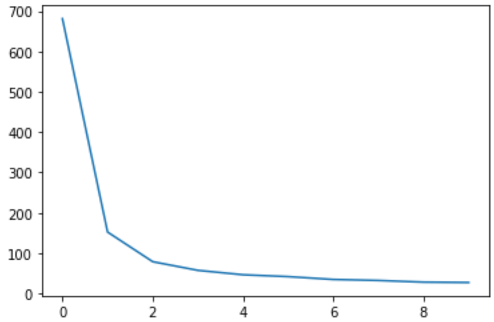
	s_length	s_width	p_length	p_width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	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

```
from sklearn.cluster import KMeans

#Init list of sse
SSE = []

#Define K
K = [1,2,3,4,5,6,7,8,9]

#For each K
for x in K:
    temporal_km = KMeans(n_clusters = x, n_init="auto")
    temporal_km.fit_predict(df[["s_length","s_width"]])
    SSE.append(temporal_km.inertia_)
plt.subplot(1,2,1)
plt.plot(range(0,10),SSE)

#Init list of sse
SSE = []

#Define K
K = [1,2,3,4,5,6,7,8,9]

#For each K
for x in K:
    temporal_km = KMeans(n_clusters = x, n_init="auto")
    temporal_km.fit_predict(df[["s_width","p_width"]])
    SSE.append(temporal_km.inertia_)
plt.subplot(1,2,2)
plt.plot(range(0,10),SSE)
plt.show()
```



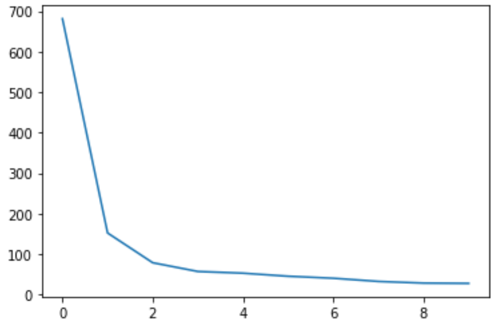
```
#Init list of sse
SSE = []

#Define K
K = [1,2,3,4,5,6,7,8,9,10]

#For each K
for x in K:
    temporal_km = KMeans(n_clusters = x, n_init="auto")
    temporal_km.fit_predict(df.iloc[:,0:4])
    SSE.append(temporal_km.inertia_)
plt.plot(range(0,10),SSE)
plt.show()
```



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

```
# 1 --
#Define n clusters
#Init list of sse
SSE = []

#Define K
K = [1,2,3,4,5,6,7,8,9,10]

#For each K
for x in K:
    temporal_km = KMeans(n_clusters = x, n_init="auto")
    temporal_km.fit_predict(df.iloc[:, 0:4])
    SSE.append(temporal_km.inertia_)

plt.plot(range(0,10),SSE)
plt.show()
```

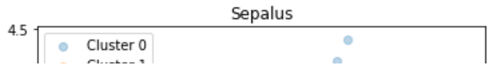



```
K = 4
km = KMeans(n_clusters = K, n_init="auto")

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

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

	s_length	s_width	p_length	p_width	Class	Clusters
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	1
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	1
149	5.9	3.0	5.1	1.8	Iris-virginica	2

150 rows × 6 columns

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


#Scatter clusters
kmc = km.cluster_centers_
#SEPALUS
plt.xlabel("Sepalus·Length")
plt.ylabel("Sepalus·Width")
plt.title("Sepalus")
plt.scatter(df1.s_length,·df1.s_width,·label="Cluster·0",·alpha·=·.3)
plt.scatter(df2.s_length,·df2.s_width,·label="Cluster·1",·alpha·=·.3)
plt.scatter(df3.s_length,·df3.s_width,·label="Cluster·2",·alpha·=·.3)
plt.scatter(df4.s_length,·df4.s_width,·label="Cluster·3",·alpha·=·.3)
plt.scatter(kmc[0],kmc[1],·label="Centroid",·marker·=·"x")
plt.legend()
plt.show()
#PETALUS
plt.xlabel("Petalus length")
plt.ylabel("Petalus Width")
plt.title("Petalus")
plt.scatter(df1.p_length, df1.p_width, label="Cluster 0", alpha = .3)
plt.scatter(df2.p_length, df2.p_width, label="Cluster 1", alpha = .3)
plt.scatter(df3.p_length, df3.p_width, label="Cluster 2", alpha = .3)
plt.scatter(df4.p_length, df4.p_width, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "x")
plt.legend()
plt.show()
```

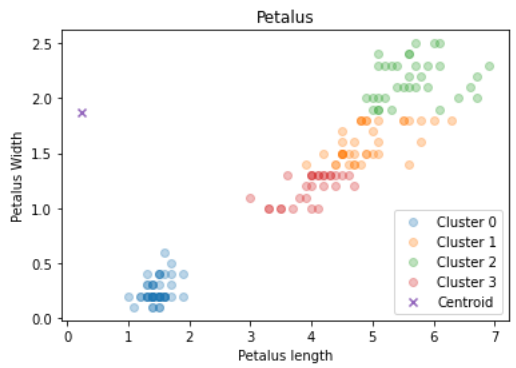



```
#2.-
K = 4
km = KMeans(n_clusters = K, n_init="auto")

#Clustering
yestimated = km.fit_predict(df.iloc[:, 3:4])

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

	s_length	s_width	p_length	p_width	Class	Clusters
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0
...	...	...	...	...	...	...
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	1

150 rows × 6 columns

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

#PETALUS
plt.xlabel("Petalus length")
plt.ylabel("Petalus Width")
plt.title("Petalus")
plt.scatter(df1.p_length, df1.p_width, label="Cluster 0", alpha = .3)
plt.scatter(df2.p_length, df2.p_width, label="Cluster 1", alpha = .3)
plt.scatter(df3.p_length, df3.p_width, label="Cluster 2", alpha = .3)
plt.scatter(df4.p_length, df4.p_width, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "x")
plt.legend()
plt.show()
```



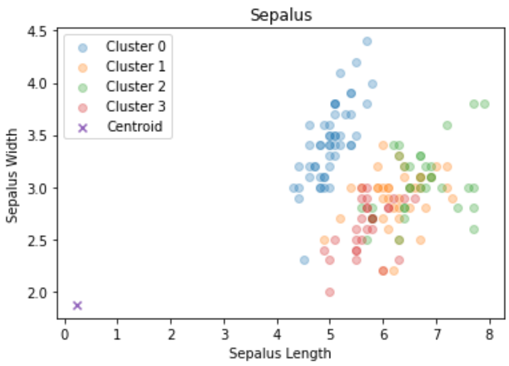

```
# 3.-
K = 4
km = KMeans(n_clusters = K, n_init="auto")

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

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

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

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

150 rows × 6 columns

```
#SEPALUS
plt.xlabel("Sepalus Length")
plt.ylabel("Sepalus Width")
plt.title("Sepalus")
plt.scatter(df1.s_length, df1.s_width, label="Cluster 0", alpha = .3)
plt.scatter(df2.s_length, df2.s_width, label="Cluster 1", alpha = .3)
plt.scatter(df3.s_length, df3.s_width, label="Cluster 2", alpha = .3)
plt.scatter(df4.s_length, df4.s_width, label="Cluster 3", alpha = .3)
plt.scatter(kmc[0],kmc[1], label="Centroid", marker = "x")
plt.legend()
plt.show()
```



Colab paid products - Cancel contracts here

✓ 0s completed at 9:37 PM