Ingeniería en Inteligencia Artificial, Aprendizaje de Máquina
Sem: 2025-1, 5BV1, Actividad 1

Actividad 1: Reporte y k-means

Docente: Garcia Floriano Andres
Unidad de aprendizaje: Aprendizaje de Máquina

Alumno: Crombie Esquinca Alexander Iain
Alumno: Alavardo Sandoval Juan Manuel
Fecha: 2024-09-10

Objetivo: Aplicar un profiling y kmeans sobre el dataset de iris
Palabras Clave: kmeans,iris,profiling

1 Parte 1

Primeramente, como se observa en la figura 1, en el archivo de jupyter, importamos las librerías de pandas, y la función de profileReport, que nos va a generar información sobre el dataset de iris.

```
[1]: import pandas as pd
     from ydata_profiling import ProfileReport
```

Figure 1: Importación de librerías

En la figura 2, se observa que se importa la información del dataset de iris, se crea el objeto relacionado al reporte, y por último, se imprime. A comparación con el .data originalmente publicado, se agrego una fila adicional al inicio para etiquetar todas las columnas. Porque sino pandas tomará la primera fila como etiquetas.

```
[2]: df = pd.read_csv("./info.csv")
     profile = ProfileReport(df, title="Reporte")
     df.head()
```

[2]:

	sl	sw	pl	pw	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

Figure 2: Lectura csv, generación de reporte, y verificación del csv

Primeramente al desplegar el reporte, observamos algunas filas repetidas, pero también notamos una correspondencia fuerte entre todas las columnas. En especial entre clase y los tamaños de petalos. Como se observa en la figura 3.

Figure 3: Muestra correlación y filas repetidas

También nos proporciona una serie de graficas que relacionan espacialmente los valores entre pl,pw,sl,sw. De una simple vista, podemos a empezar a ver ciertas agrupaciones en el dataset.

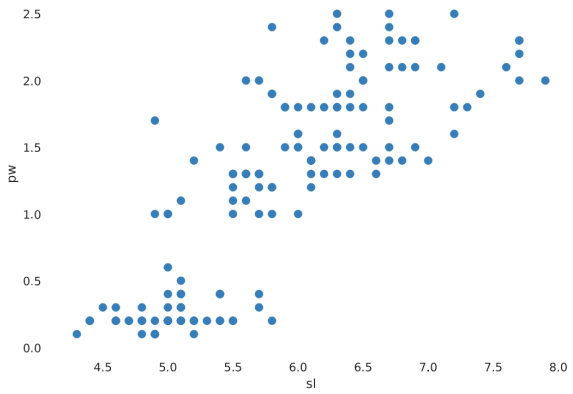

Figure 4: Muestra una grafica de disperción entre los datos pw y sl

Y en mi opinión, en la figura 5 se muestra de la mejor forma la correlación

entre la clase y sus respectivos atributos. Mostrando que efectivamente se pueden ocupar para realizar la predicción.

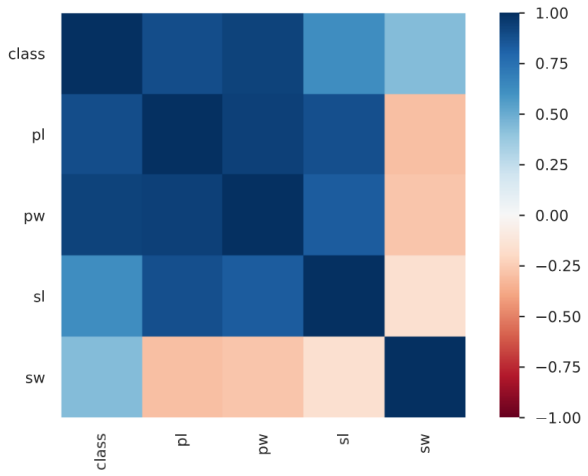

Figure 5: Cuadro de correlación

Por útlimo utilizamos la función "to_file" para crear un archivo html más fácil de compartir a personas que posiblemente no tengan jupyter. Esto lo observamos en la figura 6.

```
[4]: profile.to_file("./reporte_iris.html")
```

Figure 6: Exportar el reporte a un archivho html

2 Parte 2. Agrupamiento con K-Means

Documentación de las funciones de clustering, link: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html n_clusters:

int, default=8
Número de clusters a formar y el número de centroides a generar. Este parámetro define cuántos grupos se generarán al aplicar el algoritmo K-Means.

init: 'k-means++', 'random', callable or array-like of shape (n_clusters, n_features), default='k-means++'
Método para inicializar los centroides de los clusters.
- 'k-means++': Se eligen los centroides iniciales para acelerar la convergencia.
- 'random': Los centroides iniciales se seleccionan de forma aleatoria.

- También puedes proporcionar un array o una función personalizada para la inicialización.

n_init: 'auto' o int, default='auto'
Número de veces que se ejecutará el algoritmo con diferentes semillas de centroides. El resultado final será el mejor de las ejecuciones basadas en la inercia.

max_iter: int, default=300
Número máximo de iteraciones del algoritmo K-Means en una ejecución para encontrar los centroides.

tol: float, default=1e-4
Tolerancia relativa para declarar la convergencia del algoritmo. Se basa en la diferencia entre los centros de los clusters en iteraciones consecutivas.

verbose: int, default=0
Modo de verbosidad. Controla la cantidad de información que se imprime durante la ejecución del algoritmo.

random_state: int, RandomState instance o None, default=None
Controla la generación de números aleatorios para la inicialización de los centroides. Un valor entero asegura que los resultados sean reproducibles.

copy_x: bool, default=True
Si es True, se hace una copia de los datos antes de centrarlos. Si es False, los datos originales pueden ser modificados, lo cual puede introducir pequeñas diferencias numéricas.

algorithm: "lloyd", "elkan", default="lloyd"
Algoritmo K-means a utilizar.
- 'lloyd': Algoritmo clásico basado en Expectation-Maximization.
- 'elkan': Variante que usa la desigualdad triangular para reducir cálculos, más eficiente en datasets bien definidos, pero consume más memoria.

cluster_centers_: ndarray of shape (n_clusters, n_features)
Coordenadas de los centros de los clusters después de la convergencia del algoritmo.

labels_: ndarray of shape (n_samples,)
Etiquetas de cada muestra que indican a qué cluster pertenece cada punto.

inertia_: float
Suma de las distancias al cuadrado de cada muestra al centro del cluster más cercano. Indica la compacidad de los clusters.

n_iter_: int
Número de iteraciones realizadas por el algoritmo hasta que converge.

n_features_in_: int
Número de características (features) presentes en los datos utilizados durante el ajuste (fit).

feature_names_in_: ndarray of shape (n_features_in_,)
Nombres de las características (features) utilizadas en el ajuste. Solo está disponible si los datos de entrada tienen nombres de características.

fit: (X, y=None, sample_weight=None)
Ajusta el modelo K-Means a los datos proporcionados en X. Los datos de entrenamiento son agrupados en clusters.

fit_predict: (X, y=None, sample_weight=None)
Ajusta el modelo y devuelve los índices de los clusters a los que pertenece cada muestra. Es un atajo para aplicar 'fit' seguido de 'predict'.

fit_transform: (X, y=None, sample_weight=None)
Ajusta el modelo y transforma los datos a un espacio de distancia de los centroides de los clusters. Es más eficiente que llamar a 'fit' seguido de 'transform'.

get_feature_names_out: (input_features=None)
Devuelve los nombres de las características transformadas después de aplicar el modelo. Utilizado para validar los nombres de las características con los observados durante el ajuste.

get_metadata_routing: ()
Devuelve la ruta de metadatos del objeto, relevante solo si se usa como subestimador en un meta-estimador.

get_params: (deep=True)
Devuelve los parámetros del estimador, incluyendo los de los subestimadores si están presentes.

predict: (X)
Predice el cluster más cercano para cada muestra en X.

score: (X, y=None, sample_weight=None)
Devuelve el valor opuesto de la función objetivo del algoritmo K-Means. Usualmente mide la distancia entre puntos y centroides.

set_fit_request: (sample_weight)
Solicita metadatos que serán pasados al método 'fit'. Relevante solo en meta-estimadores como pipelines.

set_output: (transform=None)
Configura el formato de salida del transformador. Las opciones incluyen pandas, polars o el formato por defecto.

set_params: (**params)
Establece los parámetros del estimador. Puede actualizar los parámetros de subestimadores dentro de un pipeline.

set_score_request: (sample_weight)
Solicita metadatos que serán pasados al método 'score'. Relevante solo en meta-estimadores.

transform: (X)
Transforma los datos de entrada en un espacio de distancias a los centroides de los clusters.

3 Repositorio github

`https://github.com/Juan-Alvarado21/Machine-Learning.git`

4 Conclusion

En conclusión, la elaboración del profilling es una herramienta de bastante ayuda a la hora de saber diferentes meta-características de los datos en algún data set proporcionado. Nos ayuda a eliminar filas erroneas, o atributos que tienen una gran correlación con otros, que en la mayoría de los casos sse consideran como redundantes, y se pueden descartar (que también a su vez ayuda a algoritmos con problemas a la hora de expanderse dimensiones).

En la sección 2, se observa como sin nisiquiera tener los etiquetas de clase, el modelo de aprendizaje no supervizado k-means puede automaticamente agrupar los flores en los 3 grupos deseados.