DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
Inteligencia de negocios 202220 – Laboratorio 1
PROFESORA: Haydemar Nuñez

Nombres	**Apellidos**	**Código**	**Login**
María Sofía	Álvarez López	201729031	ms.alvarezl
Brenda Catalina	Barahona Pinilla	201812721	bc.barahona
Alvaro Daniel	Plata Márquez	201820098	ad.plata

Informe de laboratorio #1

Introducción

En este informe se detallan los datos, procedimientos, decisiones que se siguieron para el desarrollo de un proyecto de Inteligencia de Negocios cuyo objetivo es ayudar a la empresa SaludAlpes a agilizar sus procesos de análisis y diagnóstico de diabetes en sus pacientes. Para esto la empresa proveyó un conjunto de datos en formato CSV con información relevante sobre sus pacientes y diagnósticos anteriores. El procedimiento general del proyecto y sus resultados se muestran a continuación.

Perfilamiento de datos

Comenzaremos con el perfilamiento de los datos, para esto, se utilizará las siguientes herramientas:

- *pandas_profiling,* haciendo uso de método report=pandas_profiling.ProfileReport(data)
- Tablero de control en power BI

Al realizar el perfilamiento en pandas, los primeros datos que nos dan son los siguientes:

Dataset statistics

Number of variables	27
Number of observations	100000
Missing cells	500333
Missing cells (%)	18.5%
Duplicate rows	5842
Duplicate rows (%)	5.8%
Total size in memory	125.7 MiB
Average record size in memory	1.3 KiB

Variable types

CAT	22
UNSUPPORTED	5

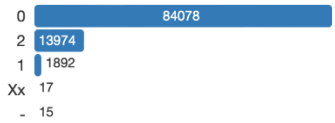
En esta tabla podemos ver que hay 22 variables categóricas y hubo problema cargando 5 de ellas. Adicional a esto, podemos evidenciar el porcentaje de filas duplicadas y de celdas que no tienen valores (o NaN), estos porcentajes son 5.8% y 18.5% respectivamente. Es importante notar que puede haber varias filas con celdas en NaN, para ello, se hace una exploración más exhaustiva más adelante.

Las características específicas de cada variable son las siguientes:

A continuación, procedemos a analizar cada una de las variables para determinar sus características y determinar qué pasos se deben seguir en la preparación y limpieza de los datos

- **Diabetes_012:** Corresponde a la variable objetivo. Sus valores válidos son 0, 1 y 2. Indica si la persona ha tenido diabetes, prediabetes o ninguna de las dos. Se puede observar que la variable cuenta con valores no válidos por el diccionario (“Xx”, “-”, “’?”), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

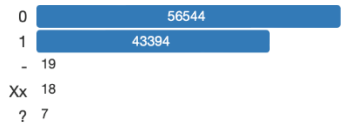
Distinct	6
Distinct (%)	< 0.1%
Missing	16
Missing (%)	< 0.1%
Memory size	781.4 KiB



- **HighBP:** Corresponde a una variable binaria. Indica si la persona ha tenido la presión arterial alta o no. Se puede observar que la variable cuenta con valores no válidos por el diccionario (“Xx”, “-”, “’?”), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y

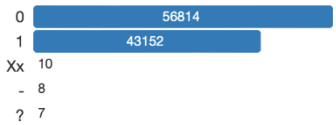
preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	18
Missing (%)	< 0.1%
Memory size	781.4 KiB



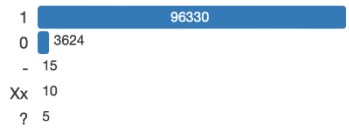
- **HighChol:** Corresponde a una variable binaria. Indica si la persona ha tenido el colesterol alto o no. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	9
Missing (%)	< 0.1%
Memory size	781.4 KiB



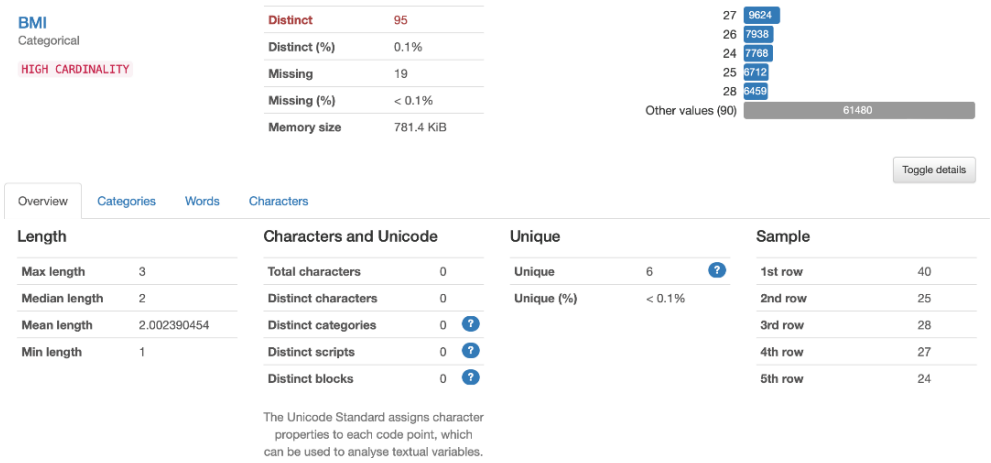
- **CholCheck:** Corresponde a una variable binaria. Indica si la persona ha revisado sus niveles de colesterol en los últimos 5 meses o no. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	16
Missing (%)	< 0.1%
Memory size	781.4 KiB



- **BMI:** Sus valores posibles son números enteros mayores a 1 y menores a 99. Indica el índice de masa corporal de la persona. Se puede observar, por el atributo de longitud máxima, que hay valores que está fuera del rango válido, al igual que casillas vacías, lo que se tendrá que corregir en el

proceso de limpieza y preparación de datos.



- **Smoker:** Corresponde a una variable binaria. Indica si la persona ha fumado al menos 100 cigarrillos en su vida. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

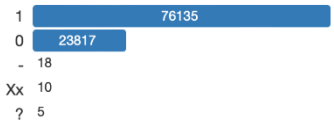
- **Stroke:** Corresponde a una variable binaria. Indica si la persona ha tenido alguna vez un derrame cerebral. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

- **HeartDiseaseorAttack:** Corresponde a una variable binaria. Indica si la persona alguna vez ha tenido enfermedad coronaria (CHD) o infarto de miocardio (MI). Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

- **PhysActivity:** Corresponde a una variable binaria. Indica si la persona ha realizado actividad física durante el último mes. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al

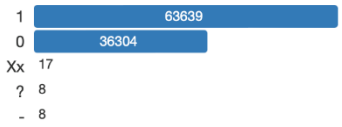
igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	15
Missing (%)	< 0.1%
Memory size	781.4 KiB



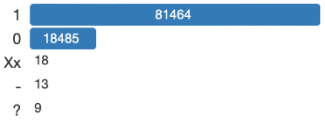
- **Fruits:** Corresponde a una variable binaria. Indica si la persona consume al menos una porción de fruta al día. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	24
Missing (%)	< 0.1%
Memory size	781.4 KiB



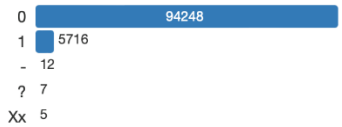
- **Veggies:** Corresponde a una variable binaria. Indica si la persona es consumidora frecuente de alcohol. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	11
Missing (%)	< 0.1%
Memory size	781.4 KiB



- **HvyAlcholConsump:** Corresponde a una variable binaria. Indica si la persona consume al menos una porción de vegetales al día. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

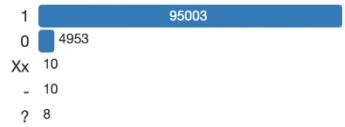
Distinct	5
Distinct (%)	< 0.1%
Missing	12
Missing (%)	< 0.1%
Memory size	781.4 KiB



- **AnyHealthcare:** Corresponde a una variable binaria. Indica si la persona tiene algún tipo de seguro o sistema de salud. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de

limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	16
Missing (%)	< 0.1%
Memory size	781.4 KiB

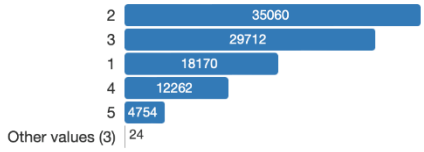


- **NoDocbcCost:** Corresponde a una variable binaria. Indica si la persona tuvo necesidad de ir con un médico, pero no lo hizo debido a su costo durante el último año. Se puede observar que la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"), al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	5
Distinct (%)	< 0.1%
Missing	22
Missing (%)	< 0.1%
Memory size	781.4 KiB

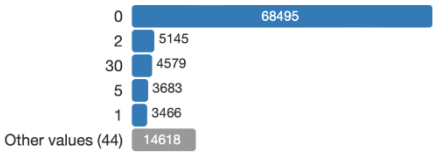
- **GenHlth:** Sus valores posibles son números enteros de 1 a 5. Indica la calificación general que le asigna la persona a su salud (1=excelente, 2=muy bueno, 3=bueno, 4=regular o 5=malo). Se puede observar que hay valores que están fuera del rango válido, al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	8
Distinct (%)	< 0.1%
Missing	18
Missing (%)	< 0.1%
Memory size	781.4 KiB



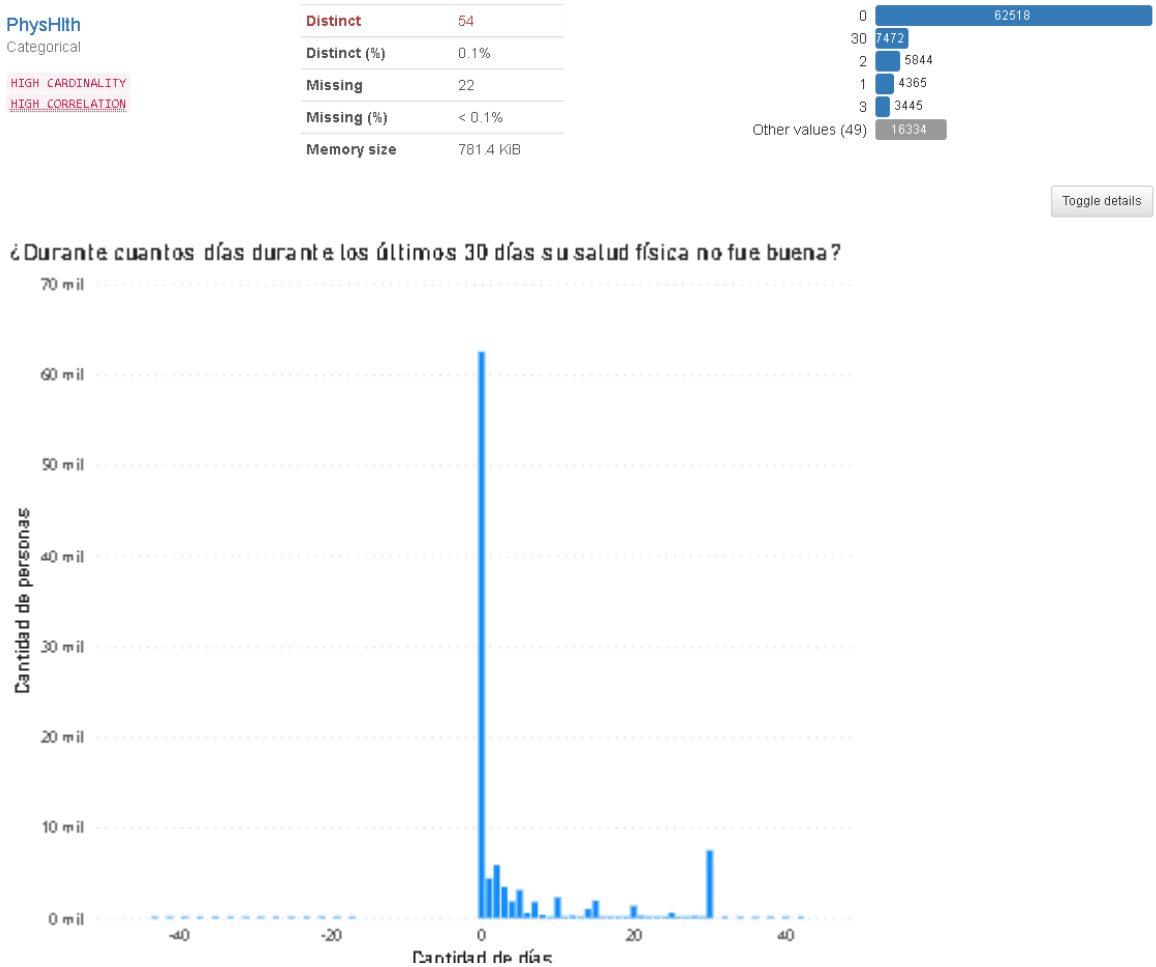
- **MentHlth:** Sus valores posibles son números enteros mayores a 1 y menores a 30. Indica la cantidad de días durante el último mes que la persona no tuvo una buena salud mental. Se puede observar que hay valores que están fuera del rango válido, al igual que casillas vacías, lo que se tendrá que corregir en el proceso de limpieza y preparación de datos.

Distinct	49
Distinct (%)	< 0.1%
Missing	14
Missing (%)	< 0.1%
Memory size	781.4 KiB

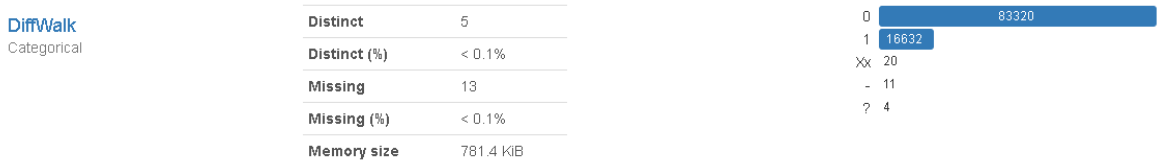


- **PhysHlth:** Esta variable entera representa cuantos días, durante el último mes, la persona tuvo una mala salud física. En esta variable se puede ver que hay una gran correlación, que hay muy pocos valores de NaN y que, además, hay una gran cardinalidad. Lo anterior es porque pandas toma como categórica esta variable, y define que hay muchas categóricas, incluso varias de las cuales solo se repiten una vez, es decir que tienen únicos valores. Para esta ocasión nos apoyaremos también en Power Bi para saber

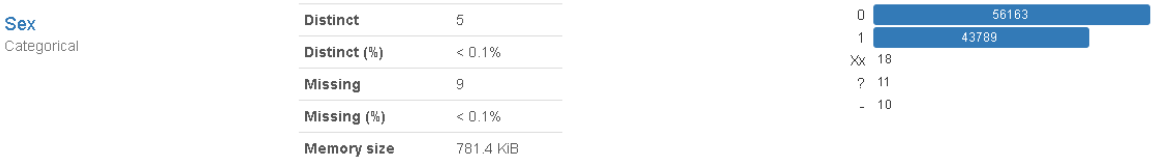
si se cumple con el rango, podemos ver que no se cumple, pues hay valores negativos.



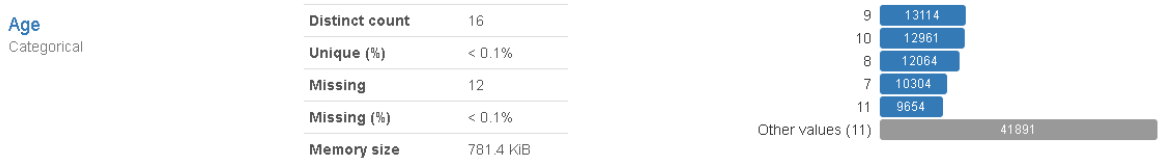
- **DiffWalk:** Esta variable binaria representa si la persona tiene serios problemas para subir o bajar las escaleras. En caso de que si los tenga, esto se representa como 1 y en caso de que no, se representa como 0. Al ver el reporte de pandas, podemos ver que hay celdas vacías, lo cual representa menos del 0.1%. Además, la variable cuenta con valores no válidos por el diccionario ("Xx", "-", "'?"). Por otro lado, los valores numéricos presentados, muestran que se cumple con el rango [0,1]. Es de anotar, que esta variable en el diccionario es binaria, pero en el reporte aparece como "Categórica", esto se da porque hay presencia de los valores no válidos mencionados anteriormente.



- **Sex:** Esta variable binaria, donde 0 es para femenino y 1 para masculino. Al ver el reporte de pandas, podemos ver que hay celdas vacías, lo cual representa menos del 0.1%. Además, la variable cuenta con valores no válidos por el diccionario (“Xx”, “-”, “’?”). Por otro lado, los valores numéricos presentados, muestran que se cumple con el rango [0,1].Es de anotar, que la variable Sex en el diccionario es binaria, pero en el reporte aparece como “Categórica”, esto se da porque hay presencia de los valores no válidos mencionados anteriormente.



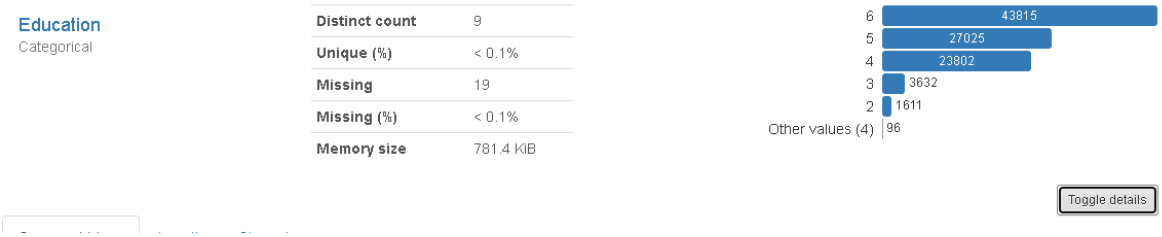
- **Age:** Variable categórica que indica la edad. Este rango va de 1 a 13, donde 1 representa de 18 a 24 años y 13 representa de 80 o más. En el reporte podemos ver que, en una primera revisión los datos no se pasan del rango establecido en el diccionario. Sin embargo, mas adelante debemos revisarlos ya que en le categoría “other values” puede haber valores que no son válidos. Adicional a esto, podemos ver que debemos tratar los valores NaN, los cuales representan menos del 0.1% de estos datos.



Toggle details

Value	Count	Frequency (%)
9	13114	13.1%
10	12961	13.0%
8	12064	12.1%
7	10304	10.3%
11	9654	9.7%
6	7659	7.7%
13	6987	7.0%
12	6361	6.4%
5	6259	6.3%
4	5310	5.3%
Other values (6)	9315	9.3%

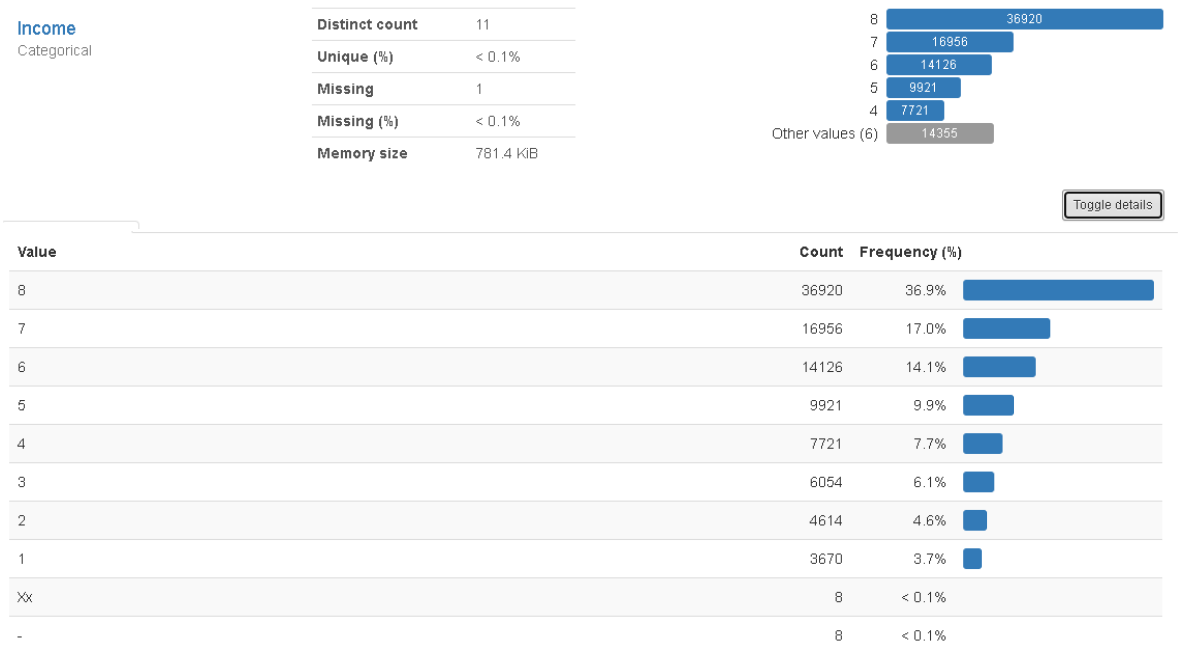
- **Education:** Esta variable es categórica. Según el diccionario, indica el nivel de educación, el cual va en un rango de enteros positivos de 1 a 6. Al ver esta variablePodemos ver que hay un porcentaje bajo de valores en blanco y también podemos ver que se cumple el rango establecido en el diccionario. También cabe anotar que hay valores no válidos como (“Xx”,”-”,”?”), cada uno de estos tiene una aparición menor al 0.1%



Toggle details

Value	Count	Frequency (%)
6	43815	43.8%
5	27025	27.0%
4	23802	23.8%
3	3632	3.6%
2	1611	1.6%
1	58	0.1%
-	21	< 0.1%
Xx	12	< 0.1%
?	5	< 0.1%
(Missing)	19	< 0.1%

- **Income**: Esta variable es categórica y que, según el diccionario, indica una escala de ingresos, este oscila en un rango de enteros positivos de 1 a 8. Al ver lo que nos dice el reporte de pandas, podemos ver que hay un pequeño porcentaje de valores vacíos. Adicionalmente, podemos ver que esta variable cumple con su rango de 1 a 8 y tiene valores que no corresponden y que no son válidos, estos son (“Xx”, “-“)



Income
Categorical

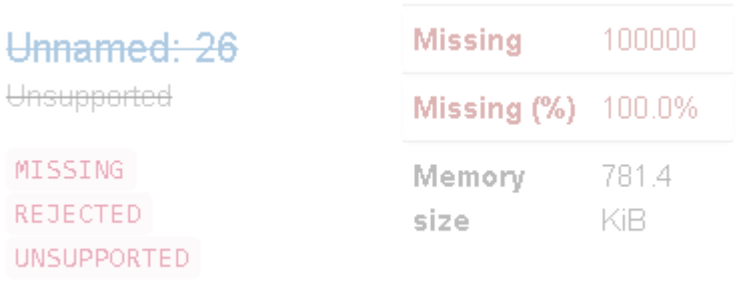
Distinct count	11
Unique (%)	< 0.1%
Missing	1
Missing (%)	< 0.1%
Memory size	781.4 KiB

8 36920
7 16956
6 14126
5 9921
4 7721
Other values (6) 14355

Toggle details

Value	Count	Frequency (%)
8	36920	36.9%
7	16956	17.0%
6	14126	14.1%
5	9921	9.9%
4	7721	7.7%
3	6054	6.1%
2	4614	4.6%
1	3670	3.7%
Xx	8	< 0.1%
-	8	< 0.1%

Unmaned 22, Unmaned 23, Unmaned 24, Unmaned 25, Unmaned 26: Para todas estas columnas, nos aparece que el 100% de los valores son NaN, por lo que no tienen ningún dato que aporte a nuestra variable de interés. Esto, relacionándolo con lo que vimos en la primera tabla mostraba, podemos afirmar que esas 5 columnas o variables que nos aparecían como "No soportadas" corresponden a estas.



Unnamed: 26
Unsupported

MISSING
REJECTED
UNSUPPORTED

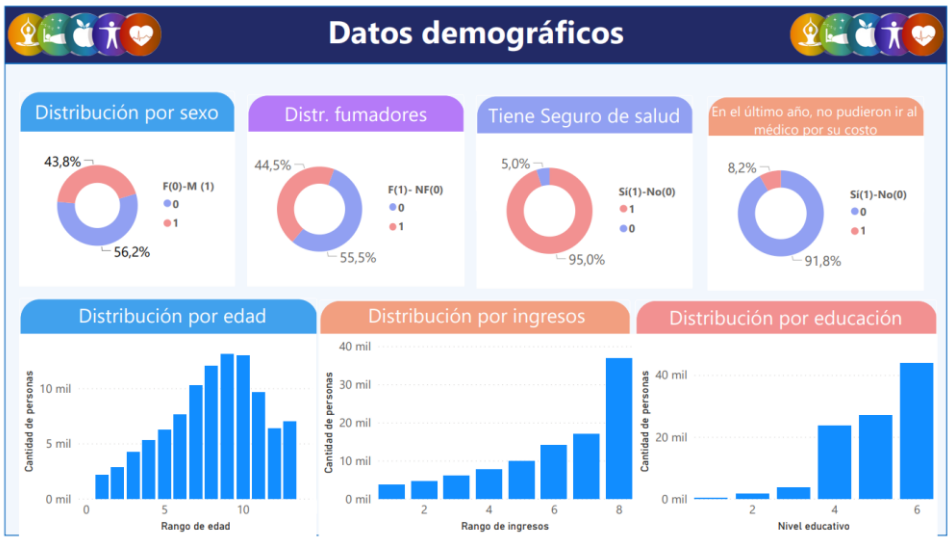
Missing	100000
Missing (%)	100.0%
Memory size	781.4 KiB

Tablero de control utilizado para el análisis de datos (perfilamiento y análisis de calidad)

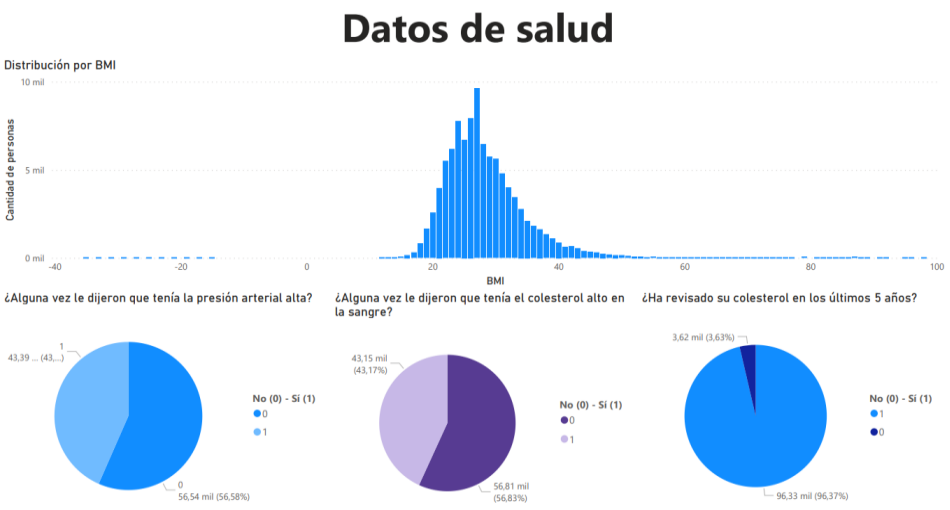
Para facilitar la visualización y comprensión de los datos y su calidad, se creó un conjunto de tableros de control usando la herramienta Power BI. En estos tableros se muestran los datos agrupados según su naturaleza en:

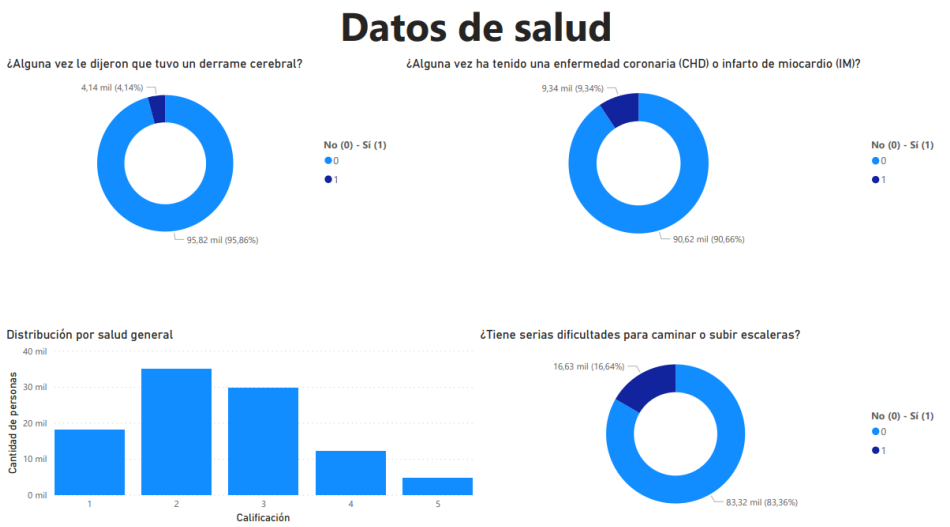
- Exploración y análisis de calidad

- Datos demográficos

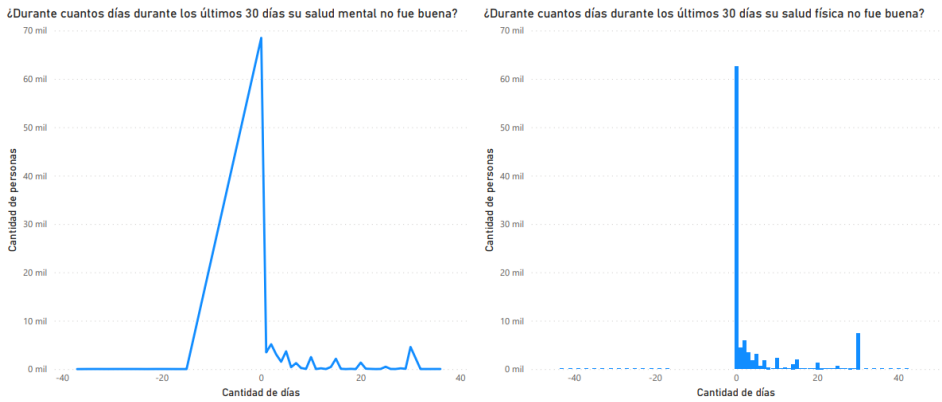


- Datos de salud



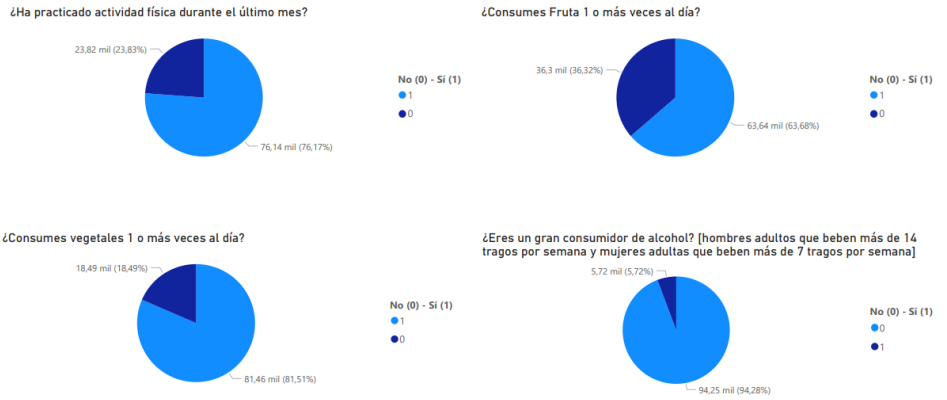


Datos de salud



- Datos de hábitos

Datos de hábitos



Preprocesamiento y procesamiento de los datos: limpieza de datos

---

1. <u>Preprocesamiento de datos:</u>

Anteriormente nos dimos cuenta de que tenemos los siguientes errores en los datos:

- Datos NaN en la totalidad de filas de las columnas unnamed
- Datos no válidos (“Xx”, “-”, “?”)
- Datos fuera de rango
- Datos NaN

Sabemos que, para cualquiera de los algoritmos que vayamos a usar, debemos limpiar los datos y librarlos de los errores que se mencionaron antes. Comenzaremos separando los datos, el 80% (es decir, 80000) de estos irán para entrenamiento y el 20% restante serán para test. Sabemos que nuestra variable de interés es Diabetes_012. En primera medida, revisaremos si esta variable tiene sus clases balanceadas, de esto podemos ver lo siguiente:

	Diabetes_012
0	67248
2	11190
1	1520
Xx	15
-	10
?	4

Notamos entonces que, en efecto, las clases están desbalanceadas. Más aún, vemos que hay mucha basura, con caracteres que no están en el diccionario y que no tienen sentido con los datos. Estos debemos eliminarlos, como se hará más adelante.Note que tenemos 67248 datos de la clase 0 (Normal), 1520 de la clase 1 (Prediabetes) y 11190 de la clase 2 (Diabetes).

- **Tratamiento de datos NaN de las columnas unnamed**

Comenzaremos eliminando las filas unnamed, que, como ya vimos en el punto anterior, estas filas están llenas de NaN y no aportan nada a la variable de interés

- **Tratamiento de valores no válidos ("Xx", "-", "?")**

Ahora, ya tenemos todas las columnas que se encuentran en el diccionario. Lo siguiente que queremos es revisar si las columnas estaban presentando otro tipo de datos no numéricos o no válidos. Pensamos que, gracias a estos valores, los tipos de valores que está tomando el programa es diferente a numérico. Esto lo comprobamos pidiento los tipos de valores para cada columna

```
HighBP                object
HighChol              object
CholCheck             object
BMI                   object
Smoker                object
Stroke                object
HeartDiseaseorAttack  object
PhysActivity          object
Fruits                object
Veggies               object
HvyAlcoholConsump     object
AnyHealthcare         object
NoDocbcCost           object
GenHlth               object
MentHlth              object
PhysHlth              object
DiffWalk              object
Sex                   object
Age                   object
Education             object
Income                object
Diabetes_012          object
dtype: object
```

Como podemos ver, efectivamente pandas no está reconociendo los tipos de valores correctos para las variables. Para solucionar esto, primero buscamos en el conjunto de datos todos los valores distintos que hay en nuestro dataframe y los convertimos en una lista. Luego, iteramos sobre ellos e intentamos convertirlos (hacer el cast) a números flotantes, usando un bloque try-except. Si se lanza una excepción, significa que el dato no es numérico y lo agregamos a la lista, para saber qué otro tipo de datos tiene nuestro dataframe. El código usado es

```
# Vemos todos los valores distintos (únicos) que tiene el dataframe.
valores_distintos = list(datos_train.apply(pd.value_counts).index)
valores_no_numericos = []
# Interamos sobre los valores distintos
for val in valores_distintos:
    try:
        # Intentamos hacer el cast a float
        numero = float(val)
    except:
        # Si hay datos no numericos, los agregamos a la lista
        valores_no_numericos.append(val)
print('Los valores no numéricos son: {}'.format(valores_no_numericos))
```

De lo anterior, confirmamos lo que vimos en el reporte de pandas, y es que hay datos no válidos, y estos son ['-', '?', 'Xx']. Luego de esto, revisamos cuántas filas tienen este tipo de error. Este valor nos da **589,** el porcentaje de estos datos inválidos de **0.74%,** lo cual es bastante reducido, por lo que procedemos a eliminarlos.

Luego de eliminarlos, quedamos con una cantidad de 79411 filas para el dataset de entrenamiento.

- **Tratamiento de los valores fuera de rango.**

Para hacer este análisis de una manera más rápida, revisaremos si las varables binarias, efectivamente tienen solo dos posibles valores en sus filas. Analizamos las variables 'HighBP', 'HighChol', 'CholCheck', 'Smoker', 'Stroke','HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'DiffWalk' y 'Sex'. Para cada una de estas variables obtuvimos que los valores únicos son [0,1,nan], lo cual, a excepción del nan, cumple con los rangos datos en el diccionario de datos.

Luego analizamos las demas columnas, obteniendo su máximo y su mínimo para identenficar cuales de estas se salen de su rango. En esto obtuvimos:

	Max	Min
BMI	98	-35
GenHlth	5	1
MentHlth	36	-37
PhysHlth	42	23
Age	13	1
Education	6	1
Income	8	1

Con lo obtenido y comparando con el diccionario que nos dan, nos damos cuenta de que las variables que están fuera de los rangos establecidos son:

- BMI (debe ser >1 y <99): Se pasa del límite inferior

- MentHlth (debe ser >1 y <30): Se pasa del límite inferior y superior
- PhysHlth (debe ser >1 y <30): Se pasa del límite inferior y superior

Luego, revisamos cuantos valores hay fuera del rango, para saber cómo manejar estos. Obtuvimos lo siguiente:

```
Datos fuera del rango para BMI:  210
Datos fuera del rango para MentHlth:  242
Datos fuera del rango para PhysHlth:  302
En total hay 754 datos fuera del rango
```

Aunque puede haber filas con más de un valor fuera de rango, que el máximo valor posible de filas con valores fuera de rango es de 0.95% (esto ocurre porque 2 o más columnas en la misma fila tienen valores fuera del rango). Como este porcentaje es muy pequeño, procederemos a eliminar estas filas de datos train.

Es importante tener en cuenta que en el diccionario de los datos se indica que el rango de valores válidos para las variables MentHlth y PhystHlth es >=2 y <=29. Sin embargo, teniendo en cuenta la naturaleza y/o el significado de estas dos variables, se tomó la decisión de utilizar el rango >=0 y <=30 ya que, no existe ninguna razón por la cual una persona no pueda indicar que tuvo 0, 1 o 30 días de mala salud (física o mental) en el mes.

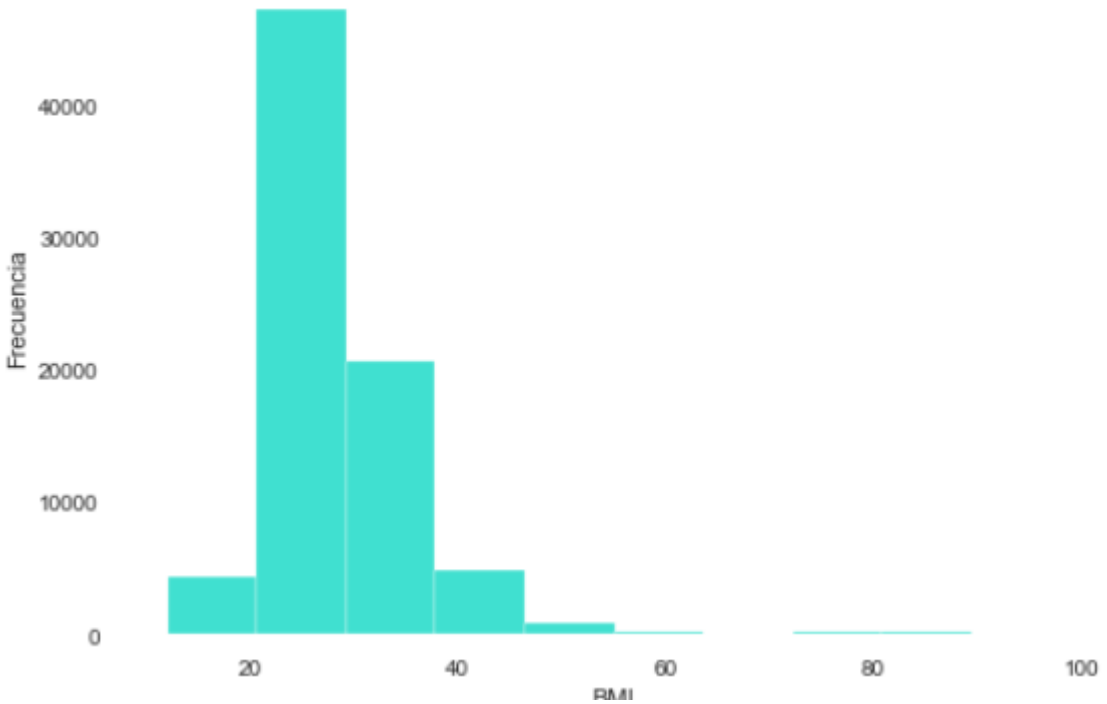
Procedemos a sacar el porcentaje que representan estos datos y vemos que es el **0.95%.** Al ver que este porcentaje es muy bajo, procedemos a eliminar estas filas. Con esto, nuestro nuevo tamaño del conjunto de los datos es de 78659.

- **Tratamiento de los valores nulos NaN.**

Para continuar con este segmento, sacamos el número de datos NaN presentes. En esto obtenemos 266 datos, lo cual representa el 0.34% de los datos. Aunque puede haber filas con más de un valor NaN, sabemos que el máximo valor posible de filas con valores NaN es de 0.34%. No obstante, por evaluar otra forma de preprocesamiento, procederemos a inputar.

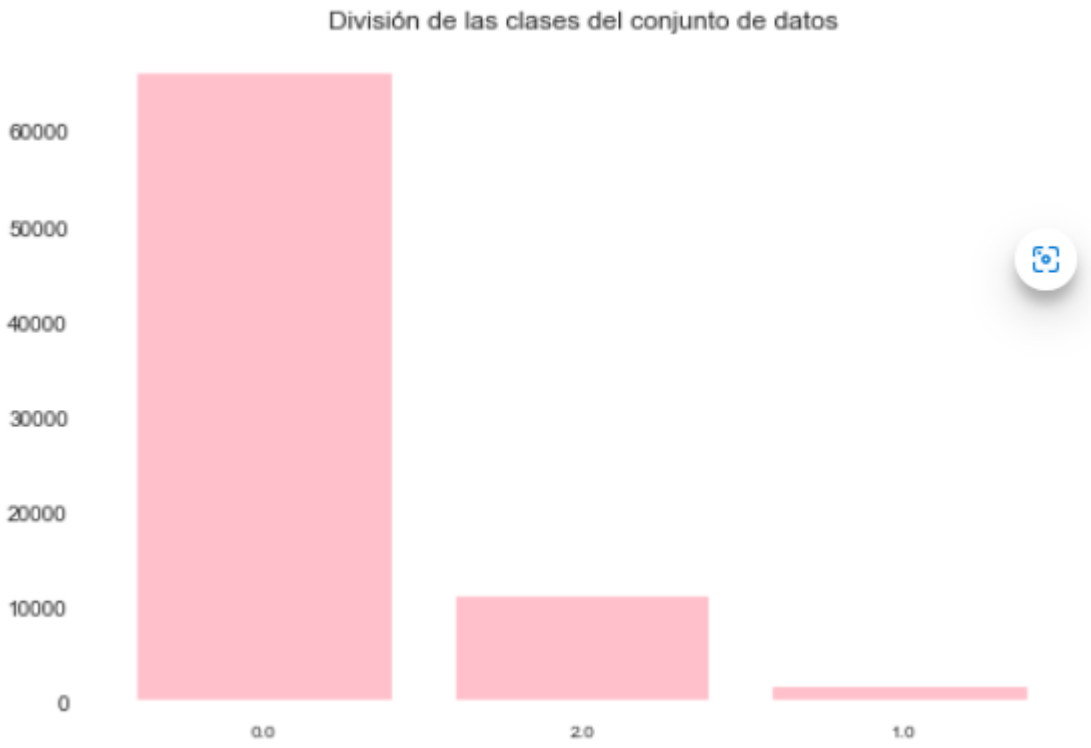
Cabe aclarar que es fundamental eliminar los registros donde haya NaN en la variable objetivo, pues no podemos imputar sobre la variable que queremos predecir. Obtuvimos que la cantidad de veces que hay datos NaN en la variable objetivo es **13**. Así que procedemos a eliminar esas filas. Con esto, nos queda un tamaño de **78646** en nuestro conjunto de datos.

Como vimos más arriba, el número de NaNs del conjunto de datos es realmente bajo. Entonces, podríamos eliminarlos. No obstante, por intentar otra técnica, podemos imputar esta variable con alguna medida de tendencia central. Debido a que tenemos variables categóricas (en su mayoría) podemos usar la moda para ellas. En el caso de BMI, podríamos usar la media al ser una variable real. No obstante, veamos primero la distribución de esta variable:



Y obtenemos una moda de **27** y una media de **28.47.** En este caso, es prácticamente indistinto el usar cualquiera de estas dos variables, así que tomemos la moda para que el procesamiento sea similar al de las otras variables del problema. Para ello, usamos el SimpleImputer() de sklearn. Hay que tener en cuenta que, el caso de cuando vayamos a preprocesar el test, NO podemos imputar. Por ello, definimos una función en que los datos con NaN en el test son eliminados.

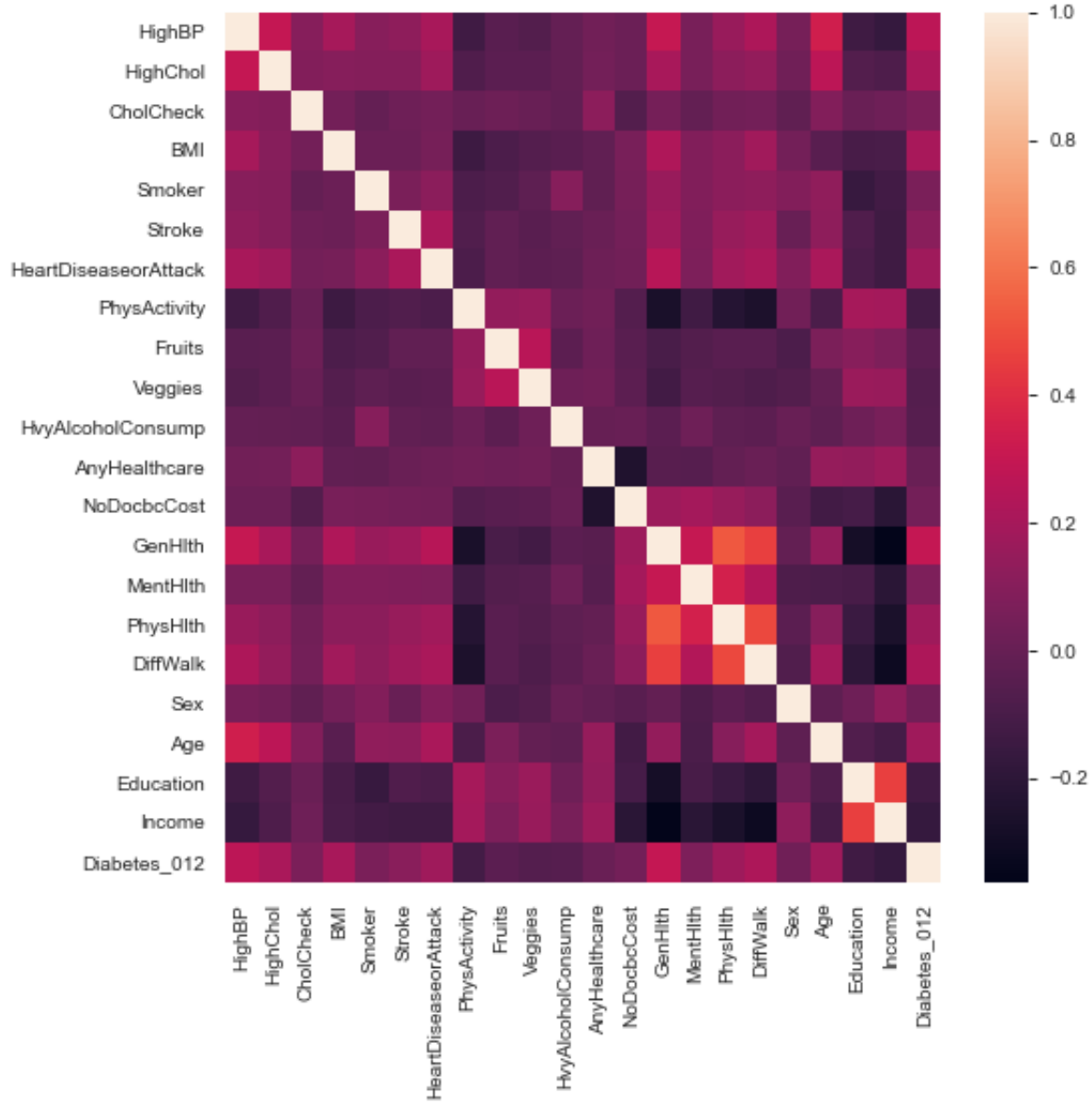
Con esto realizado, procederemos a revisar cuántas filas quitamos hasta ahora con la limpieza.



	Diabetes_012
0.0	66153
2.0	10994
1.0	1499

En comparamos con la primera gráfica que hicimos, vemos que, de la clase 2, apenas hemos eliminado 200 datos incompletos, lo cual no es malo. De la clase 0 hemos quitado alrededor de 1000. No obstante, al ser una clase mayoritaria, no es un problema mayor. La clase 1 la trataremos más adelante, pero no nos es de particular interés por la naturaleza binaria de nuestro problema.

Para finalizar esta parte de preprocesamiento, evaluaremos la correlación entre nuestras variables, una vez eliminada la basura. Esto podría ser de interés si queremos aplicar un algoritmo como regresión logística.



Podemos ver que las correlaciones positivas (ni negativas) son lo suficientemente fuertes para considerar eliminar alguna variable. Hay algunas correlaciones (leves) entre variables como DiffWalk y PhysHealth, lo cual tiene sentido en el contexto del problema, pero no son realmente considerables.

Con lo anterior, finalizamos la parte de preprocesamiento y comenzamos con el procesamiento de los datos

2. Procesamiento de datos

Ahora procedemos a realizar el procesamiento de los datos. En esta sección realizamos

- Modificación de la variable de interés
- Tratamiento de valores duplicados

- Balanceo de clases
- Modificaciones extra para hacer uso de los algoritmos

La modifcación de la variable de interés se realizará porque decidimos que, de acuerdo con el negocio, queremos saber si un paciente tiene diabetes **diagnosticada** o si no tiene diabetes. Es decir, tenemos un problema de clasificación binario. En este contexto, los datos que están clasificados con 1 (i.e. con prediabetes) son problemáticos. Con esto dicho, comencemos:

- **Modificación de la variable de interés**

La prediabetes incluye rangos de insulina entre 100 y 125 mg/dL. No obstante, no tenemos forma de saber si los datos suministrados por los usuarios están más cerca de tener niveles de insulina normales (i.e. menores a 99 mg/dL) o diabéticos (mayores a 126 mg/dL) [1]. En este contexto, es difícil decidir si se debieran juntar los datos de prediabéticos con los de diabéticos (clase 2) o los no diabéticos (clase 1). Considerando además que, tras el preprocesamiento, únicamente hay 1499 datos de clase 1, es una decisión sensata eliminarlos. Por lo tanto, procedemos a eliminar estos datos y obtenemos lo siguiente:

	Diabetes_012
0.0	66153
2.0	10994

- **Tratamiento de datos duplicados**

Lo siguiente que debemos evaluar es decidir sobre los duplicados del conjunto de datos. Los datos duplicados son un problema puesto que pueden sesgar al algoritmo y al análisis. Más aún, en el momento de usar CrossValidation, pueden obtenerse scores sesgados para las métricas usadas. Lo anterior puesto que algunos de los datos duplicados pueden usarse tanto en el entrenamiento como en la validación, haciendo que el algoritmo esté sesgado y se obtengan mejores métricas que las que realmente deberían ser (pues el algoritmo ya vio los mismos datos).

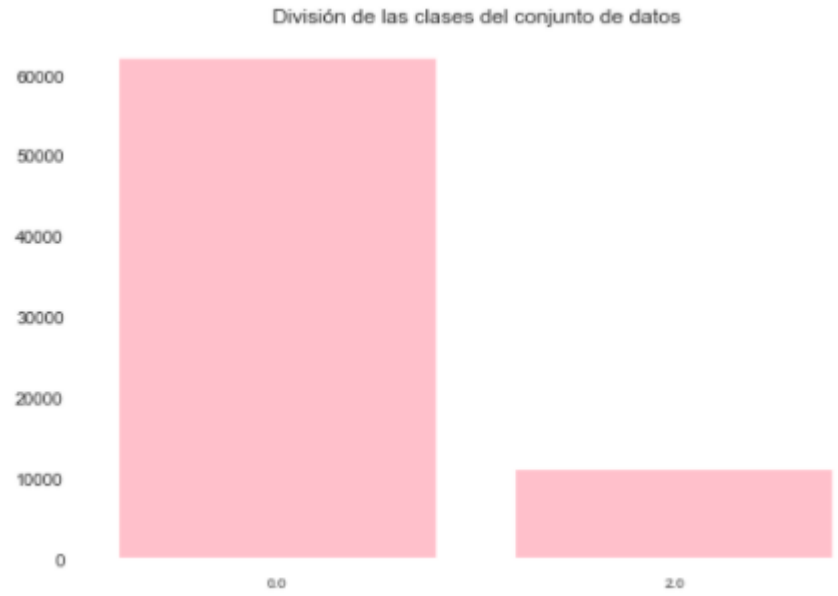
La cantidad de datos duplicados es de 4072, lo cual representa el 5.28% de los datos. Este porcentaje no es poco, procederemos a revisar por clase cuantos duplicados hay:

	Diabetes_012
0.0	4045
2.0	27

Vemos que la mayoría de los duplicados hacen parte de la clase 0, lo cual da un poco de tranquilidad pues es la clase mayoritaria (i.e. de la que más datos tenemos). Así, no es preocupante eliminar estos datos. Procedemos, entonces, a eliminar los datos duplicados del conjunto train.

- Balanceo de clases:

Nuestro conjunto de datos actual es:



En este se hace evidente que necesitamos alguna técnica de balanceo de datos. En este caso, usaremos SMOTE (Synthetic Minority Over-sampling Technique) para balancear las clases. Elegimos hacer SMOTE en lugar de undersampling, pues, aunque tenemos un set de datos considerable, no queremos disminuir nuestro espacio de muestreo de una forma tan voraz. Así las cosas, aplicamos el algoritmo de SMOTE al conjunto de datos.

Es importante mencionar que SMOTE recibe parámetros como la estrategia de remuestreo y la cantidad de vecinos que se usan para determinarlo. Aunque son parámetros interesantes, para este primer laboratorio nos lo realizaremos con la versión por defecto que incluye la librería.

- **Modificaciones extra para hacer uso de los algoritmos**

Primero que todo, es importante mencionar que tenemos variables categóricas (como education, income y GenHealth), numéricas (como BMI, MenthHlth y PhysHlth) y binarias. Es importante notar que las variables categóricas tienen un carácter ordinal: es decir, tiene sentido organizarlas de menor a mayor (de menor income a mayor income, por ejemplo). Por esto, en un principio pensamos en

mantener esta codificación ordinal para el algoritmo, y escalar los resultados entre 0 y 1 (Esto es necesario únicamente para el algoritmo que usa KNN).

No obstante, otra forma de codificar las variables es usando OneHotEncoding. Esto nos sirve para las categóricas. Decidimos probar ambos casos y pasarlo como hiperparámetro para ver cuál es la que escoge nuestro modelo.

Es importante notar que las variables numéricas deben ser escaladas en ambos casos (para KNN). Procedemos a seleccionar las columnas numéricas de las categóricas. A las numéricas, les haremos un escalamiento, en este caso dado por una normalización a 1. Esta es conveniente pues conserva la misma distribución de los datos, pero los lleva a la misma escala, tal que no haya algunas variables que, pesen más que otras al construir el modelo. Este escalamiento también lo hacemos a las categóricas después de haberlas codificado como ordinales.

Entonces, después de realizar el Smote , podemos observar los siguiente:

En este diagrama se puede observer que el oversampling se hizo adecuadamente. Ahora ambas clases tienen la misma cantidad de datos.

El dato de balanceo lo hicimos dentro de un pipeline, donde, ahí mismo escalamos los datos, por lo que, revisaremos que los datos hayan quedado correctamente escalados

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack
count	124216.000000	124216.000000	124216.000000	124216.000000	124216.000000	124216.000000	124216.000000
mean	0.575939	0.539795	0.975993	0.208992	0.482788	0.059545	0.144715
std	0.493128	0.497572	0.151629	0.082647	0.499893	0.233260	0.350437
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	0.156161	0.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000	0.197674	0.000000	0.000000	0.000000
75%	1.000000	1.000000	1.000000	0.244186	1.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Es fácil intuir que las variables binarias permanecen como tal y, las numéricas o categóricas fueron escaladas adecuadamente (esto, revisando los percentiles para cada variable). Concluimos que se realizó adecuadamente.

Podríamos usar OneHot encoding, pero en este laboratorio optaremos solo por utilizar la codificación ordinal. Especialmente, porque las categorías tienen un sentido ordinal (i.e. son ordenadas) y porque tener más variables Dummy, dadas por el OneHot, puede hacer que se ocupe mucha memoria y que el modelo tarde más en construirse.

MODELOS:

Como ya tenemos listas los dos pipelines, podemos proceder a construir nuestros modelos. Para cada modelo, hacemos un modelo básico con los dos pipelines construidos y, posteriormente, hacemos una búsqueda exhaustiva por hiperparámetros usando la funcionalidad GridSearch de CrossValidation de sklearn.

Es importante mencionar que, dado el contexto del modelo, lo que nos interesa es saber si un paciente tiene diabetes o no. En este caso, queremos minimizar la cantidad de pacientes con diabetes que son clasificados como no diabéticos. De esta forma, no nos importa tener algunos falsos positivos (i.e. no diabéticos que son clasificados como diabéticos), porque es más importante que todos los diabéticos sean clasificados así. Es así como la métrica que elegimos para evaluar nuestros modelos es el recall.

Implementación y descripción de árboles de decisión

Este modelo fue realizado por los 3 integrantes del grupo, especialmente por Alvaro Plata.

Árboles de decisión es una técnica de clasificación que consiste en la toma de decisiones en forma de árbol en donde cada nodo representa una decisión, de

forma que cada camino nos acerca a la clasificación que se está buscando.

El resultado final del proceso de construcción del modelo es un mapa de los resultados de las posibles decisiones. Este tiene un nodo superior (nodo raíz) el cual aprende a particionar en función del valor del atributo. Su estructura es similar a la de un diagrama de flujo, donde cada nodo interno representa determinada condición o regla de decisión que ayudará a clasificar el nuevo dato.

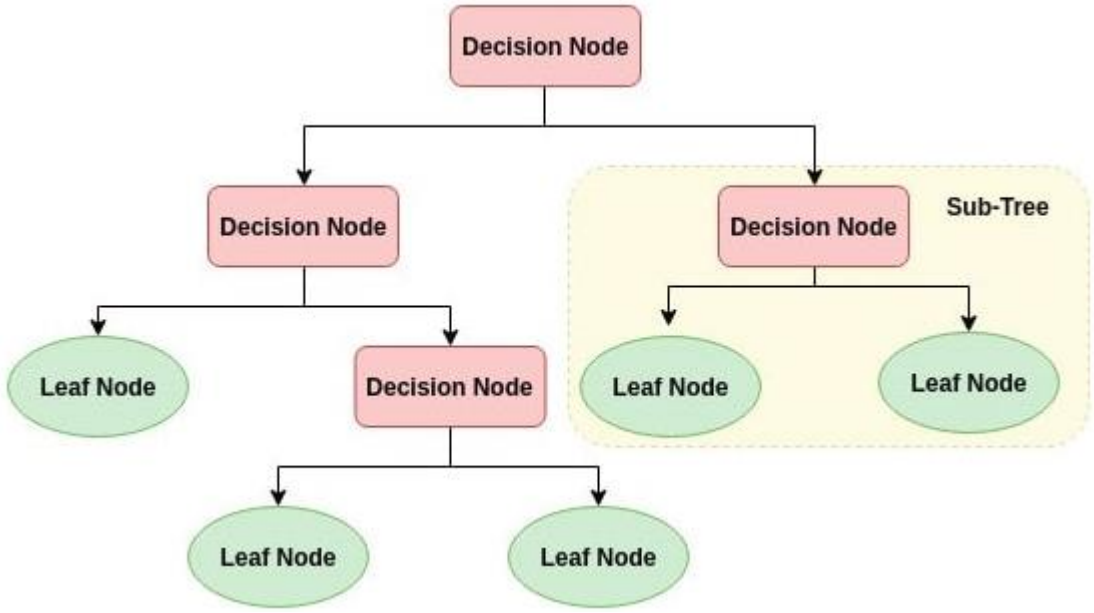


Imagen obtenida de: [Árbol de decisión en Machine Learning (Parte 1) - sitiobigdata.com]

Para realizar este modelo, se deben tener en cuenta hiperparámetros como:

- Criterio: La función para medir la calidad de una división. Los criterios admitidos son "gini" para la impureza de Gini y "entropía" para la ganancia de información.

- Profundidad máxima: La profundidad máxima del árbol. Si no se define, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos de *min_samples_split* muestras.

- Min samples split: El número mínimo de muestras requeridas para dividir un nodo interno:

Implementamos la técnica de validación cruzada para la búsqueda de los mejores valores para los hiperparámetros escogides, en donde nuestro espacio de búsqueda para cada uno de ellos fue:

- Criterio: gini y entropy
- Profundidad máxima: 3, 5, 10, 15 y 20
- Min samples split: 2, 3, 4, 5

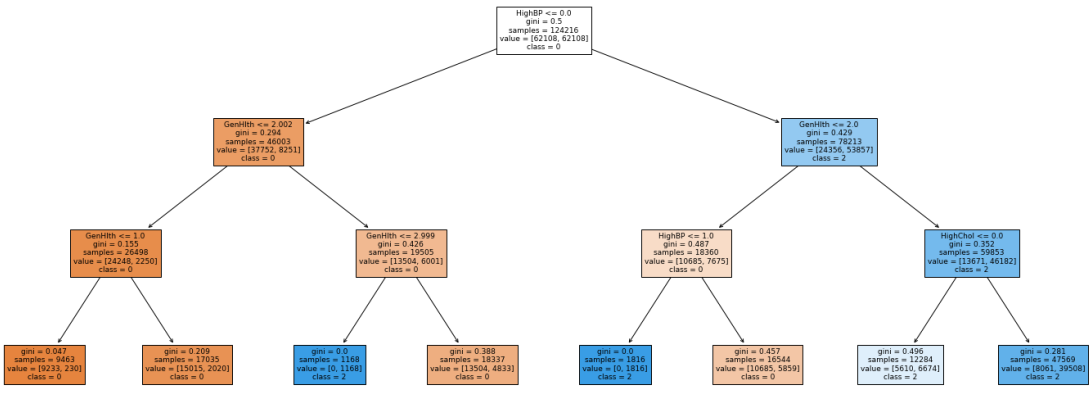
Como resultado de este proceso, encontramos que la mejor combinación de hiperparámetros fue:

- Criterio: gini
- Profundidad máxima: 3
- Min samples split: 2

Después de generar el modelo con los anteriores hiperparámetros, podemos analizar la importancia de cada variable en nuestro conjunto de datos para la toma de decisiones y clasificación. Este valor es asignado según la medida de pureza que genere la decisión con base en este atributo.

	Atributo	Importancia
0	HighBP	0.684955
1	GenHlth	0.247715
2	HighChol	0.067330
3	AnyHealthcare	0.000000
4	Education	0.000000
5	Age	0.000000
6	Sex	0.000000
7	DiffWalk	0.000000
8	PhysHlth	0.000000
9	MentHlth	0.000000
10	NoDocbcCost	0.000000
11	HvyAlcoholConsump	0.000000
12	Veggies	0.000000
13	Fruits	0.000000
14	PhysActivity	0.000000
15	HeartDiseaseorAttack	0.000000
16	Stroke	0.000000
17	Smoker	0.000000
18	BMI	0.000000
19	CholCheck	0.000000
20	Income	0.000000

Finalmente, podemos visualizar el árbol que representa el modelo construido.



En él, es posible observar cómo el valor de importancia de cada atributo concuerda con su altura en las ramas del árbol. Es decir, entre más importancia

tenga un atributo, más cerca estará de la raíz del árbol. Igualmente se puede observar cómo los atributos que tienen importancia 0 no se encuentran en el árbol y no son tomados en cuenta para tomar la decisión de clasificación de un dato.

Implementación y descripción de K-NN

Este modelo fue realizado por los 3 integrantes del grupo, especialmente por Brenda Barahona.

K vecinos cercanos (K-NN) funciona bajo el principio de funcionamiento, el cual es el de representar un objeto en un plano dimensional de tamaño n en donde cada una de las dimensiones es un atributo de ese objeto, es decir, cada dimensión es cada una de las columnas del conjunto de datos. Con esto planteado, al final tendremos una serie de puntos ubicados en este plano. El objetivo de este algoritmo es ayudar a la predicción, por lo que, cuando queramos saber si un nuevo objeto o en este caso un nuevo punto clasifica en una categoría, tomaríamos en consideración los k puntos más cercanos (clasificados con base a la distancia euclidiana o la distancia de Manhattan) y la categoría con más puntos de este conjunto sería la categoría para asignar al nuevo punto.

Para realizar este modelo, se deben tener en cuenta los hiperparámetros como:

- Los n vecinos: Este dice el número de vecinos a usar. Por default es 5.
- Peso: coloca un peso en función de la predicción. Los posibles valores que se pueden llegar a tomar son: "Uniforme", "Distancia"
- Métrica: es la métrica de distancia que va a usar el modelo.

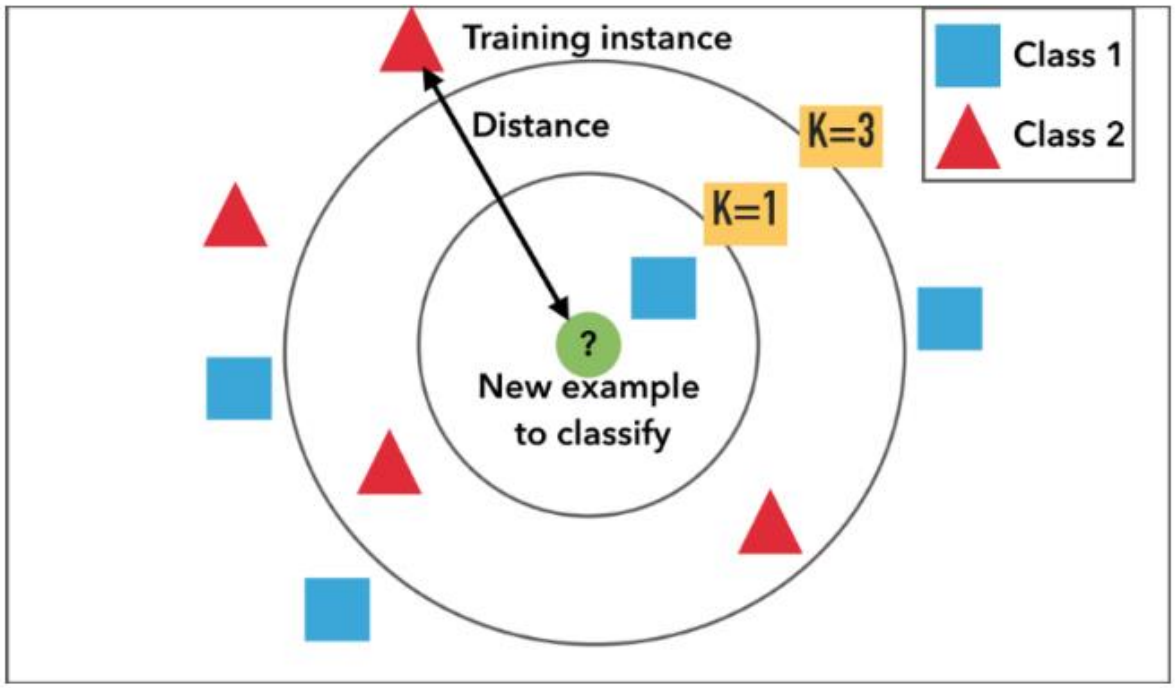


Imagen obtenida de: [k-NN — Getting to know your nearest neighbors | by Lee Schlenker | Towards Data Science]

Valores de los hiperparámetros escogidos:

- Número de vecinos: 7
- Métrica: Euclidiana
- Pesos: Uniforme
- Escala: Sin escalar

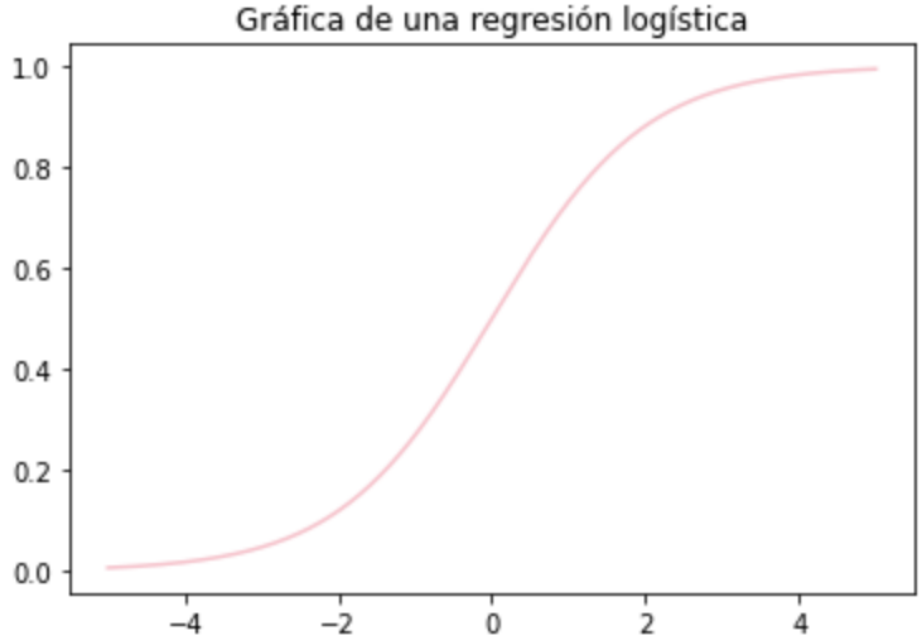
Implementación y descripción de regresión logística

Este modelo fue realizado por los 3 integrantes del grupo, especialmente por Sofía Alvarez.

Trata de un modelo utilizado generalmente en contextos de clasificación binaria. Como nuestro problema a resolver es predecir si una persona tiene diabetes o no, podemos utilizar este método. Este método usa la función logística o sigmoide, la cual está definida como

$$f(x) = \frac{1}{1 + e^{-x}}$$

Si se grafica esta función, se puede ver que retorna 1 en caso de que el dominio tienda a infinito positivo y a cero cuando tiende a infinito negativo



En estas regresiones logísticas hay 3 tipos, los cuales son: regresión logística binaria, multinomial y ordinal. En este caso nos basaremos en la regresión logística binaria, en la que usaremos dos categorías a decidir para una variable.

La regresión logística calcula una probabilidad, de acuerdo a la función sigmoide definida previamente. En el caso binario, si la probabilidad es mayor a 0.5 lo clasifica como la clase positiva (2=Diabetes, en este caso) y, si es menor, lo clasifica como de la clase negativa (0=No diabetes).

La regresión logística puede ser regularizada o no. Si es regularizada, se incluye un término que ayuda al modelo a que no haga sobreajuste (overfitting) de los datos. Construimos primero un modelo de regresión logística base, donde usamos todos los hiperparámetros por defecto de sklearn.

En estas regresiones logísticas hay 3 tipos, los cuales son: regresión logística binaria, multinomial y ordinal. En este caso nos basaremos en la regresión logística binaria, en la que usaremos dos categorías a decidir para una variable. Los hiperparámetros que hay que tener en cuenta en este método son:

- Fit intercept: Se indica si se quiere la presencia o no de la constante de regularización β_0.
- Penalty : representa la regularización. En esta hay varios tipos:
    - Regresión Lasso (L1): Su nombre es Least Absolute Shrinkage and Selection Operator. Hace que algunos valores de la constante de regularización sea 0.
    - Regresión Ridge (L2): Es la que viene por defecto, hace que los valores de la constante de regularización β_0 que acompañan a los coeficientes de las variables sean cercanos a 0.
    - Elasticnet: aplica L1 y L2 a la vez, fijando un decimal entre 0 y 1, donde 1 representa la regularización completa de L2
- Tol: Parámetro para que, luego de alcanzar cierto criterio de tolerancia, este deje de iterar.
- C: Parámetro que ayuda a reducir el overfiting. Mientras más pequeño este valor, mayor es la regularización.
- Class_weight: Dice que se va a balancear el conjunto de datos o no
- Multiclass: puede ser ovr, multinomial, auto. Esta será la estrategia que se usará para los problemas de clasificación. Por lo tanto, no lo usamos en este contexto.

Ahora, procedemos a construir el modelo por regresión logística. En este caso, usamos la librería saga porque, de acuerdo con la documentación de scikit learn, es la que mejor funciona para datasets grandes.

Sin hacer ajustes de hiperparámetros, el recall (que es la métrica seleccionada) es muy bajo. Por ello, hicimos ajuste de hiperparámetros.

En el caso de la regresión logística, los hiperparámetros principales que hay que ajustar están relacionados con aquellos que controlan la complejidad. De acuerdo a lo visto previamente, decidimos lo siguiente:

- Usar o no usar escalador: MinMaxScaler(), descrito previamente, o passthrough.
- Decidimos un espacio logaritmico entre 10^{-3} y 10^{3}, para probar valores cercanos y alejados de 0.
- Le damos al modelo la oportunidad de elegir si quiere o no hacer balanceo, basado en la frecuencia de los datos.
- Le damos al modelo la oportunidad de elegir el tipo de regresión que prefiera.

Con un modelo básico funcionando, procedemos a hacer un GridSearch junto con validación cruzada (n=3), de manera que una búsqueda más exhaustiva sobre los hiperparámetros nos provea un mejor modelo, y con la validación cruzada, resultados estadísticamente más significativos. Así las cosas, para la normalización elegimos dos opciones: no normalizar (passthrough), y escalar con el rango (MinMaxScaler). Para C (el parámetro inverso de la regularización), se hizo un barrido de diez valores en el espacio logarítmico entre 10^{-3} y 10^{3}. Entre mayor sea el valor de C, menor alta es la regularización del modelo. Asimismo, usamos dos tipos de penalización: L1 y L2. De estos dos tipos de penalización, que corresponden a Lasso y Ridge respectivamente, queremos ver cuál es más óptimo para nuestro modelo. Note que el solver saga soporta ambos tipos de penalización.

Después de correr los hiperparámetros, se obtuvo que:

De acuerdo con los hiperparámetros descritos previamente, en este caso tenemos que nuestra validación cruzada escogió los siguientes parámetros:

- C = 0.02: Como dijimos previamente, un valor de C más cercano a 0 indica que el modelo está más regularizado. En este caso, el modelo es muy regularizado.
- class_weight = None: El modelo prefirió no balancear los datos dependiendo de las frecuencias. Esto se debe seguramente al hecho de que se utilizó SMOTE.
- Penalización $L1$: Conviene utilizar una regresión Lasso, de acuerdo a lo explicado previamente, tal que se ponen en 0 algunos pesos del modelo, contrario a los parámetros configurados por defecto.
- Escalador MinMax(): Para este modelo, a diferencia de KNN, sí se prefirió tener los datos escalados, lo cual tiene sentido de acuerdo con lo que se espera de regresión logística.

Se obtuvo un recall del 75% para la clase positiva y del 72% para la negativa, mucho mejor que antes y bastante balanceado entre las dos clases.

```
----------Reporte para el Mejor Modelo Regresion Logistica----------
              precision    recall  f1-score   support

         0.0       0.94      0.72      0.82     62108
         2.0       0.32      0.75      0.45     10967

    accuracy                           0.72     73075
   macro avg       0.63      0.74      0.63     73075
weighted avg       0.85      0.72      0.76     73075
```

A diferencia del caso con KNN, vemos que el mejor modelo nos da un recall del 75% para la clase positiva, y del 72% para la clase negativa. La diferencia con KNN es que la métrica está mucho más balanceada para regresión logística (entre las dos clases 0 y 2). No obstante, 75% de recall es un muy buen valor. Sobre todo comparado con el de árboles, cuyo recall fue apenas de 60%. Esto se debe a que los árboles son, en realidad, modelos muy sencillos. Para que funcionen bien, es necesario utilizar modelos de ensamble como RandomForest que, en su versión básica, usa 100 árboles para tomar una decisión. Por las razones descritas previamente, lo más posible es que el mejor modelo sea regresión logística, porque es el que logra los mejores balances entre la clase 0 y 2 tal que seguramente no se está haciendo overfitting en los datos.

Análisis de resultados y modelo recomendado

Debido a la naturaleza del negocio, se tomó la decisión de que es más importante darle preferencia al porcentaje de personas con diabetes clasificados como positivos en nuestro modelo, que el porcentaje de personas que no tienen diabetes clasificadas como negativas. Esto nos indica que la métrica más importante para nuestros modelos es el Recall en la clase 2, por lo que ésta será la que utilizaremos para determinar cuál es el modelo más conveniente.

Los resultados de cada modelo para la métrica Recall en la clase 2 (positiva) en los datos de entrenamiento fueron los siguientes:

Modelo	Recall en datos de entrenamiento
KNN	0.98
Árbol de decisión	0.6
Regresión logística	0.75

A continuación, se detallan los resultados de cada modelo para la métrica Recall en la clase 2 (positiva)

Modelo	Recall en datos de prueba
KNN	0.69
Árbol de decisión	0.61
Regresión logística	0.76

Conclusión

Según estos resultados, podemos concluir que, a pesar de no haber sido el que tuvo el mayor valor de Recall en los datos de entrenamiento, el modelo más conveniente para este caso de uso es el de Regresión Logística, ya que fue el que presentó el mayor valor de Recall en datos de prueba.

Como recomendaciones a SaludAlpes sobre el manejo de sus datos, se menciona mantener una mayor calidad en sus datos. Es decir, realizar análisis periódicos sobre la consistencia, completitud y sentido de sus datos, de forma que sean útiles para los procesos de inteligencia de negocio y se disminuya el tiempo y recursos implementados en el proceso de limpieza y preparación de datos.