Resumen del capítulo: preparación de características

Recapitulemos. La técnica OHE nos permite transformar características categóricas en características numéricas en dos pasos:

Codificación One-Hot

Existe una técnica especial para transformar características categóricas en características numéricas. Se llama codificación one-hot (OHE).

1. Agrega una columna separada para cada valor de característica.
2. Si la categoría se ajusta a la observación, se asigna 1; de lo contrario, se asigna 0.

Las nuevas funciones (`Gender_F`, `Gender_M`, `Gender_None`) se denominan **variables dummy**.

Puedes obtener variables dummy para OHE usando la función **get_dummies()**. Se puede encontrar en la librería pandas.

```
pd.get_dummies(df['column'])
```

Trampa dummy

Cuando los datos son abundantes, tenemos la posibilidad de caer en la trampa de las características dummy. Hemos agregado tres columnas nuevas a nuestra tabla, pero su alta correlación confundirá a nuestro modelo. Para evitar esto, podemos eliminar con seguridad cualquier columna, ya que sus valores se pueden deducir fácilmente de las otras dos columnas (tiene 1 donde las otras dos columnas tienen ceros y tiene ceros en el resto). De esta manera, no caeremos en la trampa dummy.

Para eliminar la columna, llama a la función `pd.get_dummies()` junto con el parámetro **drop_first**. Si pasas `drop_first=True` entonces se elimina la primera columna. De lo contrario, es `drop_first=False` por defecto y no se descartan columnas.

```
pd.get_dummies(df['column'], drop_first=teTrue)
```

Codificación ordinal

Necesitamos una nueva técnica de codificación que permita codificar categorías textuales con números. Vamos a usar codificación ordinal. Funciona de la siguiente manera:

1. Codifica cada clase con un número.
2. Los números se ponen en las columnas.

Sklearn proporciona una clase para dicha codificación. Se llama **OrdinalEncoder**. Puedes encontrarla en el módulo `sklearn.preprocessing`.

Importa `OrdinalEncoder` de la librería:

```
from sklearn.preprocessing import OrdinalEncoder
```

La transformación se realiza en tres pasos:

1. Crea una instancia de esta clase..

```
encoder = OrdinalEncoder()
```

2. Llama al método `fit()` para obtener la lista de características categóricas, el mismo proceso que hacemos cuando entrenamos un modelo. Pásale los datos.

```
encoder.fit(data)
```

3. Utiliza el método `transform()`. Los datos transformados se almacenarán en la variable `data_ordinal`.

```
data_ordinal = encoder.transform(data)
```

Use `DataFrame()` para agregar nombres de columna:

```
data_ordinal = pd.DataFrame(encoder.transform(data),
                            columns=data.columns)
```

Si necesitamos transformar los datos solo una vez, como en nuestro caso, también puedes llamar al método `fit_transform()`. Este combina `fit()` y `transform()`.

```
data_ordinal = pd.DataFrame(encoder.fit_transform(data),
                            columns=data.columns)
```

Escalado de características

Si tenemos datos con características numéricas que tienen diferente dispersión de valores, el algoritmo puede encontrar que las características con mayores magnitudes y dispersión son más importantes. Este problema se puede solucionar con el escalado de características.

Una de las formas de escalar las características es estandarizar los datos.

Supón que todas las características se distribuyen normalmente, la media (M) y la varianza (Var) se determinan a partir de la muestra. Los valores de las características se convierten mediante esta fórmula:

$$\text{Nuevo valor} = \frac{\text{Valor antiguo} - \text{M}}{\sqrt{\text{Var}}}$$

Para la nueva característica, la media se convierte en 0 y la varianza es igual a 1.

Hay una clase de sklearn dedicada a la estandarización de datos que se llama StandardScaler. It's in the *sklearn.preprocessing* module.

```
from sklearn.preprocessing import StandardScaler
```

Crea una instancia de la clase y ajústala usando los datos de entrenamiento. El proceso de ajuste implica calcular la media y la varianza:

```
scaler = StandardScaler()
scaler.fit(df)
```

Transforma el conjunto de entrenamiento y el conjunto de validación usando transform().

```
df_scaled = scaler.transform(df)
```