Morris & Opazo

aws partner network

Advanced Consulting Partner

Data & Analytics Competency

Channel Partner

Como mejorar la performance de modelos en escenarios de clases desbalanceadas.

We Love the Cloud

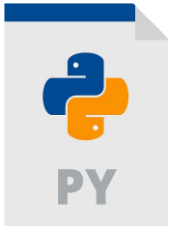
www.morrisopazo.com

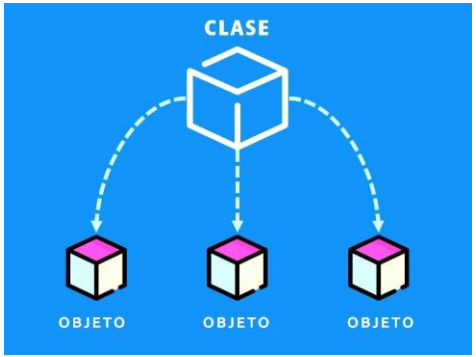
contacto@morrisopazo.com

Libreria prepImb

Primeros pasos para crear una librería local

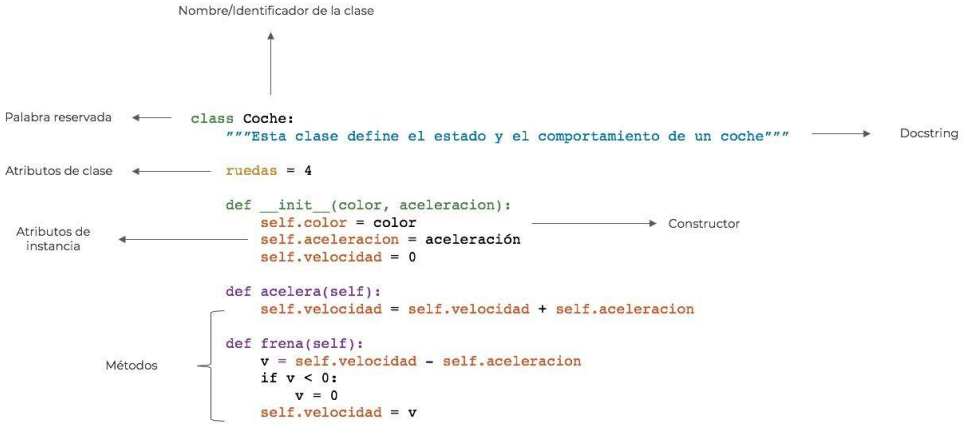
Para crear una librería tanto local como pública lo primero que se debe tener son los módulos que contienen las funciones que se desean en el paquete en archivos python.





Cada archivo para ser un módulo debe estar escrito con programación orientada a objetos (POO), es decir cada aspecto que queremos en la librería debe estar en clases y en ellas todas las funciones que requiere.

Ejemplo básico de POO



Así es la estructura que debe tener cada uno de nuestros módulos, donde los métodos contienen todas las partes que se necesitan para cumplir la función que está destinada aquella clase.

Archivo__init__.py

Si en nuestro paquete existen diversos módulos, se utiliza como intermediario entre ellos solo si un módulo necesita utilizar alguna función que está en otro módulo, si es el caso se debe importar la clase y el módulo que necesita de otro. Luego dentro de este último se importan todos aquellos que se necesitan.



También se define la variable __version_, que es para saber la versión de nuestra librería. Es para poder llamarla desde la consola donde trabajamos.

Primer paquete de librería

```
name_package/
        ├── __init__.py
        ├── modulo_1.py
        ├── modulo_2.py
        └── ...
```

Luego de crear todos los archivos anteriores los debemos guardar en una sola carpeta que llevará el nombre de nuestro primer paquete.

Cada paquete está destinado a alguna función que se quiera agregar a la librería, por lo tanto si se quiere tener más de alguna se deben crear multiples carpetas con la misma estructura antes descrita (archivos __init__y módulos).

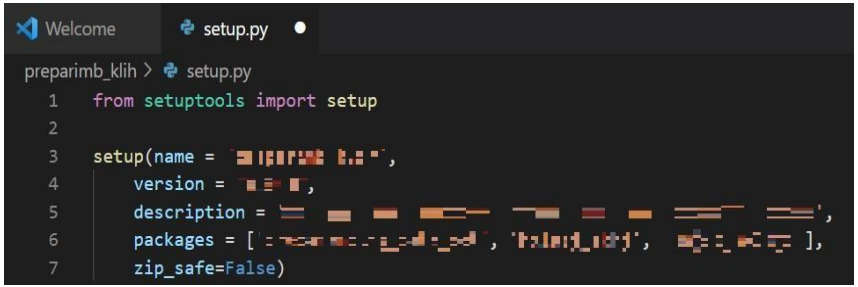
Primeros pasos

Morris & Opazo

Archivo de configuración setup.py

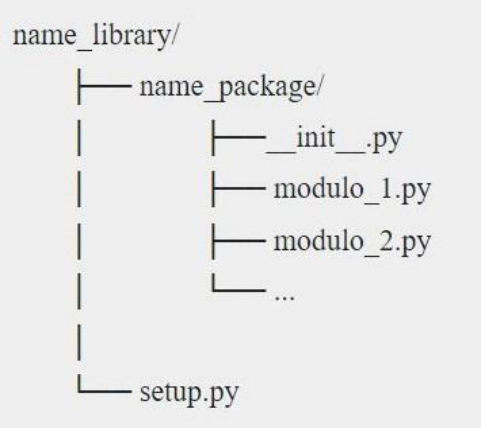
Fuera de estas carpetas con los paquetes de nuestra librería debemos generar un archivo python denominado “setup” que contiene algunas configuraciones de nuestra librería.

Este archivo contiene el nombre de la librería, la versión (la cual debe actualizarse con el tiempo), una breve descripción de lo que realiza (todas las funciones que contiene), los paquete que contiene (los nombres de todas las carpetas que creamos) y algunas otras configuraciones que queramos en ella.



Nombre librería

Finalmente luego de crear todos los archivos anteriormente descritos todos los empaquetamos en una carpeta con el nombre de nuestra nueva librería, aquí se debe tener especial cuidado ya que no debe coincidir con ninguno antes creado por lo menos si se desea publicar. Por tanto nuestro directorio de librería debe estar de la siguiente forma.



Librería prepImb

Librería que prepara conjuntos de datos no equilibrados para la creación de modelos transformando diferentes tipos de datos. En otras palabras balancea aquellos conjuntos que lo requieran, separa el conjunto de entrenamiento, el de prueba y el de validación, además de transformar los tipos de datos numéricos, categóricos y ordinales.

Esta librería requiere de otras que ya están publicadas, algunas de ellas son:







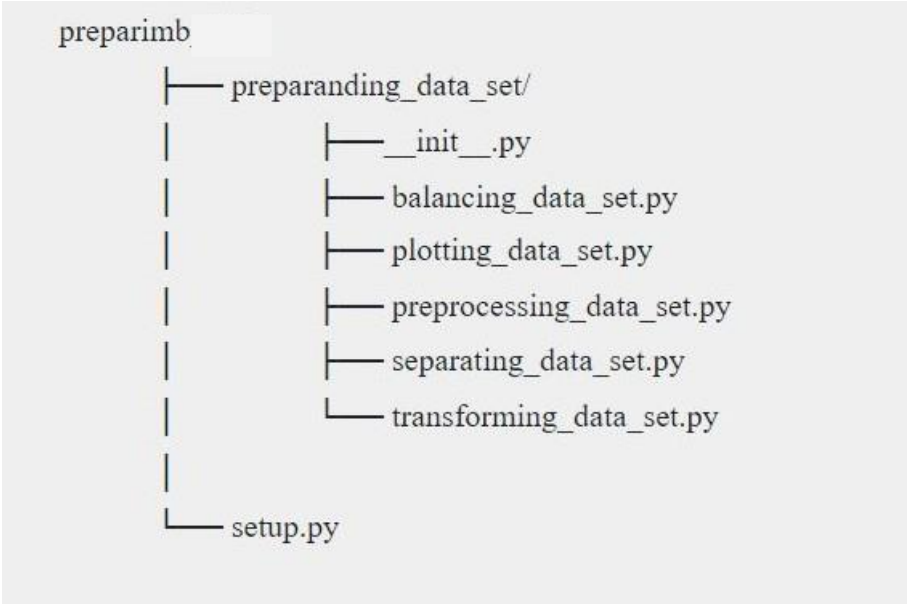






Librería prepImb

El directorio de la librería prepImb



Morris & Opazo | aws

Descripción funciones librería

separating_data_set.py

Módulo que divide el conjunto de datos de la columna que contiene las etiquetas de clase de estas.

plotting_data_set.py

Módulo que grafica circularmente las clases de un conjunto de datos.

balancing_data_set.py

Módulo que disminuye la desproporción entre clases de conjuntos previamente etiquetados que lo requieran. Agregando nuevos registros sintéticos a clase minoritaria.

preprocessing_data_set.py

Módulo que realiza la preparación de un conjunto de datos, divide el conjunto de datos en los conjuntos de entrenamiento, prueba y validación, transforma diversos tipos de datos, balancea si es que el conjunto lo requiere y guarda todas estas aplicaciones. En este se utilizan todos los demás módulos.

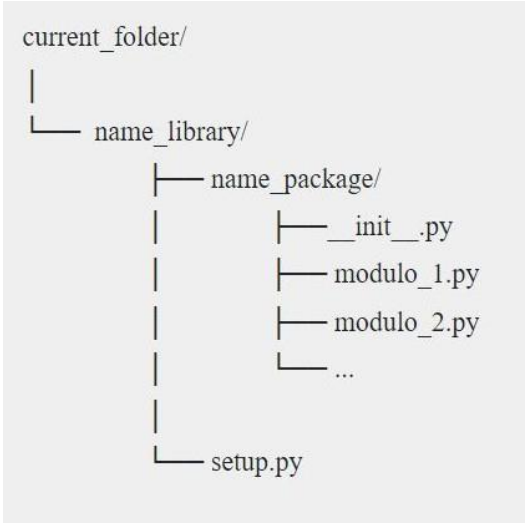
transforming_data_set.py

Módulo que transforma diversos tipos de datos, guarda aquellas transformaciones en formato joblib, además divide el conjunto en entrenamiento, prueba y validación para luego guardarlos.

Librería prepImb

Instalación de librería local

Para la instalación de una nueva librería de forma local es necesario estar en la misma carpeta donde se encuentra nuestra librería. Es decir al abrir el programa con el que trabajaremos debemos cambiar el directorio a donde esta la carpeta con el nombre de nuestra librería, siguiendo la imagen deberíamos estar ubicados en la carpeta “current_folder”.



Librería prepImb

Instalación de librería local

Luego de verificar nuestro directorio debemos ejecutar la siguiente línea de código, luego del comando -e va el nombre de la librería.

```
In [2]: pip install -e preparimb
```

Si todo salio bien obtendremos la siguiente salida por la consola de nuestro programa.

```
Installing collected packages: preparimb
  Running setup.py develop for preparimb
Successfully installed preparimb
Note: you may need to restart the kernel to use updated packages.
```

Morris & Opazo

aws

Amazon SageMaker

Amazon S3

EJEMPLO CON LIBRERIA preplmb

Ejemplo con libreria prepImb

Morris & Opazo

aws

Amazon SageMaker

Amazon S3

Módulo separación

In [1]:
```
#Importamos el archivo desde amazon s3
import pandas as pd

df = pd.read_csv('s3://kinostroza-fraud-datasets/data/sanitarias_MORRIS.csv')
df['DIAMETRO']=df['DIAMETRO'].astype(object)
```

Importamos el conjunto de datos

Importamos el módulo de separación de nuestra librería.

In [3]:
```
#importamos de nuestra libreria el modulo de separacion
from preparimb                          import separating_data_set

#dividimos la data con una de las funciones de nuestra libreria.
clase_separacion = separating_data_set.split_data_and_target()

[X, y] = clase_separacion.fit(df, 'target')
```

llamamos a la clase que contiene la función de separación

Llamamos y aplicamos la función para separar

Ejemplo con libreria prepImb

Morris & Opazo

aws

Amazon SageMaker

Amazon S3

Módulo para graficar

```
In [1]: #Importamos el archivo desde amazon s3
        import pandas as pd

        df = pd.read_csv('s3://kinostroza-fraud-datasets/data/sanitarias_MORRIS.csv')
        df['DIAMETRO']=df['DIAMETRO'].astype(object)
```

Importamos el conjunto de datos

Importamos el módulo para graficar de nuestra librería.

```
In [10]: #Ahora utilizamos el modulo para graficar la variable target

         from preparimb                    import plotting_data_set

         graficar_target = plotting_data_set.class_ratio()
         graficar_target.plot_pie(y, 'desbalanced', 'yes')
```

llamamos a la clase que contiene la función para graficar

Llamamos y aplicamos la función.

```
.preparanding_data_set import plotting_data_set
```

Ejemplo con libreria prepImb

Morris & Opazo

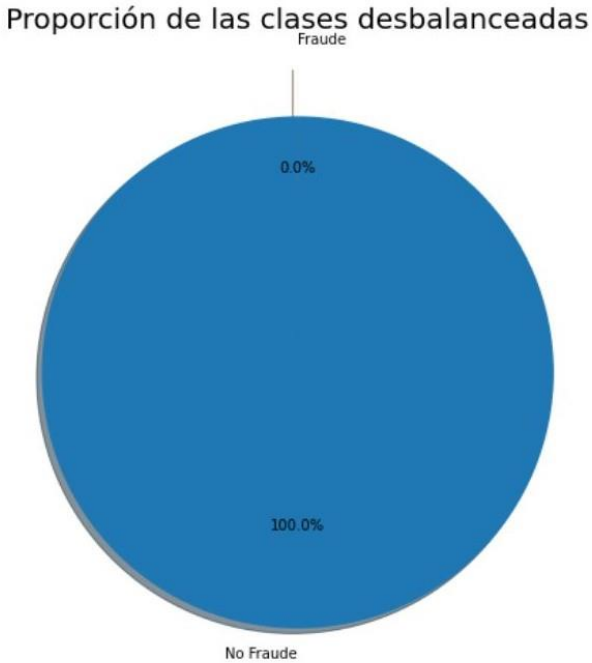
aws

Amazon SageMaker

Amazon S3

Módulo para graficar

Output función



Ejemplo con libreria prepImb

Morris & Opazo

aws

Amazon SageMaker

Módulo para balancear dataset's

Amazon S3

In [1]:
```
#Importamos el archivo desde amazon s3
import pandas as pd

df = pd.read_csv('s3://kinostroza-fraud-datasets/data/sanitarias_MORRIS.csv')
df['DIAMETRO']=df['DIAMETRO'].astype(object)
```

Importamos el conjunto de datos

Importamos el módulo de balanceo de nuestra librería.

In [5]:
```
#Como esta claramente desbalanceado, utilizaremos el modulo para balancear.
from preparimb                          import balancing_data_set

balancing_prueba = balancing_data_set.balance_target(target_plot = 'no')
[x_balan, y_balan] = balancing_prueba.fit_balance(df, 'target')
```

llamamos a la clase que contiene la función de balanceo

Llamamos y aplicamos la función para separar

Ejemplo con libreria prepImb

Morris & Opazo

Amazon SageMaker

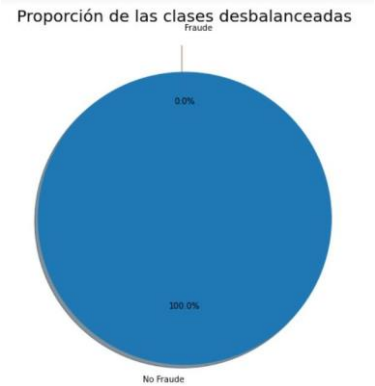
Módulo para balancear dataset's

Amazon S3

Output función

Sí "target_plot" = 'yes'

- Proporción_clases_balanceadas.png
- Proporción_clases_desbalanceadas.png



Output función

Sí "target_plot" = 'no', no se mostraran estas graficar por consola, pero lo demas todo se guardará



Ejemplo con libreria prepImb

Morris & Opazo

aws

Amazon SageMaker

Amazon S3

Módulo preprocesamiento

Importamos el módulo de preprocesamiento de la librería.

llamamos a la clase que contiene la función para preprocesar.

```
In [1]: from preparimb                    import preprocessing_data_set

        preprocessing_prueba = preprocessing_data_set.apply_tranform(target_plot = 'yes')
        preprocessing_prueba.fit('/home/ec2-user/SageMaker/data', 'target', ordinal_Category = ['DIAMETRO'])
```

```
/home/ec2-user/SageMaker/preparimb_klih/preparimb_klih/preparanding_data_set/preprocessing_data_set.py:32: UserWarning: All the
.csv and .txt files in the path you set below will be read and concatenated. Make sure that the folder contains only those that
you require for preprocessing or data balancing.
  return warnings.warn('All the .csv and .txt files in the path you set below will be read and concatenated. Make sure that the
folder contains only those that you require for preprocessing or data balancing.')
```

Llamamos y aplicamos la función

Ejemplo con libreria prepImb

Morris & Opazo

aws

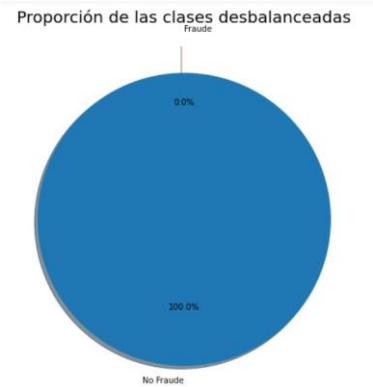
Amazon SageMaker

Amazon S3

Módulo preprocesamiento

Output función

Sí “target_plot” = ‘yes’



Output función

Sí “target_plot” = ‘no’, no se mostraran estas graficar por consola, pero lo demas todo se guardará



Ejemplo con libreria prepImb

Morris & Opazo

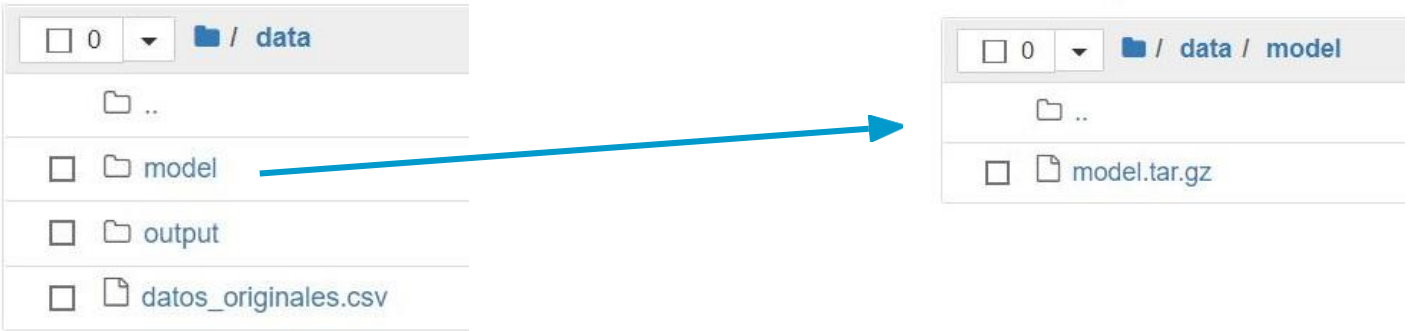
Amazon SageMaker

Amazon S3

Módulo preprocesamiento

Output función

Sí “target_plot” = ‘yes’



Ejemplo con libreria prepImb

Morris & Opazo

aws

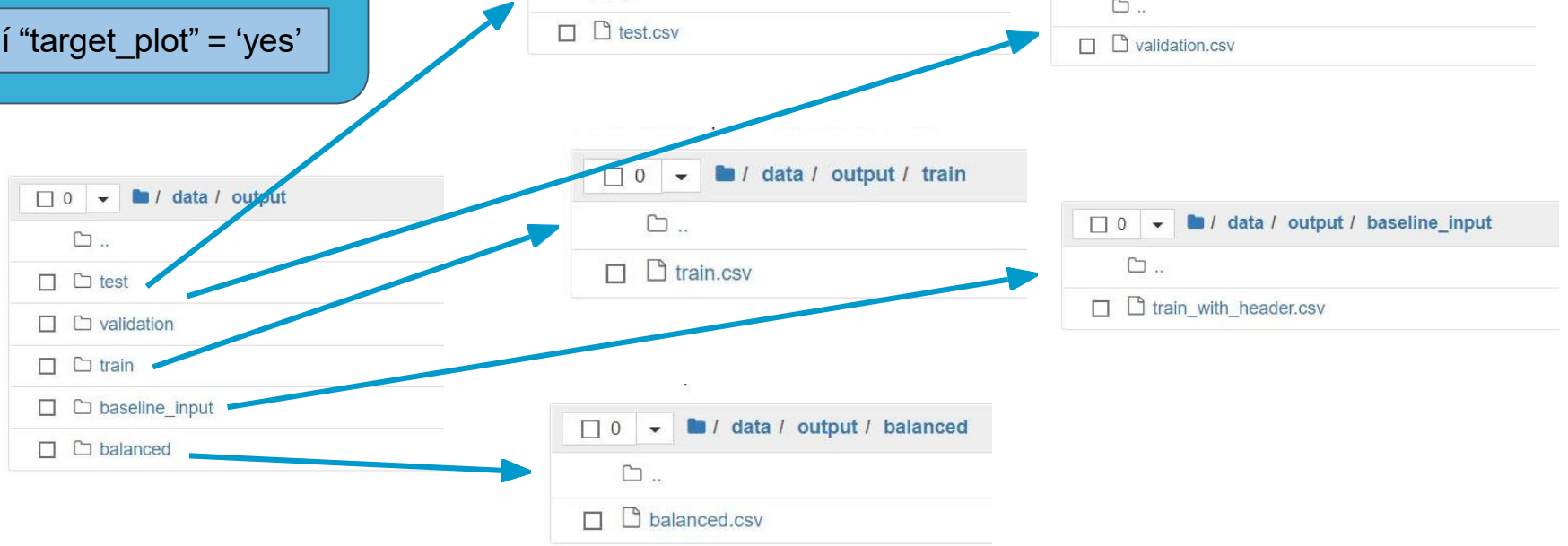
Amazon SageMaker

Módulo preprocesamiento

Amazon S3

Output función

Sí “target_plot” = ‘yes’



Morris & Opazo

aws

¡MUCHAS GRACIAS!