Reporte practica 2

Quiñones Mayorga Rodrigo

Introducción

Es esta practica vamos a realizar el algoritmo de K-mans, y en la pagina dada por ell profesor se nos describe el funcionamiento de el mismo, siendo esta una técnica para agrupar en grupos, que aquí se llaman clusters según que similitudes contengan los datos.

Síntesis

Organiza datos en un número predeterminado de clústeres (k). El algoritmo busca minimizar la suma de las distancias al cuadrado entre los puntos de datos y el centroide del clúster al que pertenecen.
Luego se itera ajustando los centroides y mandando a los datos a los grupos mas cercanos, hasta que no cambien de manera significativa los centroides.

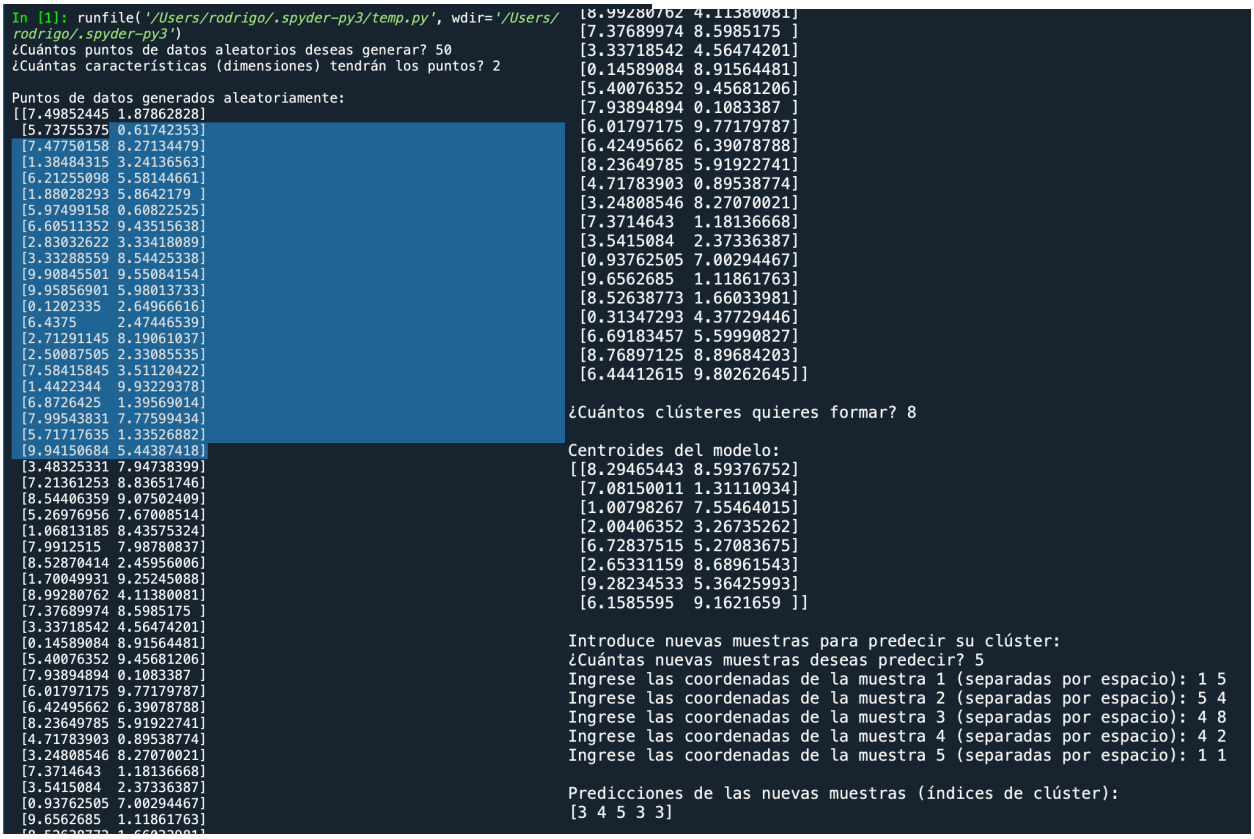
```
In [1]: runfile('/Users/rodrigo/.spyder-py3/temp.py', wdir='/Users/
rodrigo/.spyder-py3')
Ingresa el número de puntos de datos: 25
Ingresa el número de clústeres: 8
Puntos generados:
[[8.62771275 9.78457217]
 [9.49177607 3.20306283]
 [8.37180958 7.08254584]
 [4.89952429 4.24244328]
 [2.44747576 4.49831257]
 [0.52058314 4.62128704]
 [4.36693274 0.05622786]
 [0.08495375 2.63271516]
 [5.79577018 7.75992809]
 [4.99572806 9.12482726]
 [7.78014111 1.55385134]
 [1.53957556 6.01543474]
 [6.21636241 2.13112076]
 [1.20274437 7.34136883]
 [8.08836518 4.34683867]
 [1.56878743 0.71747455]
 [9.93006858 1.81234831]
 [2.09794969 1.3640902 ]
 [7.93776419 1.84380889]
 [9.23045545 7.51216469]
 [7.08388146 9.05534866]
 [5.87632925 8.6747682 ]
 [6.50299767 8.43657201]]
Score: -37.571183799113555

In [2]: runfile('/Users/rodrigo/.spyder-py3/temp.py', wdir='/Users/
rodrigo/.spyder-py3')
Ingresa el número de puntos de datos: 27
Ingresa el número de clústeres: 27
Puntos generados:
[[8.00369537 7.19862922]
 [4.14233891 8.2494982 ]
 [0.41277626 6.68653519]
 [1.19725794 6.77479639]
 [8.57047925 7.8666617 ]
 [6.3058366  4.32614426]
 [2.06404026 9.52024373]
 [2.85840935 7.55878012]
 [5.38705159 2.93782453]
 [4.72002569 8.71142472]
 [3.76502099 6.29236904]
 [2.63130908 4.98858953]
 [2.55223854 4.33182207]
 [8.37323762 6.59499976]
 [1.14710028 9.73442829]
 [4.90560754 8.55503829]
 [6.11132916 7.76145053]
 [1.41477651 4.93997229]
 [0.1752129  3.1523151 ]
 [6.79120301 3.43930013]
 [0.10501854 6.8340036 ]
 [4.80476272 7.15753552]
 [1.63754676 4.27590976]
 [9.53144053 7.91750288]
 [0.5226804  7.6615873 ]
 [3.05259814 6.83145003]
 [9.93489025 1.86940579]]
Score: -1.4791141972893971e-31
```

El primer ejemplo que implemente fue el de SCORE, que básicamente mide que tan bien están agrupados los valores, aquí el score mas cercano a 0 significa que esta mejor agrupado, mientras que este menor sea al numero negativo, significa que esta mas disperso dentro de los grupos.

Lo que realice fue un código donde el usuario ingresa el numero de datos ademas de el numero de grupos, luego genera números aleatorios y te da finalmente el score.

El segundo código fue tomando el ejemplo de predicciones, donde primero se generan unos datos, se agrupan en cluster para posteriormente que el usuario ingrese nuevos números los cuales se agregaran a grupos usando la información anterior con la que se entrenó.



Este código genera puntos de datos aleatorios, entrena un modelo KMeans con ellos, y permite al usuario predecir a qué clústeres pertenecen nuevas muestras.

Para nuestro tercer código, haremos el características polinomiales, el cual transforma características en combinaciones mas complejas, haremos el GET_FEATURE_NAMES

La razón por la cual hacemos esto es para darle más "información" al modelo que estamos entrenando.

Cómo funciona el algoritmo en el ejemplo

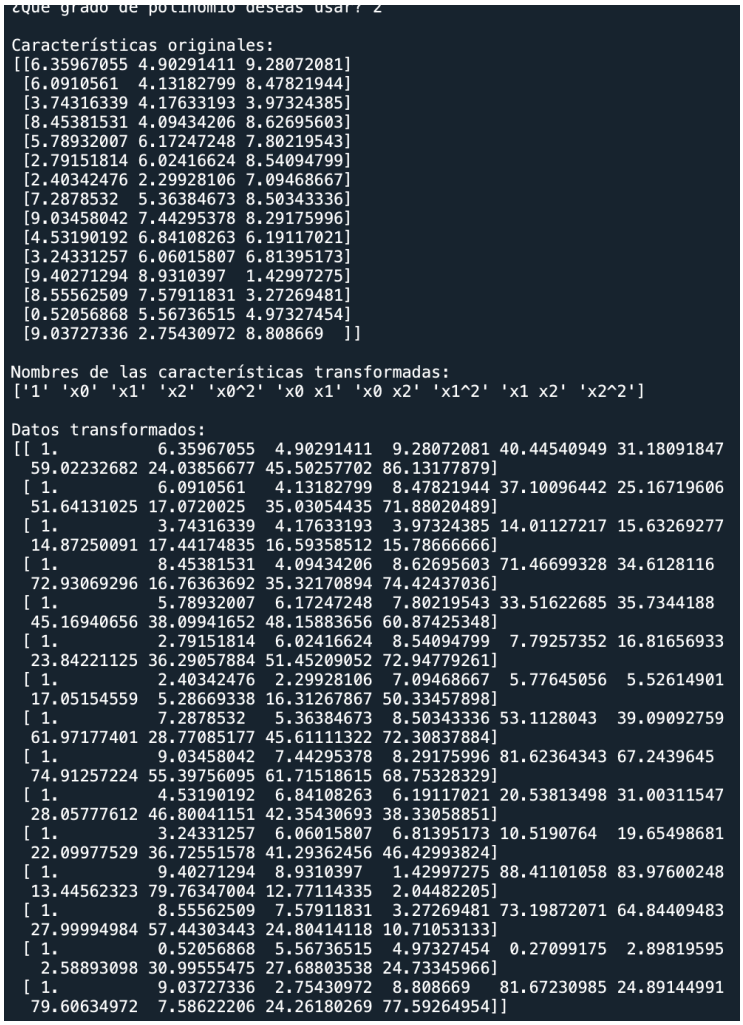
En el código que te di, los pasos son los siguientes:

1. **Generar datos aleatorios**: En lugar de decir manualmente qué características tienen nuestras "pelotas", el programa genera números aleatorios para simular las características (tamaño, peso, etc.).

2. **Transformar las características**: Después, aplicamos el algoritmo **PolynomialFeatures**, que toma las características originales y genera nuevas combinaciones de ellas (cuadrados, productos, etc.).

3. **Obtener los nombres de las características nuevas**: Luego, usamos la función get_feature_names_out() para obtener los nombres de estas nuevas características, por ejemplo: "Tamaño^2", "Tamaño x Peso", etc.

4. **Transformar los datos**: Finalmente, los datos originales (que eran simplemente el tamaño y peso de las pelotas) se transforman en un nuevo conjunto de datos que incluye todas las combinaciones nuevas de las características. Ahora tenemos más información con la que el modelo puede trabajar.



Repositorio
https://github.com/RodQM/
Practica2-Machine-LEARNING