

25 DE MAYO DE 2023

PRÁCTICA 18

Segmentación y Clasificación Múltiple- Modelos Combinados - K-Means - Bosques Aleatorios



Objetivo de la práctica

Por Angel Damian Monroy Mendoza

No. de Cuenta: 316040707

Generar un modelo de clasificación múltiple de un conjunto de datos seleccionado por el alumno, con la finalidad de segmentar datos del comportamiento de compra en una tienda física y hacer una clasificación múltiple en función de los parámetros disponibles.



Características:

- El conjunto de datos contiene información sobre el comportamiento de compra de 2000 personas en una tienda física. Todos los datos se han recopilado a través de las tarjetas de fidelización que utilizan al finalizar la compra.
- Los datos han sido preprocesados y anonimizado para proteger la privacidad de los clientes.

1. ID: Es el identificador único de un cliente.
2. Sex: Género de un cliente (0 = hombre, 1 = mujer).
3. Marital status: Estado civil de un cliente (0 = soltero, 1 = no soltero - divorciado/separado/casado/viudo-).
4. Age: Es la edad del cliente (18 años es el valor mínimo y 76 es el valor máximo).
5. Education: Nivel de educación del cliente (0 = otro/desconocido, 1 = escuela secundaria, 2 = universidad; 3 = posgrado).
6. Income: Es el ingreso anual en dólares autorreportado por el cliente.
7. Occupation: Ocupación del cliente (0 = desempleado/no calificado, 1 = empleado calificado/funcionario, 2 = directivos/autónomos/empleado altamente cualificado/funcionario).
8. Settlement size: Es el tamaño de la ciudad en la que vive el cliente (0 = pequeña ciudad, 1 = ciudad mediana, 2 = gran ciudad).

Desarrollo

Para el desarrollo de esta práctica se realizaron varios procesos que se engloban en los siguientes puntos:

1. Importación de bibliotecas, acceso a datos y selección de características.
2. Modelo Híbrido: Segmentación Particional - (K - Means)
3. Aplicación del algoritmo - Bosques Aleatorios.
4. Nuevos pronósticos.

1. Bibliotecas, datos y selección de características

En este apartado se instalaron las bibliotecas necesarias para la manipulación de los datos, creación de vectores y matrices, generación de gráficas, la implementación de K - Means y, posteriormente, en la sección de *Aplicación del Algoritmo*, se instalaron las bibliotecas para los *Bosques Aleatorios*, así como lo necesario para saber la eficiencia del modelo y sus diferentes métricas de la matriz de confusión como *classification_report*, *confusion_matrix*, *accuracy_score* y *model_selection*.

Posteriormente, para la parte de cargar los datos se realizó mediante un link que se obtiene desde el archivo de GitHub, el cuál se instanció en la variable 'url' y se le pasó como argumento a la función de pandas para leer archivos con extensión csv. Después, se realizó un análisis exploratorio de datos muy superficial, en donde se observó la estructura de los mismos, si contaban con valores nulos o faltantes, una primera extracción de los datos totales de la variable Sexo ('Sex'), una eliminación de la columna de ID con el método '.drop()' y una impresión del mapa de calor de las correlaciones entre variables.

Para la selección de características se decidió utilizar todas las variables debido a que la correlación que se encontró entre la variable 'Income' y la variable 'Occupation' no era muy alta, además de que se cuentan con pocas variables (7), lo cual aumenta el riesgo de caer en sobreajuste. Las variables que se eligieron se muestran en la siguiente página.



The screenshot shows a Jupyter Notebook cell with the following code:

```
url='https://raw.githubusercontent.com/aDamianMo/.../Clients.csv'
Clientes = pd.read_csv(url)
Clientes
```

Below the code, the resulting DataFrame is displayed:

	ID	Sex	Marital status	Age	Education	Inco...
0	100000001	0	0	67	2	124
1	100000002	1	1	22	1	150
2	100000003	0	0	49	1	89
3	100000004	0	0	45	1	171
4	100000005	0	0	53	1	149
...
1995	100001996	1	0	47	1	123
1996	100001997	1	1	27	1	117
1997	100001998	0	0	31	0	86
1998	100001999	1	1	24	1	97
1999	100002000	0	0	25	0	68
2000 rows × 8 columns						

Desarrollo



Variables seleccionadas:

Ante una sola presencia de correlaciones altas (fuertes), se consideran a todas las variables para la construcción de los modelos, pues las dos variables se consideran de alta relevancia para el caso de estudio.

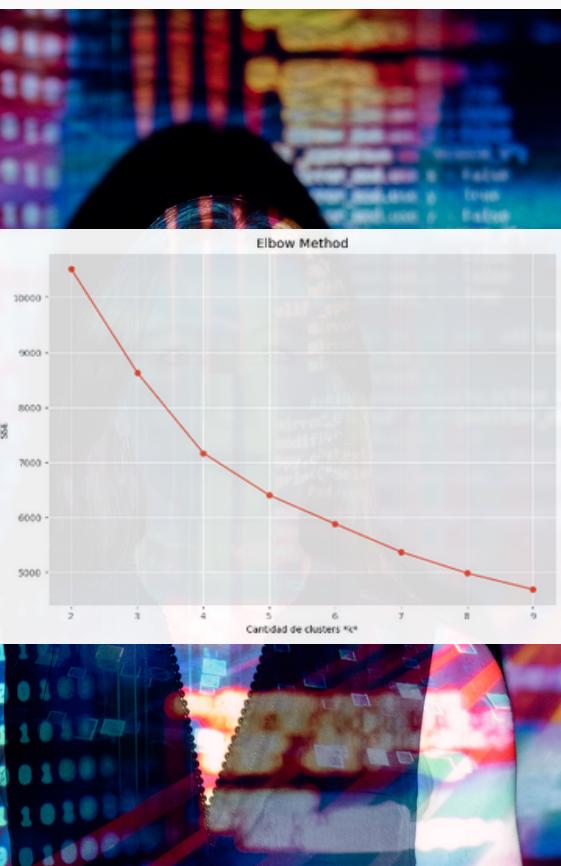
- Sex.
- Marital status.
- Age.
- Education.
- Income.
- Occupation.
- Settlement size.

2. K-Means

Para esta parte se utilizó la biblioteca `sklearn.preprocessing` para importar el método de normalizar y escalar, pues al estar trabajando con datos numéricos que difieren mucho entre ellos, es preciso realizar la estandarización para que cada uno de los valores tenga el mismo peso y aporte de forma equilibrada a los resultados.

Así pues, se eligió el método de estandarización para los datos (escalado) y, posteriormente, se instanció una nueva matriz estandarizada "MEstandarizada" ajustando los datos de la matriz obtenida de la selección de características, al escalado.

Luego, recordemos que con K-Means se tiene que proponer el número de clusters. Por esta razón, se utiliza el "método del codo" en donde se usa la suma total de la distancia al cuadrado dentro de cada grupo (SSE) y se realiza un recorrido entre 2 y 10 en donde se instancia el algoritmo `KMeans`, se ajusta este algoritmo a la matriz `MEstandarizada` y se realiza una lista sobre SSE con elementos de `km` con su atributo `.inertia_`. En seguida, se realiza una gráfica sobre estos SSE vs No. de Clústeres y, dado que en la práctica no siempre se tiene un punto de inflexión notable, se realiza un apoyo con el "método de la rodilla".

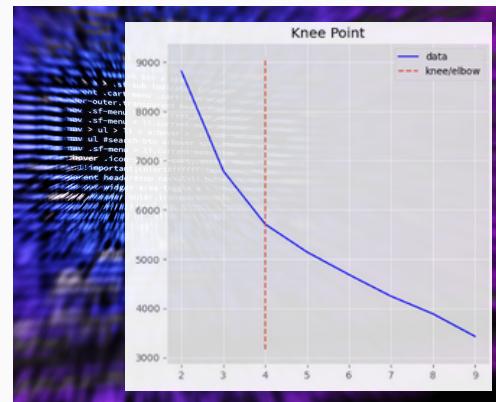




Desarrollo

Para el "método de la rodilla" se instaló `kneed` y, después, se importó la biblioteca de `KneeLocator`. Luego se instanció un localizador en el mismo rango de la SSE y la curva convexa y se imprimió el valor de **cuatro clusters** que arrojó el localizador.

Posteriormente, se crean las etiquetas de los elementos en los clústeres y se imprimen. Se agrega otra columna al conjunto de datos en donde se guardan los valores de las etiquetas para el clustering particional. Se hace una agrupación y conteo de los elementos en cada uno de los cluster y, por último, se obtienen los centroides para hacer el análisis de cada clúster.



3. Aplicación del algoritmo - Bosques Aleatorios

Para este apartado se instalaron las bibliotecas necesarias para la aplicación del algoritmo de bosques aleatorios explicadas al principio de la práctica. Posteriormente, se seleccionan las variables predictoras (X) y la variable a pronosticar (Y)

1er Modelo:

Variables predictoras: 'Sex','Marital status','Age','Education','Income','Occupation','Settlement size'.

Variable a pronosticar: 'clusterP'

Cabe mencionar que, dado que fue un modelo mixto, la salida que se obtuvo del algoritmo de clusterización particional "K-Means", se convierte en la variable objetivo (entrada) de los algoritmos de Árboles y Bosques. En este entendido, se hace la división de los datos utilizando la función de `model_selection` con el método `train_test_split`, en donde se le dan las variables predictoras y la variable a pronosticar; el tamaño del conjunto de testeo, el cual en este caso fue de 20% de prueba (p) y 80% de entrenamiento (e), recordando que es un parámetro que podemos modificar a nuestro criterio.



*Nota: no se presentaron los análisis de los clústeres debido a que éstas se encuentran en el cuaderno.

Desarrollo

3. Aplicación del algoritmo - Bosques Aleatorios

Asimismo, se agrego un estado random para volver reproducible los mismos datos obtenidos todas las veces que corramos nuestro código y un `shuffle = 'True'`, que indica que los datos estén barajeados, es decir, que no agarre los primeros datos para el conjunto de entrenamiento y los últimos para el de prueba, sino que sea al azar la asignación de los datos de prueba y entrenamiento.

Posteriormente, se entrena el modelo creando un objeto para instanciar la función `RandomForestClassifier()` junto con los hiperparámetros ya modificados y luego, a ese objeto se le ajustan los datos de entrenamiento. Los parámetros modificados fueron los siguientes:

```
n_estimators = 105, max_depth = 8, min_samples_split = 4,  
min_samples_leaf = 2, random_state = 1234
```

A continuación, con estos modelos ya entrenados se generan los pronósticos con el método `.predict` y las variables predictoras del testeo o prueba y se imprimen en un `DataFrame` de pandas.

Por último, se obtienen las características necesarias para poder armar nuestros modelos de árbol de decisión, así como para poder evaluar qué tan buenos son. En este sentido, se imprime la exactitud, la matriz de clasificación, la matriz de confusión y la importancia de las variables:

accuracy_score(Y_validation, Y_ClasicacionBA)				
0.9875				
Clasificación	0	1	2	3
Reales				
0	141	1	0	0
1	0	93	0	2
2	0	2	113	0
3	0	0	0	48

Importancia variables:				
[0.12102911 0.26985737 0.11054991 0.14932479 0.07764031 0.08059688 0.19100163]				
Exactitud: 0.9875				
precision recall f1-score support				
0	1.00	0.99	1.00	142
1	0.97	0.98	0.97	95
2	1.00	0.98	0.99	115
3	0.96	1.00	0.98	48
accuracy				
macro avg	0.98	0.99	0.99	400
weighted avg	0.99	0.99	0.99	400

*Nota: no se presentaron los análisis de las configuraciones debido a que éstas se encuentran en el cuaderno.

Desarrollo

Nuevos pronósticos.

Por último, para el modelo de bosques aleatorios se realizó un pronóstico de un cliente ficticio con ciertos valores de entrada (variables predictoras) y con el modelo que se guardó en el objeto de *ClasificacionBA*, se pronosticó la variable objetivo.

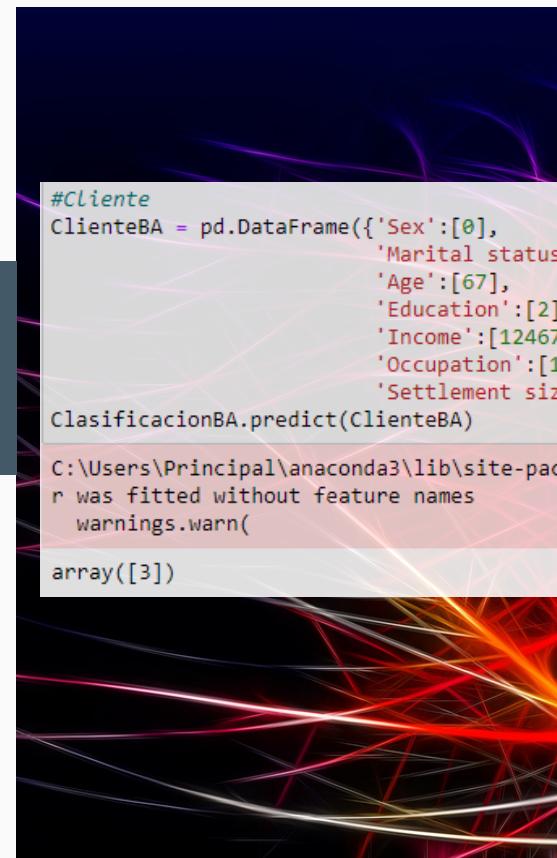
5. Comparación y Conclusiones

Además, se realizaron las validaciones del modelo de bosques aleatorios, observando que, como de costumbre, el bosque aleatorio presenta un buen desempeño. Cabe mencionar que se realizaron más modelos no documentados. Algunos fueron quitando la variable 'Occupation' por presentar alta correlación, sin embargo, la exactitud era del 100% lo que indicaba sobre ajuste.

Por último, se analizó el rendimiento de cada una de las clases del modelo de bosque aleatorio obteniendo lo siguiente:

```
AUC para la clase 1: 0.9997816355497325
AUC para la clase 2: 0.999792924935289
AUC para la clase 3: 0.9995423340961098
AUC para la clase 4: 1.0
```

Lo cual nos dice cosas interesantes del rendimiento, pues nuestro modelo puede identificar perfectamente a la clase 4, mientras quelas demás clases se pueden diferenciar delas demás de una forma excepcional. No obstante, estamos a expensas del especialista para catalogar las clases que se obtuvieron de la clusterización. En conclusión, los modelos combinados generan muy buenos resultados, aunque le restan un poco de comprensión, pues al no tener experiencia en el campo de estudio, se tiene que apoyar con un especialista para hacer más concreto el significado de cada una de las clases generadas.



```
#Cliente
ClienteBA = pd.DataFrame({'Sex':[0],
                           'Marital status': [1],
                           'Age': [67],
                           'Education': [2],
                           'Income': [12467],
                           'Occupation': [1],
                           'Settlement size': [1]})

ClasificacionBA.predict(ClienteBA)

C:\Users\Principal\anaconda3\lib\site-packages\sklearn\ensemble\_base.py:415: UserWarning: Estimator was fitted without feature names
  warnings.warn(
array([3])
```

