Preprocesamiento de datos en R

El preprocesamiento de datos es la fase inicial de Aprendizaje Máquina en la que los datos se preparan para los modelos de aprendizaje automático. Esta parte es crucial y debe realizarse de manera adecuada y sistemática. De lo contrario, terminaremos construyendo modelos que no son precisos para su propósito.

En este documento, aprenderemos todos los pasos que están involucrados en la etapa de preprocesamiento de datos. Se utilizo la versión 4.1.2.

Requisitos previos

Para seguir este documento, el lector debe tener lo siguiente:

1. R instalado en su computadora.
2. Paquete ('caTools') instalados.

Pasos en el preprocesamiento de datos

- Paso 1: Importación del conjunto de datos
- Paso 2: Manejo de los datos que faltan
- Paso 3: Codificación de datos categóricos.
    - Salida
- Paso 4: dividir el conjunto de datos en conjuntos de entrenamiento y prueba
    - Conjunto de entrenamiento
    - Conjunto de prueba
- Paso 5: Escalado de características
    - conjunto de entrenamiento (training_set)
    - conjunto de prueba (test_set)

Paso 1: Importación del conjunto de datos

Antes de comenzar a preparar nuestros datos, primero debemos cargarlos en R.
Aquí está cómo lograr esto. No olvidar que los archivos se guardan en la carpeta Documentos de Windows.

```
> install.packages("readr")
> library(readr)
> Dataset = read_csv('dataset.csv')
```

Este código importa nuestros datos almacenados en formato CSV.
Podemos echar un vistazo a nuestros datos usando la función 'View()':

Al ejecutar, obtenemos nuestro conjunto de datos como se muestra a continuación.

> View(Dataset)

Salida

	Country	Age	Salary	Purchased
1	France	44	72000	No
2	Spain	27	48000	Yes
3	Germany	30	54000	No
4	Spain	38	61000	No
5	Germany	40	NA	Yes
6	France	35	58000	Yes
7	Spain	NA	52000	No
8	France	48	79000	Yes
9	Germany	50	83000	No
10	France	37	67000	Yes

Nuestro conjunto de datos tiene cuatro columnas y diez observaciones, muestra cómo los clientes de tres países diferentes con diferentes edades y salarios respondieron a la compra de un determinado producto.

Paso 2: Manejo de los datos que faltan

Del conjunto de datos, la columna Edad (Age) y Salario (Salary) contienen datos que faltan. Antes de implementar nuestros modelos de aprendizaje automático, este problema debe resolverse; de lo contrario, causará un problema grave a nuestros modelos de aprendizaje automático. Por lo tanto, es nuestra responsabilidad asegurarnos de que estos datos faltantes se eliminen de nuestro conjunto de datos utilizando la técnica más adecuada.

Se presentan dos técnicas que podemos usar para manejar los datos faltantes:

1. Elimina la observación que reporta los datos faltantes:

Esta técnica es adecuada cuando se trata de grandes conjuntos de datos y con muy pocos valores faltantes, es decir, eliminar una fila de un conjunto de datos con miles de observaciones no puede afectar la calidad de los datos. Cuando en el conjunto de datos reportados faltan muchos valores, puede ser muy peligroso utilizar esta técnica. Eliminar

muchas filas de un conjunto de datos puede provocar la pérdida de información crucial contenida en los datos.

Para garantizar que esto no suceda, hacemos uso de una técnica adecuada que no daña la calidad de los datos.

2. Reemplaza los datos que faltan con el promedio de la función en la que faltan los datos:

Esta técnica es la mejor forma hasta ahora de tratar con los valores faltantes. Muchos estadísticos hacen uso de esta técnica sobre la primera.

Ahora que conocemos las técnicas utilizadas para tratar los datos faltantes, resolvamos este problema a partir de nuestros datos. En nuestro caso, haremos uso de la segunda técnica.

Comencemos reemplazando los datos que faltan en la columna Edad con la media de esa columna.

El siguiente código lleva a cabo tal tarea.

```
> Dataset$Age = ifelse(is.na(Dataset$Age),
+ ave(Dataset$Age, FUN = function (x)mean(x, na.rm = TRUE)),
+ Dataset$Age)
```

¿Qué hace realmente el código anterior?

Dataset$Age: simplemente toma la columna Age de nuestro conjunto de datos.

En la columna Age, acabamos de tomar eso de nuestro conjunto de datos, necesitamos reemplazar los datos que faltan y, al mismo tiempo, mantener los datos que no faltan.

Este objetivo se logra mediante el uso de la instrucción if-else.

Nuestra declaración ifelse está tomando tres parámetros:

- El primer parámetro es si la condición es verdadera.
- El segundo parámetro es el valor que ingresamos si la condición es verdadera.
- El tercer parámetro es la acción que tomamos si la condición es falsa.

Nuestra condición es is.na(Dataset$Age). Esto nos dirá si falta o no un valor en Dataset$Age. Devuelve una salida lógica, YES si falta un valor y NO si no falta un valor. El segundo parámetro, la función 'ave()', encuentra la media de la columna Edad.

Debido a que esta columna informa valores NA, necesitamos excluir los datos nulos en el cálculo de la media, de lo contrario, obtendremos la media como NA.

Esta es la razón por la que pasamos na.rm = TRUE en nuestra función de media para declarar los valores que deben usarse y aquellos que deben excluirse al calcular la media del vector Age.

La tercera condición es el valor que se devolverá si no falta el valor de la columna Age del conjunto de datos.

Utilizando nuevamente:
> View(Dataset)

	Country	Age	Salary	Purchased
1	France	44.00000	72000	No
2	Spain	27.00000	48000	Yes
3	Germany	30.00000	54000	No
4	Spain	38.00000	61000	No
5	Germany	40.00000	NA	Yes
6	France	35.00000	58000	Yes
7	Spain	38.77778	52000	No
8	France	48.00000	79000	Yes
9	Germany	50.00000	83000	No
10	France	37.00000	67000	Yes

El valor faltante que estaba en la columna Age de nuestro conjunto de datos se reemplazó correctamente con la media de la misma columna.

Hacemos lo mismo para la columna Salary ejecutando el siguiente código:

```
> Dataset$Salary = ifelse(is.na(Dataset$Salary),
+ ave(Dataset$Salary, FUN = function (x)mean(x, na.rm = TRUE)),
+ Dataset$Salary)
```

> View(Dataset)

	Country	Age	Salary	Purchased
1	France	44.00000	72000.00	No
2	Spain	27.00000	48000.00	Yes
3	Germany	30.00000	54000.00	No
4	Spain	38.00000	61000.00	No
5	Germany	40.00000	63777.78	Yes
6	France	35.00000	58000.00	Yes
7	Spain	38.77778	52000.00	No
8	France	48.00000	79000.00	Yes
9	Germany	50.00000	83000.00	No
10	France	37.00000	67000.00	Yes

El valor faltante que estaba en la columna Salary se reemplazó correctamente con la media de la misma columna.

Paso 3: Codificación de datos categóricos

La codificación se refiere a la transformación de datos de texto en datos numéricos. Codificar datos categóricos simplemente significa que estamos transformando datos que caen en categorías en datos numéricos.

En nuestro conjunto de datos, la columna Country son datos categóricos con 3 niveles, es decir, Francia, España y Alemania. La columna Purchased también tiene datos categóricos con 2 categorías, es decir, SÍ y NO.

Los modelos de máquinas que construimos en nuestro conjunto de datos se basan en ecuaciones matemáticas y solo toman números en esas ecuaciones.

Mantener textos de una variable categórica en la ecuación puede causar algunos problemas a los modelos de aprendizaje automático y es por eso qué codificamos esas variables. Para transformar una variable categórica en numérica, usamos la función factor().

Comencemos codificando la columna Country.

```
> Dataset$Country = factor(Dataset$Country,
+ levels = c('France','Spain','Germany'),
+ labels = c(1.0, 2.0 , 3.0 ))
>
> View(Dataset)
```

	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	No
2	2	27.00000	48000.00	Yes
3	3	30.00000	54000.00	No
4	2	38.00000	61000.00	No
5	3	40.00000	63777.78	Yes
6	1	35.00000	58000.00	Yes
7	2	38.77778	52000.00	No
8	1	48.00000	79000.00	Yes
9	3	50.00000	83000.00	No
10	1	37.00000	67000.00	Yes

Los nombres de nuestros países fueron reemplazados con éxito por números.

Hacemos lo mismo para la columna Purchased.

```
> Dataset$Purchased = factor(Dataset$Purchased,
+ levels = c('No', 'Yes'),
+ labels = c(0, 1))
> Dataset$Purchased[is.na(Dataset$Purchased)] <- 0
> as.factor(Dataset$Purchased)
 [1] 0 1 0 0 1 1 0 1 0 1
Levels: 0 1
> View(Dataset)
```

	Country	Age	Salary	Purchased
1	1	44.00000	72000.00	0
2	2	27.00000	48000.00	1
3	3	30.00000	54000.00	0
4	2	38.00000	61000.00	0
5	3	40.00000	63777.78	1
6	1	35.00000	58000.00	1
7	2	38.77778	52000.00	0
8	1	48.00000	79000.00	1
9	3	50.00000	83000.00	0
10	1	37.00000	67000.00	1

Nuestra columna Purchased se codificó con éxito en 0,sy 1,s.

Paso 4: Dividir el conjunto de datos en el conjunto de entrenamiento y prueba

En el aprendizaje automático, dividimos los datos en dos partes:

- Conjunto de entrenamiento: la parte de los datos en la que implementamos nuestro modelo de aprendizaje automático.
- Conjunto de prueba: la parte de los datos en la que evaluamos el rendimiento de nuestro modelo de aprendizaje automático.

La razón por la que dividimos estos datos es para asegurarnos de que nuestro modelo de aprendizaje automático no sobre aprenda la correlación de los datos en los que está entrenado. Si dejamos que aprenda demasiado sobre los datos, puede tener un desempeño deficiente cuando se prueba en un nuevo conjunto de datos con una correlación diferente.

Por lo tanto, siempre que estemos construyendo un modelo de aprendizaje automático, la idea es implementarlo en el conjunto de entrenamiento y evaluarlo en el conjunto de prueba. Esperamos que el rendimiento en el conjunto de entrenamiento y el conjunto de prueba sea diferente y, si este es el caso, el modelo puede adaptarse a nuevos conjuntos de datos.

Usando nuestro conjunto de datos, dividámoslo en conjuntos de entrenamiento y prueba.

Para empezar, primero instalamos y cargamos la biblioteca requerida.

```
> install.packages("caTools")
> library(caTools)
> set.seed(123)
> split = sample.split(Dataset$Purchased, SplitRatio = 0.8)
> training_set = subset(Dataset, split == TRUE)
> test_set = subset(Dataset, split == FALSE)
> training_set
# A tibble: 8 x 4
  Country   Age Salary Purchased
  <fct>   <dbl>  <dbl> <fct>
1 1        44   72000  0
2 2        27   48000  1
3 3        30   54000  0
4 2        38   61000  0
5 3        40   63778. 1
6 2        38.8 52000  0
7 1        48   79000  1
8 1        37   67000  1
```

```
> test_set
# A tibble: 2 x 4
  Country   Age Salary Purchased
  <fct>   <dbl>  <dbl> <fct>
1 1          35  58000 1
2 3          50  83000 0
```

A partir de la salida, queda claro que dos observaciones fueron al conjunto de prueba.

Paso 5: Escalado de características

Es un caso común en la mayoría de los conjuntos de datos, las características también conocidas como entradas no están en la misma escala. Muchos modelos de aprendizaje automático se basan en la distancia euclidiana.

Sucede que, las entidades con unidades grandes dominan a las de unidades pequeñas a la hora de calcular la distancia euclidiana y será como si esas entidades con unidades pequeñas no existieran.

Para asegurarnos de que esto no ocurra, necesitamos codificar nuestras funciones para que todas estén en el rango entre -3 y 3. Hay varias formas que podemos usar para escalar nuestras funciones. La más utilizada es la técnica de estandarización y normalización.

La técnica de normalización se usa cuando los datos se distribuyen normalmente, mientras que la estandarización funciona tanto con datos distribuidos normalmente como con datos que no se distribuyen normalmente.

La fórmula para estas dos técnicas se muestra a continuación.

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

Ahora, vamos a escalar tanto el conjunto de entrenamiento como el conjunto de prueba de nuestro conjunto de datos por separado.

Así es como logramos esto:

```
> training_set[, 2:3] = scale(training_set[, 2:3])
> test_set[, 2:3] = scale(test_set[, 2:3])
> training_set
# A tibble: 8 x 4
  Country     Age Salary Purchased
  <fct>     <dbl>  <dbl> <fct>
1 1        0.901   0.939 0
2 2       -1.59   -1.34  1
3 3       -1.15   -0.768 0
4 2        0.0224 -0.104 0
5 3        0.315   0.159 1
6 2        0.136  -0.958 0
7 1        1.49    1.60  1
8 1       -0.124   0.465 1
> test_set
# A tibble: 2 x 4
  Country    Age Salary Purchased
  <fct>    <dbl>  <dbl> <fct>
1 1       -0.707 -0.707 1
2 3        0.707  0.707 0
```

Nuestro conjunto de entrenamiento y prueba se escalaron con éxito.

Conclusión

Este es el final de nuestro viaje de preprocesamiento de datos. Nuestros datos ahora están bien preparados para brindar modelos de aprendizaje automático que pueden predecir resultados futuros de manera efectiva. Espero que hayas disfrutado cada paso hasta este final.