Autor:
Iván García Santillán

El preprocesamiento de datos

Introducción

El procesamiento de datos implica la preparación de los datos crudos para su posterior análisis utilizando algoritmos de aprendizaje automático. En general, algunas de las tareas comunes de preprocesamiento de datos incluyen:

- Limpieza de datos para tratar valores faltantes (nulos), valores atípicos y datos inconsistentes.
- Escalado o normalización de datos numéricos.
- Transformación (codificación) de datos categóricos en un formato adecuado para su análisis.
- Reducción de la dimensionalidad si fuese necesario.

A continuación, se detallan las tareas mencionadas utilizando ejercicios prácticos con el lenguaje Python (InteractiveChaos, 2023).

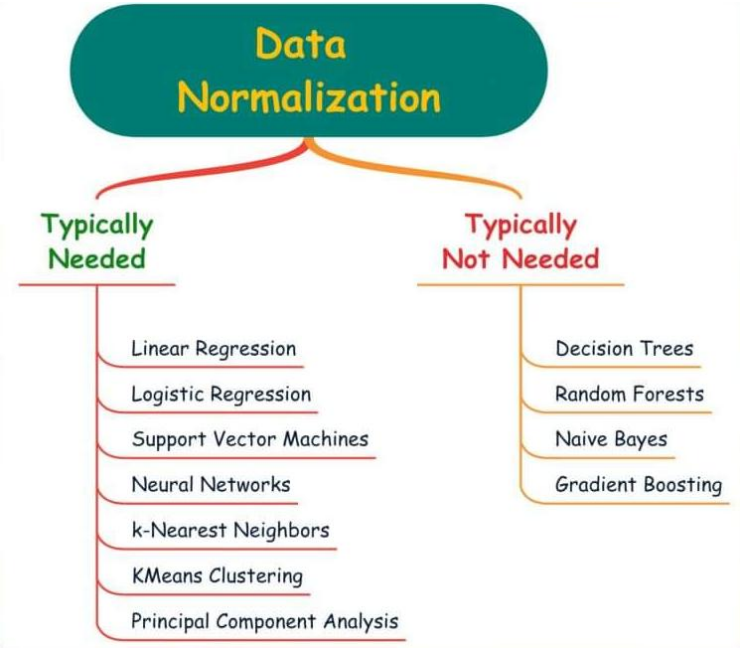
Contenido

Tareas en la preparación de los datos:

Escalado o normalización de datos numéricos:
Antes de entrenar un algoritmo con nuestros datos de entrenamiento debemos asegurarnos de que éstos se encuentran en la forma adecuada.
Algunos algoritmos (como RNA, SVM, K-NN, Logistic regression, ...) son muy sensibles a la escala de los datos (cm, m, km, miles, millones, etc.) (a diferencia de Decision Tree, random forests que son más robustos en este sentido), tendiendo a dar mayor o menor importancia a aquellas características que tomen valores mayores, ver Figura 6. Desde este punto de vista, representar una distancia en kilómetros o en metros no va a modificar el concepto representado

(distancia), pero puede tener un impacto indeseable en el comportamiento de ciertos algoritmos. Por lo que resulta conveniente transformarlos de forma que todas las características (predictoras) tenga una escala semejante.

Figura 6 Normalización de datos



Fuente: InteractiveChaos (2023)

Para aplicar un escalado semejante a las diferentes características (variables predictoras, no es necesario escalar la variable objetivo) tenemos dos enfoques diferentes: el escalado y la normalización.

El **escalado** va a transformar los valores de las características de forma que estén confinados en un rango [a, b], típicamente [0, 1] o [-1, 1].

La **normalización** va a transformar las características de forma que todas compartan un mismo valor medio ($u=0$) y una misma desviación estándar ($\sigma=1$). Las variables categóricas no son susceptibles de aplicar normalización.

En python, *Scikit-Learn* library ofrece diferentes escaladores: *Standard Scaler* ($u=0$, $\sigma=1$), *MinMaxScaler* (*rango [a, b]*).

from sklearn.preprocessing import ***StandardScaler***
from sklearn.preprocessing import ***MinMaxScaler***

Gestión de valores nulos:

La posible existencia de valores nulos en nuestro conjunto de datos es un problema que también deberemos solucionar (imputación de datos).
Veamos algunas técnicas para resolver este problema:

- **Eliminación de muestras o de características** (x): Este método puede ser aplicable solo si el tamaño de los datos es suficientemente grande. De otra forma podemos estar perdiendo demasiada información. Aquí podemos verificar el % de nulos que hay en cada variable (< 10% de nulos podría ser aceptable).

- **Reemplazo por la media, mediana o moda (imputación de datos)**: Es aplicable solo cuando la característica es numérica (la moda también es aplicable a características categóricas). Una versión más sofisticada es el reemplazo de aquellos valores de la característica que pertenecen a un determinado grupo. Debemos ser conscientes, en todo caso, que estamos añadiendo información al modelo que puede añadir *sesgo o varianza* a los datos. En Python, S*cikit-Learn library* ofrece la clase:

  sklearn.impute.SimpleImputer

- **Asignación a una categoría exclusiva:** Si estamos trabajando con una variable categórica, podemos crear una categoría nueva (ej. *unknow*) y asignarla a todos los valores nulos.

- **Predicción de los valores nulos:** usando alguna técnica de predicción como p. ej. regresión lineal o múltiple. Un problema que puede surgir es que los valores imputados pertenezcan a un rango que no pueda darse en la realidad (edad o sueldo con valores negativos).

```
from sklearn.linear_model import LinearRegression
```

Gestión de características categóricas:

Ya se ha comentado que los algoritmos generalmente necesitan valores numéricos para poder ser entrenados, lo que nos obliga a codificar adecuadamente los posibles valores categóricos que existan en nuestro conjunto de datos.

Frecuentemente nos encontramos con *datasets* que incluyen características **categóricas** (nominal), normalmente almacenadas con formato de texto (género, estado civil, ciudad, país, …). Generalmente, los algoritmos de aprendizaje automático y las redes neuronales de aprendizaje profundo (Deep Learning) requieren que las variables de entrada y salida sean números..

Aunque hay ciertos algoritmos (como *decision tree*) capaces de trabajar con variables de tipo texto, la mayor parte exigen valores numéricos, por lo que la pregunta que debemos hacernos es ¿cómo convertir estos valores en números? Esto significa que los datos categóricos deben codificarse en números antes de que podamos usarlos para ajustar y evaluar un modelo. Como suele ocurrir en el área de *Data Science*, la respuesta no es única. Cada enfoque tiene sus ventajas e inconvenientes, y será el entorno (nuestros datos, problema, modelo) el que nos lleve en cada caso a decantarnos por una u otra opción. Revisemos los dos enfoques más frecuentemente usados: ***Label Encoding*** y ***One-Hot Encoding***. El uso de algún enfoque tiene un gran impacto en la precisión de la predicción.

Label Encoding:

En este enfoque cada etiqueta única se asigna a un número entero. Identificar los distintos valores existentes (etiquetas/clases) y sustituir cada uno de ellos por un número (0, 1, 2, …. N-1). De esta forma, podríamos sustituir "male" por el valor 0 y "female" por el valor 1, por ejemplo.

El método de *Label Encoding* tiene la ventaja de ser sencillo de implementar. Sin embargo, tiene un cierto problema: los valores numéricos pueden ser malinterpretados por algunos algoritmos (como RNA): si hemos codificado varias ciudades con los valores 0, 1, 2 y 3 ¿significa que la ciudad correspondiente al valor 3 (digamos Ecuador) es el triple (o mayor) que la que ha recibido el valor 1 (digamos Colombia).

from sklearn.preprocessing import LabelEncoder

One-Hot Encoding (OHE):

One-Hot Encoding es el proceso de creación de variables ficticias donde cada etiqueta se asigna a un vector binario. La estrategia que implementa

es crear una columna para cada valor distinto que exista en la característica que estamos codificando y, para cada registro, marcar con un 1 la columna a la que pertenezca dicho registro y dejar las demás con 0 (Tabla 2).

Tabla 2. One-Hot encoding sobre la variable género

gender	
gender_male	gender_female
1	0
0	1

Scikit-Learn implementa esta funcionalidad en la clase: sklearn.preprocessing.OneHotEncoder

Aplicamos codificación One-Hot cuando:

- La característica categórica (X) es nominal al igual que los países (donde no existe relación entre categorías).
- La cantidad de características categóricas es menor, por lo que la codificación one-hot se puede aplicar de manera efectiva.

Aplicamos codificación Label-Encoding cuando:

- El rasgo categórico es ordinal como Likert.
- El número de categorías es bastante grande, ya que la codificación one-hot puede provocar un alto consumo de memoria.

Taller:

Realice el preprocesamiento de datos en el dataset "***Titanic***" dado por la librería *seaborn que contiene 15 columnas y 891 registros* (Tabla 3):

```
import seaborn as sns
titanic = sns.load_dataset("titanic")
titanic.head()
```

Tabla 3. Variables del dataset Titanic

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

A continuación, se presenta un código básico de Python con la explicación (comentarios) respectiva (adaptado de Gonzalez, 2023):

```
########## librerías a utilizar ##########

#Se importan la librerías a utilizar
import numpy as np
import pandas as pd


########## Importando la data ##########

#Importar los datos de los archivos .csv almacenados
df_test = pd.read_csv('titanic_test.csv')
df_train = pd.read_csv('titanic_train.csv')

print(df_test.head())
print(df_train.head())

########## Entendimiento de la data ##########

#Verifica la cantidad de datos que hay en los dataset
print('Cantidad de datos:')
print(df_train.shape)
print(df_test.shape)

#Verifica el tipo de datos contenida en ambos dataset
print('Tipos de datos:')
print(df_train.info())
print(df_test.info())

#Verifica los datos faltantes de los dataset
print('Datos faltantes:')
print(pd.isnull(df_train).sum())
print(pd.isnull(df_test).sum())

#Verifica las estadísticas básicas del dataset
print('Estadísticas del dataset:')
print(df_train.describe())
print(df_test.describe())

########## Preprocesamiento de la data ##########

# Transforma los datos de la variable sexo (categórico) en números
df_train['Sex'].replace(['female','male'],[0,1],inplace=True)
df_test['Sex'].replace(['female','male'],[0,1],inplace=True)
```

```
#Transforma los datos de embarque (categórico) en números
df_train['Embarked'].replace(['Q','S', 'C'],[0,1,2],inplace=True)
df_test['Embarked'].replace(['Q','S', 'C'],[0,1,2],inplace=True)

#Reemplazo los datos faltantes en la edad por la media de esta variable
print(df_train["Age"].mean())
print(df_test["Age"].mean())
promedio = 30
df_train['Age'] = df_train['Age'].replace(np.nan, promedio)
df_test['Age'] = df_test['Age'].replace(np.nan, promedio)

#Crea varios grupos/rangos de edades
#Rangos: 0-8, 9-15, 16-18, 19-25, 26-40, 41-60, 61-100
bins = [0, 8, 15, 18, 25, 40, 60, 100]
names = ['1', '2', '3', '4', '5', '6', '7']
df_train['Age'] = pd.cut(df_train['Age'], bins, labels = names)
df_test['Age'] = pd.cut(df_test['Age'], bins, labels = names)

#Se elimina la columna de "Cabin" ya que tiene muchos datos perdidos
df_train.drop(['Cabin'], axis = 1, inplace=True)
df_test.drop(['Cabin'], axis = 1, inplace=True)

#Elimina las columnas que se considera que no son necesarias para el analisis
df_train = df_train.drop(['PassengerId','Name','Ticket'], axis=1)
df_test = df_test.drop(['Name','Ticket'], axis=1)

#Se elimina las filas con datos perdidos
df_train.dropna(axis=0, how='any', inplace=True)
df_test.dropna(axis=0, how='any', inplace=True)

#Verifica los datos
print(pd.isnull(df_train).sum())
print(pd.isnull(df_test).sum())

print(df_train.shape)
print(df_test.shape)

print(df_test.head())
print(df_train.head())
```

El tiempo dedicado al preprocesamiento de datos en un proyecto de análisis de datos varía, pero comúnmente se estima que representa una gran parte del esfuerzo total del proyecto. Según diversas fuentes y la experiencia en la industria, el preprocesamiento de datos puede consumir entre el 50% y el 80% del tiempo total del proyecto. Las otras fases del

análisis de datos pudiendo resultar así: Análisis Exploratorio de Datos (10-20%), Modelado (10-20%), Implementación y Comunicación (5-10%). Esta distribución puede variar dependiendo del proyecto específico, la calidad y cantidad de los datos, y las herramientas utilizadas.
En el siguiente meme nos deja una moraleja/mensaje del esfuerzo requerido en el preprocesamiento de datos.

Fuente: (Sociedad Ecuatoriana de Estadística, 2023)

Trabajo autónomo: Este ejemplo nos ayuda a comprender el impacto de Label y One-Hot encoder en un modelo para un problema de clasificación binaria usando una MLP (multi-layer perceptron) y el dataset *Breast-cancer* que contiene 30 columnas (clic aquí)

Conclusiones

En esta lectura se ha revisado los fundamentos de la ciencia de datos, respecto al preprocesamiento de datos. Antes de realizar el análisis de los datos, es necesario

realizar un preprocesamiento de los datos. Este paso implica principalmente la limpieza de datos, Escalado o normalización de datos numéricos y Transformación (codificación) de datos categóricos en un formato adecuado para su análisis a través de algoritmos de aprendizaje automático como los que se verán más adelante. Se motiva a los estudiantes a revisar material y ejemplos adicionales disponibles libremente en Internet.

ANEXOS

Referencias:

Gonzalez, L. (2023). Predecir la supervivencia del Titanic. http://ligdigonzalez.com/predecir-la-supervivencia-del-titanic-utilizando-python/

InteractiveChaos. (2023). Preprocesamiento de datos. https://interactivechaos.com/en/node/916