17 Mar Dataset Desbalanceado: Prediciendo eventos muy poco frecuentes

PUBLICADO A LAS 11:24H EN ANÁLISIS DE DATOS, BUSINESS, DATOS, MACHINE LEARNING, PYTHON, TRATAMIENTO DATOS POR EDITOR

Sucesos infrecuentes de alto impacto en el negocio

En nuestros procesos de negocio gestionamos **eventos** que ocurren con relativa **poca frecuencia** pero tienen un **gran impacto** en nuestros resultados.

Algunos de estos eventos raros son **favorables:**

- Cerrar una venta cuantiosa, pero difícil
- Identificar un cliente que nos trae otro
- Recuperar un cliente que va a abandonar

Y otros sucesos son muy **desfavorables:**

- Un error en un proceso
- Una máquina que deja de funcionar correctamente
- Una enfermedad rara

- La materialización de un impago u otro tipo de riesgo
- La ocurrencia de un fraude que nos perjudica

Surge la necesidad de identificar estos sucesos para **potenciar** los favorables y para **prevenir, corregir o mitigar** los desfavorables.

Identificando los sucesos con la ciencia de datos: modelos predictivos

La ciencia de datos o **data science** nos ayuda a identificar con más precisión dónde, cuándo y porqué ocurren esos sucesos vitales para el **éxito** de nuestro negocio y alcanzar los **objetivos.** Utilizamos algoritmos de inteligencia artificial o **modelos predictivos** que transforman el conocimiento acumulado en las bases de datos en estas reglas que identifican esos sucesos.

Entrenando modelos de eventos raros

Ocurre a veces, que hay que identificar estos sucesos relativamente escasos de una gran cantidad de situaciones en las que no ocurre nada destacable, como en los ejemplos que mencionábamos antes: venta exitosa / no venta; error / funcionamiento normal; fraude / transacción normal. Vemos que una de las dos categorías opuestas es mucho más frecuente que la otra.

Los **modelos predictivos** tienden a aprender mejor las situaciones que más se les muestran. Dicho de otra forma, si no hacemos nada, tendremos problemas para predecir las situaciones más infrecuentes.

Estas bases de datos con eventos muy raros, técnicamente, se llaman conjuntos de datos o **dataset desblanceados**.

Técnicamente, ¿qué es un Dataset desbalanceado?

Es un conjunto de datos en el que el número de observaciones no es el mismo para las distintas clases que deseamos distinguir en el dataset. Algunas de las clases son muy frecuentes y otras muy raras.

¿Qué ocurre si trabajamos con este tipo de Dataset?

Si queremos realizar de una forma correcta la clasificación de esa clase

minoritaria (por ejemplo, personas que van a realizar fraude) debemos tener unos datos con un número de observaciones similar para cada clase. De no ser así los algoritmos tienden a favorecer la clase con mayor proporción de observaciones pudiendo obtener un modelo que no predice de forma correcta la clase minoritaria.

En este post, trabajaremos con un dataset en el cual encontramos dos clases:

Clase 0: No fraude.
Clase 1: Fraude.

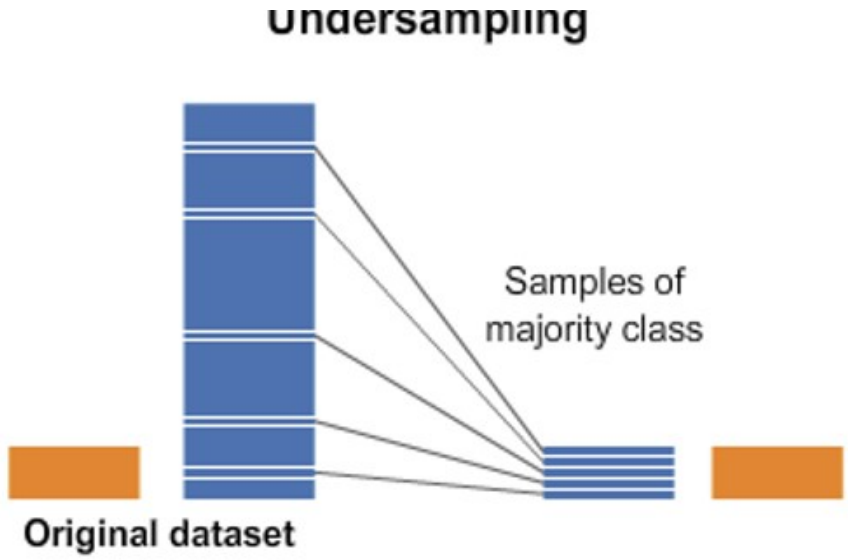
Análisis exploratorio de los datos

Nuestro conjunto de datos está desbalanceado, tenemos más registros pertenecientes a la clase 0 (no fraude) y prácticamente ninguno en la clase 1 (fraude) que es la que queremos predecir. El modelo predictivo que montemos sobre estos datos tendrá dificultades para identificar que caracteriza a la clase 1 de fraude.

¿Qué podemos hacer?

En primer lugar, intentaremos conseguir más datos. En ocasiones, conseguir un mayor número de registros no es posible. Para resolver el problema disponemos de varias técnicas:

1. **Undersampling:** Consiste en **reducir** el número de casos de la categoría más abundante con el fin de obtener un conjunto de datos balanceado.



Para realizar esta técnica en nuestros datos, proponemos dos maneras (de ambas formas obtenemos resultados similares):

1.1 Utilizando el método **resample** de la librería **sklearn**:

Código Python

Se divide el dataset según la clase (0,1)

Se usa la función resample

Unión de dataframe

```
not_fraud = df[df['Class']==0]
fraud = df[df['Class']==1]

no_fraud_under = resample (not_fraud,
                           replace = False,
                           n_samples = len (fraud),
                           random_state = 123)

df_under = pd.concat([no_fraud_under,fraud])
```

1.2 Utilizando el método **RandomUnderSampler** de la librería **imblearn:**

Código Python

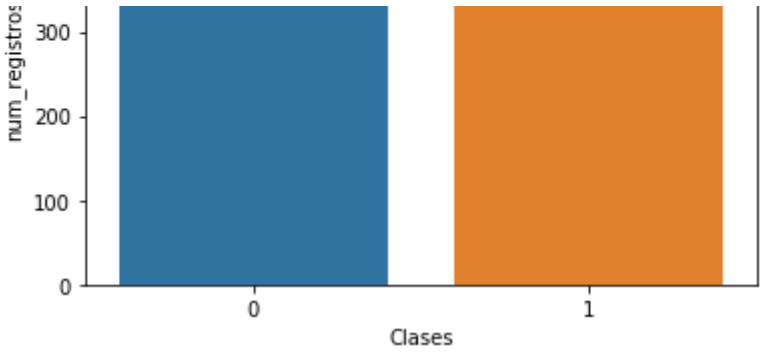
Se divide el dataset en:
- Variables explicativas (x)
- Variable objetivo (y)

Creación y uso del modelo

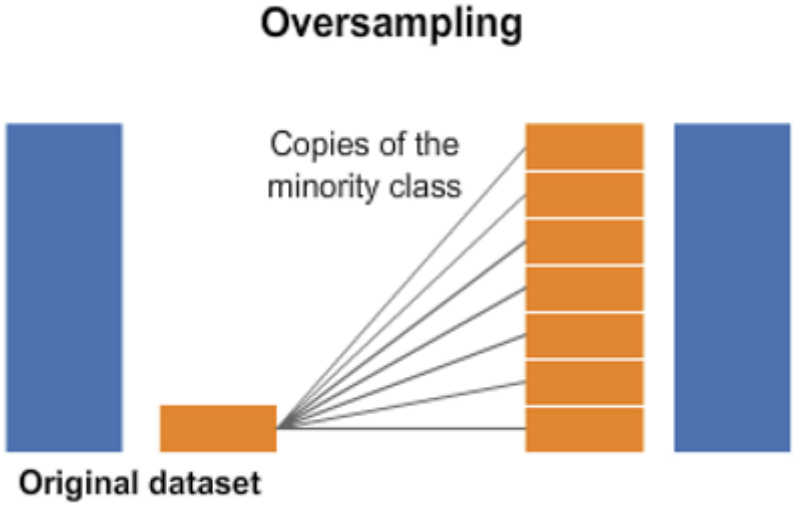
Formación del dataframe

```
x = df[['Time','V1','V2','V3','V5','V6','V7','V8','V9','V10','V11','V12',
  'V13','V14','V15','V16','V17','V18','V19','V20','V21','V22',
  'V23','V24','V25','V26','V27','V28','Amount']]

y = df['Class']

random_under = RandomUnderSampler(sampling_strategy='auto',
                                  random_state=123)

X_under, y_under = random_under.fit_resample(x, y)

df_under = X_under
df_under['Class'] = y_under
```

En ambos casos se obtiene un nuevo dataset balanceado cuyo diagrama de barras seria:



2. **Oversampling**: Consiste en **generar datos sintéticos** con unas características parecidas a las observaciones de la clase minoritaria.



Para realizar esta técnica proponemos dos maneras (de ambas formas obtenemos resultados similares):

2.1 Utilizando el método **Smote (Synthetic Minority Over-sampling)** de la librería **imblearn:**

Código Python

Se divide el dataset en:
- Variables explicativas (x)
- Variable objetivo (y)

```
x = df[['Time','V1','V2','V3','V5','V6','V7','V8','V9','V10','V11','V12',
 'V13','V14','V15','V16','V17','V18','V19','V20','V21','V22',
 'V23','V24','V25','V26','V27','V28','Amount']]

y = df['Class']
```

Creación y uso del modelo

```
sm = SMOTE(sampling_strategy='auto',
           random_state=123)

x_res, y_res = sm.fit_resample(x, y)
```

Formación del dataframe

```
df_over = x_res
df_over['Class'] = y_res
```

2.2 Utilizando el método **RandomOversample** de la librería **imblearn:**

Código Python

Se divide el dataset en:
- Variables explicativas (x)

```
x = df[['Time','V1','V2','V3','V5','V6','V7','V8','V9','V10','V11','V12',
 'V13','V14','V15','V16','V17','V18','V19','V20','V21','V22',
 'V23','V24','V25','V26','V27','V28','Amount']]
```

- Variable objetivo (y)

Creación y uso del modelo

Formación del dataframe

```
y = df['Class']

random_over = RandomOverSampler(sampling_strategy='auto',
                                random_state=123)

X_over, y_over = random_over.fit_resample(x, y)

df_over = X_over
df_over['Class'] = y_over
```

El nuevo dataset balanceado obtenido aplicando las dos fórmulas anteriores, presenta el siguiente gráfico:

Nota: Se recomienda aplicar estas familias de técnicas solo en los datos de entrenamiento y no sobre los datos de test.

De esta forma lograremos modelos que no pasen por alto esas clases menos frecuentes pero tan valiosas

Desde Data Equity esperamos que la información de este artículo sea de utilidad para tratar con dataset desbalanceados. No dudéis en poneros en contacto con nosotros.

Etiquetas: Análisis de datos, Data Scientist, dataset desbalanceado, manchine learning, tratamiento de datos