Universidad Autónoma de Querétaro

Facultad de Ingeniería
Maestría en Inteligencia Artificial
Machine Learning
Sheila Leyva López
Cecilia Gabriela Rodríguez Flores

Creación de datos sintéticos.

25 de noviembre del 2022

Objetivo

Desarrollar un algoritmo enfocado al aumento de datos, mediante el lenguaje de programación de python, a fin de conseguir un mejor balance mediante los datos sintéticos.

Introducción

Dentro del área de la Inteligencia Artificial (IA) existe una sub-área llamada aprendizaje automático o Machine Learning (ML), cuyas herramientas permiten a un sistema aprender patrones y comportamientos de los datos en lugar de aprender mediante la programación explícita.

Normalmente los algoritmos de aprendizaje automático suelen entrenarse con una gran cantidad de datos, lo cual podría mejorar su desempeño considerablemente, sin embargo, no siempre se cuenta con suficientes instancias para entrenarlos, y simplemente obtener más registros de manera real no es viable. Por tales motivos, se aumenta la cantidad de datos de manera sintética o artificial. De esta forma, se puede observar el rendimiento de los algoritmos mucho antes de hacer pruebas con datos reales.

Existen distintos tipos de datos sintéticos, por lo general suelen ser del tipo, texto, multimedia (videos, imágenes o audios) o tabulares. En el presente trabajo se desarrollan las funciones, en lenguaje de Python, necesarias para la generación de datos sintéticos tabulares. Mediante el uso de la base de datos: *Indicadores personales clave de enfermedad cardíaca.*

Marco Teórico

Es importante definir conceptos básicos implementados en este trabajo a fin brindar un mejor entendimiento del contexto en el que se trabaja.

Datos sintéticos

Los datos sintéticos son datos que son creados artificialmente en lugar de ser obtenidos mediante mediciones o eventos. Generalmente, se utilizan algoritmos para aumentar el número

de datos, ya sea porque las pruebas lo requieren o no existen suficientes instancias para entrenar un algoritmo.

Existen varias razones por las cuales los datos sintéticos son importantes, algunas de ellas son:

- Cuando se requieren datos para prueba, pero no están disponibles en ese momento o no existen.
- Cuando se tiene una cantidad pequeña de instancias y no alcanzan para hacer tanto el entrenamiento como las pruebas.
- Cuando se requieren datos de entrenamiento, sin embargo, es costoso o inviable realizarlos.
- Cuando el porcentaje de las clases no está equilibrado.

Existen muchos beneficios de crear y usar datos sintéticos, por ejemplo:

- Obtención de flexibilidad para probar un algoritmo.
- Completa libertad sobre la cantidad y la distribución de los datos de un modelo.
- Realización de pruebas cuando las clases están desbalanceadas.
- Proteger la privacidad de los datos reales mientras se trabaja con el modelo utilizando datos sintéticos.

Algoritmo de la ruleta

El método estocástico de ruleta se utiliza generalmente como un operador genético de selección en cómputo evolutivo [1]. Y básicamente sigue los siguientes pasos:

1. Conocer las observaciones del atributo a partir del cuál se desea generar los datos sintéticos.
2. Probabilidad de que este valor aparezca en la base de datos.
3. Se suman las probabilidades para priorizar los valores existentes.
4. Se genera un porcentaje aleatorio.

5. Se buscan el porcentaje más cercano alcanzado por los valores únicos (observaciones) y se obtienen los índices de cada valor.
6. Finalmente, se obtiene el valor para generar el dato sintético y se agrega a la base de datos.

Resulta importante mencionar que, se debe comparar la distribución de los datos antes y después de la generación de datos sintéticos, para siempre asegurar que no se están arrastrando sesgos.

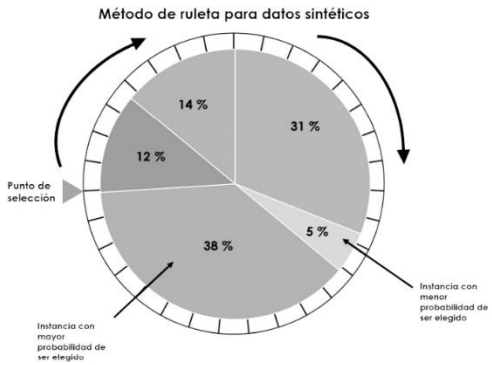


Ilustración 1.Método de ruleta para datos sintéticos.

Técnica de sobremuestreo de minorías sintéticas (SMOTE)

La técnica de sobremuestreo de minorías sintéticas (SMOTE) es un enfoque de preprocesamiento bien conocido para manejar conjuntos de datos desequilibrados, es decir, aumenta el número de casos de un conjunto de datos de forma equilibrada, en donde se generan observaciones sintéticas para la clase minoritaria en un rango aleatorio entre la observación y sus k vecinos de clase minoritaria más cercanos [2].

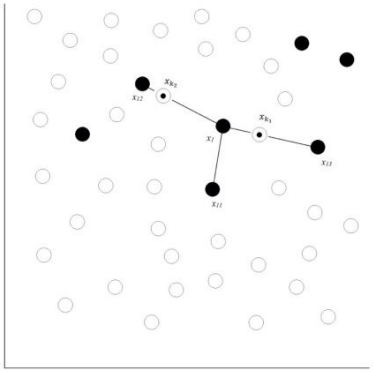


Ilustración 2.Procedimiento SMOTE sobre un conjunto de datos desequilibrados en un espacio bidimensional.

La Ilustración 2 muestra la generación de observaciones sintéticas según SMOTE sobre datos desequilibrados.

Sin embargo, muchos trabajos han demostrado que el deterioro del rendimiento del modelo se debe a otras razones vinculadas a la distribución de la muestra de clase minoritaria. El sobremuestreo ciego de SMOTE conduce a dos problemas principales: ruido, son aquellos de una clase ubicada en la zona segura de la otra; y ejemplos límite, son aquellos ubicados en la vecindad del límite de clase, en donde los ejemplos sintéticos se crean sin considerar la clase mayoritaria, lo que posiblemente resulte en ejemplos ambiguos si hay una fuerte superposición de las clases.

Etapas en la implementación de modelos

Es importante definir conceptos básicos del aprendizaje máquina que permitan entender el contexto en el que se trabaja, como son las etapas fundamentales para la implementación de técnicas de aprendizaje máquina: Preparación de datos, la cual es una etapa fundamental en cualquier proyecto, debido a que por medio de ella se pueden inicializar correctamente los datos para su posterior procesamiento y análisis; Analizar, en esta sección se utilizaran (que dato se van a utilizar) o dividirá los datos ya preparados para el siguiente paso, para este trabajo se optó

por crear subconjuntos de datos, en los cuales el 80% de los datos se considera para el entrenamiento, mientras que un 20% corresponde a pruebas del modelo de clasificación. Posteriormente, se prosiguió con la implementación del aumento de datos para el conjunto de entrenamiento por medio del método de ruleta y la técnica SMOTE.

Materiales y métodos

Herramientas utilizadas

- Google Collaboratory
- Conjunto de datos: *Indicadores personales clave de enfermedad cardíaca*
- AMD Ryzen 9 5900HS with Radeon Graphics 3.30 GHz
- RAM 16.0 GB
- 64-bit operating system, x64-based processor

Conjunto de datos

En este trabajo, se utilizará el conjunto de datos: *Indicadores personales clave de enfermedad cardíaca*, datos provenientes de 400,000 adultos, obtenidos durante la encuesta anual 2020 de los Centros para el Control y prevención de Enfermedades (CDC, por sus siglas en inglés) pertenecientes al departamento de salud y servicios humanos en los Estados Unidos. Originalmente el conjunto de datos contenía alrededor de 300 atributos, sin embargo, se redujo a solo 18 variables, los cuales son los que se encuentran disponibles públicamente en la plataforma Kaggle [3].

Casi la mitad de los estadounidenses (47%), incluyendo afroamericanos, indios americanos, nativos de Alaska y blancos; tienen al menos de 1 a 3 factores de riesgo de padecer alguna enfermedad cardíaca. A continuación, se agrega una breve descripción de los atributos incluidos en este conjunto de datos:

- HeartDisease: (atributo de decisión): personas encuestadas que informaron alguna vez haber padecido alguna enfermedad coronaria (CHD, por sus siglas en inglés) o infarto al miocardio(IM, por sus siglas en inglés).
- BMI: Índice de Masa Corporal.
- Smoking: personas encuestadas que han fumado al menos 100 cigarros en su vida entera.
- AlcoholDrinking: corresponde a hombres adultos que beben más de 14 tragos por semana y mujeres adultas que beben más de 7 tragos por semana.
- Stroke: responde a la pregunta: ¿alguna vez le dijeron o usted tuvo un derrame cerebral?
- PhysicalHealth: incluyendo enfermedades y lesiones físicas, responde a la pregunta: ¿durante cuántos días en los últimos 30 días su salud física no fue buena? (de 0 a 30 días)
- MentalHealth: ¿durante cuántos días en los últimos 30 días su salud mental no fue buena? (de 0 a 30 días).
- DiffWalking: responde a ¿tiene serias dificultades para caminar o subir escaleras?
- Sex: hombre o mujer.
- AgeCategory: 14 rangos de edad.
- Race: valor de raza / etnicidad imputada.
- Diabetic: responde a ¿alguna vez ha sido diagnosticada con diabetes?
- PhysiclActivity: adultos que informaron haber realizado actividad física o ejercicio en los últimos 30 días, no incluyendo su trabajo habitual.
- GenHealth: responde a ¿cómo calificarías tu salud en general?
- SleepTime: responde a un promedio de horas que duerme, en un periodo de 24 horas, la persona encuestada.
- Asthma: responde a ¿alguna vez ha sido diagnosticado con asma?
- KidneyDisease: responde a ¿alguna vez le dijeron que tenía una enfermedad renal?, sin incluir cálculos renales, infección de vejiga o incontinencia.
- SkinCancer: responde a ¿alguna vez ha sido diagnosticado de cáncer de piel.

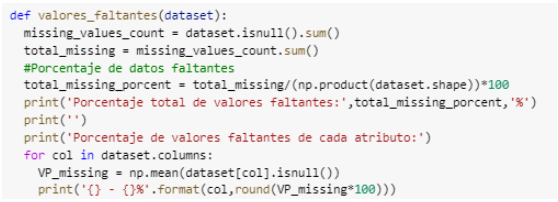
Métodos

En esta sección se presentan las funciones de los métodos de aumento de datos seleccionados, en donde se incluye de igual forma las funciones para la preparación de los datos y la función para la división de entrenamiento y prueba.

Primeramente, para la realización de esta tarea, fue necesario importar las librerías de Pandas, numpy, etc., a fin de poder hacer uso de la base de datos propuesta.

Posteriormente, se optó por crear diversas funciones a fin de distribuir las tareas de una manera más eficiente como se muestra a continuación:

- Función valores_faltantes: obtiene el porcentaje total de los valores faltantes, así como el porcentaje de valores faltantes para ca uno de los atributos que comprende la base de datos en cuestión. Esta función requiere como datos de entrada únicamente la base de datos.



```
def valores_faltantes(dataset):
  missing_values_count = dataset.isnull().sum()
  total_missing = missing_values_count.sum()
  #Porcentaje de datos faltantes
  total_missing_porcent = total_missing/(np.product(dataset.shape))*100
  print('Porcentaje total de valores faltantes:',total_missing_porcent,'%')
  print('')
  print('Porcentaje de valores faltantes de cada atributo:')
  for col in dataset.columns:
    VP_missing = np.mean(dataset[col].isnull())
    print('{} - {}%'.format(col,round(VP_missing*100)))
```

Ilustración 3.Función propuesta para la obtención del total de valores faltantes.

- Función norm_min_max: realiza la normalización de la base de datos. Esta función requiere como datos de entrada la base de datos en cuestión.

```
def norm_min_max(datos):
    lim_sup = []
    lim_inf = []
    rangoDatos =[]
    maxNorm = 1
    minNorm = 0
    rango = maxNorm - minNorm
    for i in range (0,datos.columns.size):
        lim_sup.append(datos.iloc[:,i].max())
        lim_inf.append(datos.iloc[:,i].min())
        rangoDatos.append(lim_sup[i] - lim_inf[i])
    nombres = datos.columns.values.tolist()
    datosNorm = pd.DataFrame(columns = nombres)

    for j in range(len(datos.columns)):
        varNorm = []
        var = datos.iloc[:,j]
        for i in range(len(datos)):
            D = var[i] - lim_inf[j]
            DPct = D/rangoDatos[j]
            dNorm = rango*DPct
            varNorm.append(minNorm+dNorm)
        datosNorm.iloc[:,j] = varNorm
    datos = datosNorm
    return datos
```

Ilustración 4.Función propuesta para la normalización.

- Función matriz_cov: se encarga de obtener la matriz de covarianza de los elementos de la base de datos en cuestión.

```
def matriz_cov(data):
    atributos = data.columns
    n = len(atributos)
    m = np.zeros((n,n))
    for i in range(n):
        for j in range(n):
            X = data[atributos[i]]
            Y = data[atributos[j]]
            m[i][j] = (((X-X.mean())*(Y-Y.mean())).sum())/(len(X)-1)
    return m
```

Ilustración 5.Función propuesta para la matriz de covarianza.

- Función PCA: brinda los porcentajes de cada uno de los atributos de acuerdo con su relevancia dentro de la base de datos, a fin de discernir los atributos con mayor peso.

```
def PCA(datos,col_decision):
  datos1 = datos.drop([col_decision],axis=1) #Eliminando el atributo de decisión
  #Ajustar los datos restando la media a cada atributo
  datos_A = pd.DataFrame(columns=datos1.columns,index=range(len(datos1)))
  for i in datos_A.columns:
      datos_A[i] = datos1[i] - datos1[i].mean()
  #datos_A
  matrix = matriz_cov(datos_A)
  #sns.heatmap(matrix)
  L,V =  np.linalg.eig(matrix)
  #Obtener el porcentaje de covarianza de cada uno de los atributos
  total = L.sum()
  p = (L/total)*100
  pca =[]
  columnas1 = datos_A.columns.values
  for index, row in enumerate(p):
    print(columnas1[index] + ':',row)
```

Ilustración 6.Función propuesta para el algoritmo de PCA.

- Función division_80_20: realiza la separación del conjunto de datos de acuerdo al porcentaje de 80% para el conjunto de entrenamiento y 20% para el conjunto de prueba. Esta función requiere como entradas los datos y las etiquetas.

```
def division_80_20(datos, etiquetas):
  n = len(datos)
  indices = np.arange(n)
  np.random.shuffle(indices)
  lim = int(n * 0.80) + 1
  x_train = datos.loc[indices[0:lim]]
  x_test = datos.loc[indices[lim:]]
  y_train = etiquetas.loc[indices[0:lim]]
  y_test = etiquetas.loc[indices[lim:]]
  return x_train, x_test, y_train, y_test
```

Ilustración 7. Función propuesta para la división del conjunto de datos.

- Función probabilidad: brinda la suma acumulativa de la posibilidad para priorizar valores de la base de datos. Esta función requiere como entradas: el conjunto de datos, la columna y el número que le corresponde a la columna en cuestión

```
def probabilidad(datos,columns_df,i):
  item_counts=datos[columns_df[i]].value_counts(normalize =True)       #probabilidad de que este valor aparezca en la base de datos
  return (item_counts.cumsum()).to_numpy() #suma acumulativa de la posibilidad para priorizar valores
```

Ilustración 8.Función propuesta para obtención de la probabilidad.

- Función Ruleta: se encarga de llevar a cabo la implementación del método, en esta función se ira generando un porcentaje aleatorio, el cual se buscará un aproximado del porcentaje brindado entre los porcentajes de las observaciones. Así bien, los datos sintéticos se irán agregando al conjunto de datos de entrenamiento. La presente función requiere como datos de entrada: la base de datos, el porcentaje de datos a aumentar y la lista de las columnas de la base de datos.

```
def Ruleta(datos,Porc_datos,columns_df):
  #generar datos sinteticos a partir de método ruleta
  total_cells =np.product(datos.shape)
  N_datos = total_cells*(int(Porc_datos)/100)
  syn = []
  for j in tqdm(range(int(N_datos))):
    row = []
    for i in range(len(columns_df)):
      obs = datos[columns_df[i]].value_counts().index.tolist()       #valor unico
      Prob_np = probabilidad(datos,columns_df,i)
      rand_value = random.random()       #Generar un porcentaje aleatorio
      flecha = np.argwhere(rand_value <= Prob_np)       #Buscar el porcentaje más cercano alcanzado por los valores unicos
      new_value = obs[int(flecha[0])]       #Obtener el valor para generar dato sintetico
      row.append(new_value)
    syn.append(row)
  d_syn = pd.DataFrame(syn, columns = columns_df)   #Crear un macro de datos de los datos sinteticos
  syn_final = pd.concat([datos,d_syn],ignore_index=True)   #Agregar datos sintéticos a la copia de los datos originales
  return syn_final
```

Ilustración 9.Función propuesta para el algoritmo de Ruleta.

- Función SMOTE: se encarga identificar la clase del atributo minoritario y la clase del atributo mayoritario, a fin igual el número de la clase mayoritaria por medio de la creación

de datos sintéticos, los cuales se agregarán a la base de datos. Esta función requiere como entradas la base de datos y el número de vecinos a considerar.

```
def SMOTE(df, k):
    total_data = df.values.tolist()
    labelCont = Counter(df[data.columns[-1]].values)
    majority_num = max(labelCont.values())

    for label, num in tqdm(labelCont.items()):
        #print('label',label, ' num', num)
        if num < majority_num:
            to_add = majority_num - num #diferencia entre cada clase del atributo de decisión
            print('# Atributo de decisión con mayor numero:',majority_num, ' # Atributo de decisión con menor numero',num)
            last_column = df[df.columns[-1]] #Identificación del atributo de decisión
            data_w_label = df.loc[last_column == label] #Extracción de las filas con el atributo de decisión menor
            #print(label)
            data_no_label = data_w_label[df.columns[:-1]].values
            #print(data_no_label)
            if len(data_no_label) < k:
                k = len(data_no_label) # void # of neighbors >= sample size
            #print('K',k)

            M=data_no_label
            t = M.shape[0] # number of minority class samples
            numattrs = M.shape[1]

            synthetic = []
            for _ in range(to_add):
                _, indices = get_neigbours(M, k)
                for i in range(t):
                    neighbour = randrange(0, k)
                    diff = M[indices[i, neighbour]] - M[i]
                    gap = random.uniform(0, 1)
                    synthetic.append(M[i] + gap*diff)
            M_S = synthetic[:to_add]
            M_S = pd.DataFrame(M_S)
            M_S['-']=label

            for n in range(len(M_S)):
                new_row =M_S.iloc[n]
                total_data.append(new_row)
            df_final =pd.DataFrame(total_data)
    return df_final,M_S
```

Ilustración 10.Función propuesta para la técnica de SMOTE.

Diagrama de metodología

Para el análisis de cualquier base de datos se deben seguir los siguientes pasos:

1. Recolección: Definir y cargar la base de datos a utilizar.
2. Preparar: Convertir las características lingüísticas a valores categóricos, comprobar la no existencia de datos faltantes, identificación de outliers, así como la normalización de los datos.
3. Analizar: Conocer la distribución de la información por cada atributo, reducción de dimensionalidad, división de los conjuntos de entrenamiento y prueba del modelo, asignando el 80% de ellos para el entrenamiento y el 20 % restante a la etapa de prueba; e Implementación del modelo de ruleta y la técnica de SMOTE para el conjunto de entrenamiento.

A continuación, se muestra el diagrama de flujo que representa el proceso de desarrollo de la presente tarea, el cual fue implementado en un programa basado en lenguaje Python.

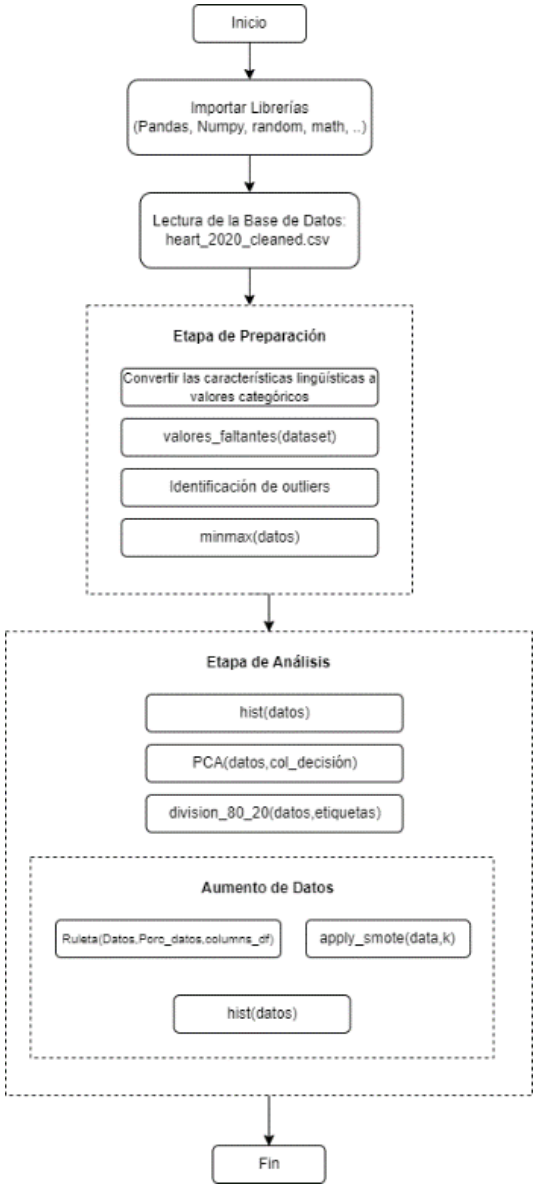


Ilustración 11.Diagrama de flujo para el análisis del conjunto de datos.

Resultados y discusión

A continuación, se presentan las distribuciones de los atributos de la base de datos en cuestión, después de haberle realizado un subsampling del 5%, al cual se le efectuó una preparación, la cual comprende los siguientes puntos: la conversión de características lingüísticas a valores categóricos, la comprobación de la no existencia de datos faltantes, la identificación de outliers y la normalización de los datos.

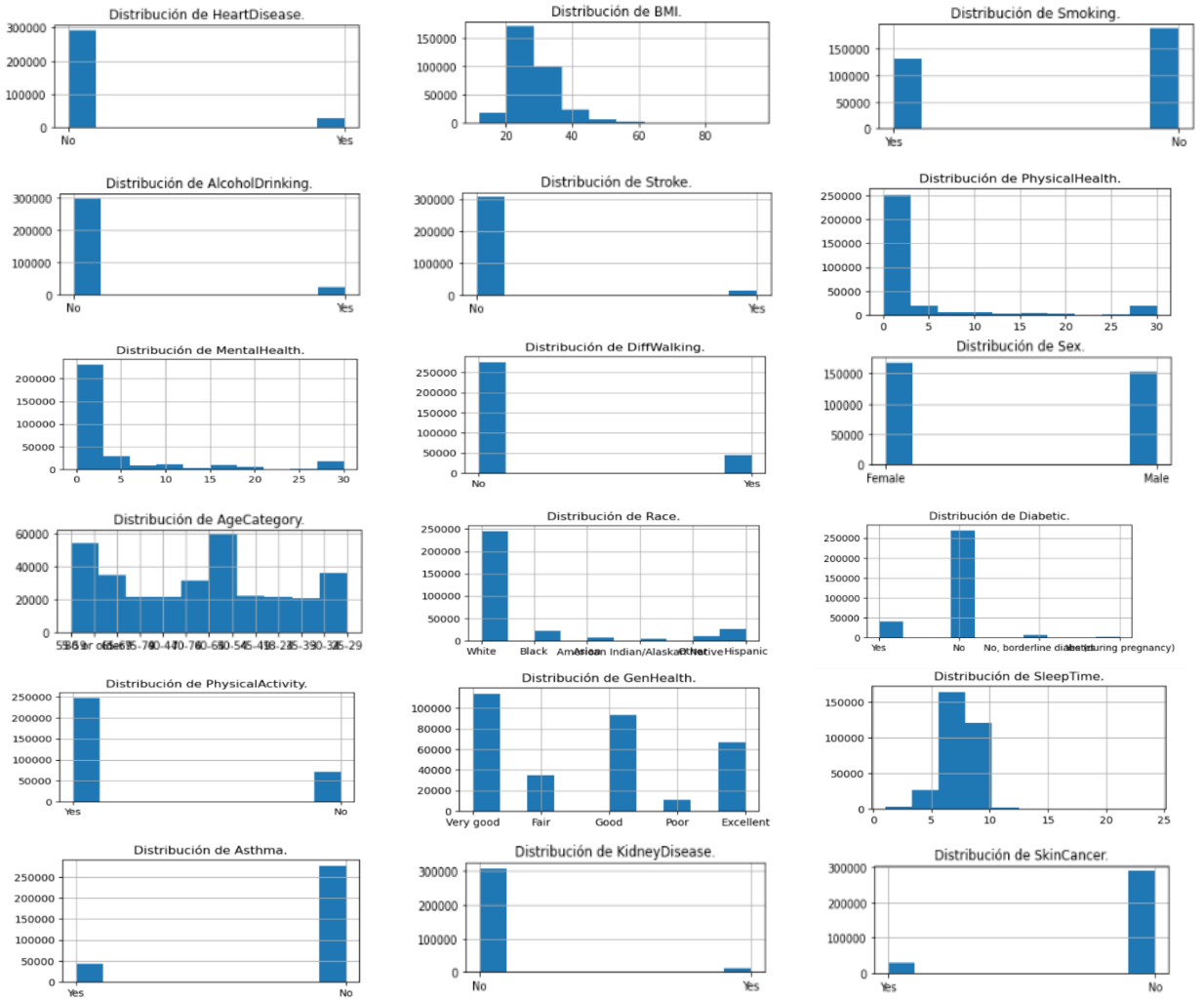


Ilustración 12. Distribución de los atributos de la base de datos.

A partir de haber implementado la técnica de análisis de componentes principales (PCA) en la base de datos en cuestión, se logró reducir la cantidad de atributos de 18 a 12, entre los atributos que contenían la mayor información se encuentran: BMI, PhysicalHealth, MentalHealth, AgeCategory, Race, Diabetic, Smoking, AlcoholDrinking, Stroke, DiffWalking y PhysicalActivity. Así bien, para los siguientes pasos se incluye a estos atributos seleccionados el atributo de decisión HeartDisease.

Antes de proseguir con la creación de datos sintéticos será necesario dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba, a fin de realizar el aumento de datos únicamente en el conjunto de entrenamiento. En la Tabla 1 se presentan el total de los datos para cada uno de los conjuntos, en donde se utilizó la división 80/20.

Tabla 1.Conjunto de Entrenamiento y Prueba.

	Conjunto de Entrenamiento	Conjunto de Prueba
Total de datos	12792	3198

Aumento de Datos

En la Ilustración 13 se presenta la distribución de las clases del atributo de decisión, en donde se puede observar el desbalance que existe al contar con un total de 14,607 casos de pacientes sin problemas cardiacos (No) y un total de 1,383 casos de pacientes con problemas cardiacos (Yes). Así bien, también se presentan en la Ilustración 14 la distribución del conjunto de entrenamiento, en donde de igual manera sigue presente este desbalance.

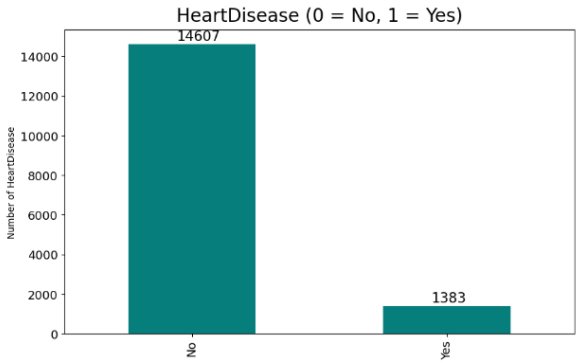


Ilustración 13.Balance del Atributo de Decisión.

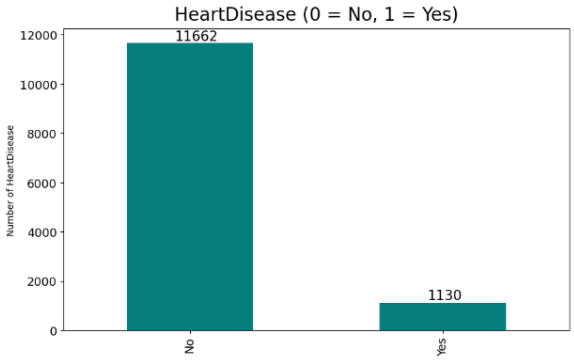


Ilustración 14.Balance del Atributo de Decisión del conjunto de entrenamiento.

Así mismo, se presentan los resultados obtenidos de cada uno de los métodos de aumento de datos abordado en esta tarea, los cuales se implementaron únicamente para el conjunto de entrenamiento.

Modelo de Ruleta

A continuación, se pueden observar los resultados que el método de Ruleta brindó con un porcentaje de datos sintéticos del 10%.

```
100%|██████████| 15350/15350 [03:51<00:00, 66.30it/s]
```

	BMI	PhysicalHealth	MentalHealth	AgeCategory	Race	Diabetic	Smoking	AlcoholDrinking	Stroke	DiffWalking	PhysicalActivity	HeartDisease
0	0.207171	0.033333	0.000000	0.525424	1.0	0.000000	0.0	0.0	0.0	0.0	1.0	No
1	0.119643	0.100000	0.066667	0.864407	0.4	0.000000	0.0	0.0	0.0	0.0	1.0	No
2	0.268985	0.100000	0.100000	0.694915	1.0	0.000000	1.0	0.0	0.0	0.0	1.0	No
3	0.188458	1.000000	0.166667	0.949153	1.0	0.333333	1.0	0.0	0.0	0.0	1.0	No
4	0.223832	0.000000	0.000000	0.694915	0.4	0.000000	1.0	0.0	0.0	0.0	1.0	No
...	...	...	...	...	...	...	...	...	...	...	...	...
28137	0.501751	0.000000	0.000000	0.440678	0.6	0.000000	0.0	0.0	0.0	0.0	1.0	No
28138	0.098515	0.000000	0.000000	1.000000	1.0	0.000000	0.0	0.0	0.0	0.0	1.0	No
28139	0.135217	0.000000	0.333333	1.000000	1.0	0.000000	1.0	0.0	0.0	1.0	1.0	No
28140	0.166365	0.000000	0.166667	0.610169	1.0	0.333333	0.0	0.0	0.0	0.0	1.0	No
28141	0.517325	0.000000	0.000000	0.271186	1.0	0.000000	0.0	0.0	1.0	0.0	1.0	No

28142 rows × 12 columns

Ilustración 15.Resultados del modelo de Ruleta.

En la Ilustración 16 se puede observar el desbalance aun presente en el atributo de decisión posterior a la implementación del método de Ruleta, debido a que como su nombre lo indica, se realizó el método de selección de una manera probabilística, dado que, no considera si existe un desbalance inicial en las clases.

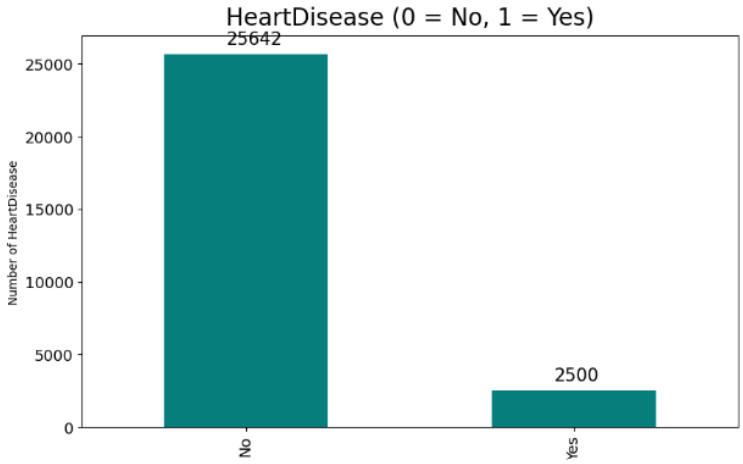


Ilustración 16.Balance del Atributo de Decisión posterior a la implementación del método de ruleta.

En la Tabla 2 se presenta la comparación entre la desviación estándar de la base de datos y la desviación estándar posterior a la implementación del método de Ruleta.

Tabla 2. Desviación Estándar previo y posterior a la implementación del modelo de Ruleta.

	Base de datos sin aumento de datos	**Base de datos con aumento de datos**
BMI	0.076276	0.076578
PhysicalHealth	0.263234	0.264886
MentalHealth	0.268345	0.268425
AgeCategory	0.303308	0.302981
Race	0.243580	0.246275
Diabetic	0.165617	0.166645
Smoking	0.491930	0.493366
AlcoholDrinking	0.252362	0.253289
Stroke	0.186040	0.187957
DiffWalking	0.342890	0.341343
PhysicalActivity	0.418917	0.418732

A partir de la Tabla 2 se puede observar que la desviación estándar de los atributos con datos sintéticos es bastante similar a la original, dado que, varía en cuestión de milésimas, lo cual nos indica que la distribución de los datos no se vio afectada por el método de la ruleta, tal como se muestra en las siguientes figuras.

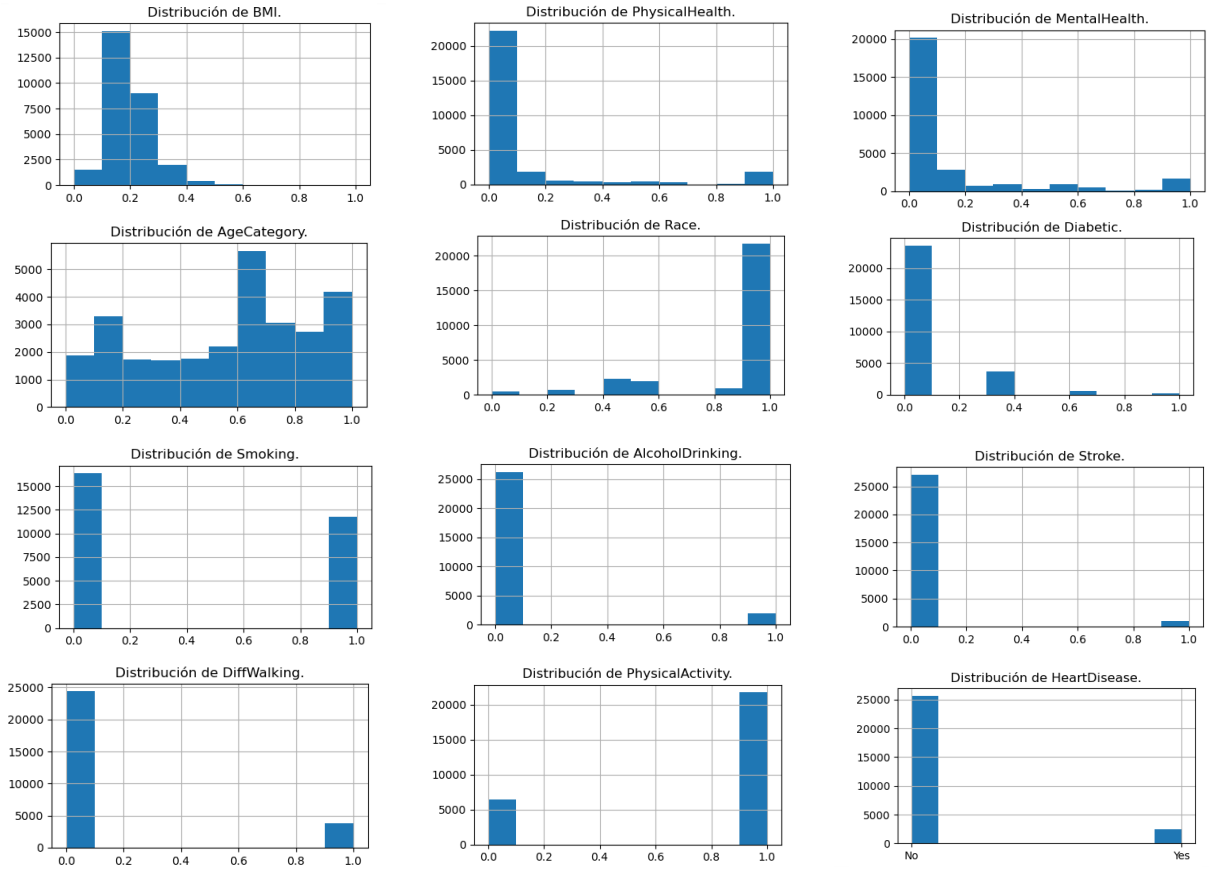


Ilustración 17.Distribución de la implementación del modelo de Ruleta.

Técnica SMOTE

A continuación, se pueden observar los resultados que el método de SMOTE brindo.

100%|██████████| 2/2 [05:41<00:00, 170.61s/it]

	BMI	PhysicalHealth	MentalHealth	AgeCategory	Race	Diabetic	Smoking	AlcoholDrinking	Stroke	DiffWalking	PhysicalActivity	HeartDisease
0	0.207171	0.033333	0.000000	0.525424	1.0	0.000000	0.0	0.0	0.0	0.0	1.0	No
1	0.119643	0.100000	0.066667	0.864407	0.4	0.000000	0.0	0.0	0.0	0.0	1.0	No
2	0.268985	0.100000	0.100000	0.694915	1.0	0.000000	1.0	0.0	0.0	0.0	1.0	No
3	0.188458	1.000000	0.166667	0.949153	1.0	0.333333	1.0	0.0	0.0	0.0	1.0	No
4	0.223832	0.000000	0.000000	0.694915	0.4	0.000000	1.0	0.0	0.0	0.0	1.0	No
...	...	...	...	...	...	...	...	...	...	...	...	...
23319	0.114099	0.000000	0.000000	0.993975	1.0	0.000000	0.0	0.0	0.0	0.0	0.0	Yes
23320	0.296107	0.000000	0.000000	0.949153	1.0	0.000000	0.0	0.0	0.0	0.0	1.0	Yes
23321	0.206131	0.000000	0.000000	1.000000	1.0	0.000000	0.0	0.0	0.0	0.0	1.0	Yes
23322	0.209626	0.000000	0.000000	0.949272	1.0	0.333333	1.0	0.0	0.0	0.0	1.0	Yes
23323	0.169447	0.000000	0.000000	0.949153	1.0	0.000000	1.0	0.0	0.0	0.0	1.0	Yes

23324 rows × 12 columns

Ilustración 18.Resultados de la técnica SMOTE.

En la Ilustración 19 se puede observar el balance en el atributo de decisión posterior a la implementación del método de SMOTE, ya que este método se enfoca en la categoría minoritaria.

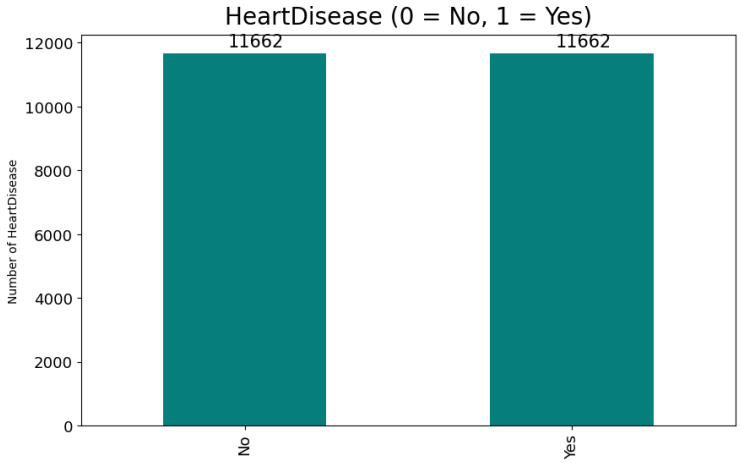


Ilustración 19. Balance del Atributo de Decisión posterior a la implementación de la técnica SMOTE.

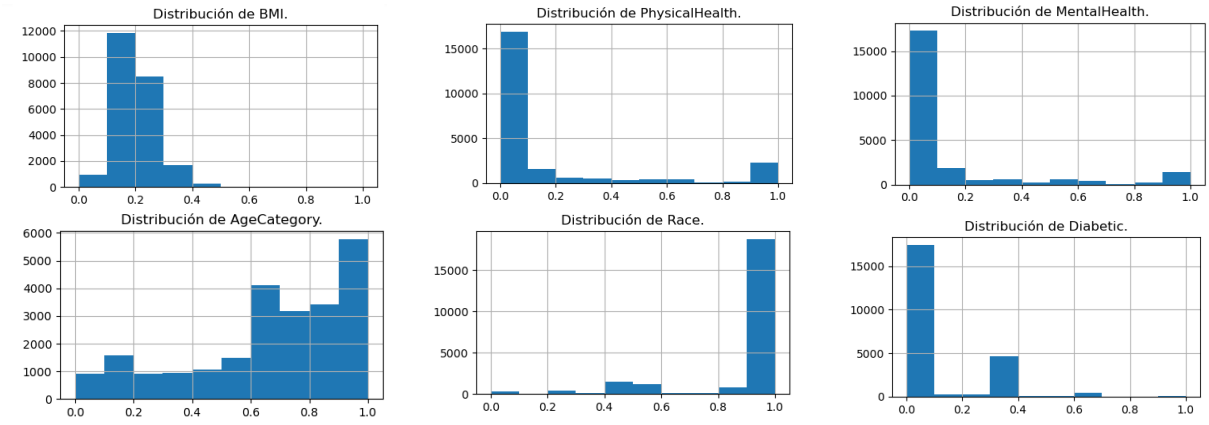
En la Ilustración 19 se muestra como de las 1130 instancias que se tenían originalmente para la clase Yes, se crearon 10,532 datos sintéticos para igualar la cantidad de datos de la clase No.

Así bien, en la Tabla 3 se presentan la comparación entre la desviación estándar de la base de datos y la desviación estándar posterior a la implementación del método de SMOTE.

Tabla 3. Desviación Estándar previo y posterior a la implementación de la técnica SMOTE.

	Base de datos sin aumento de datos	**Base de datos con aumento de datos**
BMI	0.076276	0.072162
PhysicalHealth	0.263234	0.316526
MentalHealth	0.268345	0.273796
AgeCategory	0.303308	0.282288
Race	0.243580	0.224651
Diabetic	0.165617	0.172332
Smoking	0.491930	0.499709
AlcoholDrinking	0.252362	0.231902
Stroke	0.186040	0.293293
DiffWalking	0.342890	0.422544
PhysicalActivity	0.418917	0.450429

A partir de las desviaciones estándar mostradas en la Tabla 3 se puede observar que los valores de esta métrica antes y después de los datos sintéticos. Así mismo, en la Ilustración 20 se presentan las distribuciones de los atributos, las cuales no se ven afectadas radicalmente.



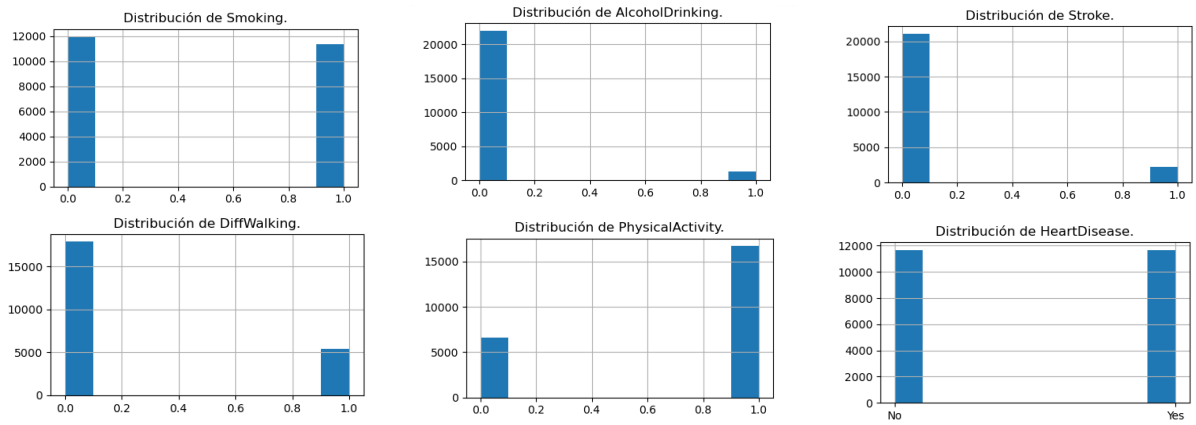


Ilustración 20 Distribución de la implementación del método SMOTE.

Por otra parte, se muestra en la siguiente tabla la comparación entre los dos métodos implementados.

Tabla 4. Comparación de las Desviaciones Estándar de los métodos implementados: Ruleta y SMOTE.

	Base de datos sin aumento de datos	**Método Ruleta**	**Método SMOTE**
BMI	0.076276	0.076578	0.072162
PhysicalHealth	0.263234	0.264886	0.316526
MentalHealth	0.268345	0.268425	0.273796
AgeCategory	0.303308	0.302981	0.282288
Race	0.243580	0.246275	0.224651
Diabetic	0.165617	0.166645	0.172332
Smoking	0.491930	0.493366	0.499709
AlcoholDrinking	0.252362	0.253289	0.231902
Stroke	0.186040	0.187957	0.293293
DiffWalking	0.342890	0.341343	0.422544
PhysicalActivity	0.418917	0.418732	0.450429

Los resultados presentados en la Tabla 4 demuestran que el método para la generación de datos sintéticos SMOTE afecta ligeramente más la desviación estándar de los datos originales si se compara con el método de la ruleta. Ya que, si bien con el método de la ruleta varían los valores en cuestión de milésimas, con SMOTE varia en medida de centésimas y décimas. La razón por lo

que se realiza la comparación entre las desviaciones estándares de los dos métodos en cuestión es debido a que es importante mantener una distribución muy parecida a la original a fin de no sesgar la etapa de entrenamiento y no obtener resultados diferentes a los esperados en la etapa de prueba.

Conclusiones

En este programa generalizado inicialmente se generaron funciones para preparar la base de datos en cuestión a fin de poder implementar posteriormente dos métodos para la creación de datos sintéticos, esto con el fin de identificar los puntos fuertes de cada uno de ellos.

Comenzando con el método de la ruleta, por su naturaleza, al ser un método estocástico agrega aleatoriedad a los datos sintéticos generados, sin embargo, no considera si existe un desbalance de clases en la base datos, por lo que pudiera agravar este problema. Por otro lado, el método SMOTE, a pesar de que cambia ligeramente la desviación estándar de los datos, soluciona el problema del desbalance de clases. Por lo tanto, se puede concluir que, como en todos los métodos de Machine Learning, el uso de estos métodos depende mucho de las aplicaciones y necesidades específicas del conjunto de datos con el cual se trabajará.

Así mismo, actualmente, es muy común contar con base de datos no balanceadas como es en el caso del área médica donde suelen presentarse casos muy esporádicos de pacientes con alguna enfermedad en cuestión, lo cual termina siendo un desafío si es que se desea utilizar dicha información en modelos de inteligencia artificial, por lo cual la implementación de técnicas de aumento de datos suele ser una vía prometedora para solucionar esta problemática.

Referencias

[1] M. Antonio and A. Fernández, *Inteligencia artificial para programadores con prisa by Marco Antonio Aceves Fernández - Books on Google Play*. Universo de Letras. [Online]. Available: https://play.google.com/store/books/details/Inteligencia_artificial_para_programadores_con_pri?id=ieFYEAAAQBAJ&hl=en_US&gl=US

[2] Chawla, N. V. et al. (2002). SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357. Doi: 10.1613/jair.953

[3] "Personal Key Indicators of Heart Disease | Kaggle." https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease/code (accessed Aug. 25, 2022). https://learn.microsoft.com/es-es/azure/machine-learning/component-reference/smote