# Universidad Industrial de Santander

**Facultad de Ciencias**

**Tópicos en Estadística**

**Parte Practica Tercer Parcial**

Edwin José Ávila - 2212512

Celin Amparo Libreros - 2211369

Dania Stefanny Rivera - 2211973


## Introducción

Según el Ministerio de Salud y Protección Social (2021), los controles médicos periódicos durante la infancia son fundamentales para detectar enfermedades tempranas, garantizar el desarrollo integral de los niños y reducir desigualdades en salud a lo largo de la vida. Sin embargo, en contextos de vulnerabilidad, el acceso a estos servicios puede verse limitado por factores sociales, económicos y geográficos.

El presente informe tiene como propósito aplicar técnicas de análisis estadístico y aprendizaje automático para estudiar el acceso a controles médicos por parte de menores en Colombia, con base en los datos suministrados por el archivo `D. MENORES.txt` proveniente de la Gran Encuesta Integrada de Hogares (GEIH) del DANE. En particular, se busca predecir si un menor ha recibido control médico preventivo en el último año (variable `P6161`) y el número de citas recibidas (variable `P6161s1`), utilizando como predictores variables demográficas y sociales relevantes.

Para ello, se desarrollan modelos de clasificación y regresión empleando algoritmos como redes neuronales (MLP), máquinas de soporte vectorial (SVM), Random Forest y Naive Bayes, bajo el enfoque de interés práctico en la precisión de predicción y la comprensión de los factores asociados al acceso a servicios de salud. Se realiza limpieza, transformación y análisis de los datos, así como la evaluación comparativa del desempeño de cada modelo, con el fin de interpretar resultados que puedan orientar decisiones basadas en datos.

## Metodología

Este estudio se desarrolló mediante un enfoque secuencial que combinó técnicas de ciencia de datos y aprendizaje automático para analizar los determinantes de la falta de controles médicos en menores. Se utilizaron datos de la Encuesta Longitudinal de Protección Social (ELPS 2012), enfocándose en variables relacionadas con el cuidado infantil, como la ubicación del menor durante el día (`acu casa`, `acu trab`), hábitos alimenticios (`si desa`, `no alm`), factores socioeconómicos (`costoso`, `sin cupo`) e interacción parental (`no comp madre`, `si comp padre`). La variable objetivo fue la asistencia a controles de crecimiento y desarrollo (`P6161`), categorizada en "Sí" (2210 casos) y "No" (514 casos).

Durante el preprocesamiento se trataron los valores faltantes mediante imputación estratégica: por ejemplo, en las variables `P6159s8` y `P6163s8`, relacionadas con actividades maternas y paternas, los valores nulos se reemplazaron por un código neutro (2 = "No aplica") para evitar pérdida de información. Luego, se aplicó codificación one-hot (`pd.get_dummies`) a variables categóricas y se estandarizaron las escalas numéricas, conservando el desbalance natural de clases para reflejar adecuadamente la realidad observada.

Para el modelado predictivo se implementaron cuatro algoritmos supervisados: una red neuronal multicapa (MLP) con seis capas ocultas de 50 nodos y activación ReLU, un bosque aleatorio (Random Forest) con 100 estimadores, un clasificador Naive Bayes Gaussiano sin ajuste de hiperparámetros y una máquina de soporte vectorial (SVM) con kernel RBF. Los datos se dividieron en un 80 % para entrenamiento y 20 % para validación usando `train_test_split`, priorizando la detección de la clase minoritaria.

La evaluación de modelos se centró en exactitud y sensibilidad, complementadas con matrices de confusión. Aunque MLP y Random Forest alcanzaron una exactitud del 80 %, mostraron limitaciones al identificar casos críticos ("No"). En contraste, Naive Bayes, con 74 % de exactitud, presentó mayor sensibilidad frente a la clase minoritaria, siendo más útil desde un enfoque interpretativo.
Los resultados revelaron dos patrones relevantes: uno de negligencia extrema, donde los menores estaban solos en casa, sin alimentación adecuada ni interacción parental; y otro de carencia de acompañamiento, con necesidades básicas cubiertas pero sin actividades compartidas con los padres. A partir de estos hallazgos, se recomendaron acciones de política pública como ampliar cupos en guarderías, promover programas de acompañamiento parental y campañas de concientización sobre la importancia de controles médicos preventivos. El uso de librerías como `seaborn` para visualizar matrices de confusión y `scikit-learn` para validación iterativa aseguró la transparencia y replicabilidad del análisis. Este flujo metodológico permitió no solo predecir, sino también comprender los factores sociales que explican la ausencia de controles médicos en menores.

## Resultados y Conclusiones

```
from google.colab import files # Para cargar archivos en google.colab
uploaded = files.upload()
```

Upload widget is only available when the cell has been executed in the current browser session. Please return this cell to enable.
Saving D. MEMORES.txt to D. MEMORES (2).txt
La base de datos es una encuesta longitudinal de Protección Social-ELPS 2012. Cargamos la base de datos, le decimos a Python que es un archivo txt y lo transformamos en un archivo tipo pandas.

```
import pandas as pd
import io
datos_memores = pd.read_csv(io.BytesIO(uploaded['D. MEMORES (2).txt']), delimiter='\t')
datos_memores.head(4)
```

Cuadro 1: Primeras 4 filas de los datos

| Direccion | No.encuesta | Secuencia.encuesta | Secuencia.p | Orden | PS1 | PS2 | PS2s1 | PS3 | PS4 | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 447 | 247 | 4 | 1 | 4 | 4 | 5.0 | NaN | NaN | ... |
| 1 | 17355 | 247 | 3 | 1 | 3 | 1 | NaN | NaN | 1.0 | ... |
| 2 | 373 | 247 | 5 | 1 | 5 | 1 | NaN | NaN | 4.0 | ... |
| 3 | 373 | 247 | 4 | 1 | 4 | 1 | NaN | NaN | 4.0 | ... |

Vamos a determinar donde se encuentran los valores faltantes en nuestra base de datos.

```
datos_memores.PS2.isnull()
```

Tomamos los elementos cuya variable respuesta no sean nulos, ubicando filas y columnas. A continuación, seleccionamos la columna P52 y las filas donde esta variable no tiene valores.

```
datos_menores.loc[datos_menores.P52.notnull(), "P52"]
```

Cuadro 2: Valores no nulos de P52

| Índice | P52 |
|---|---|
| 0 | 5.0 |
| 4 | 5.0 |
| 5 | 5.0 |
| 6 | 5.0 |
| 7 | 4.0 |
| ⋮ | ⋮ |
| 4085 | 4.0 |
| 4086 | 4.0 |
| 4087 | 1.0 |
| 4089 | 5.0 |
| 4090 | 5.0 |

2724 rows × 1 columns

Vamos a repetir el procedimiento anterior, pero con columnas específicas. Recordemos que P52 denota la razón por la cual el menor no va a una guardería, y con el resultado, notamos que en efecto ya no hay valores faltantes en la variable P52, los cuales brindaban información sobre las razones de que los menores sí asisten a una guardería, pero esta información no es de nuestro interés respecto a la variable respuesta. Por otro lado, tenemos variables como P6163a8, que se denota como: el padre no realiza ninguna actividad con el menor y cuyos valores faltantes son relevantes, pero sin un valor cuantitativo.

```
tabla_menores = datos_menores[["P51", "P52", "P55", "P56", "P57",
                               "P6182a8", "P6161", "P6161a1"]]
tabla_menores = tabla_menores[tabla_menores.P52.notnull()]
tabla_menores1.head(4)
```

Cuadro 3: Tabla de menores (primeras 4 filas)

| Índice | P51 | P52 | P55 | P56 | P57 | P6159a8 | P6163a8 | P6161 | P6161a1 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 5.0 | 1 | 1 | 1 | NaN | NaN | 1 | 6.0 |
| 4 | 2 | 5.0 | 1 | 1 | 1 | NaN | NaN | 1 | 3.0 |
| 5 | 2 | 5.0 | 1 | 1 | 1 | 1.0 | 1.0 | 1 | 1.0 |
| 6 | 2 | 5.0 | 1 | 1 | 1 | NaN | NaN | 1 | 4.0 |

```
table_memores1.P52.value_counts()
```

Cuadro 4: Conteo de valores en P52

| P52 | Conteo |
|---|---|
| 5.0 | 1597 |
| 4.0 | 647 |
| 1.0 | 229 |
| 3.0 | 119 |
| 2.0 | 70 |
| 6.0 | 62 |

```
table_memores1.P6159s8.isnull() # Las madres que no hacen ninguna actividad con los menores
table_memores1.loc[table_memores1.P6159s8.isnull(), "P6159s8"] = 2 # 2 denota las madres que sí hacen

actividades con los hijos
table_memores1.P6163s8.isnull() # Los padres que no hacen ninguna actividad con los menores
table_memores1.loc[table_memores1.P6163s8.isnull(), "P6163s8"] = 2 # 2 denota los padres que sí
hacen actividades con los hijos
table_memores1.P6163s8.value_counts()
```

Cuadro 5: Conteo de valores en P6163s8

| P6163s8 | Conteo |
|---|---|
| 2.0 | 2024 |
| 1.0 | 700 |

En las líneas de código anteriores estamos cambiando aquellos valores que tenemos faltantes en la columna P6159s8 y en P6163s8, los cuales se reemplazan por un 2. La idea de este estudio es hacer un algoritmo predictivo para establecer si el niño es llevado al médico para el control de crecimiento y desarrollo o no (P6161), más específicamente, vamos a determinar bajo qué condiciones de las variables explicativas los niños son o no llevados a control de crecimiento y desarrollo.
Las variables explicativas son:

- **P51**: ¿Dónde o con quién permanece el menor durante la mayor parte del tiempo entre semana?
- **P52**: ¿Cuál es la razón principal por la que el menor no asiste a una guardia, hogar comunitario o jardín?
- **P55**: ¿Recibe o toma desayuno en el lugar donde permanece la mayor parte del tiempo entre semana?
- **P56**: ¿Recibe o toma onces en el lugar donde permanece la mayor parte del tiempo entre semana?
- **P57**: ¿Recibe o toma almuerzo en el lugar donde permanece la mayor parte del tiempo entre semana?
- **P6159s8**: La madre no realiza ninguna actividad con el menor
- **P6163s8**: El padre no realiza ninguna actividad con el menor

Finalmente, determinamos el uso del algoritmo de aprendizaje de máquinas, pues tenemos 2024 datos, donde el conjunto de datos de entrenamiento debe tener un porcentaje de explicación del 70 %-80 % de los datos.

```
dummies_memores = pd.get_dummies(table_memores1,
                                 columns=["P51", "P52", "P55", "P56", "P57",
                                          "P6159s8", "P6163s8"],
                                 dtype=float)
```

```
dummim_menores.head(5)
```

Cuadro 6: Primeras 5 filas de datos dummy

| | P6181 | P61811 | P6182 | P6183 | P6184 | P6185 | P6186 | P6187 | P6188 | P6189 | P6181 | P6182 | P6183 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 60 | 00 | 00 | 10 | 00 | 00 | 00 | 00 | - | 10 | 00 | 10 |
| 4 | 1 | 30 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | - | 10 | 00 | 10 |
| 5 | 1 | 10 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | - | 10 | 00 | 10 |
| 6 | 1 | 40 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | - | 10 | 00 | 10 |
| 7 | 1 | 10 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | - | 10 | 00 | 10 |

5 rows × 25 columns

Transformamos las variables categóricas a variables tipo Dummy.

```
tabla_menores1.P6161.value_counts()
```

Cuadro 7: Conteo de valores en P6161

| P6161 | Conteo |
|---|---|
| 1 | 2210 |
| 2 | 514 |

```
dummim_menores.columns
```

```
Index(['P6161', 'P6161s1', 'P581_2', 'P581_2', 'P581_4', 'P581_5', 'P581_6',
       'P581_7', 'P581_8', 'P582_1.0', 'P582_2.0', 'P582_2.0', 'P582_1.0',
       'P582_5.0', 'P582_6.0', 'P585_1', 'P585_2', 'P586_1', 'P586_2',
       'P587_1', 'P587_2', 'P6159a8_1.0', 'P6159a8_2.0', 'P6159a8_1.0',
       'P6159a8_2.0'], dtype='object')
```

Cuando ya tenemos las variables listas para implementar a un modelo, es más cómodo renombrar las columnas según su descripción en la encuesta.

```
dummim_menores.rename(columns = {
    'P581_2':'acu casa',
    'P581_3':'acu trab',
    'P581_4':'nafacer',
    'P581_5':'mayor',
    'P581_5':'mencer',
    'P581_7':'solo',
    'P581_8':'otra persona'}, inplace = True)

dummim_menores.rename(columns = {
    'P582_1.0':'lejos',
    'P582_2.0':'costoso',
    'P582_2.0':'sin cupo',
    'P582_4.0':'casa',
    'P582_5.0':'edad',
    'P582_6.0':'otro motivo'}, inplace = True)

dummim_menores.rename(columns = {
    'P585_1':'si desa',
    'P585_2':'no desa',
    'P585_1':'si alm',
    'P585_2':'no alm',
    'P585_1':'si oncea',
    'P585_2':'no oncea'}, inplace = True)

dummim_menores.rename(columns = {
    'P6159a8_1.0':'no comp madre',
    'P6159a8_2.0':'si comp madre',
    'P6159a8_1.0':'no comp padre',
    'P6159a8_2.0':'si comp padre'}, inplace = True)

dummim_menores.head(5)
```

```
# "PS1", "PS2", "PS5", "PS6", "PS7", "PS15s8" y "PS16s8", y la salida es la variable "PS16i"
x = dumm_i_memores.drop(['PG16i', 'PG16is1'], axis=1)
y = dumm_memores["PG16i"].astype(str)
y.loc[(y == "1")] = "Sí"
y.loc[(y == "2")] = "No"
y.value_counts()
```

Cuadro 8: Conteo de valores en PS16i

| PS16i | Conteo |
|---|---|
| Sí | 2210 |
| No | 514 |

Ahora vamos a determinar el conjunto de datos de entrenamiento y el conjunto de datos de validación. Primero, tomamos las variables de entrada y la variable de salida (PS16i) que se cambió a una variable tipo `str`, pues toma valores de 1 y 2 que son valores numéricos, pero que en la encuesta son cualidades. Por esta razón, se cambia a ese tipo, para que Python sepa que esos valores se deben interpretar como caracteres. En segundo lugar, se da una cualidad para cada valor de la variable de salida.
En la primera línea se toman las variables explicativas que son todas las variables menos PS16i, la cual denota si el menor recibe control de crecimiento y desarrollo. Según los resultados hay 514 menores a los

que no llevan a controles de crecimiento y desarrollo y 2210 que sí asisten a dichos controles.

```
from sklearn.model_selection import train_test_split
trainX, testX, trainY, testY = train_test_split(x, y, test_size=0.2)
```

```
from sklearn.neural_network import MLPClassifier
mlp_clf = MLPClassifier(hidden_layer_sizes=(50,50,50,50,50,50),
                        max_iter=300,
                        activation='relu',
                        solver='adam')
```

Este código es para implementar el modelo de la red neuronal, que tiene 6 capas ocultas con 50 nodos cada una.

```
mlp_clf.fit(trainX, trainY) # Estimación de los parámetros sobre los datos de entrenamiento
y_pred = mlp_clf.predict(testX) # Predicción sobre los datos de validación
```

Luego, se entrena el modelo; prosigue la predicción sobre los datos de validación. Lo que hacemos es calcular la exactitud respecto a los datos de validación, esto es el porcentaje de veces que los datos coinciden con su predicción, para con ello determinar si el modelo es adecuado o no, lo que implica que comparemos la predicción con los datos reales.

```
from sklearn.metrics import accuracy_score
print('Accuracy: {:.2f}'.format(accuracy_score(y_pred, testY)))
```

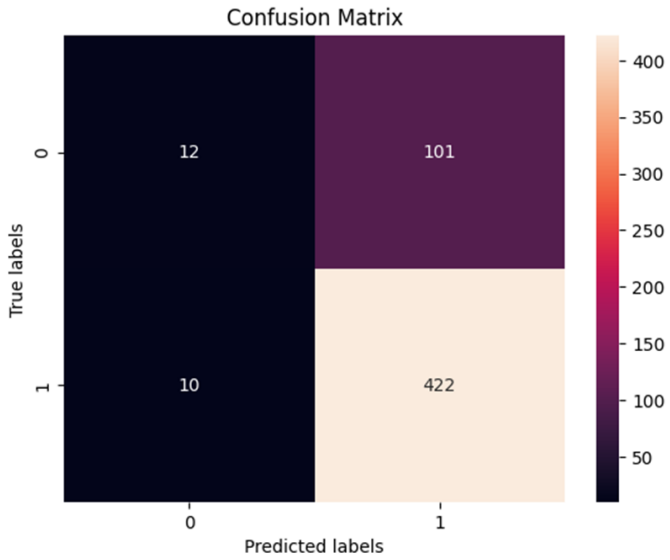
```
Accuracy: 0.80
```

La exactitud nos dio un 80 %, lo que significa que no hay ni underfitting ni overfitting, es decir, hay buen ajuste con el modelo de la red neuronal.

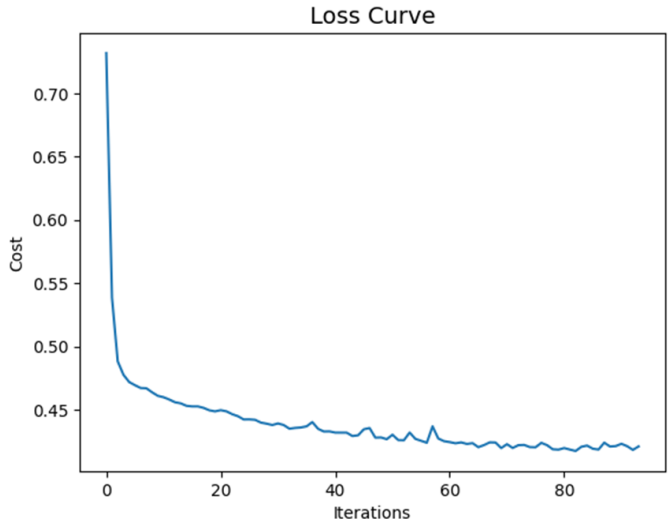
```
from sklearn.metrics import confusion_matrix
import pylab as pl
import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(testY, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

Implementada la matriz de confusión, tenemos que ver que los valores de las diagonales sean los más altos respecto a los otros, esto para confirmar que todos los datos tienen una buena predicción. La red neuronal multicapa hasta el momento hace una buena predicción para los menores que no reciben buena atención médica, en otras palabras, no reciben control de crecimiento y desarrollo.




```
plt.plot(mlp_clf.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```




A pesar de que la exactitud y la matriz de confusión arrojen resultados favorables, la curva de pérdida nos permite decir que el modelo de la red neuronal puede no llegar a una adecuada convergencia. Tiene que aparecer el cero y ahí no aparece, es decir que la función de costo llegue a cero, esto es, que la función de pérdida sea lo suficientemente pequeña para garantizar una buena convergencia. En este caso llega a 0.45, la asíntota sería el cero, pero no lo alcanza. Además, no basta con generar más capas y más nodos, pues según el comportamiento de los datos, no converge. Nos toca implementar otro modelo hasta que los datos se ajusten bien.

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(trainX, trainY)
y_pred_random = random_forest.predict(testX)
print('Accuracy: {:.2f}'.format(accuracy_score(y_pred_random, testY)))
```

**Accuracy: 0.80**

Random Forest es otro algoritmo de clasificación, lo usamos con 100 estimaciones para ver si mejora la exactitud o no, todo para que esté por encima del 80 % y quedó justo en este porcentaje. La idea es jugar con los hiperparámetros para una mejor precisión. Ahora nos preocupa la matriz de confusión.

```
from sklearn.metrics import confusion_matrix
import pylab as pl
import seaborn as sns
import matplotlib.pyplot as plt

cm1 = confusion_matrix(testY, y_pred_random)
plt.figure(figsize=(8,6))
sns.heatmap(cm1, annot=True, fmt='g', cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```

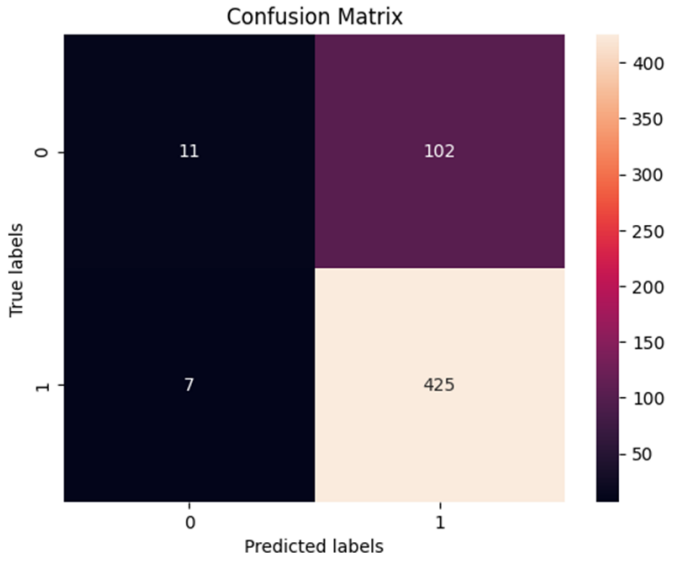



Figura 1: Visualización de la matriz de confusión

Otra matriz con un algoritmo nuevo y de nuevo nos interesa la diagonal principal. Nuestro interés es decirle al trabajador social bajo qué condiciones los niños no son atendidos.
Entonces nos fijamos en el primer cuadrante, este algoritmo empeoró la predicción de los niños que no asisten al control de crecimiento en comparación con la red neuronal multicapa.

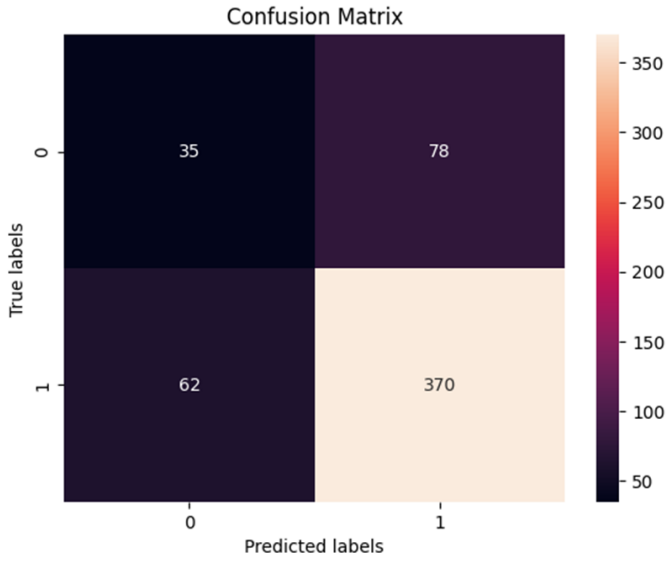
```
from sklearn.naive_bayes import GaussianNB
class_bayes = GaussianNB()
class_bayes.fit(trainX, trainY)
y_pred_bayes = class_bayes.predict(testX)
print('Accuracy: {:.2f}'.format(accuracy_score(y_pred_bayes, testY)))
```

```
Accuracy: 0.74
```

Ahora usamos el algoritmo clasificador Gaussiano. Por más que se modificaron los hiperparámetros, el modelo empeoraba, entonces se dejó sin ninguna modificación y con una exactitud del 74 %, lo cual indica underfitting, es decir, este algoritmo tiene un bajo ajuste para los datos presentados.

```
from sklearn.metrics import confusion_matrix
import pylab as pl
import seaborn as sns
import matplotlib.pyplot as plt

cmd = confusion_matrix(testY, y_pred_bayes)
plt.figure(figsize=(8,6))
sns.heatmap(cmd, annot=True, fmt="g", cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```




Aunque el clasificador Gaussiano no sea una buena implementación y presente un bajo ajuste, la matriz de confusión brinda la información que queremos, pues mejoró la predicción, garantizando que el clasificador tipo Naive Bayes (GaussianNB) es mejor para los individuos que no reciben control de crecimiento y desarrollo en contraste con la red neuronal y Random Forest.

```
from sklearn.svm import SVC

soporte_vector = SVC(decision_function_shape='ovr',
                     kernel="rbf",
                     degree=3)
soporte_vector.fit(trainX, trainY)
y_pred_sopor = soporte_vector.predict(testX)
print('Accuracy: {:.2f}'.format(accuracy_score(y_pred_sopor, testY)))
```

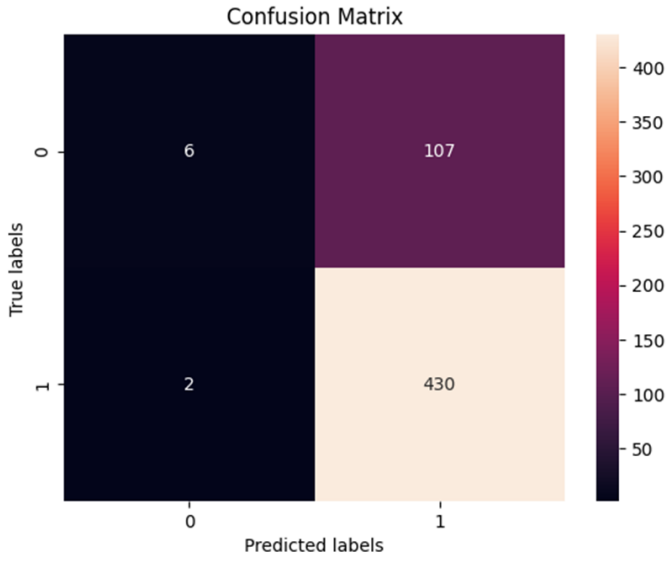
```
Accuracy: 0.80
```

Aplicamos un último algoritmo, que es: Máquina de Soporte Vectorial, la exactitud es adecuada y arroja buen ajuste del modelo.

```
from sklearn.metrics import confusion_matrix
import pylab as pl
import seaborn as sns
import matplotlib.pyplot as plt

cmd = confusion_matrix(testY, y_pred_sopor)
plt.figure(figsize=(8,6))
sns.heatmap(cmd, annot=True, fmt="g", cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix - SVM')
plt.show()
```

```
sns.heatmap(cmd, annot=True, fmt="g", ax=ax)  # annot=True para anotar celdas
plt.xlabel('Etiquetas Predichas')
plt.ylabel('Etiquetas Verdaderas')
plt.title('Matriz de Confusión')
```



El modelo demostró un buen ajuste, pero la matriz de confusión empeoró drásticamente en comparación con los algoritmos anteriormente implementados. Por tanto, nos quedamos con el clasificador Gaussiano, específicamente con los datos de la matriz de confusión.

```
trainX.columns
```

```
Index(['acu casa', 'acu trab', 'nafacer', 'mayor', 'mencer', 'solo',
       'otra persona', 'lejos', 'costoso', 'sin cupo', 'casa', 'edad',
       'otro motivo', 'si desa', 'no desa', 'si alm', 'no alm',
       'si oncea', 'no oncea', 'no comp madre', 'si comp madre',
       'no comp padre', 'si comp padre'], dtype='object')
```

```
import numpy as np
no_obs = np.array([[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1]])
class_bayes.predict(no_obs)
```

```
array(['No'], dtype='<U2')
```

El resultado [No] evidencia que tenemos un menor que no recibe atención médica. Para saber las características de este individuo es necesario mirar los valores de las columnas que tienen nuestros elementos. Un menor no recibe atención médica/control cuando:

[leftmargin=*]Se queda solo en casa. No puede ir a una guardería o jardín por falta de cupos. No desayuna. No almuerza. Toma onces. No comparte ninguna actividad con su madre. Comparte tiempo con su padre.

**Recomendaciones**

Estas condiciones sugieren negligencia en el cuidado del menor, y exposición a amenazas que pueden influir en su desarrollo integral, como se evidencia en la falta de atención médica. Es una urgencia que los padres cuiden a sus hijos y les brinden necesidades básicas como compañía, comida y salud.

```
no_obs1 = np.array([[1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
class_bayes.predict(no_obs1)
```

- `array(['No'], dtype='<U2')`

Por otro lado, tenemos otro individuo que no es llevado al control de crecimiento y desarrollo con las siguientes características:

- Está con su madre en la casa.
- No asiste a una guardería porque es muy costoso.
- Sí desayuna.
- Sí almuerza.
- Sí toma onces.

- La madre no realiza ninguna actividad con el menor.
- El padre no realiza ninguna actividad con el menor.

**Recomendaciones**

Aunque este perfil cuenta con condiciones básicas, es importante estar al pendiente de la salud de los niños y cómo se desarrollan. Los padres deben entender que los controles médicos no solo son para emergencias, sino una herramienta para asegurar que el menor crezca sano y con las oportunidades que merece.

Finalmente, con estos casos de estudio podemos concluir que actualmente los menores se enfrentan a diferentes riesgos desde casa. Este tipo de encuesta y recomendaciones no trata de culpar, sino de promover acción. Las familias colombianas deben saber que no están solas - existen redes de apoyo, programas sociales y profesionales dispuestos a ayudar para asegurar el bienestar de los niños.

## Referencias


DANE. (2024). *Gran Encuesta Integrada de Hogares – GEIH.* https://microdatos.dane.gov.co/index.php/catalog/819

Ministerio de Salud y Protección Social. (2021). *Lineamientos para la atención integral en salud.* https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/PP/Lineamiento-atencion-i pdf