TECNOLÓGICO NACIONAL DE MÉXICO

Alumno:

RUIZ CHAVEZ ABEL EDMUNDO

Docente:

EDUARDO ANTONIO HINOJOSA PALAFOX

Materia:

MINERIA DE DATOS

Grupo:

S7C

Grupo:

PROYECTO FINAL

INTRODUCCION

La minería de datos la podemos definir como la extracción de conocimiento a través de la información de datos procesados. Esta tiene una utilidad muy importante para el futuro, ya que gracias al Big Data, todos los datos que se generan por medio de las redes sociales, paginas, blogs, es decir, el internet, es necesario tener una forma de como representar toda esta información.

El machine Learning tiene distintos modelos que se utilizan con el fin de obtener la información de los datos.

Existen los supervisados y los no supervisados. Algunos ejemplos de ambos, son los modelos de clasificación, estos tienen por defecto ya unas etiquetas que permiten tomar decisiones. En estos se encuentran los árboles de decisiones o los bosques aleatorios. Por otro lado, los no supervisados, tienden a hacer agrupamientos de datos basándose en sus características, por ello, un algoritmo que se destaca por esto es el de K-means, que será el que utilizaremos en este proyecto.

DESCRIPCION DEL PROBLEMA

El conjunto de datos recopila el precio de costo de Internet en más de 200 países, en este caso, usando como referencia desde el año 2020 al año de 2021.

La principal razón de la existencia de este dataset es el de mostrar al mundo que la India tiene el Internet más barato, por lo que el autor de este le interesa mucho el tema.

El internet en la India comenzó en 1986 y fue solamente accesible para la educación y la comunidad de investigación. Para que el público general pudiese acceder a él tuvieron que pasar años hasta el 15 de agosto de 1995 y en 2020, se ha alcanzado la cantidad de 718.78 millones de usuarios activos en internet, número que representa el 54.29% de la población mundial.

En mayo de 2014, el internet en la India, tuvo 9 formas diferentes de fibras por las cuales podía acceder y 5 puntos de tierra por los que lograba llegar el Internet.

El gobierno de la India ha realizado en proyectos como BharatNet, Digital India, Brand India y Startup India para acelerar aún más su crecimiento en los ecosistemas basados en Internet.

Lo que se busca en este dataset, como bien se dijo, es mostrar que cantidad de los países tienen dentro del internet, mediante su porcentaje de población, características como los usuarios activos de internet y el precio que estos tuvieron los últimos 2 años.

DESCRIPCION DEL CONJUNTO DE DATOS

Existen 3 datasets dentro del archivo, pero el importante es el llamado all_csv sorted.csv, porque junta los otros dos archivos dentro de este para poder maniobrar mejor con los datos.

Este tiene de características las siguientes columnas:

- S. NO: Este sirve como un ID para cada País.
- Country code: Abreviación del nombre del país para agilizar su rápida comprensión.
- Country: Nombre completo del país sin ningún tipo de abreviatura.
- Continental región: Donde está ubicado geográficamente dentro de los continentes.
- NO. OF Internet Plans: Numero de planes de Internet que maneja cada país.

- Average Price of 1GB (USD): Precio promedio por GB usado en dólares americanos.
- Cheapest 1GB for 30 days (USD): Precio más barato por GB para 30 días.
- Most expensive 1GB (USD): Precio más caro por GB.
- Average Price of 1GB (USD at start of 2021): Precio promedio por GB en los inicios de 2021.
- Average Price of 1GB (USD at start of 2020): Precio promedio por GB en los inicios de 2020.
- Internet users: Cantidad de usuarios activos en internet.
- Population: Población del país.
- Avg \n(Mbit/s)Ookla: Precio Promedio por megabit sobre segundo.

DESCRIPCION DE LA SOLUCION PROPUESTA

La solución que se propone es el utilizar el algoritmo de K-means para agrupar los países en distintos conjuntos y mostrar cuales de estos son los que tienen el internet más barato y la cantidad de usuarios activos de internet. Cada uno de estos es divido por supuesto por las características que comparten con otros países, así que es de esperar que puedan clasificarse en varios conjuntos, como lo son, Precio más alto y menor cantidad de usuarios hasta los más baratos y con más usuarios.

DESCRIPCIÓN DEL CÓDIGO UTILIZADO

1. Primero que todo, al comenzar con nuestra libreta de Python, tenemos que importar las librerías necesarias para no tener problema a la hora de trabajar. En este caso, usaremos las librerías de matplotlib, pandas, numpy, sys para graficar, sklearn para standarizar los datos, importar KMeans y calcular la silueta.

```
[309] import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import matplotlib.cm as cm
      import sys
      from sklearn.preprocessing import StandardScaler
      from sklearn.cluster import KMeans
      from sklearn.metrics import silhouette_samples, silhouette_score
```

2. Ya con las librerías importadas, podemos seguir a leer los datos del dataset usando read_csv.

```
df = pd.read_csv("all_csv sorted.csv")
df
```

:ountry code	Country	Continental region	NO. OF Internet Plans	Average price of 1GB (USD)	Cheapest 1GB for 30 days (USD)	Most expensive 1GB (USD)	Average price of 1GB (USD at the start of 2021)	Average price of 1GB (USD – at start of 2020)	Internet users	Population	Avg \n(Mbit/s)Ookla
IL	Israel	NEAR EAST	27.0	0.05	0.02	20.95	0.11	0.9	6,788,737	8,381,516	28.01
KG	Kyrgyzstan	CIS (FORMER USSR)	20.0	0.15	0.10	7.08	0.21	0.27	2,309,235	6,304,030	16.30
FJ	Fiji	OCEANIA	18.0	0.19	0.05	0.85	0.59	3.57	452,479	883,483	25.99
IT	Italy	WESTERN EUROPE	29.0	0.27	0.09	3.54	0.43	1.73	50,540,000	60,627,291	37.15

3. Como trabajaremos con datos numéricos, es necesario no tomar en cuenta las columnas que tengan un formato categórico y eliminar los registros que sean nulos. Para ello, utilizaremos drop para eliminar las columnas que no necesitamos. Y dropna elimina los registros que no tienen nada, que están vacíos dentro del dataset.

```
[311] df.dropna()
      df.drop('Avg \n(Mbit/s)Ookla', axis=1,inplace=True)
      df.drop('S.NO', axis=1,inplace=True)
      df.drop('Country code', axis=1,inplace=True)
      df.columns

      Index(['Country', 'Continental region', 'NO. OF Internet Plans',
             'Average price of 1GB (USD)', 'Cheapest 1GB for 30 days (USD)',
             'Most expensive 1GB (USD)',
             'Average price of 1GB (USD  at the start of 2021)',
             'Average price of 1GB (USD – at start of 2020)', 'Internet users',
             'Population'],
            dtype='object')
```

4. Usando el comando isnull más sum, podemos observar en que columnas siguen habiendo registros vacíos, por lo que debemos eliminarlos de una forma distinta y más especifica.

```
df.isnull().sum()
```

```
Country                                                    0
Continental region                                         0
NO. OF Internet Plans                                     11
Average price of 1GB (USD)                                 0
Cheapest 1GB for 30 days (USD)                            11
Most expensive 1GB (USD)                                  11
Average price of 1GB (USD  at the start of 2021)          11
Average price of 1GB (USD – at start of 2020)             11
Internet users                                            30
Population                                                32
dtype: int64
```

5. Para ello, usaremos drop con la diferencia que ahora sacaremos a todos los registros que sean categóricos dentro de Average price of 1Gb(USD at the start of 2021), con los distintos registros que llegan a tener. Después de esto, sacaremos los nulls que esten dentro de las demás columnas.
6. Tambien cabe aclarar, que para saber que registros aún quedan, usamos unique() para identificarlos y por ende, eliminar los que no son numéricos.

```
df.drop(df[df['Average price of 1GB (USD  at the start of 2021)'] == 'NO PROVIDERS'].index,inplace = True)
df.drop(df[df['Average price of 1GB (USD  at the start of 2021)'] == 'HYPERINFLATION'].index,inplace = True)
df.drop(df[df['Average price of 1GB (USD  at the start of 2021)'] == "Prices listed in non-convertible 'units'"].index,inplace = True)
```

```
df.drop(df[df['NO. OF Internet Plans'].isnull()].index, inplace=True)
df.drop(df[df['Population'].isnull()].index, inplace=True)
```

```
df['Average price of 1GB (USD  at the start of 2021)'].unique()
df.drop(df[df['Average price of 1GB (USD  at the start of 2021)'] == 'NO PACKAGES'].index,inplace=True)
df.drop(df[df['Average price of 1GB (USD – at start of 2020)'] == 'NO PACKAGES'].index,inplace=True)
```

7. Después de ya haber limpiado el dataset, ahora falta convertir los datos que supuestamente son numéricos dentro de nuestro dataset. En algunos casos, estos llegan a ser strings por las comas que los dividen, así que tenemos que reemplazar con replace todas las comas antes de transformarlos a valores numéricos.

```
Columnas = ['Population','Internet users']
df[Columnas] = df[Columnas].replace({',':''},regex=True)
df[Columnas] = df[Columnas].astype(int)

df['Internet users'] = df['Internet users'].replace({',':''},regex=True)
df['Internet users'] = df['Internet users'].astype(int)

df['NO. OF Internet Plans'] = df['NO. OF Internet Plans'].astype(int)

Columnas = ['Average price of 1GB (USD)','Cheapest 1GB for 30 days (USD)',
            'Most expensive 1GB (USD)','Average price of 1GB (USD  at the start of 2021)',
            'Average price of 1GB (USD – at start of 2020)']

df[Columnas] = df[Columnas].astype(float)
```

8. Ahora que ya tenemos listo nuestro dataset, procedemos a verificar los tipos de las columnas.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 0 to 230
Data columns (total 10 columns):
 #   Column                                          Non-Null Count  Dtype  
---  ------                                          --------------  -----  
 0   Country                                         200 non-null    object 
 1   Continental region                              200 non-null    object 
 2   NO. OF Internet Plans                           200 non-null    int64  
 3   Average price of 1GB (USD)                      200 non-null    float64
 4   Cheapest 1GB for 30 days (USD)                  200 non-null    float64
 5   Most expensive 1GB (USD)                        200 non-null    float64
 6   Average price of 1GB (USD  at the start of 2021)  200 non-null    float64
 7   Average price of 1GB (USD – at start of 2020)   200 non-null    float64
 8   Internet users                                  200 non-null    int64  
 9   Population                                      200 non-null    int64  
dtypes: float64(5), int64(3), object(2)
memory usage: 17.2+ KB
```

9. Con nuestro dataset limpio, podemos estandarizar los datos y crear nuestros clústeres de manera segura, pero antes de proseguir, para maniobrar con los datos de una manera más cómoda, solo elegiremos los registros que tengan mayor o igual a la población de 10 000 000.

```
df = df[(df['Population'] >= 10000000)]
X = df.iloc[:,7:]
sc = StandardScaler()
X = sc.fit_transform(X)
X
```

```
[ 6.85103900e-01, -3.55326829e-01, -3.27304648e-01],
[-7.83067542e-01, -3.83252515e-01, -3.51192799e-01],
[ 3.11895706e-01,  5.37956932e-01,  5.09681806e-01],
[-7.15659895e-01, -2.41714200e-01, -2.35140323e-01],
[-7.15659895e-01, -2.19369230e-01, -2.56113620e-01],
[-7.07439450e-01, -3.54379737e-01, -2.85524147e-01],
[-4.13147526e-01,  7.00596021e-01,  5.73027858e-01],
[-4.49317483e-01, -3.60407473e-01, -3.29147390e-01],
[-7.25524429e-01, -3.64710669e-01, -2.29815162e-01],
[-7.43609407e-01, -1.85502113e-02,  5.55953015e-02],
[ 1.50774988e-01, -3.27459974e-01, -3.23415875e-01],
[-5.31521931e-01, -2.33119242e-02, -7.95114935e-02],
[-3.85198014e-01, -3.48778909e-01, -3.49133133e-01],
```

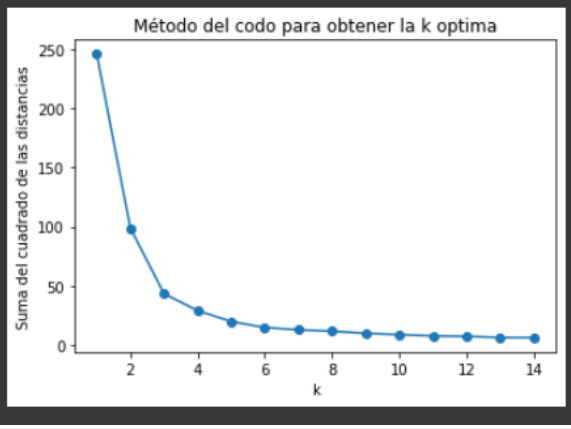
10. Utilizando el algoritmo de KMeans, le expresarmoes que utilize un numero de 6 clusters, para que sea el máximo a utilizar y agrupe nuestros datos de tal forma. Ajustamos los datos al algoritmo con fit y sacamos los nombres de las columnas con .labels.

```
clusterNum = 6
k_means = KMeans(init = "k-means++", n_clusters = clusterNum, n_init = 6)
k_means.fit(X)
labels = k_means.labels_
print(labels)
```

```
[2 2 0 0 2 2 2 0 2 5 1 0 2 2 5 2 2 2 2 1 2 5 2 2 4 2 0 2 2 2 0 2 2 2 5 2 2
 2 5 2 2 2 2 3 5 5 2 2 5 5 2 2 5 5 4 5 2 5 5 5 4 2 4 0 5 4 3 4 4 5 4 5 5 2
 4 4 5 3 5 4 3 2]
```

11. Por consiguiente, para no utilizar cualquier cantidad de clusters, utilizaremos el método del codo para obtener el numero de clusters optimo para nuestro dataset. Cabe mencionar que dependiendo de los resultados, se puede elegir otra cantidad, depende mucho del contexto y el objetivo.

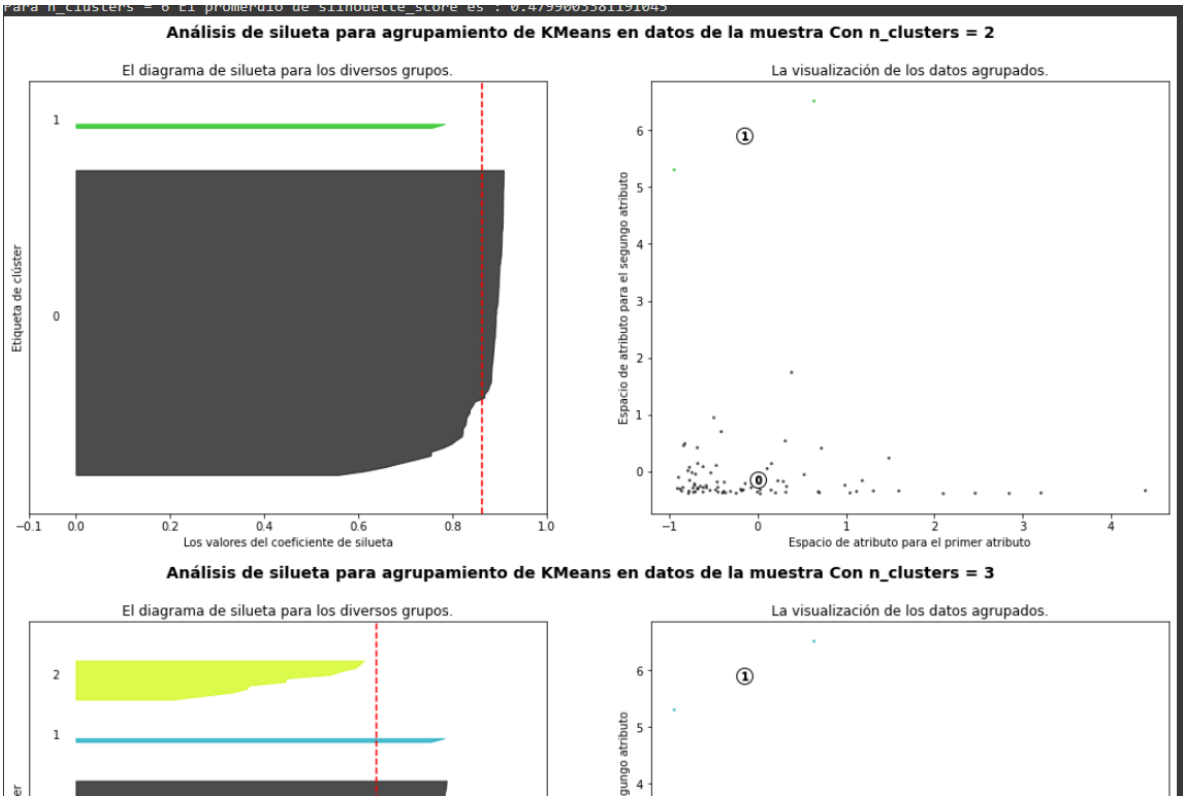
```
Sum_of_squared_distances = []

# Se usa k de 1 to 15
K = range(1,15)
for k in K:
    km = KMeans(n_clusters=k, init='random', n_init=10, max_iter=500,  tol=1e-04, random_state=42)
    km = km.fit(X)
    # Se obtiene la suma de las distancias al cuadrado aplicando km.inertia_
    Sum_of_squared_distances.append(km.inertia_)

# Gráfica de los resultados
plt.plot(K, Sum_of_squared_distances, marker='o')
plt.xlabel('k')
plt.ylabel('Suma del cuadrado de las distancias')
plt.title('Método del codo para obtener la k optima')
plt.show()
```



12. Como se vio en el método del codo, la cantidad de clusters optima es de 2, pero podemos usar también 3 o hasta 4, porque siguen siendo codos que marcan una diferencia con los demás. A continuación, usamos un código que nos calcula la silueta para saber cual cantidad es la óptima, es decir, una forma distinta para saber cual puede ser la mejor cantidad por medio de el promedio de la silueta.

```
range_n_clusters = [2, 3, 4, 5, 6]

for n_clusters in range_n_clusters:
    # Crear una sub gráfica con 1 fila y 2 columnas
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # La primera subtrama es la trama de silueta.
    # El coeficiente de silueta puede oscilar entre -1, 1 pero en este ejemplo todos
    # se encuentran dentro de [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # El (n_clusters+1)*10 es para insertar un espacio en blanco entre la silueta
    # Gráficos de conglomerados individuales, para delimitarlos claramente.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # Inicialice el clúster con valor n_clusters y un generador aleatorio
    # semilla de 10 para la reproducibilidad.
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X)

    # El silhouette_score da el valor promedio de todas las muestras.
    # Esto da una perspectiva de la densidad y separación de los formados.
    # clúster
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("Para n_clusters =", n_clusters,
          "El promerdio de silhouette_score es :", silhouette_avg)

    # Calcule las puntuaciones de la silueta para cada muestra
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # Agregue los puntajes de silueta para las muestras pertenecientes a
        # clúster i, y se ordenan
```

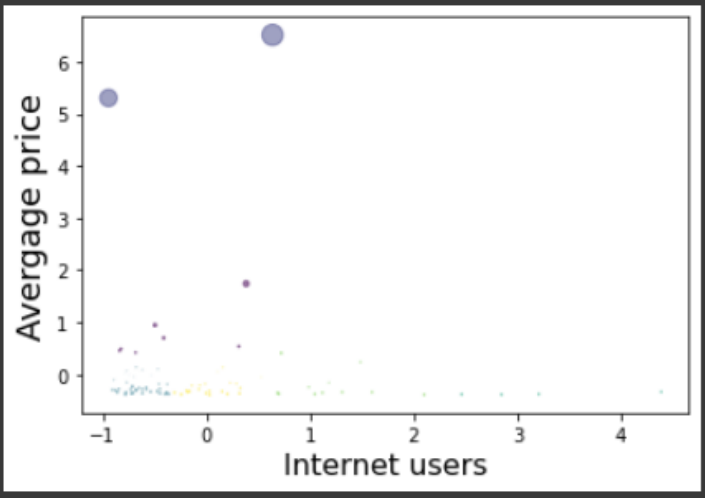
```
[ ]     ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

        # 2nd Plot showing the actual clusters formed
        colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
        ax2.scatter(X[:, 0], X[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                    c=colors, edgecolor='k')

        # Labeling the clusters
        centers = clusterer.cluster_centers_
        # Draw white circles at cluster centers
        ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                    c="white", alpha=1, s=200, edgecolor='k')

        for i, c in enumerate(centers):
            ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                        s=50, edgecolor='k')

        ax2.set_title("La visualización de los datos agrupados.")
        ax2.set_xlabel("Espacio de atributo para el primer atributo")
        ax2.set_ylabel("Espacio de atributo para el segungo atributo")

        plt.suptitle(("Análisis de silueta para agrupamiento de KMeans en datos de la muestra "
                      "Con n_clusters = %d" % n_clusters),
                     fontsize=14, fontweight='bold')

    plt.show()

    Para n_clusters = 2 El promerdio de silhouette_score es : 0.8619050442865095
    Para n_clusters = 3 El promerdio de silhouette_score es : 0.6371036599004852
    Para n_clusters = 4 El promerdio de silhouette_score es : 0.5061314541146998
    Para n_clusters = 5 El promerdio de silhouette_score es : 0.527609950718134
    Para n_clusters = 6 El promerdio de silhouette_score es : 0.4799003381191045
```



13. Por último, usaremos una gráfica para ver de una manera similar como se comportan los datos, tomando en cuenta la cantidad de usuarios conforme a los países.

```
[ ]  area = np.pi * ( X[:, 1])**2
     plt.scatter(X[:, 0], X[:, 1], s=area, c=labels.astype(np.float), alpha=0.5)
     plt.ylabel('Avergage price', fontsize=18)
     plt.xlabel('Internet users', fontsize=16)

     plt.show()
```

```
<ipython-input-322-315d2661b4a1>:2: DeprecationWarning:

`np.float` is a deprecated alias for the builtin `float`. To silence this warn
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/dev
```



RESULTADOS

Ser puede observar que China y la India, efectivamente son los mejores candidatos a tener un internet barato y que suele tener mucha población usándolo.

```
X = df.sort_values(by = 'Internet users')['Country']
Y = df.sort_values(by = 'Internet users')['Internet users']
f = plt.figure()
f.set_size_inches(20, f.get_figheight()*5, forward=True)
plt.barh(X,Y)
plt.xlabel('Internet users')
plt.ylabel('Country')
plt.show()
```

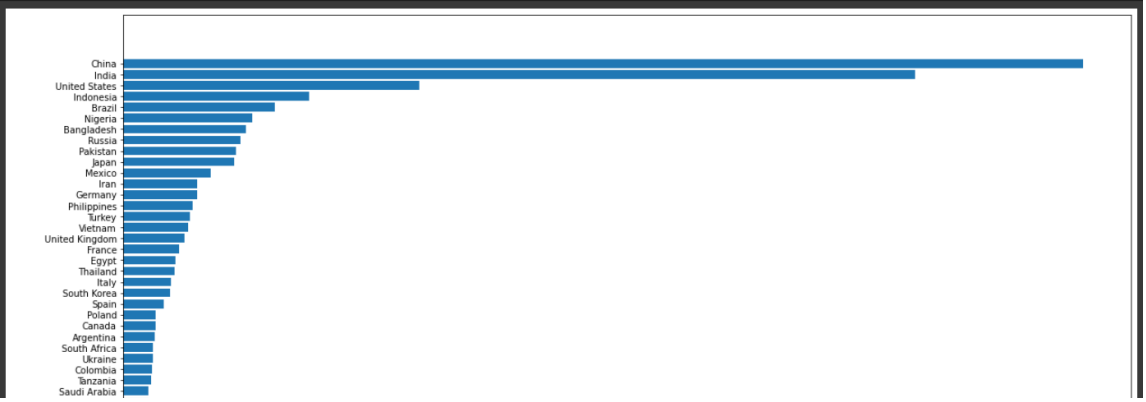
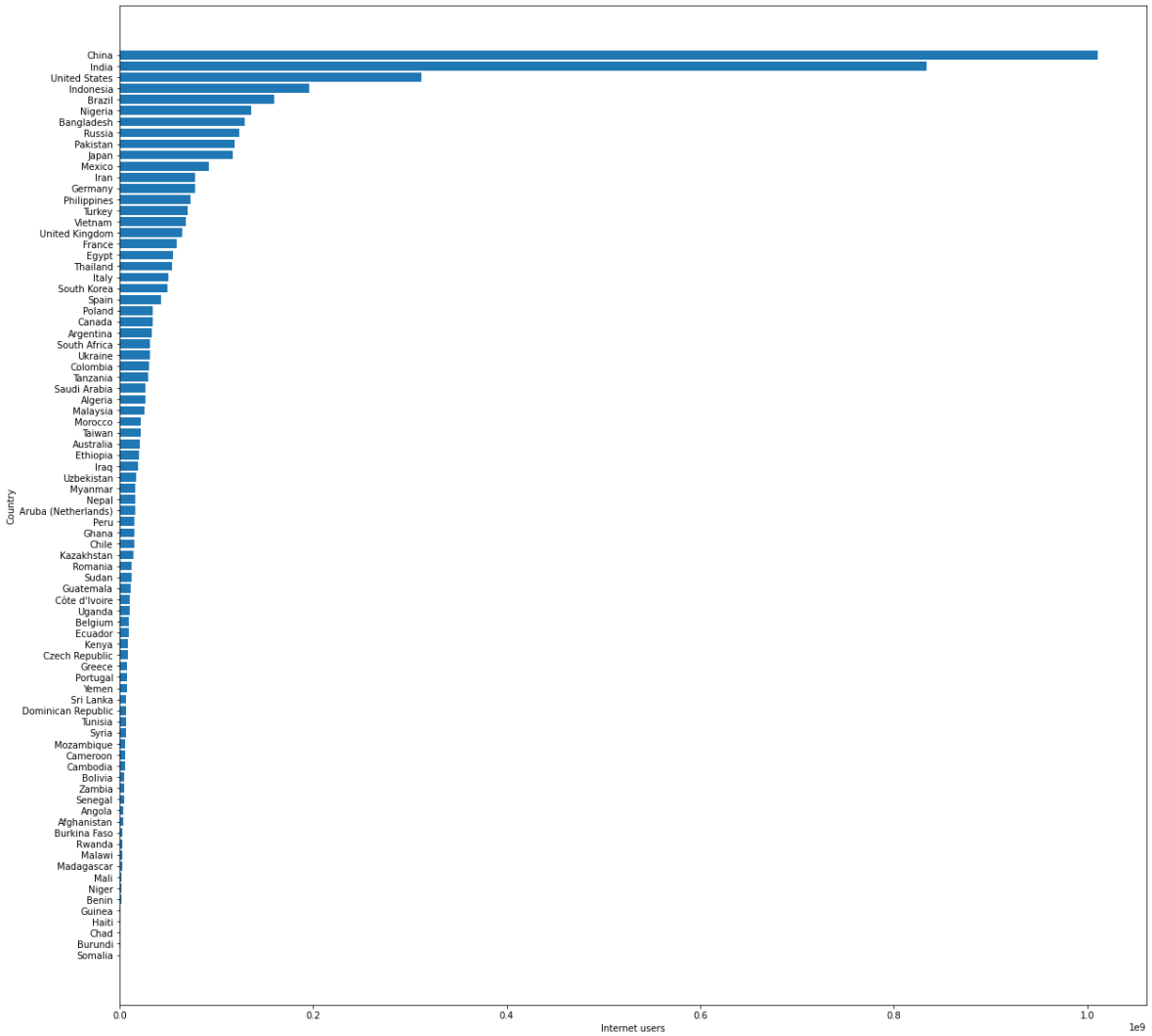


Imagen original



CONCLUSIONES

El algoritmo de KMeans es una gran forma de representar los datos en grupos pequeños o grandes, dependiendo del contexto y su objetivo. Permitiendo revisar de una manera más digerible de ver datos a mayores escalas. En nuestro problema, pudimos confirmar como es que la India, llega a tener una gran cantidad de usuarios en internet gracias a que el precio por internet es muy bajo, siendo así uno de los que mayor población se tiene dentro del mundo en internet.

Enlace a repositorio:

https://github.com/RuizChavezAbelEdmundo/MINERIA-DE-DATOS.git