Modelo de priorización de usuarios para atenciones nutricionales en programas de primera infancia del ICBF en el Banco Magdalena

Juan José Amaris Vanegas

Julio Edien Roa Pinzón

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor: Javier Fernando Botía Valderrama, Doctor (Ph.D.) en Ingeniería Electrónica

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Posgrados en Ingeniería

Medellín, Colombia

2022

C ita	Amaris Vanegas y Roa Pinzón [1]
Referencia	[1] J. A. Amaris Vanegas y J.E. Roa Pinzón, “Modelo de priorización de usuarios para atenciones nutricionales en programas de primera infancia del ICBF en el Banco Magdalena” , Trabajo de grado especialización, Especialización en Analítica y Ciencia de Datos, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.
Estilo IEEE (2020)	

TABLA DE CONTENIDOS

I. RESUMEN EJECUTIVO

El Instituto Colombiano de Bienestar Familiar implementa diferentes programas en procura de defender los derechos de los niños en general en especial de aquellos que se encuentran en condiciones que impliquen algún tipo de vulnerabilidad. Entre los programas existentes que están siendo implementados actualmente por el ICBF se encuentran los que están enfocados en la Primera Infancia, es decir, para niños y niñas entre los 0 años y 6 meses de edad hasta los 5 años y 11 meses de edad.

En el municipio de El Banco, Magdalena funciona el Centro Zonal El Banco, dependiente de la Regional Magdalena del ICBF. Desde este centro zonal se supervisan los Operadores de Servicios de Primera Infancia, los cuales a través de licitaciones públicas ofertan para obtener la ejecución de los contratos de atención a usuarios de esta modalidad. Los contratos se ofertan por lo general con una duración de máximo un año, las atenciones van desde febrero hasta diciembre y por cada contrato se tiene un número fijo de cupos a atender. Para este caso de estudio, se está analizando usuarios provenientes de la Modalidad Institucional Centro de Desarrollo Infantil y Modalidad Desarrollo Infantil en Medio Familiar. Al iniciar cada año de atención se requiere vincular la totalidad de cupos a fin de garantizar una atención plena, y durante el curso del periodo de atención, mensualmente se realizan retiros (Cuando el usuario lo solicita por traslado de ciudad, barrio o tránsito a colegio) y posterior ingreso de usuarios.

Una de las finalidades del programa es dar atención a los usuarios cuando presenten condiciones nutricionales. Para saber si un niño o niña cuenta con alguna condición de este tipo existen relaciones como Relación Talla-Peso, Edad-Talla y Edad-Peso, a diferencia de los adultos donde basta con hacer el cálculo el Índice de Masa Corporal para saber si existe alguna condición anormal. Cuando se trata de usuarios nuevos, por medio del Registro Civil se puede conocer la edad, pero hasta no dar la primera atención no se puede conocer su talla y su peso. Una condición nutricional anormal que no se atiende a tiempo y forma puede ocasionar daños irreparables a nivel de desarrollo cognitivo y físico en los niños. Por el contexto de la atención puede pasar bastante tiempo hasta que un usuario sea valorado por primera vez por los apoyos en Salud, tiempo valioso que luego puede ser irrecuperable [1].

Por medio del modelo se realiza la tarea de clasificación para priorizar los usuarios inscritos en el ICBF en el programa de primera infancia de la Regional Magdalena, zona el Banco buscando establecer una clasificación que priorice los usuarios que ingresen al programa de primera infancia. Teniendo en cuenta factores como: Sisben, zona en que residan, Edad, Régimen de Salud, EPS, Discapacidad, Nacionalidad, Grupo Étnico, curvas de desarrollo y crecimiento, entre otros.

Como resultado se obtuvo una clasificación en la que se priorizan los usuarios más propensos a presentar alguna condición nutricional y que estos reciban una valoración más pronta por parte de enfermeros y nutricionistas, ganándole un tiempo muy importante a su condición nutricional y evitar problemas a largo plazo de salud en los usuarios, que finalmente se convierten en costos y gastos de atención tanto para el gobierno como para la persona que pueda ver afectada su salud por malnutrición a temprana edad.

Los datos que se tienen para trabajar en esta monografía se exportaron directamente desde el Portal Cuéntame (https://rubonline.icbf.gov.co/) del Instituto Colombiano de Bienestar Familiar. Para obtener acceso a esta información, se estableció un acuerdo con un Operador de un Programa de Primera Infancia, respetando la **Política de Tratamiento de Datos del ICBF** [2]. El sistema entrega una Hoja de Cálculo en Excel con la información organizada en filas y columnas. No está de más acotar que esta información es digitada en el portal Cuéntame por los Auxiliares administrativos y los apoyos en salud del operador de Servicio.

II. DESCRIPCIÓN DEL PROBLEMA

La desnutrición en Colombia es un tema de gran importancia y una problemática de salud pública que si no se detecta a tiempo y se corrige perjudica el buen funcionamiento del organismo, retrasa el desarrollo psicomotor, generando trastornos de crecimiento, motores y cognitivos, generando una menor inmunocompetencia y aumento en la mortalidad, lo que pretendemos con un modelo que estima la prioridad de atención a los niños que estén inscritos en programas del ICBF y que tengan alguna tendencia o mayor probabilidad de estar en estado de desnutrición es brindarle al ICBF una herramienta de clasificación que puedan utilizar para generar alarmas cuando se detecten niños en alto riesgo y se mejore su atención y por ende su estado de salud y desarrollo [7].

A. *Problema de negocio*

En la actualidad no existe en el instituto colombino de bienestar familiar ICBF Regional Magdalena, zona el Banco, una clasificación de los usuarios menores a 5 años, todos son atendidos por igual y con las mismas frecuencias sin importar su estado nutricional, estado socioeconómico y curvas de desarrollo, por lo cual un niño con desnutrición y una clasificación del Sisben baja es atendido de la misma manera que un niño que cuente con buen estado de salud y nivel socioeconómico estable

B. *Aproximación desde la analítica de datos*

Por medio del modelo clasificatorio se quiere establecer una clasificación de los usuarios que ingresen al programa de primera infancia del ICBF, Regional Magdalena, zona el Banco. Teniendo en cuenta factores como: Sisben, zona en que residan, Edad, Régimen de Salud, EPS, Discapacidad, Nacionalidad, Grupo Étnico, ubicación en las curvas de crecimiento y desarrollo según la edad.

Como resultado se obtendrá un listado en el que se priorizan los usuarios más propensos a presentar alguna condición nutricional desfavorable y que estos reciban una valoración más pronta por parte de enfermeros y nutricionistas, ganándole un tiempo muy importante a su desarrollo psicomotor y prevenir problemas a largo plazo de salud en los usuarios, que finalmente se convierten en costos y gastos de atención tanto para el gobierno como para la persona que pueda ver afectada su salud por malnutrición a temprana edad.

Inicialmente se realizará una exploración de datos para determinar las variables categóricas y numéricas, también se realizará un análisis de datos faltantes, detección de datos atípicos por medio de gráficas y de ser necesario se llevará a cabo un escalamiento de los datos que puedan tener un mayor peso para realizar el modelo. En este caso se tratará de un problema de clasificación utilizando métodos como regresión logística, árbol de decisión, bosques aleatorios, máquina de soporte vectorial y redes neuronales artificiales, se buscará obtener el mejor resultado. Para entrenar el algoritmo se utilizará una relación 70% para entrenamiento -30% para validación del conjunto de datos inicial. También se realizará un proceso de segmentación con las variables de interés obtenidas en el análisis inicial.

C. *Origen de los datos*

Los datos que se tienen para trabajar en esta monografía se exportaron directamente desde el Portal Cuéntame (https://rubonline.icbf.gov.co/) del Instituto Colombiano de Bienestar Familiar. Para obtener acceso a esta información, se estableció un acuerdo con un Operador de un Programa de Primera Infancia, respetando la **Política de Tratamiento de Datos del ICBF**.

D. *Métricas de desempeño*

Las métricas de desempeño varían según el tipo de aprendizaje que estemos afrontando. Cuando se trate de no supervisado se tendrán en cuenta métricas como el método del codo, y los scores de la Silueta, Calinski-Harabasz y Davies-Bouldin. Cuando se trate de aprendizaje supervisado se utilizarán las métricas de la librería Scikit-Learn cómo: precisión, precisión balanceada, F1 Score, Curva ROC [6].

En el caso de estudio que estamos enfrentando, se requiere una alta precisión en los modelos ya que se trata de la salud de niños entre los 0 y los 5 años. Por esto se buscan modelos que sean capaces de tener métricas superiores al 95%. Para esto se pondrán a disposición todos los recursos computacionales posibles además de la gama de complejidad de algoritmos existentes para hacer clasificación. La finalidad es obtener resultados que sean de utilidad en la práctica para el operador de servicio que amablemente suministró la información.

III. DATOS

A. *Datos originales*

El sistema entrega una Hoja de Cálculo en Excel con la información organizada en filas y columnas. No está de más acotar que esta información es digitada en el portal Cuéntame por los Auxiliares administrativos y los apoyos en salud del operador de Servicio.

Dentro de la base de datos se encuentran datos cuantitativos y cualitativos los cuales le brindan una versatilidad importante. Existen columnas con gran mayoría de información en blanco y otras con información irrelevante, las cuales junto a los datos que restringió el operador serán omitidos durante el proceso de limpieza y preparación de datos.

Entre las características más importantes incluidas en la base de datos se encuentran:

- *Sisben:* El Sisben actualmente establece una clasificación alfanumérica, las letras van desde la A (Extrema Pobreza) a la D y dentro de cada escalafón hay 51 escalafones donde se mide el nivel de pobreza.
- *Edad:* Se encuentra en un rango entre 6 meses y 5 años y 11 meses.
- *Régimen de Salud:* Régimen subsidiado, Régimen contributivo, Régimen Especial
- *EPS:* Mutual Ser, Nueva EPS, Coosalud, Magisterio.
- *Nacionalidad:* Colombiana, Venezolana
- *Grupo étnico:* Ninguno, Indígena, afrocolombiano, raizal
- *Zona de residencia:* Urbana, rural

- *Barrio*

La base de datos inicial cuenta con un total de 3561 filas y 66 columnas, con los siguientes tipos de datos y tamaño de archivo:

dtypes: datetime64[ns](1), float64(14), int64(9), object(43)
memory usage: 1.8+ MB

Como parte del acuerdo, se definió que la base de datos elaborada a partir de esta hoja de cálculo **NO contendría de ninguna forma datos como: Nombres, apellidos y número de documento de los usuarios, nombres, apellidos y número de documento de los acudientes, foto del usuario o acudiente y ningún tipo de dato que pueda exponer o relacionar de una u otra manera a alguno de los usuarios inscritos al programa.**

B. *Base de datos*

Una vez tenemos el archivo para trabajar empezamos con una exploración inicial de la data, esto para ver qué tipo de datos tenemos, el tamaño y la calidad para empezar a diseñar la manera más adecuada de tratar esta data, el tipo de datos de la data lo relacionamos a continuación.

TABLA I
TIPOS DE DATOS UTILIZADOS

Dtype	Describe cómo se deben interpretar los bytes en el bloque de memoria
object	Se refiere a las variables de tipo categórico, por lo general son cadenas de texto o *strings.*
Int 64	Hace referencia a un número entero y al espacio de memoria que este puede tener.
datetime64[ns]	Es una clase que permite operar de manera aritmética entre fechas, el 64 hace referencia al espacio de memoria asignado
float64	Hace referencia a la cantidad de memoria asignada para representar un número real, es decir con decimales

Una vez terminada la exploración, realizamos un filtrado a los datos eliminando los datos que no podemos mostrar por políticas de tratamientos de datos y aquellos que por su naturaleza o cantidad no nos aportan ningún tipo de información que sea concluyente.

```
dataf =
data.drop(data.columns[[0,1,2,3,4,5,6,8,10,11,12,13,14,17,18,19,20,23,27,28,29,30,31,32,33,34,35,36,37,38,39
,40,43,44,45,46,47,48,49,50,51,56,59,63,65]], axis='columns')
```

Aquí seleccionamos las columnas con las que vamos a crear la base de datos de entrenamiento.

C. *Descriptiva*

Después de tener la base de datos filtrada empezamos a codificar las variables que así lo requieren para terminar de crear la base de datos de entrenamiento, codificamos la variable tipo sisben para poder ponderar y clasificar la población de la base de datos original

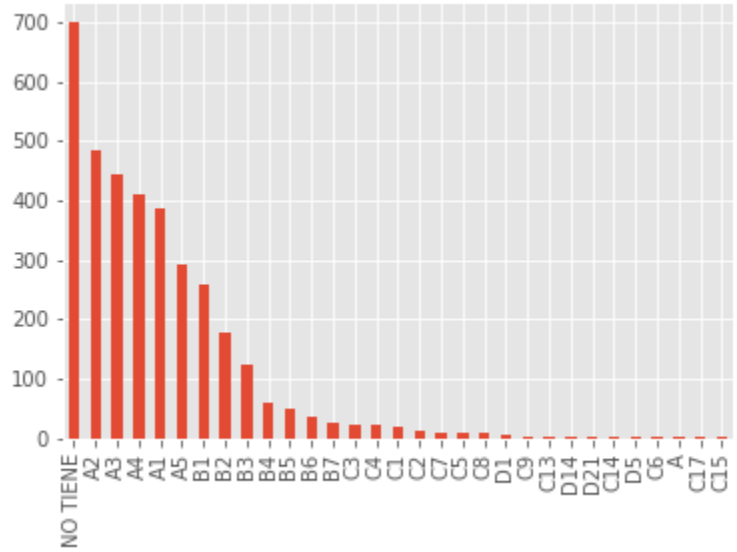


Fig. 1. Codificación de la variable Sisben.

Nota: *fuente autores*.

	Edad en Meses	Meses Lactancia Materna Exclusiva	Meses Lactancia Materna Total	Peso	Talla	Sisben	Nombre del Servicio_DESARROLLO INFANTIL EN MEDIO FAMILIAR SIN ARRIENDO - FAMILIAR	Nombre del Servicio_HOGARES INFANTILES - INSTITUCIONAL	Tipo Documento Beneficiario_SIN DOCUMENTO	Sexo_Mujer	...	Estado IMC_sobrepeso
0	12	7	7	10.0	73.0	4.0	1	0	0	1	...	0
1	1	1	1	6.2	58.0	1.0	1	0	1	0	...	0
2	58	6	18	14.3	99.6	1.0	0	0	1	0	...	0
3	61	6	18	15.5	102.1	1.0	0	0	1	0	...	0
4	0	0	0	3.8	50.0	1.0	1	0	1	1	...	0

5 rows × 46 columns

Fig. 2. Descripción de algunas variables de la base de datos
Nota: *fuente autores.*

IV. PROCESO DE ANALÍTICA

A. *Pipeline principal*

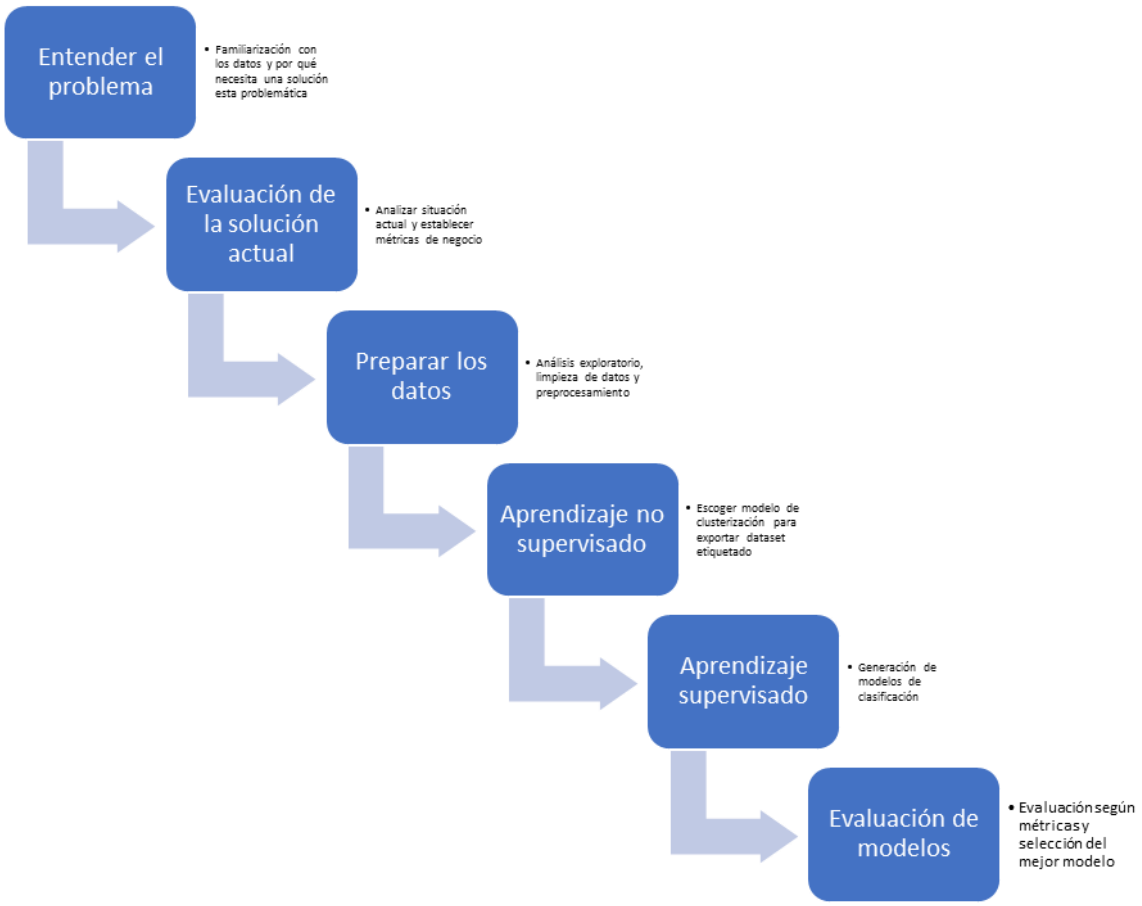


Fig. 3. Flujo grama de trabajo
Nota: *fuente autores*

B. *Preprocesamiento*

1) *Preprocesamiento en aprendizaje no supervisado:*

```
data = pd.read_excel("ICBFCUEGeneralPorToma.xlsx", na_values=0) # Read data
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3561 entries, 0 to 3560
Data columns (total 67 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Numero Contrato                                       3561 non-null   int64
 1   Regional                                              3561 non-null   object
 2   Centro Zonal                                          3561 non-null   object
 3   Municipio                                             3561 non-null   object
 4   Nombre Entidad Contratista                            3561 non-null   object
 5   Codigo Unidad                                         3561 non-null   int64
 6   Nombre Unidad                                         3561 non-null   object
 7   Nombre del Servicio                                   3561 non-null   object
 8   Tipo Beneficiario                                     3561 non-null   object
 9   Tipo Documento Beneficiario                           3561 non-null   object
 10  Numero Documento Beneficiario                         3561 non-null   object
 11  Primer Apellido Beneficiario                          3561 non-null   object
 12  Segundo Apellido Beneficiario                         3554 non-null   object
 13   Primer Nombre Beneficiario                           3561 non-null   object
 14  Segundo Nombre Beneficiario                           2783 non-null   object
 15  Sexo                                                  3561 non-null   object
 16  País de Nacimiento                                    3561 non-null   object
 17  Fecha Nacimiento                                      3561 non-null   object
 18  Fecha Ingreso al programa                             3561 non-null   object
 19  Fecha Valoración Nuricional                           3561 non-null   object
 20  Municipio Residencia                                  3561 non-null   object
 21  Edad en Meses                                         3561 non-null   int64
 22  Grupo Edad                                            3561 non-null   object
 23  Hijo de mujer gestante atendida en la modalidad       0 non-null      float64
 24  Meses Lactancia Materna Exclusiva                     3561 non-null   int64
 25  Meses Lactancia Materna Total                         3561 non-null   int64
 26  Grupo Étnico                                          3561 non-null   object
 27  Ubicación Pertenecia                                  2 non-null      object
 28  Discapacidad                                          3561 non-null   object
 29  Diagnóstico Discapacidad                              2 non-null      object
 30  Grupo en el Cual se encuentra (Páralisis Cerebral)    0 non-null      float64
```

Fig. 4. Cargue de la base de datos

Nota: fuente autores.

```
dDum= dataf.copy()
dDum =pd.get_dummies(dDum, drop_first=1)
```

Por medio de la aplicación del método dummy para variables categóricas se consigue expresar las variables categóricas en variables numéricas desglosadas por cada atributo de la categoría. De esta forma se pueden expresar como variables numéricas de tipo discreto, cuyo valor será 0 o 1.

2) *Preprocesamiento en aprendizaje supervisado:*

```
data = pd.read_csv("Dataset_With_Classes.csv", sep=';')
display(data)
```

	Edad en Meses	Meses Lactancia Materna Exclusiva	Meses Lactancia Materna Total	Peso	Talla	Sisben	Nombre del Servicio_DESARROLLO INFANTIL EN MEDIO FAMILIAR SIN ARRIENDO - FAMILIAR	Nombre del Servicio_HOGARES INFANTILES - INSTITUCIONAL	Tipo Documento Beneficiario_SIN DOCUMENTO	Sexo_Mujer	...	Estado IMC_sobrepeso	¿En el sistema general de seguridad social en salud usted es?_AFILIADO REGIMEN ESPECIAL	¿En el sistema general de seguridad social en salud usted es?_AFILIADO REGIMEN SUBSIDIADO	¿En el sistema general de seguridad social en salud usted es?_NO AFILIADO(A)
0	12	7	7	10.0	73.0	4.0	1	0	0	1	...	0	0	1	0
1	1	1	1	6.2	58.0	1.0	1	0	1	0	...	0	0	0	1
2	58	6	18	14.3	99.6	1.0	0	0	1	0	...	0	0	0	1
3	61	6	18	15.5	102.1	1.0	0	0	1	0	...	0	0	0	1
4	0	0	0	3.8	50.0	1.0	1	0	1	1	...	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3556	52	6	18	13.6	100.8	1.0	0	0	1	0	...	0	0	0	1
3557	56	6	18	14.6	102.5	1.0	0	0	1	0	...	0	0	0	1
3558	13	8	8	10.0	72.0	1.0	1	0	1	0	...	0	0	0	1
3559	53	6	14	15.0	101.0	1.0	0	1	1	1	...	0	0	0	1
3560	57	6	14	15.4	105.0	1.0	0	1	1	1	...	0	0	0	1

3561 rows × 46 columns

Fig. 5. Cargue de la base de datos después de aplicar dummies
Nota: fuente autores.

Para el cargue de la base de datos obtenido como salida del proceso de aprendizaje no supervisado se utilizó el método *pandas.read_csv.*

```
count_Clases=data.Clases.value_counts()
count_Clases
```

```
0    2210
1    1351
Name: Clases, dtype: int64
```

Posteriormente, se estableció la cantidad de datos que se tenían por cada una de las dos clases, para así determinar si sería necesario utilizar librerías o métodos especializados en tratamiento de bases de datos desbalanceadas. Para esto se utilizó el método *value.counts()* para realizar el conteo de las clases. Teniendo en cuenta los resultados obtenidos, no se considera necesario utilizar métodos para el balanceo de la base de datos como *oversampling* o *subsampling*. Si se tendrá en cuenta el parámetro *class weight* a la hora de crear los modelos para entrenamiento.

3) *Escalamiento de los datos:*

```
MM = MinMaxScaler(feature_range=(0, 1))
data_nuevo_MM = MM.fit_transform(data_nuevo)
data_nuevo_MM = pd.DataFrame(data_nuevo_MM, columns = ['Edad en Meses', 'Meses Lactancia Materna Exclusiva',
       'Meses Lactancia Materna Total', 'Peso', 'Talla', 'Sisben',
       'Nombre del Servicio_DESARROLLO INFANTIL EN MEDIO FAMILIAR SIN ARRIENDO - FAMILIAR',
       'Nombre del Servicio_HOGARES INFANTILES - INSTITUCIONAL',
       'Tipo Documento Beneficiario_SIN DOCUMENTO', 'Sexo_Mujer',
       'País de Nacimiento_Venezuela', 'Grupo Edad_Grupo2',
       'Grupo Edad_Grupo3', 'Grupo Étnico_INDIGENA',
       'Grupo Étnico_NO SE AUTORECONOCE EN NINGUNO DE LOS ANTERIORES',
       'Estado Talla Edad_Retraso en talla',
       'Estado Talla Edad_Riesgo de baja talla',
       'Estado Talla Edad_Talla adecuada para la edad',
       'Estado Peso Edad_Desnutricion global severa',
       'Estado Peso Edad_Grande para la edad Gestacional',
       'Estado Peso Edad_No Aplica',
       'Estado Peso Edad_Peso adecuado para la edad',
       'Estado Peso Edad_Riesgo de peso bajo para la edad',
       'Estado Peso Talla_Desnutricion aguda severa',
       'Estado Peso Talla_Obesidad',
       'Estado Peso Talla_Peso adecuado para la talla',
       'Estado Peso Talla_Riesgo de Sobrepeso',
       'Estado Peso Talla_Riesgo de desnutrición aguda',
       'Estado Peso Talla_Sobrepeso', 'Estado Peso Talla__',
       'Estado IMC_Delgadez', 'Estado IMC_Obesidad',
       'Estado IMC_Riesgo de Sobrepeso', 'Estado IMC_Riesgo para la delgadez',
       'Estado IMC_Sobrepeso', 'Estado IMC__', 'Estado IMC_sobrepeso',
       '¿En el sistema general de seguridad social en salud usted es?_AFILIADO REGIMEN ESPECIAL ',
       '¿En el sistema general de seguridad social en salud usted es?_AFILIADO REGIMEN SUBSIDIADO',
       '¿En el sistema general de seguridad social en salud usted es?_NO AFILIADO(A)',
       '¿El beneficiario cuenta con el carnet de vacunación?_S',
       '¿El carnet de vacunación se encuentra al día en las vacunas y dosis que corresponden a la edad del niño o niña?_SI',
       'El beneficiario cuenta con Carnet de Crecimiento y Desarrollo_SI',
       'Estado_VINCULADO', 'Talla inferior última toma_SI'])
display(data_nuevo_MM)
```

Fig. 6. Escalonamiento de los datos
Nota: *fuente autores*

Se utilizó un método de normalización (MinMaxScaler) de scikit-learn, con el cual se escalan las variables numéricas entre 0 y 1. Es el método de escalamiento más simple disponible en la librería mencionada.

4) *Separación en conjuntos de entrenamiento y validación:*

```
validation_size = 0.30
seed = 17
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, y, test_size=validation_size, random_state=seed)
```

Utilizando el método *train_test_split()* disponible en la librería de Scikit Learn, se realizó la separación de nuestra base de datos inicial en dos conjuntos. Al método en sus parámetros se le entrega las variables dependientes como **X** y la variable independiente como **Y**, además de esto se le asigna un tamaño de validación del 30%. Como resultado de esto se obtendrán dos conjuntos, uno llamado *train* que tiene el 70% de los datos elegidos aleatoriamente y el otro llamado *validation* que contiene los datos restantes. Con el primero se realizará el entrenamiento del algoritmo y con el segundo se realizará la validación de este a través de las métricas de negocio elegidas para este caso.

C. *Modelos*

TABLA II
MODELOS IMPLEMENTADOS

Aprendizaje no supervisado	Aprendizaje supervisado
KMeans	Regresión logística
Clustering Jerárquico	Arbol de decisión
PCA + KMeans	Random Forest
IPCA + KMeans	Máquina de Soporte Vectorial
TSNE + KMeans	MLP Classifier (Red Neuronal)

D. *Métricas*

1) *Métricas de aprendizaje no supervisado:*

Para realizar el proceso inicial de generación de etiquetas en nuestra base de datos se seleccionaron las siguientes métricas. Es importante utilizar la mayor cantidad de métricas que sean representativas y permitan hacer una correcta selección del modelo final. Como resultado de cada una de estas métricas se obtuvo un número de clusters recomendados que a su vez se traducirán en las clases que tendremos en el posterior paso de aprendizaje supervisado.

TABLA III
DESCRIPCIÓN DE MODELOS DE APRENDIZAJE NO SUPERVISADO IMPLEMENTADOS

Métrica	Función de Scikit-Learn	Descripción
Método del codo	modelo_kmeans.inertia_	Es un método más visual. Se busca el punto en la gráfica Número de clusters vs varianza intra cluster donde se evidencie un abrupto cambio en la pendiente [5].
Score de la Silueta	silhouette_score(X, cluster_labels)	Se calcula con el promedio de la distancia intra-cluster y el promedio de la distancia con el grupo más cercano. En esta métrica se calcula el máximo valor de la silueta que correspondía al número óptimo de clusters [5].
Score de Calinski-Harabasz	calinski_harabasz_score(X_scaled, cluster_labels)	Se define como el radio de la suma de la distancia inter-cluster sobre la distancia intra-cluster. El máximo valor de esta métrica permite obtener el número óptimo de clusters [5].

Score de Davies-Bouldin	davies_bouldin_score(X_scaled, cluster_labels)	Mide la similitud promedio entre los clusters, entendiendo la similitud como la distancia entre centros de los clusters. Para esta métrica se busca el puntaje más cercano a cero que indique mayor compactación de los grupos [5].

2) *Métricas de aprendizaje supervisado:*

TABLA IV
DESCRIPCIÓN DE MODELOS DE APRENDIZAJE SUPERVISADO IMPLEMENTADOS

Métrica	Función de Scikit-Learn	Descripción
Precisión	accuracy_score(Y_validation, predictions)	Como su nombre lo indica calcula la precisión del modelo. Compara la cantidad de etiquetas clasificadas por el modelo correctamente en relación a las etiquetas verdaderas. Es la métrica más básica del aprendizaje supervisado [5].
Precisión Balanceada	balanced_accuracy_score(Y_validation, predictions)	Similar a la precisión, pero esta vez tiene en cuenta el desbalanceo de clases existente en la base de datos. Gana mayor importancia cuando las clases se encuentran muy desbalanceadas [5].
F1 Score	f1_score(Y_validation, predictions, average='weighted')	Esta métrica mide el balance entre la precisión y la memorización del modelo, en otras palabras, se define como el promedio armónico entre la precisión y el recall. En esta métrica se prefieren los valores cercanos a 1 [5].
Área bajo la curva ROC	roc_auc_score(Y_validation, predictions)	Calcula el área bajo la curva característica operativa del receptor, comparando la cantidad de valores verdaderos positivos con respecto a la cantidad de valores de falsos positivos [5].

V. METODOLOGÍA

A. *Baseline*

Para la primera iteración tuvimos que realizar una limpieza inicial y un filtrado ya que la base de datos que nos fue suministrada no estaba etiquetada. Inicialmente realizamos un proceso de aprendizaje no supervisado donde aplicamos diferentes algoritmos y basado en las métricas de negocio definidas previamente, tomar una decisión de cual modelo escoger. Durante el análisis de las columnas contenidas en la base de datos, encontramos variedad en el tipo de datos, algunas eran numéricas, tanto discretas como continuas, de igual modo de tipo entero o real, además de esto columnas con fechas y otras en su mayoría categóricas, lo cual sugería a priori el uso de métodos de codificación de estas.

Investigando más a fondo, encontramos que algunas columnas en su totalidad o gran parte estaban compuestas por *missing values*, por datos redundantes, sin sentido o poco confiables. En el inicio del ejercicio se acudió con el proveedor de los datos para hallar la forma de minimizar esta incertidumbre, ya que de antemano se conocía de los posibles errores ocurridos en el origen y consolidación de esta información. En algunos casos fue posible mejorar la calidad de los datos y mantener en consideración estas variables, en otros casos, no fue posible y se trató de las primeras características que debimos prescindir para nuestro estudio. En el análisis previo, se definió como parte del ejercicio la importancia de contar con una variable que fuera confiable en su origen y la vez permitiera caracterizar el contexto socioeconómico de los usuarios, indagando el asunto, se encontró que con la información contenida en la base de datos y previa autorización del operador de servicio que la suministró, se procedió a generar la columna con esta información. Este proceso se realizó de manera manual, requirió una buena cantidad de tiempo para su conclusión, pero también se garantiza confiabilidad en el conjunto de datos.

Fig. 7. Clasificacion Sisben
Nota: fuente https://www.sisben.gov.co/Paginas/consulta-tu-grupo.aspx

La clasificación actual del Sisben fue establecida en 2021 y pasó de un valor numérico de tipo continuo a un escalafón alfanumérico como se aprecia en la ilustración anterior. El escalafón actual se compone de una letra que va de la A a la D, siendo la A el grupo de pobreza extrema, la B corresponde a pobreza moderada, la C a la población vulnerable y la D aquella población que no se encuentra en condición de pobreza y vulnerabilidad. A su vez cada grupo se subdivide según un número que varía según cada grupo inicial, en cada caso el número 1 corresponde al subgrupo de menor condición socioeconómica en relación con los demás subgrupos. Posteriormente se realizó un proceso de codificación de esta variable alfanumérica convirtiendo la en una variable numérica de tipo discreto, desde 0 que representa el subgrupo de los que no se conoce su estado en el Sisben hasta 52 que representa al subgrupo D21. Al realizarse este proceso en parte manual, es susceptible al error humano, por ejemplo, algunas casillas quedaron con la letra A que identifica a la extrema pobreza, pero no se conoce con exactitud en cual subgrupo se hallaba.

En cuanto a las variables con datos poco confiables como la de Discapacidad, que contaba con una variabilidad muy baja, ya que la mayoría absoluta de los datos pertenecían a un mismo tipo. Al finalizar este proceso, se pasó de un número de variables inicial de 66 a un reducido 16 pero de mayor valor concentrado para la posterior generación de modelos. Este condensado contaba con 5 variables numéricas, de las cuales 2 son de tipo real y 3 de tipo entero, por otra parte, las variables de tipo categórico son 11. Al contar con mayoría de variables categóricas, se hizo necesario el uso del método dummy de la librería pandas (dDum = pd.get_dummies(dDum, drop_first=1)) con el cual se desglosaba estas variables hasta obtener otras variables que describiera de manera más sencilla la información sin perder calidad. El siguiente paso, ya consistía en generar, entrenar y evaluar los modelos, para ello se definió utilizar los siguientes algoritmos de aprendizaje no supervisado:

- ***K-Means*** es un algoritmo clásico de agrupamiento basado en métricas de distancia que mide la relación entre los datos con los centros de los clusters. Se utiliza cuando tenemos una gran cantidad de datos sin etiquetar. Este algoritmo empieza definiendo aleatoriamente una cantidad de centros indicados en los parámetros del código y en cada iteración estos centros van migrando en procura de minimizar la distancia intra cluster y optimizar la distancia entre clusters, por medio de esto se garantiza la cohesión y similitud entre clases. El objetivo de este algoritmo es el de encontrar "K" grupos (clusters) entre los datos crudos [5].

- ***Agrupamiento jerárquico***, construye una jerarquía de clusters para realizar el análisis y existen dos categorías para este tipo de clusterización: aglomerante y divisivo. Para representar los resultados de la jerarquía de grupos se usa el dendograma que muestra las jerarquías de acuerdo con las distancias que existen entre los elementos del conjunto de datos, las cuales se pueden representar en una matriz de distancias [5].
- **PCA + KMeans,** Se conoce como el análisis de componentes principales. Por medio de este método de reducción dimensional, se logra condensar una gran cantidad de columnas en un número menor que conserva la representatividad del conjunto inicial. En nuestro caso de estudio, se definió reducirlo a dos variables *p1* y *p2* que lograban representar el 90% de los datos iniciales. Con esto se conseguía disminuir los recursos computacionales requeridos, reducir los tiempos de ejecución del algoritmo y potencializar las diferencias entre las variables para mejorar el proceso de agrupamiento a realizarse posteriormente con KMeans [5].

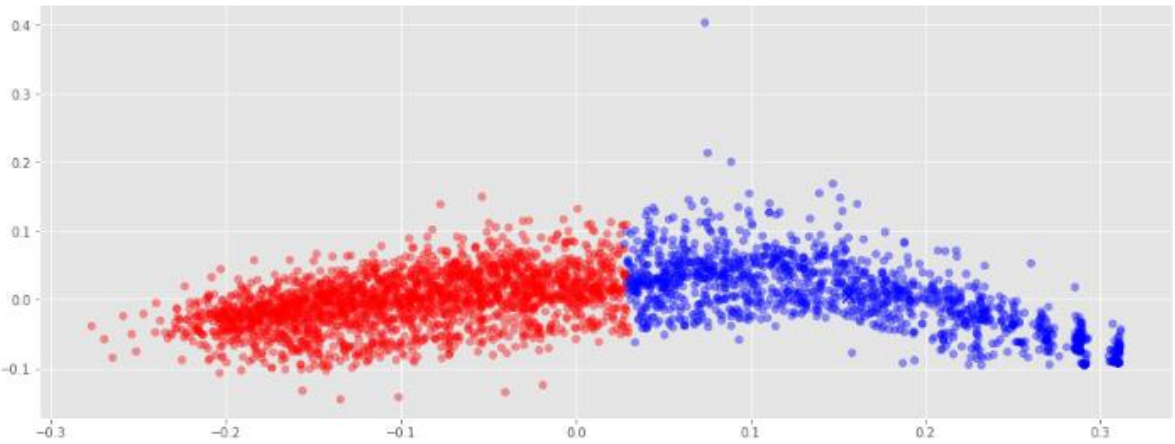


Fig. 8. Grafica de PCA+KMeans
Nota: *fuente autores*

- **IPCA + KMeans,** se utiliza como reemplazo del PCA cuando el número de datos es demasiado grande, este crea una aproximación de rango bajo para los datos de entrada utilizando una menor cantidad de memoria. Se puede decir que es una optimización del método PCA, que, si bien requiere de mayores recursos computacionales, por lo general permite retribuirlos en una mejor segmentación. Al igual que con el método PCA, se utilizó la salida como entrada de un proceso de segmentación con KMeans [5].

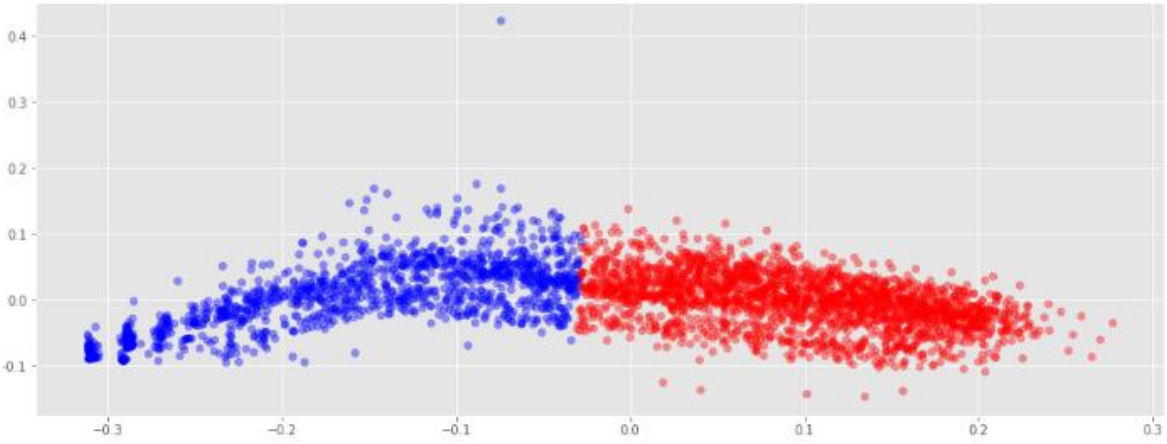


Fig. 9. Grafica de IPCA+KMeans
Nota: *fuente autores*

- **T-SNE + KMeans,** es una técnica de reducción de dimensionalidad utilizada para representar un conjunto de datos de alta dimensión en un espacio de baja dimensión de dos o tres dimensiones para que podamos visualizarlo. A diferencia de otros algoritmos de reducción de dimensionalidad como PCA, que simplemente maximiza la varianza, t-SNE crea un espacio de características reducido donde las muestras similares se calculan mediante una relación entre los vecinos más cercanos y las muestras diferentes que se modelan mediante puntos distantes con alta probabilidad, para determinar cuál es el mínimo error de la reducción de dimensionalidad el algoritmo tsne utiliza

una métrica de entropía llamada divergencia Kullback-Leibler el cual el algoritmo busca minimizar ese valor de entropía para garantizar la mejor representación de los datos reducidos. Este método se caracteriza por carácter estocástico, es decir que en cada ejecución podía presentar resultados diferentes. A diferencia de los métodos de PCA y PCA Incremental, requiere de un mayor tiempo de ejecución. La salida obtenida en este método de reducción de dimensionalidad, se utilizó posteriormente como entrada en una segmentación con K Means [5].

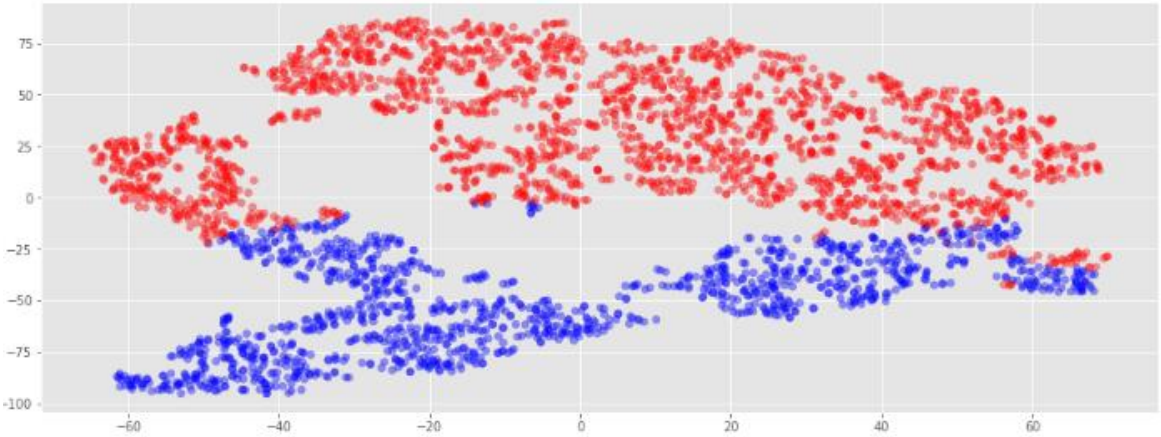


Fig. 10. Grafica T-SNE+KMeans
Nota: *fuente autores*

En este caso de estudio hablar de primera iteración es algo ambiguo, porque de antemano se definió utilizar cinco modelos distintos y que el Machine Learning hiciera su tarea. Teniendo etiquetada la base de datos a la salida del módulo de aprendizaje no supervisado de este proyecto, se comenzó con la tercera y última etapa que consistió en generar modelos de clasificación. Para esto también se utilizaron diferentes herramientas, desde la más simple de Regresión Logística hasta una Red Neuronal básica con el MLP Classifier de la librería Scikit-Learn. En el problema de negocio que enfrentamos al tratarse de la salud de seres humanos y en especial al tratarse de niños en primera infancia, más específicamente entre los 0 y los 5 años, se hizo crítico obtener un modelo con altos estándares en cuanto a las métricas, por ello se definió utilizar varias de ellas que permitieran evaluar los modelos desde distintas perspectivas. El procedimiento de aprendizaje no supervisado ayudó a mejorar estás métricas, ya que entregó una base de datos etiquetada de manera apropiada con dos clases bien cohesionadas entre sí y distinguibles una de la otra.

Los métodos de aprendizaje supervisado utilizados fueron los siguientes:

- Regresión Logística
- Arbol de Decisión
- Random Forest
- Máquina de Soporte Vectorial
- Red Neuronal Artificial [4]

B. *Validación*

Inicialmente el conjunto de datos se partió en dos, un conjunto de entrenamiento llamado *train* y otro conjunto de prueba del modelo llamado *validation.* Se definió la proporción 70-30 para estos conjuntos para disminuir el riesgo de presentar sobreajuste o subajuste. Al final del proyecto, se elevó la petición al Operador de Servicios para obtener datos nuevos que permitieran realizar una validación más exhaustiva de los modelos obtenidos. Por medio de esta petición se obtuvo una pequeña base de datos de 100 filas con datos correspondientes al año 2022 que sirvieron para poner a prueba los modelos finalistas y tomar la decisión final de con cual quedarse.

C. *Iteraciones y evolución*

1) *Aprendizaje no supervisado:*

Se comenzó por identificar y familiarizarse con la base de datos y luego pasar a un preprocesamiento donde se analizó a profundidad el mismo conjunto de datos, se eliminaron o trataron nulos, se codificaron variables categóricas y finalmente se normalizaron. Inicialmente se buscó etiquetar la base de datos por lo cual se implementaron 5 métodos de agrupamiento y basado en 3 métricas previamente seleccionadas tomar la decisión de cual método escoger [6]. Para conseguir este objetivo, anteriormente se había realizado un análisis exploratorio de los datos, limpiado la base de datos y realizado un preprocesamiento [3].

2) *Aprendizaje supervisado:*

En esta parte del proyecto se fueron implementando métodos de menor a mayor según la complejidad en el cómputo del modelo, de esta forma se inició con el clásico de modelo de regresión lineal y se llegó hasta una red neuronal básica utilizando MLP Classifier. En este proceso se pasó por un árbol de decisión, luego por uno de random forest y otro de máquina de soporte vectorial, siempre en procura de obtener las mejores métricas que a su vez estuviesen alineadas con los requerimientos de negocio [3].

Después de realizadas las 5 iteraciones, se preseleccionaron 2 modelos según las métricas obtenidas. Para tomar la decisión final se tuvo en cuenta la ejecución de estos dos modelos un nuevo conjunto de datos que fue obtenido.

D. *Herramientas*

Para el desarrollo de este caso de estudio se utilizaron las siguientes herramientas:

- Google Colaboratory: La mayoría de trabajo se realizó en este medio, las 3 etapas de la monografía aquí
- Microsoft Excel: Se utilizó para visualizar inicialmente los datos obtenidos, ya que este fue el medio en el cual lo entregó el operador de servicios.
- Microsoft Power BI: Se realizaron algunas visualizaciones utilizando esta herramienta

VI. RESULTADOS

A. *Metricas*

1) *Configuración de modelos de aprendizaje no supervisado:*

a) *KMeans:*

```
#Se define el algoritmo junto con el valor de K
k=2
algoritmo = KMeans(n_clusters = k, init = 'k-means++', max_iter = 300, n_init = 10)

#Se entrena el algoritmo
algoritmo.fit(X_scaled)
```

```
KMeans(n_clusters=2)
```

```
nro=[]
silueta=[]
calinski=[]
davies=[]
j=2
for n_clusters in range_n_clusters:
    modelo_kmeans = KMeans(
                        n_clusters   = n_clusters,
                        #n_init       = 20,
                        random_state = 10
                    )
    cluster_labels = modelo_kmeans.fit_predict(X_scaled)
    nro.append(j)
    silueta.append(silhouette_score(X_scaled, cluster_labels))
    calinski.append(calinski_harabasz_score(X_scaled, cluster_labels))
    davies.append(davies_bouldin_score(X_scaled, cluster_labels))
    j=j+1
```

Fig. 11. Ejemplo de algoritmo Kmeans
Nota: *fuente autores*

b) *Clustering jerárquico:*

```
modelo_hclust_ward = AgglomerativeClustering(
                             affinity = 'euclidean',
                             linkage  = 'ward',
                             distance_threshold = 0,
                             n_clusters         = None
                     )
modelo_hclust_ward.fit(X=X_scaled)
```

```
AgglomerativeClustering(distance_threshold=0, n_clusters=None)
```

```
def plot_dendrogram(model, **kwargs):
    '''
    Esta función extrae la información de un modelo AgglomerativeClustering
    y representa su dendograma con la función dendogram de scipy.cluster.hierarchy
    '''

    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot
    dendrogram(linkage_matrix, **kwargs)
```

Fig. 12. Ejemplo de Clustering jerárquico

Nota: *fuente autores.*

c) *PCA + KMeans:*

```
modelo_pca = PCA(n_components = 2)
pca = modelo_pca.fit_transform(X_scaled)
#Se aplicar la reducción de dimsensionalidad a los centroides
centroides_pca = modelo_pca.transform(centroides)
pca_data=np.vstack((pca.T, cluster, colores_cluster))
pca_df=pd.DataFrame(pca_data.T, columns=['P1', 'P2', 'cluster', 'color'])
pca_df.head(20)
```

	P1	P2	cluster	color
0	0.13075801672700002	0.002609404377063834	0	blue
1	0.2862715621219628	-0.07392143357710398	0	blue
2	-0.2277431679121862	-0.024860834191216367	1	red
3	-0.23778997919982614	-0.030525579943262292	1	red
4	0.3094318306024583	-0.09101466416655173	0	blue
5	-0.06720083039099217	0.016344685281983358	1	red
6	-0.03924130342487911	0.031161572328274753	1	red
7	-0.07777498622340419	0.011062649593339604	1	red
8	-0.11599713768804552	0.016516084723314565	1	red
9	-0.14582558895755646	0.00749603449486054	1	red
10	-0.03565571191978493	-0.012075523994832402	1	red

```
nro=[]
silueta=[]
calinski=[]
davies=[]
j=2
for n_clusters in range_n_clusters:
    modelo_pca = KMeans(
                        n_clusters   = n_clusters,
                        #n_init       = 20,
                        random_state = 10
                       )
    cluster_labels = modelo_pca.fit_predict(pca_data.T[:,0:2])
    nro.append(j)
    silueta.append(silhouette_score(pca_data.T[:,0:2], cluster_labels))
    calinski.append(calinski_harabasz_score(pca_data.T[:,0:2], cluster_labels))
    davies.append(davies_bouldin_score(pca_data.T[:,0:2], cluster_labels))
    j=j+1
```

Fig. 13. Ejemplo de PCA+KMeans

Nota: *fuente autores.*

d) *IPCA + KMeans:*

```
modelo_ipca = IncrementalPCA(n_components=2, batch_size=32)
ipca = modelo_ipca.fit_transform(X_scaled)
ipca_data=np.vstack((ipca.T, cluster, colores_cluster))
ipca_df=pd.DataFrame(ipca_data.T, columns=['P1', 'P2', 'cluster', 'color'])
ipca_df.head(20)
```

	P1	P2	cluster	color
0	-0.13079765431891371	0.002100619085880972	0	blue
1	-0.2862897816507166	-0.0715332237912037	0	blue
2	0.22768258897058744	-0.021101702987836927	1	red
3	0.2377350995523257	-0.0270496872557278	1	red
4	-0.30940946439983585	-0.08881670276371456	0	blue
5	0.06708006468312085	0.02191908842389305	1	red
6	0.03910503031136193	0.037238783355921276	1	red
7	0.07765930962462557	0.016410063001065695	1	red
8	0.11588260151954971	0.021894415645919877	1	red
9	0.14572358221445805	0.012528792384724216	1	red
10	0.035551986884498496	-0.007843198725121424	1	red

```
nro=[]
silueta=[]
calinski=[]
davies=[]
clases_ipca=[]
j=2
for n_clusters in range_n_clusters:
    modelo_ipca = KMeans(
                        n_clusters   = n_clusters,
                        #n_init       = 20,
                        random_state = 10
                    )
    cluster_labels = modelo_ipca.fit_predict(ipca_data.T[:,0:2])
    clases_ipca.append(cluster_labels)
    nro.append(j)
    silueta.append(silhouette_score(ipca_data.T[:,0:2], cluster_labels))
    calinski.append(calinski_harabasz_score(ipca_data.T[:,0:2], cluster_labels))
    davies.append(davies_bouldin_score(ipca_data.T[:,0:2], cluster_labels))
    j=j+1
```

Fig. 14. Ejemplo de algoritmo IPCA+KMeans
Nota: *fuente autores.*

e) *TSNE + KMeans:*

```
modelo_tsne = TSNE(n_components = 2, perplexity=30, method='exact', random_state=1234)
tsne = modelo_tsne.fit_transform(X_scaled)
tsne_data=np.vstack((tsne.T, cluster, colores_cluster))
tsne_df=pd.DataFrame(tsne_data.T, columns=['P1', 'P2', 'cluster', 'color'])
tsne_df.head(8)
```

	P1	P2	cluster	color
0	-23.331285	-36.303757	0	blue
1	-44.829697	-93.60273	0	blue
2	-27.59082	82.793465	1	red
3	-28.229973	82.02977	1	red
4	-54.891052	-93.18078	0	blue
5	52.15812	51.159023	1	red
6	57.006256	37.607502	1	red
7	51.893597	51.237995	1	red

```
nro=[]
silueta=[]
calinski=[]
davies=[]
clases_tsne=[]
j=2
for n_clusters in range_n_clusters:
    modelo_tsne = KMeans(
                         n_clusters   = n_clusters,
                         #n_init       = 20,
                         random_state = 10
                        )
    cluster_labels = modelo_tsne.fit_predict(tsne_data.T[:,0:2])
    clases_tsne.append(cluster_labels)
    nro.append(j)
    silueta.append(silhouette_score(tsne_data.T[:,0:2], cluster_labels))
    calinski.append(calinski_harabasz_score(tsne_data.T[:,0:2], cluster_labels))
    davies.append(davies_bouldin_score(tsne_data.T[:,0:2], cluster_labels))
    j=j+1
```

Fig. 15. Ejemplo de algoritmo T-SNE+Kmeans

Nota: *fuente autores.*

TABLA V

SCORE DE METRICAS UTILIZADAS

	# de Clusters	Score Silhouette	Score Calinski-Harabasz	Score Davies-Bouldin
0	2	0.491766	5042.164551	0.768223
1	3	0.402277	4706.577695	0.863599

2	4	0.340266	4298.638174	0.986444
3	5	0.290371	3737.570210	1.140388
4	6	0.298675	3440.543181	1.073030
5	7	0.252655	3178.464687	1.165482
6	8	0.241174	2945.193641	1.213691
7	9	0.249084	2779.073055	1.199140
8	10	0.232733	2632.310780	1.250760
9	11	0.234336	2510.171009	1.228064

En la tabla anterior tenemos los resultados obtenidos de la interacción realizada del algoritmo de K-means y los diferentes puntajes obtenidos con los diferentes cluster. Para nuestro caso trabajamos con un valor de K=2 ya que fue una de las mejores distribuciones de los datos alrededor de los centroides

- ***Score de Sillhouette:*** este score va de -1 a +1, en donde un valor elevado significa la cohesión de los grupos entre más cercano a 1 mucho mejor la métrica
- ***Score Calinski-Harabasz:*** un valor alto significa que las agrupaciones son densas y están bien separadas, se utiliza para hallar un pico de decisión.
- ***Score Davies-Bouldin:*** este mide la similitud promedio de cada grupo con su grupo más similar, los grupos más separados y menos dispersos son mejores, el mejor valor es el más cercano a cero.

2) *Configuración de modelos de aprendizaje supervisado:*

Terminado el proceso de etiquetado de los datos, se procedió a realizar los modelos de aprendizaje supervisado bajo las siguientes configuraciones:

a) *Regresión logística:*

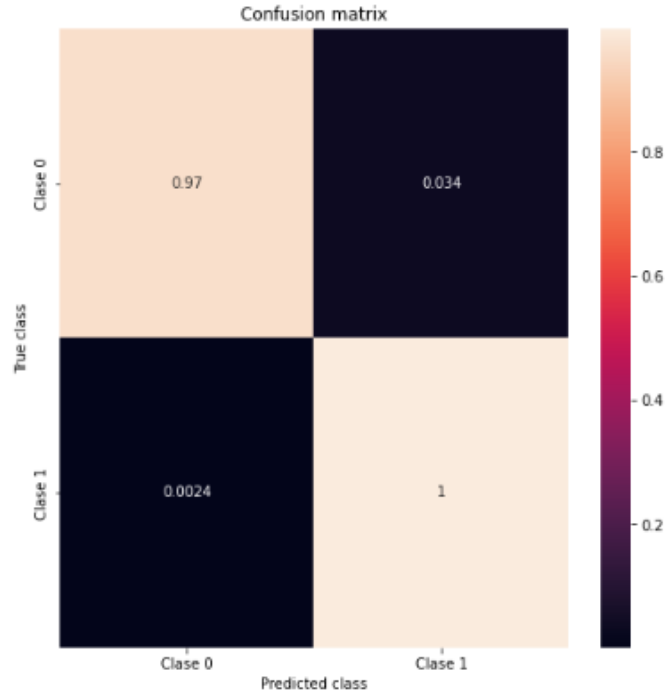
```
[8] X = np.array(data_nuevo_MM)
    y = np.array(data['Clases'])
    X.shape

    (3561, 45)
```

```
[9] validation_size = 0.30
    seed = 17
    X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, y, test_size=validation_size, random_state=seed)
```

```
[10] model = linear_model.LogisticRegression()
     model.fit(X_train,Y_train)

     LogisticRegression()
```

```
[11] predictions = model.predict(X_validation)
     print(predictions[0:15])

     [1 1 0 0 0 0 1 0 1 0 1 0 1 0 1]
```

```
[12] model.score(X_train,Y_train)

     0.9755216693418941
```

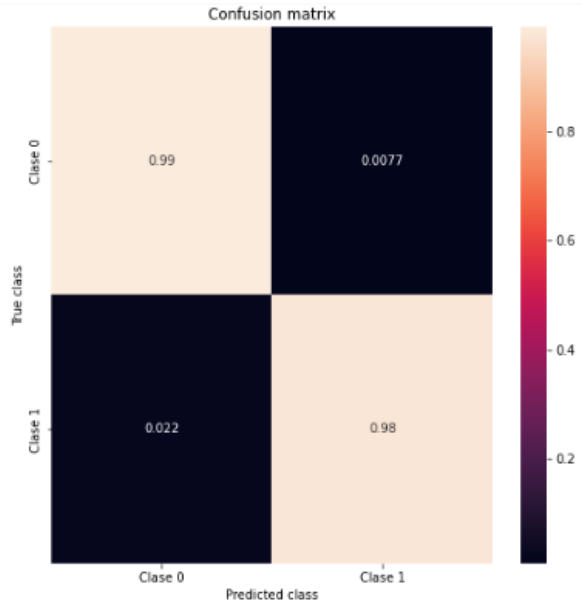


	precision	recall	f1-score	support
0	1.00	0.97	0.98	653
1	0.95	1.00	0.97	416
accuracy			0.98	1069
macro avg	0.97	0.98	0.98	1069
weighted avg	0.98	0.98	0.98	1069

Fig. 16. Matriz de confusión del modelo de regresión logística
Nota: *fuente autores.*

b) *Árbol de decisión:*

```
[21] cv = KFold(n_splits=10) # Numero deseado de "folds" que haremos
     accuracies = list()
     max_attributes = len(list(data))
     depth_range = range(1, max_attributes + 1)

     # Testearemos la profundidad de 1 a cantidad de atributos +1
     for depth in depth_range:
         fold_accuracy = []
         tree_model = tree.DecisionTreeClassifier(criterion='entropy',
                                                  min_samples_split=20,
                                                  min_samples_leaf=5,
                                                  max_depth = depth,
                                                  class_weight={1:1.6})
         for train_fold, valid_fold in cv.split(data):
             f_train = data.loc[train_fold]
             f_valid = data.loc[valid_fold]

             model_tree_decision = tree_model.fit(X = f_train.drop(['Clases'], axis=1),
                                    y = f_train["Clases"])
             valid_acc = model_tree_decision.score(X = f_valid.drop(['Clases'], axis=1),
                                     y = f_valid["Clases"]) # calculamos la precision con el segmento de validacion
             fold_accuracy.append(valid_acc)

         avg = sum(fold_accuracy)/len(fold_accuracy)
         accuracies.append(avg)

     # Mostramos los resultados obtenidos
     df = pd.DataFrame({"Max Depth": depth_range, "Average Accuracy": accuracies})
     df = df[["Max Depth", "Average Accuracy"]]
     print(df.to_string(index=False))
```

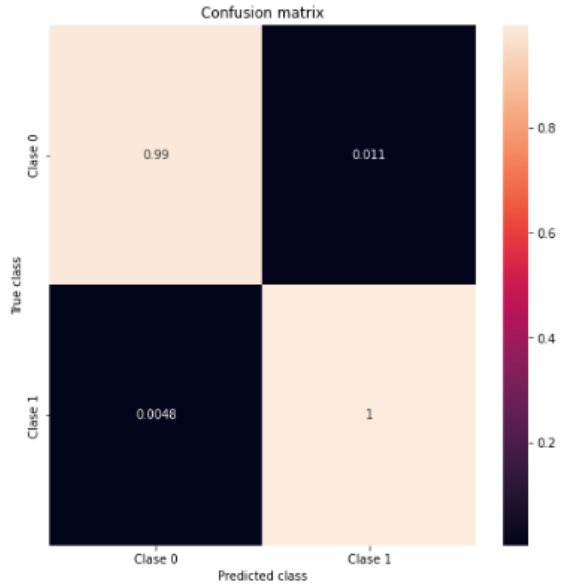


	precision	recall	f1-score	support
0	0.99	0.99	0.99	653
1	0.99	0.98	0.98	416
accuracy			0.99	1069
macro avg	0.99	0.99	0.99	1069
weighted avg	0.99	0.99	0.99	1069

Fig. 17. Matriz de confusión con el modelo de árbol de decisión
Nota: *fuente autores.*

c) *Random forest:*

```
# Crear el modelo con 100 arboles
model_random_forest = RandomForestClassifier(n_estimators=100,
                                             criterion='gini',
                                             bootstrap = True,
                                             verbose=2,
                                             max_features = 'sqrt',
                                             class_weight={1:1.6})
# entrenar!
model_random_forest.fit(X_train, Y_train)
```



	precision	recall	f1-score	support
0	1.00	0.99	0.99	653
1	0.98	1.00	0.99	416
accuracy			0.99	1069
macro avg	0.99	0.99	0.99	1069
weighted avg	0.99	0.99	0.99	1069

Fig. 18. Matriz de confusión con el modelo Random Forest
Nota: *fuente autores.*

d) *Máquina de soporte vectorial:*

```
import matplotlib
X=X_train
y=Y_train
# Creamos el modelo SVM para clasificación con kernel lineal/rbf y entrenamos el modelo
model_svm = svm.SVC(kernel='linear', C=100, class_weight={1:1.6},probability=True).fit(X, y)
```

```
# Veamos el efecto del truco kernel (aumentando la dimensionalidad del espacio de entradas)
from mpl_toolkits import mplot3d

# Aplicamos una operación de kernel gaussiano para separar las clases
# Gamma controla el efecto del kernel, si es muy pequeño el modelo se parece al lineal
gamma = 100
Xr = np.exp(-gamma*(X ** 2).sum(1))

# Graficamos el espacio de características mapeado por el kernel
ax = plt.subplot(projection='3d')
cmap   = matplotlib.colors.ListedColormap( [ 'k', 'g' ] )
ax.scatter3D(X[:, 0], X[:, 1], Xr, c=y, s=50, cmap=cmap)
```

```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x7f34ec1953d0>
```

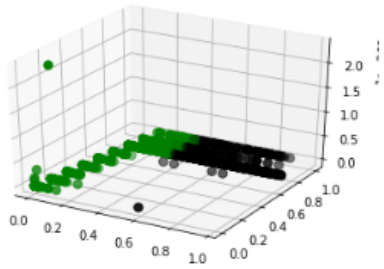
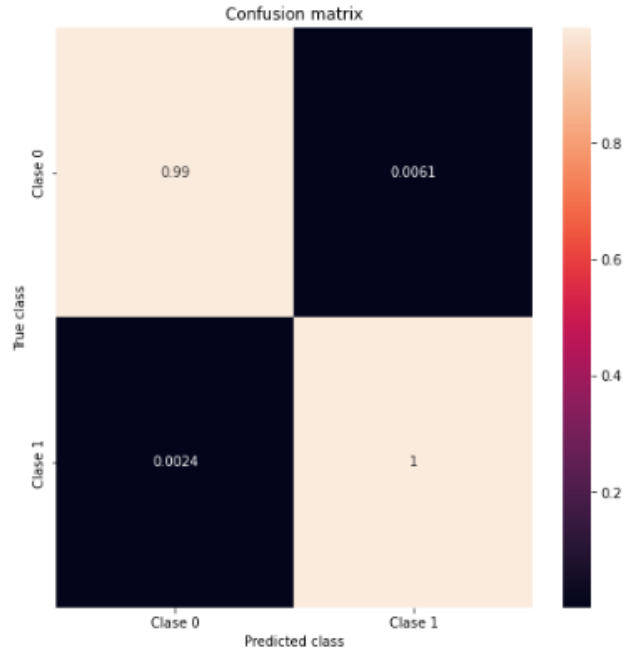


Fig. 19. Gráfica espacio de características Kernel.
Nota: *fuente autores.*



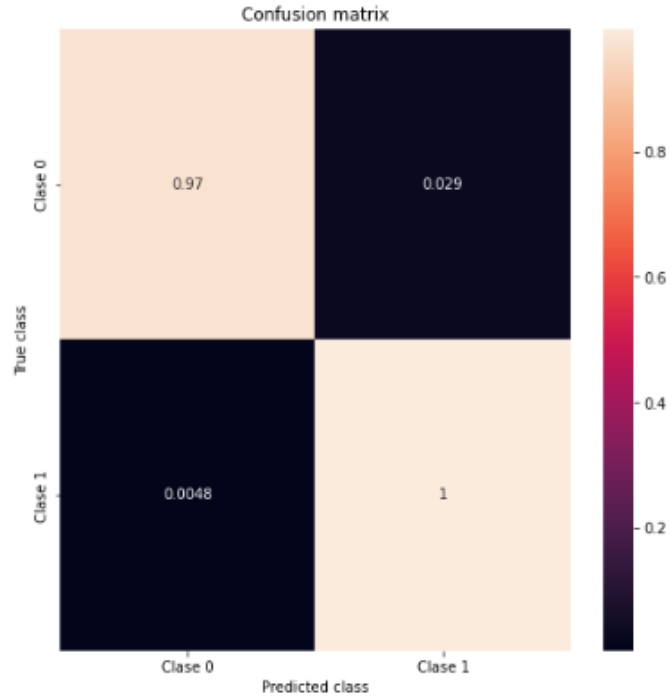
	precision	recall	f1-score	support
0	1.00	0.99	1.00	653
1	0.99	1.00	0.99	416
accuracy			1.00	1069
macro avg	0.99	1.00	1.00	1069
weighted avg	1.00	1.00	1.00	1069

Fig. 20. Matriz de confusión de máquina vectorial aplicada al modelo
Nota: *fuente autores.*

e) *MLP Classifier:*

```
model_mlp = MLPClassifier(hidden_layer_sizes=100,
                          activation='relu',
                          solver='sgd',
                          learning_rate='constant',
                          max_iter=1000)
model_mlp.fit(X_train, Y_train)
```

```
MLPClassifier(hidden_layer_sizes=100, max_iter=1000, solver='sgd')
```



	precision	recall	f1-score	support
0	1.00	0.97	0.98	653
1	0.96	1.00	0.98	416
accuracy			0.98	1069
macro avg	0.98	0.98	0.98	1069
weighted avg	0.98	0.98	0.98	1069

Fig. 21. Matriz de confusión con el modelo MLP Classifier
Nota: *fuente autores.*

Generados todos los modelos y evaluados según las métricas de negocio, se obtuvo la siguiente tabla de métricas:

TABLA VI
METRICAS

Métrica	Regresión Logistica	Arbol de Decisión	Random Forest	Maquina de Soporte Vectorial	Clasificador MLP
Precisión	0.978485	0.986904	0.991581	0.995323	0.980355
Precisión Balanceada	0.981953	0.985354	0.992236	0.995735	0.983048
F1 Score	0.978574	0.986892	0.991590	0.995326	0.980423
AUC Score	0.981953	0.985354	0.992236	0.995735	0.983048

En cada métrica se prefiere el valor más alto.

B. *Evaluación cualitativa*

Los resultados obtenidos con el modelo fueron bastante satisfactorios ya que se consiguió la meta inicial de obtener métricas superiores al 95%. Culminado el procedimiento de aprendizaje supervisado la decisión se redujo a dos modelos que se destacaron por obtener las mejores métricas. Por una parte, el de Random

Forest y por otra el de Máquina de Soporte Vectorial. En un principio se validaron con el conjunto de prueba y las métricas no sugerían la presencia de sobreajuste por lo cual, y para obtener una mayor seguridad en el caso de estudio, se procedió a solicitar datos nuevos al Operador de Servicios, ya que los datos iniciales correspondían a información de octubre de 2021. Después de un largo proceso con la entidad, se obtuvo la información que posteriormente debió organizarse en forma de base de datos con la misma cantidad de variables y organizados exactamente de la misma forma que la base de datos con que se generaron los modelos. Exactamente fueron 100 datos nuevos, con los cuales se pusieron a prueba los modelos, resultando el de Random Forest con evidentes signos de sobreajuste, ya que al guardar el modelo con la librería *joblib* y luego ejecutarlo con estos datos nuevos, el modelo solamente clasificaba los datos con la clase 0 que corresponde a los usuarios que no requieren atención priorizada.

```
[87] real_clases = Datos_Nuevos['Clases']
     del Datos_Nuevos['Clases']
```

```
[88] modelopredictivo = load('modelo_icbf.joblib')
     #Aqui se cargan los nuevos datos, EXACTAMENTE en el mismo orden y la misma magnitud
     class_predicted = modelopredictivo.predict(Datos_Nuevos)

     print("Clase = ", class_predicted)

     Clase =  [0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0
      0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
      0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0]
     /usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but SVC was fitted without feature names
       f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
[89] Puntaje = accuracy_score(real_clases, class_predicted)
     print(Puntaje)

     0.96
```

Después de esto se realizó el mismo ejercicio guardando y luego cargando el modelo obtenido con Máquina de soporte vectorial, se ejecutó con los datos nuevos y se obtuvo que de las 100 muestras, 96 fueron clasificadas correctamente, lo cual ratifica la condición satisfactoria de ejercicio realizado.

```
[90] Puntaje2 = balanced_accuracy_score(real_clases, class_predicted)
     print(Puntaje2)

     0.9259259259259259
```

```
[91] Puntaje3 = f1_score(real_clases, class_predicted, average='weighted')
     print(Puntaje3)

     0.9589333333333333
```

```
[92] Puntaje4 = roc_auc_score(real_clases, class_predicted)
     print(Puntaje4)

     0.9259259259259259
```

También se evaluó el modelo ejecutado con los datos nuevos utilizando las otras métricas de negocio, en todos los casos se obtuvo resultados muy prometedores.

C. *Consideraciones de producción*

Considerando el entorno en que se genera y utiliza la información utilizada en este caso de estudio, pensar en integración con servicios en la nube sería casi utópico, por lo general las empresas contratistas que se hacen cargo de estos contratos de atención a primera infancia, cuentas llanamente con el recurso suficiente para dar una atención decente a los niños. Dentro de los presupuestos que le asigna el Instituto Colombiano de Bienestar Familiar no se encuentran contemplados gastos referentes a analítica descriptiva o predictiva de datos, estos procesos a menudo se dan en las oficinas centrales del Instituto y no tiene la granularidad suficiente para llegar a poblaciones tan recónditas como El Banco, Magdalena. Viéndolo de este modo, resultaría interesante hacer la propuesta de contar con un pequeño presupuesto que permita actualizar con cierta periodicidad este modelo obtenido. Si se consiguiera poner en producción este modelo, se requiere articular otros procesos que garanticen la calidad de los resultados obtenidos. Como ya se mencionó anteriormente, existen variables numéricas que son susceptibles al error humano o a una mala calibración de los equipos con que se toman, es el caso del peso y la talla. Es importante capacitar al personal para realizar correctamente el proceso de pesaje y tallaje, ya que, si bien puede parecer algo rutinario, en la realidad se cometen errores simples que atrofian la medida correcta. A la hora de digitar los valores sobre todo en áreas alejadas donde no se tiene acceso al computador, es importante tener una caligrafía clara que no deje lugar al error cuando el digitalizador ingrese la información al sistema.

Estos contratos de servicio por lo general llevan todo el año, y constantemente están ingresando nuevos usuarios al programa, por lo cual el flujo de información es constante. En una puesta en producción sería fundamental alimentar constantemente el modelo y renovarlo mensualmente para mantener cimientos reales. Este caso de estudio es altamente susceptible a fenómenos sociales y culturales de la región por ejemplo la migración de extranjeros o las posibles inundaciones que con frecuencia se presentan en esta zona bañada por el Río Magdalena. Todas estas variaciones afectan las características socioeconómicas de la población y de igual manera la calidad de vida de los niños, aumentando enfermedades que pueden actuar como agravantes de condiciones nutricionales anormales.

Enlace de acceso a repositorio en GitHub con los notebooks y bases de datos del proyecto:
https://github.com/juanjoamaris/Monografia-UDEA

VII. CONCLUSIONES

- El proyecto mostró unos resultados prometedores, se consiguieron los objetivos planteados al principio y el potencial para implementarlo es bastante grande obviando el hecho de la falta de financiamiento.
- Se pudo generar un modelo confiable el cual fue probado con datos nuevos, que no guardan vinculación con los utilizados a la hora de entrenar y testear el modelo.
- El ejercicio fue satisfactorio, ya que mientras se perseguía el objetivo de generar un modelo útil, se pudo implementar la mayoría de las técnicas aprendidas durante la especialización, esto permitió fortalecer conocimientos para un desempeño futuro.
- En este tipo de ejercicios donde se trata la salud de niños menores de 5 años, es fundamental obtener métricas de negocio altas, si esto no está garantizado, difícilmente se consiga poner en producción el modelo generado.
- En un futuro proyecto similar, se recomienda conseguir una base de datos más grande, preferiblemente con más variables que caractericen condiciones socioeconómicas.

VIII. AGRADECIMIENTOS

En primer lugar, a Dios por la oportunidad de haber cursado esta especialización en la que adquirimos invaluables conocimientos referentes al mundo de la analítica descriptiva, predictiva y prescriptiva. Agradecer a nuestro docente asesor, el profesor Ph.D. Javier Fernando Botía Valderrama, por el tiempo dedicado y la paciencia para orientarnos en el proceso que llegó a la consecución de esta monografía, también una mención especial al Instituto Colombiano de Bienestar Familiar y el Operador de Servicios Corporación Comunitaria Nutrir por habernos suministrado la base de datos con que se realizó nuestra monografía. De igual manera expresar gratitud a nuestras familias por el apoyo brindado en el curso de esta especialización. Este resultado es el fruto de un profundo esmero por poner a prueba los conocimientos adquiridos de parte de nuestros docentes.

IX. REFERENCIAS

[1] A. Ortiz-Andrellucchi, L. Peña Quintana, A. Albino Beñacar y F. Mönckeberg Barros, "Desnutrición infantil, salud y pobreza: intervención desde un programa integral", *Nutricion Hospitalaria*, vol. 21, n.º 4, 2006.

[2] Instituto Colombiano de Bienestar Familiar. "Política De Tratamiento De Datos Personales". https://www.icbf.gov.co/sites/default/files/politica_de_tratamiento_de_datos_personales_0.pdf.

[3] Haykin, S. S., Haykin, S. S., Haykin, S. S., & Haykin, S. S. (2009). Neural networks and learning machines (Vol. 3). Upper Saddle River, NJ, USA:: Pearson.

[4] J. Alzubi, A. Nayyar, and A. Kumar, “Machine Learning from Theory to Algorithms: An Overview,” in Journal of Physics: Conference Series, Nov. 2018, vol. 1142, no. 1, pp. 1–2, doi: 10.1088/1742-6596/1142/1/012012

[5] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

[6] D. Carvalho, E. Pereira y J. Cardoso, "Machine Learning Interpretability: A Survey on Methods and Metrics", *Electronics*, vol. 8, n.º 8, p. 832, 2019.

[7] Universidad de La Sabana ¿Cómo está la desnutrición infantil en Colombia? [en linea] Disponible en: https://www.unisabana.edu.co/portaldenoticias/al-dia/como-esta-la-desnutricion-infantil-en-colombia/

X. GLOSARIO

- Malnutrición: Hace referencia al déficit en la ingesta calórica
- Operador de Servicio: Es la entidad que administra el contrato de primera infancia, fueron quienes suministraron la base de datos.
- Sisben: Es el mecanismo mediante el cual el Estado colobiano clasifica a las familias según su situación económica.