Especialización en Inteligencia Artificial

Trabajo Semana 4 Algoritmos No Supervisados

Asignatura: Machine Learning
Realizado por: Michael Andrés Mora Poveda
Universidad Minuto de Dios
Abril 2023

Propósito:

Trabajo Semana 4 - K-means Algorithm

Asignatura: Machine Learning

Especialización en Inteligencia Artificial

Realizado por: Michael Andrés Mora Poveda

Hola a todos, el objectivo de este trabajo es aplicar algoritmos **no supervisados** a los datasets trabajados en la semana dos: **Congressional Voting Records Data Set** y **adults**

Los siguientes links serán útiles para conocer la estructura de los datasets:

- https://archive.ics.uci.edu/ml/datasets/adult
- https://www.kaggle.com/datasets/devvret/congressional-voting-records
- https://github.com/micmorap/Machine_Learning_Portfolio/blob/main/AI%20Specialization%20-%20U%20of%20Minuto%20de%20Dios/Machine_Learning/Semana_2/Análisis%20exploratorio%20de%20datos%20-%20ML.ipynb

Imagen y código de autoría propia

Notas:

Esta presentación contiene partes del script generado en Jupyter Notebook de acuerdo a la actividad de la semana 4 de la Especialización en IA de la Universidad Minuto de Dios. El script completo se encuentra en el siguiente repositorio de Github:

https://github.com/micmorap/Machine_Learning_Portfolio/blob/main/AI%20Specialization%20-%20U%20of%20Minuto%20de%20Dios/Machine_Learning/Semana_4/K-means_Algorithm.ipynb

1. K-Modes para Congressional Voting Records dataset

Con el dataset ya revisado y depurado, vamos a realizar la iteración para encontrar el número óptimo de clústers:

In [37]:
```
# Elbow curve to find optimal K
cost = []
K = range(1,5)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "random", n_init = 5, verbose=1)
    kmode.fit_predict(vote_categorical_data)
    cost.append(kmode.cost_)

plt.figure(1, figsize = (7,3))
<font size='3'>
<br>
Ahora, vamos a realizar validaciones generales como número de columnas, cantidad de valores nulos, el tipo de variab
</br>
</font>plt.plot(K, cost, 'bx-')
plt.xlabel('No. of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal k')
plt.show()
```

```
Starting iterations...
Run 3, iteration: 1/100, moves: 243, cost: 1470.0
Run 3, iteration: 2/100, moves: 32, cost: 1460.0
Run 3, iteration: 3/100, moves: 15, cost: 1460.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 4, iteration: 1/100, moves: 71, cost: 1460.0
Run 4, iteration: 2/100, moves: 25, cost: 1452.0
Run 4, iteration: 3/100, moves: 3, cost: 1452.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 5, iteration: 1/100, moves: 43, cost: 1556.0
Run 5, iteration: 2/100, moves: 0, cost: 1556.0
Best run was number 2
```

Imagen y código de autoría propia

A continuación podemos ver el número óptimo de clústers para este ejercicio, es decir, **k=2**:

Con el dataset ya revisado y depurado, vamos a realizar la iteración para encontrar el número óptimo de clústers:

In [69]:
```
# Elbow curve to find optimal K
cost = []
K = range(1,5)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "random", n_init = 5, verbose=1)
    kmode.fit_predict(vote_categorical_data)
    cost.append(kmode.cost_)

plt.figure(1, figsize = (7,3))

plt.plot(K, cost, 'bo-', c= 'steelblue')
plt.xlabel('No. of clusters')
plt.ylabel('Cost')
plt.title('Elbow Method For Optimal k', fontsize = 15, weight='bold', color= 'midnightblue')
plt.show()
```

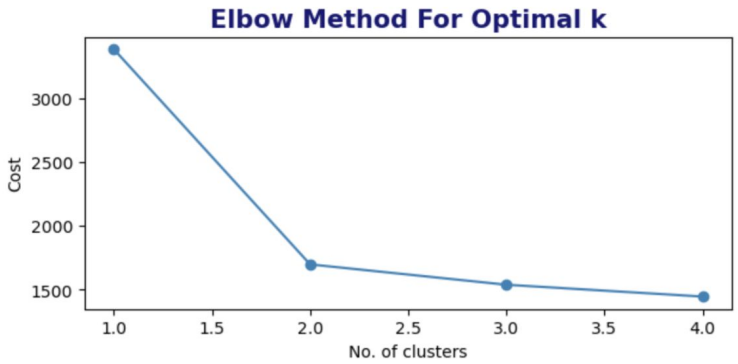


Imagen y código de autoría propia

Luego, entrenamos el modelo con base al resultado anterior, por medio del algoritmo K-Modes:

Dada la gráfica anterior, el número óptimo de clústers es 2. Por lo tanto, vamos a aplicar el algoritmo con base a esta información.

```
In [39]: # Building the model with 2 clusters
         kmode = KModes(n_clusters=2, init = "random", n_init = 5, verbose=1)
         clusters = kmode.fit_predict(vote_categorical_data)
         clusters

         Starting iterations...
         Run 3, iteration: 1/100, moves: 73, cost: 1705.0
         Run 3, iteration: 2/100, moves: 7, cost: 1705.0
         Init: initializing centroids
         Init: initializing clusters
         Starting iterations...
         Run 4, iteration: 1/100, moves: 29, cost: 1699.0
         Init: initializing centroids
         Init: initializing clusters
         Starting iterations...
         Run 5, iteration: 1/100, moves: 20, cost: 1699.0
         Best run was number 2

Out[39]: array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
                0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
                0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
```

Imagen y código de autoría propia

Visualización y agrupación los clústers 0 y 1:

In [50]:
```
# Asignamos el cluster correspondiente a cada uno de los registros del dataset:
vote_categorical_data.insert(0, "Cluster", clusters)
vote_categorical_data
```

Out[50]:

Class Name	Cluster	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	religious-groups-in-schools	anti-satellite-test-ban	aid-to-nicaraguan-contras	mx-missile	immigration	synfuels-corporation-cutback	educatior spendin
republican	0	n	y	n	y	y	y	n	n	n	n	n	
democrat	0	?	y	y	?	y	y	n	n	n	n	y	
democrat	0	n	y	y	n	?	y	n	n	n	n	y	
democrat	0	y	y	y	n	y	y	n	n	n	n	y	
democrat	0	n	y	y	n	y	y	n	n	n	n	n	
...	...	...	...	...	...	...	...	...	...	...	...	...	
republican	0	n	n	y	y	y	y	n	n	y	y	n	

In [47]:
```
# Distribución por conteo de cada uno de los clústers:
vote_categorical_data.groupby('Cluster').count()
```

Out[47]:

Cluster	handicapped-infants	water-project-cost-sharing	adoption-of-the-budget-resolution	physician-fee-freeze	el-salvador-aid	groups-in-schools	satellite-test-ban	nicaraguan-contras	mx-missile	immigration	synfuels-corporation-cutback	education-spending	superfund-right-to-sue
0	212	212	212	212	212	212	212	212	212	212	212	212	212
1	222	222	222	222	222	222	222	222	222	222	222	222	222

click to unscroll output; double click to hide

Imagen y código de autoría propia

Chi-cuadrado test:

En términos porcentaules, el 48% pertenece al primer clúster y el 52% restante al segundo clúster.
Ahora, vamos a ver un ejemplo del test Chi cuadrado para ver la relación entre dos variables categóricas de nuestro dataset:

In [61]:
```
# importamos librería stats de SciPy y tomamos dos columnas aleatorias del dataset:
from scipy.stats import chi2_contingency
data_chi_test = pd.crosstab(index=vote_categorical_data['immigration'], columns=vote_categorical_data['handicapped-i
```

In [62]:
```
# ejecutamos la función chi2_contingency para calcular el Chi cuadrado,
# P-value, grados de libertad y valores esperados:
chi2, pval, dof, expected = chi2_contingency(data_chi_test)
print('Chi-squared test statistic:', chi2)
print('P-value:', pval)
print('Degrees of freedom:', dof)
print('Expected values:\n', expected)
```

```
Chi-squared test statistic: 45.766653948826736
P-value: 2.754179185429524e-09
Degrees of freedom: 4
Expected values:
 [[  0.19354839   3.79032258   3.01612903]
 [  5.86175115 114.79262673  91.34562212]
 [  5.94470046 116.41705069  92.63824885]]
```

In [63]:
```
#Condicional para saber si existe asociación significativa entre dichos features:
if pval < 0.05:
    print('There is a significant association between the two categorical variables.')
else:
    print('There is no significant association between the two categorical variables.')
```

```
There is a significant association between the two categorical variables.
```

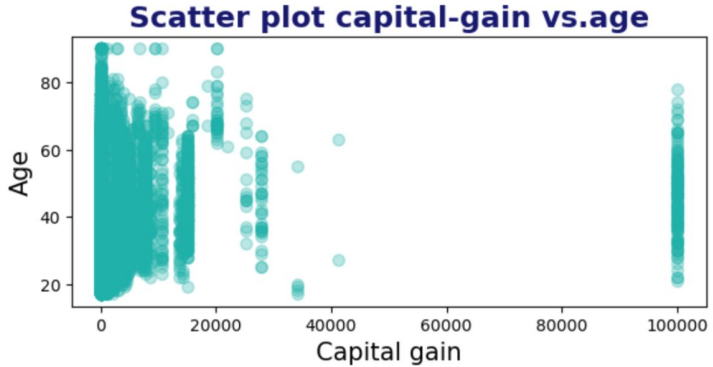
Imagen y código de autoría propia

2. K-Means para Adults dataset

Para este segundo ejercicio, aplicaremos el algoritmo **K-means (no supervisado) para adults dataset:**

Ahora, aplicaremos K-means en **2D** basados entre las variables capital-gain y hours-per-week:

In [12]:
```
# Creamos un scatterplot en 2D entre capital ganado y horas trabajadas por semana:
plt.figure(1, figsize = (7,3))
plt.title("Scatter plot capital-gain vs.age", fontsize = 18, weight='bold', color= 'midnightblue')
plt.xlabel("Capital gain", fontsize = 15)
plt.ylabel("Age", fontsize = 15)
plt.scatter(x= "capital-gain", y="age", data = dfAdults, s = 50, c='lightseagreen', alpha = 0.3)
plt.show()
```



Luego, vamos a enfocarnos en buscar el número correcto de clústers **k** para este ejercicio, teniendo en cuenta dos features: **capital gain** y **Age**

Imagen y código de autoría propia

Estandarización:

In [13]:
```
# Hacemos el scaling de las variables para evitar sesgos:
# Seleccionamos las columnas e instanciamos un objeto StandardScaler:
cols_to_scale = ['capital-gain', 'age']
scaler = StandardScaler()
```

In [14]:
```
# Hacemos la estandarización con los métodos fit y transform:
scaler.fit(dfAdults[cols_to_scale])
dfAdults[cols_to_scale] = scaler.transform(dfAdults[cols_to_scale])
```

In [15]:
```
# Chequeamos esta transformación:
dfAdults[cols_to_scale].head(3)
```

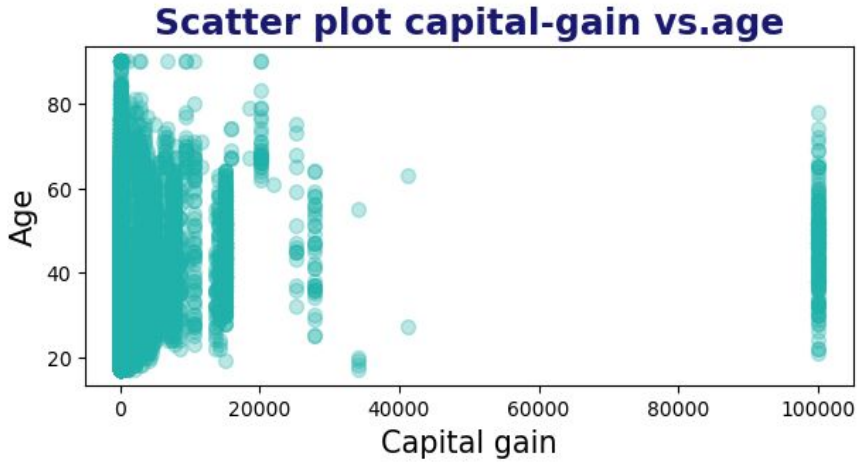
Out[15]:

	capital-gain	age
0	-0.145914	0.837097
1	-0.145914	-0.042640
2	-0.145914	1.057031

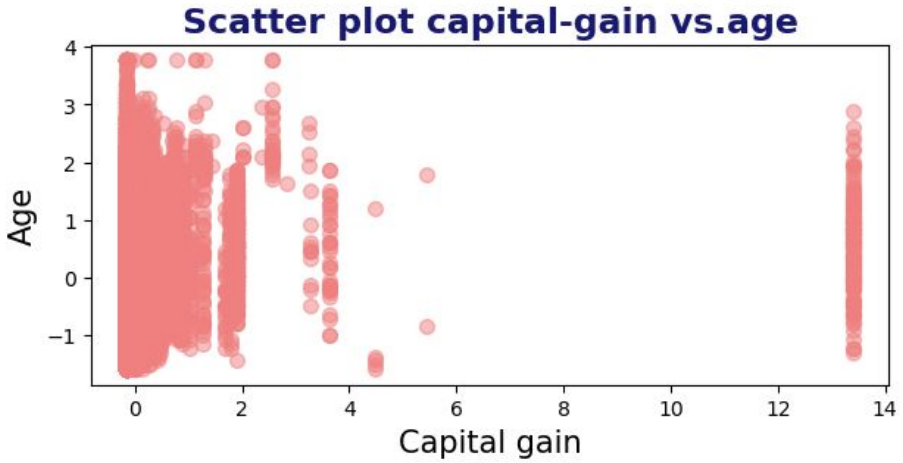
In [16]:
```
# Creamos un scatterplot en 2D entre capital ganado y horas trabajadas por semana:
plt.figure(1, figsize = (7,3))
plt.title("Scatter plot capital-gain vs.age", fontsize = 17, weight='bold', color= 'midnightblue')
plt.xlabel("Capital gain", fontsize = 15)
plt.ylabel("Age", fontsize = 15)
plt.scatter(x= "capital-gain", y="age", data = dfAdults, s = 50, c='lightcoral', alpha = 0.5)
plt.show()
```

Imagen y código de autoría propia

Estandarización: Con las siguiente imágenes, comprobamos que no es necesario estandarizar los features:



Scatterplot original



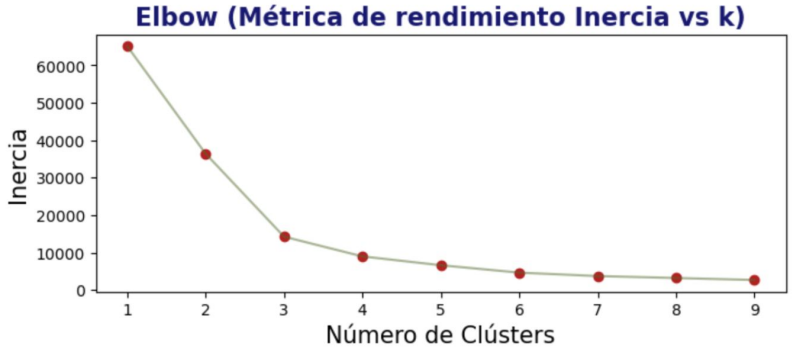
Scatterplot estandarizado

Imagen y código de autoría propia

Métrica de performance (Inercia) y número óptimo de clústers (**k=4**):

```
# Esta calcula distancia promedio al cuadrado entre cada punto y su centroide más cercano:
inertia = []
for k in range(1, 10):
    iteration = KMeans(n_clusters = k ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001,
                       random_state= 111)
    iteration.fit(X)
    inertia.append(iteration.inertia_)
```

In [19]:
```
# Luego graficamos el scatterplot entre el número de clústers k y la métrica de inercia:
plt.figure(1 , figsize = (8 ,3))
plt.title("Elbow (Métrica de rendimiento Inercia vs k)", fontsize = 16, weight='bold', color= 'midnightblue')
plt.plot(np.arange(1 , 10) , inertia , 'o', color = "firebrick")
plt.plot(np.arange(1 , 10) , inertia , '-' , alpha = 0.5, color = "darkolivegreen")
plt.xlabel('Número de Clústers', fontsize = 15) , plt.ylabel('Inercia', fontsize = 15)
plt.show()
```



De acuerdo al gráfico anterior y al "elbow" que nos indica el número óptimo de clústers para este dataset, vamos a crear la visualización de los centroides y sus correspondientes regiones. Además con los scatterplots para la data con o sin estandarizar, comprobamos que para este ejercicio no es necesario aplicar el StandardScaler a nuestros datos.

Imagen y código de autoría propia

Segmentación final:

In [20]:
```
# Tomamos de nuestro backup la data original para estas dos variables:
X1 = df_Adults_2[['capital-gain', 'age']].values
```

In [21]:
```
# Ejecutamos el algoritmo para 4 clústers:
four_groups = KMeans(n_clusters = 4 ,init='k-means++', n_init = 10 ,max_iter=300, tol=0.0001, random_state= 111 )
four_groups.fit(X1)
labels = four_groups.labels_
centroides = four_groups.cluster_centers_
```

In [22]:
```
# Por último graficamos los clúster y los centroides correspondientes:
plt.figure(1 , figsize = (7, 3))
plt.clf()
plt.title("Clústers y centroides", fontsize = 18, weight='bold', color= 'midnightblue')
plt.scatter(x="capital-gain" , y= "age", data = df_Adults_2, s = 50, c=labels, alpha = 0.5)
plt.scatter(x = centroides[: , 0] , y =  centroides[: , 1] , s = 300 , c = 'red' , alpha = 0.5)
plt.ylabel('Age', fontsize = 15)
plt.xlabel('Capital gain', fontsize = 15)
plt.show()
```

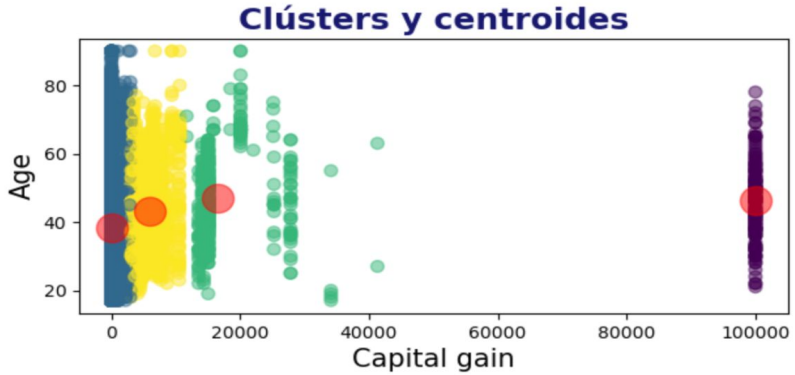


Imagen y código de autoría propia

Conclusiones

- A nivel general este es uno de los algoritmos no supervisados más prácticos de aplicar en problemas de clasificación.
- Como apreciamos en los scatterplots, no siempre es necesario estandarizar o normalizar la data, se debe revisar este paso con detenimiento para evitar sesgos e imprecisiones.
- La métrica de performance de inercia calcula el mean square distance entre cada observación y su centroide más cercano.
- El concepto de **Elbow** o codo nos ayuda a determinar el número **k** de clústeres más adecuado para el ejercicio intentando evitar overfitting ó underfitting teniendo en cuenta la relación entre los valores que tome **k** y su inercia asociada.
- Para el algoritmo K-Modes es importante tener en cuenta el uso de la distancia Hamming en vez de la euclidiana (como se usa normalmente para K-Means) y del uso de la moda como medida de tendencia central en vez de la media aritmética, dado que está diseñado para variables categóricas.
- El **Chi-Squared test** es el análogo del coeficiente de correlación de Pearson para las variables categóricas.

Imagen y código de autoría propia

Referencias:

[1]. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2]. Congressional Voting Records. (2020).Kaggle.Taken from:https://www.kaggle.com/datasets/devvret/congressional-voting-records

[3]. Pedregosa, F., Varoquaux, Gael, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[4]. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

[5]. Heeral, D. (2021)."KMeans Clustering for Customer Data", Taken from: https://www.kaggle.com/code/heeraldedhia/kmeans-clustering-for-customer-data/notebook

[6]. Bonthu, H. (2022). KModes Clustering Algorithm for Categorical data. Taken from: https://www.analyticsvidhya.com/blog/2021/06/kmodes-clustering-algorithm-for-categorical-data/?#

[7]. K-Mode Clustering in Python. (2023). K-Mode Clustering in Python. Taken from: https://www.geeksforgeeks.org/k-mode-clustering-in-python/

[8]. Kaplan, D. (2022). K Mode Clustering Python (Full Code). Taken from: https://enjoymachinelearning.com/blog/k-mode-clustering-python/