Actividad 1: Laboratorio De Aprendizaje Automático, UNIR

- Realizado por: Nicolás Felipe Trujillo Montero

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
```

Primer Paso

Análisis Descriptivo de los Datos

In [2]:

```
df_base = pd.read_csv("pima-indians-diabetes.csv")
df_base.dtypes
```

Out[2]:

```
nEmbarazos                  int64
concentracionGlucosa        int64
presionArterialSistolica    int64
pliegueCutaneo              int64
insulinaSerica              int64
IMC                       float64
funcionPediDiabe          float64
edad                        int64
diabetes                    int64
dtype: object
```

a) Comentar de manera general qué se puede observar

Pues nos encontramos ante una función que lee de un DataFrame, nos dice el tipo de datos que contiene cada columna y el tipo del Dataframe que es de tipo objeto. En este caso todas las variables son númericas de tipo entero de 64 bits, excepto IMC y funcionPediDiabe que son númericas de tipo punto flotante de 64 bits (punto flotante es equivalente al conjunto de los números reales)

In [3]:

```
df_base.describe()
```

Out[3]:

	nEmbarazos	concentracionGlucosa	presionArterialSistolica	pliegueCutaneo	insulinaSerica	IMC	funcionPediDiabe	edad	76
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	

b) Se pueden observar las estadísticas de las columnas numéricas. ¿Si se tienen 768 observaciones, a qué conclusiones podríamos llegar con estos datos? ¿Podríamos eliminar alguna variable?

Este dataframe resultante nos indica por cada variable (columna) el número de registros/observaciones que hay, la media aritmética, la desviación estándar, el elemento más pequeño, el elemento que se encuentra en el primer cuartil, el elemento que se encuentra en el segundo cuartil, el elemento que se encuentra en el tercer cuartil y el elemento más grande.

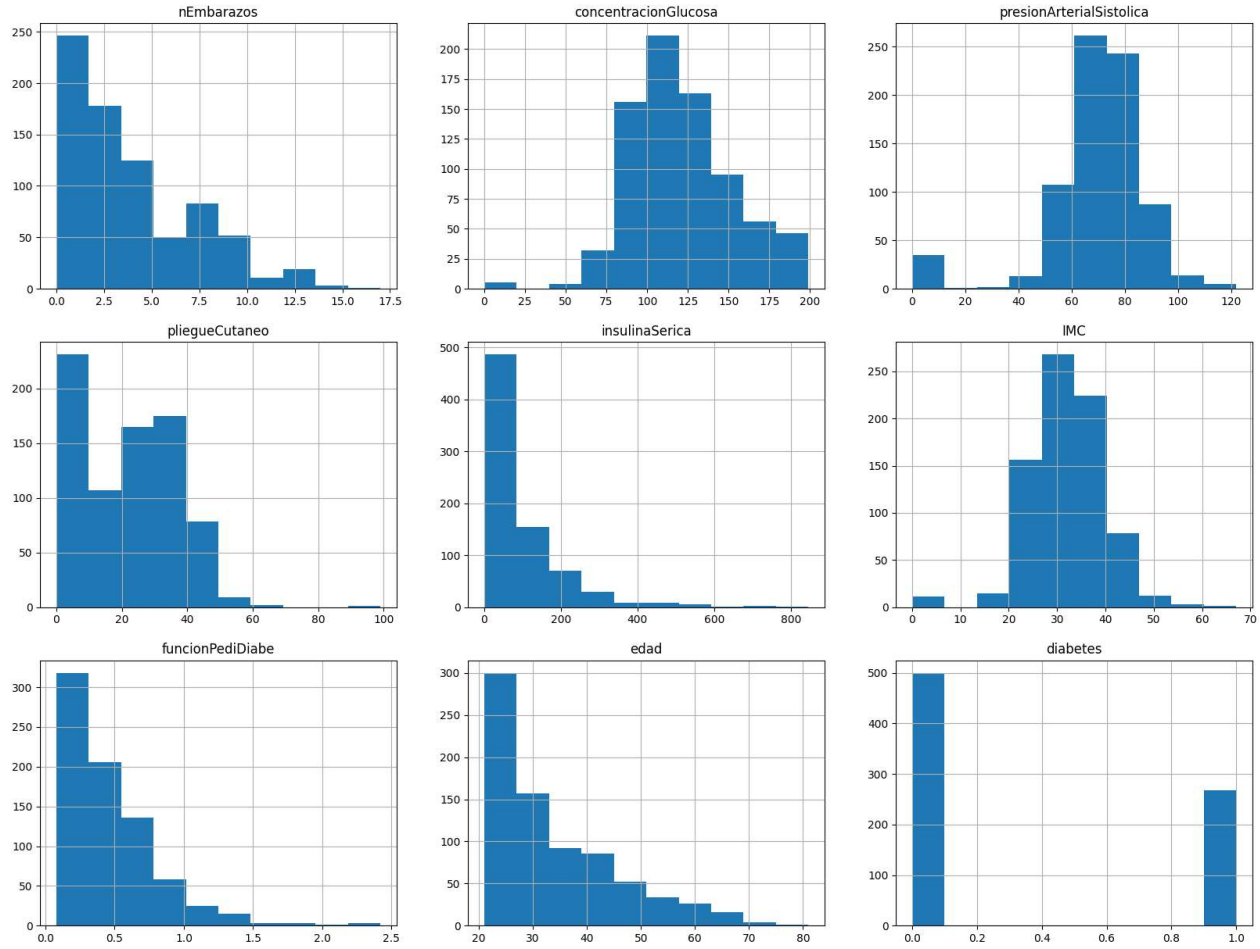
Gracias a los datos podemos ver como están distribuido los datos o si tienen un sesgo, es decir, si se agrupa la mayoría de los datos en alguna zona. También podemos observar que cada dato tiene un intervalo distinto, por ejemplo, el número de embarazos se mueve entorno a [0,17], y el de insulinaSerica se mueve entorno a [0,846].

Si eliminar alguna variable significa quitar la columna entera de nuestro marco de datos, entonces tendríamos que investigar a posteriori si nos interesa quitarla porque a primera vista, teniendo en cuenta la falta de base de conocimientos de salud, no creo que haga falta.

Si eliminar alguna variable significa quitar la observacion, entonces si eliminaria algunas observaciones ya que que una mujer tenga de IMC 0 no tiene ningún tipo de sentido al igual que la presión arterial, concentracion, pliegue e insulina (A no ser que las mujeres que se observan estén difuntas cosas que no tiene mucho sentido).

In [4]:

```
nvar = df_base.shape[1]
nrow = 3
ncol = 3

fig, ax = plt.subplots(nrow, ncol, figsize=(20, 15))

for irow in range(0, nrow):
    for icol in range(0,ncol):
        ax[irow, icol].hist(df_base[ df_base.columns[irow*nrow + icol] ])
        ax[irow, icol].grid()
        ax[irow, icol].set_title(df_base.columns[irow*nrow + icol])
```



c) Se muestran los histogramas de cada una de las columnas. ¿Qué se puede decir de la distribución de las variables?

Pues se ve que la distribución de la mayoría de las variables están sesgadas hacia la izquierda, es decir, que su kurtosis es positiva, y que la moda es un valor menor al de la media ya que la media es un valor muy influenciado por los outlier.

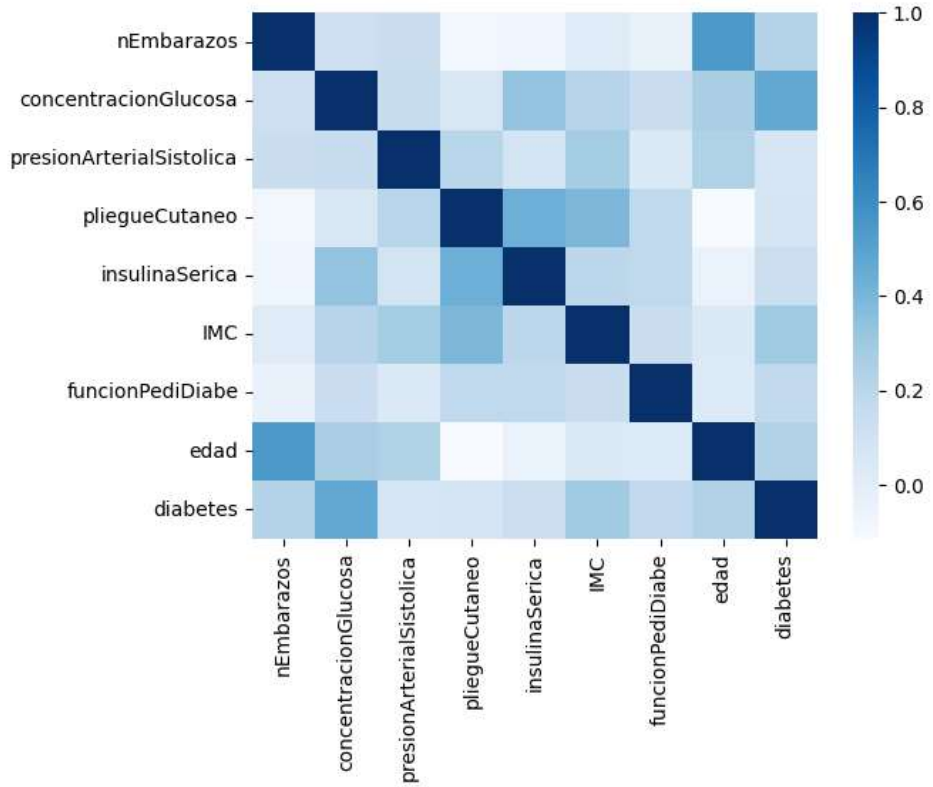
De las variables concentracionGlucosa, presionArterialSistólica y el IMC podemos decir que siguen una distribucion normal de los datos.

In [5]:

```
sns.heatmap(df_base.corr(), cmap="Blues")
```

Out[5]:

```
<AxesSubplot: >
```



d) Tenemos el mapa de calor de la matriz de correlaciones, por favor revise cuáles son las variables que mayor correlación tienen y si se puede eliminar alguna columna con base en este mapa de calor.

Gracias a la matriz de correlación podemos ver que la variable nEmbarazos y la variable edad están muy correlacionadas. Esto es un aspecto que tiene bastante sentido ya que es muy extraño que teniendo poca edad hayas tenido mucho número de embarazos, por lo que, conforme aumenta la edad, aumenta el número de embarazos.

Uno de los momentos clave para eliminar variables para nuestro análisis es la visión de la matriz de correlación. Si en una fila en este caso, sin contar la relación de una variable consigo misma, los valores son muy cercanos a 0 eso significa que la variable no están correlacionadas y son independientes, como en el caso de pliegueCutaneo que la mayoría de las correlaciones son por debajo de 0.5, por lo que cabría la posibilidad de eliminarla (Habría que analizar la existencia de relaciones no lineales ya que la matriz de correlación analiza las relaciones lineales).

Tratamiento de valores faltantes

In [6]:

```
col_total_nulos = df_base.isnull().sum()
serie_col_nombres = col_total_nulos[col_total_nulos > 0]
display(serie_col_nombres)
```

```
Series([], dtype: int64)
```

No existen valores faltantes

In [7]:

```
(df_base==0).sum(axis=0)
```

Out[7]:

```
nEmbarazos                  111
concentracionGlucosa          5
presionArterialSistolica     35
pliegueCutaneo              227
insulinaSerica              374
IMC                          11
funcionPediDiabe              0
edad                          0
diabetes                    500
dtype: int64
```

a) Podemos ver que existen columnas con ceros. Puede comentar ¿Qué puede estar ocurriendo con este conjunto de datos?

Pues significa que no existe ningún valor igual a 0 en la columna funcionPediDiabe, ni en la columna edad.

In [8]:

```
cols = ['concentracionGlucosa','presionArterialSistolica','pliegueCutaneo','insulinaSerica','IMC']
df_base[cols] = df_base[cols].replace({'0':np.nan, 0:np.nan})
print(df_base.isnull().sum())
```

```
nEmbarazos                    0
concentracionGlucosa          5
presionArterialSistolica     35
pliegueCutaneo              227
insulinaSerica              374
IMC                          11
funcionPediDiabe              0
edad                          0
diabetes                      0
dtype: int64
```

b) ¿Puede completar la descripción? ¿Qué acciones se realizan?

Lo que se ha realizado es cambiar de las columnas del array cols en el dataframe original, los valores 0 (esten guardados en string o en númerico) por valores "Not a number" y los contabilizamos. Aquellos valores que están a 0 significan que no tienen ningún valor a 0.

3. Entrenamiento de algoritmos

En este apartado intentaré aplicar las formas de llegar a los resultados, pero adjuntaré los del word y los explicaré

In [9]:

```
def graficar_accuracy_scores(estimator, train_x, train_y, test_x, test_y, nparts=5, seed=None, jobs=None):
    kfold = KFold(n_splits=nparts,shuffle=True, random_state=seed)
    fig,axes = plt.subplots(figsize=(7, 3))
    axes.set_title("Ratio de éxito(Accuracy)/Nro. Fold")
    axes.set_xlabel("Nro. Fold")
    axes.set_ylabel("Accuracy")
    train_scores = cross_val_score(estimator, train_x,train_y, cv = kfold, n_jobs=jobs, scoring="accuracy")
    test_scores = cross_val_score(estimator, test_x, test_y, cv = kfold, n_jobs=jobs, scoring="accuracy")
    train_sizes = range(1,nparts+1,1)
    axes.grid()
    axes.plot(train_sizes, train_scores, 'o-', color="r",label="Datos Entrenamiento")
    axes.plot(train_sizes, test_scores, 'o-', color="g",label="Datos de Test")
    axes.legend(loc="best")
    return train_scores
```

3.1 Eliminar todas las filas faltantes

In [10]:

```
df_base_res = df_base.dropna(axis=0)
df_base_res
```

Out[10]:

	nEmbarazos	concentracionGlucosa	presionArterialSistolica	pliegueCutaneo	insulinaSerica	IMC	funcionPediDiabe	edad	diabetes
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1
6	3	78.0	50.0	32.0	88.0	31.0	0.248	26	1
8	2	197.0	70.0	45.0	543.0	30.5	0.158	53	1
13	1	189.0	60.0	23.0	846.0	30.1	0.398	59	1
...	...	...	...	...	...	...	...	...	...
753	0	181.0	88.0	44.0	510.0	43.3	0.222	26	1
755	1	128.0	88.0	39.0	110.0	36.5	1.057	37	1
760	2	88.0	58.0	26.0	16.0	28.4	0.766	22	0
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0

392 rows × 9 columns

In [11]:

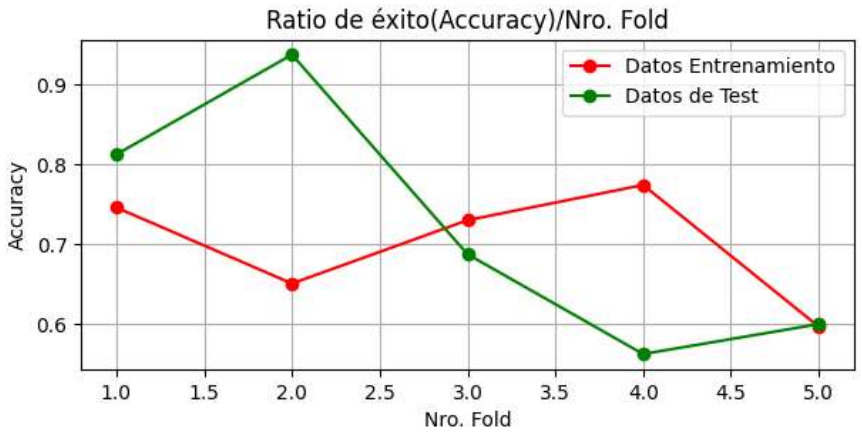
```
# Variables Globales
endog = df_base_res["diabetes"]
exog_names = list(set(df_base_res.columns) - set(["diabetes"]))
exog = df_base_res[exog_names]
n_folds = 5
seed=0
```

In [12]:

```
#########################################################
# Decision Tree Classifier
#########################################################

modelo = DecisionTreeClassifier(random_state=0)
X_train, X_test, Y_train, Y_test = train_test_split(exog, endog, test_size=0.2, random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[12]:

```
array([0.74603175, 0.65079365, 0.73015873, 0.77419355, 0.59677419])
```



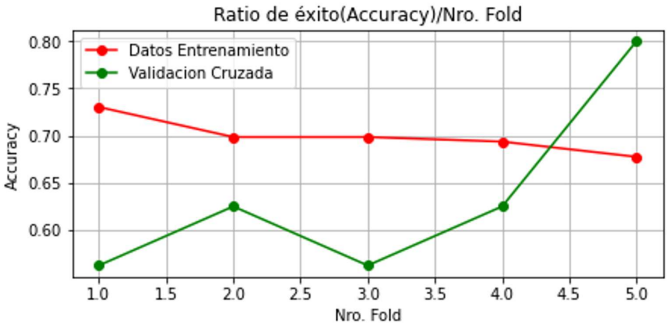
In [13]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        54
           1       0.71      0.60      0.65        25

    accuracy                           0.80        79
   macro avg       0.77      0.74      0.75        79
weighted avg       0.79      0.80      0.79        79
```

3.1 Solucion Decision Tree Classifier

```
              precision    recall  f1-score   support

           0       0.79      0.76      0.78        55
           1       0.50      0.54      0.52        24

    accuracy                           0.70        79
   macro avg       0.65      0.65      0.65        79
weighted avg       0.70      0.70      0.70        79
```



Comentario Decision Tree Classifier:

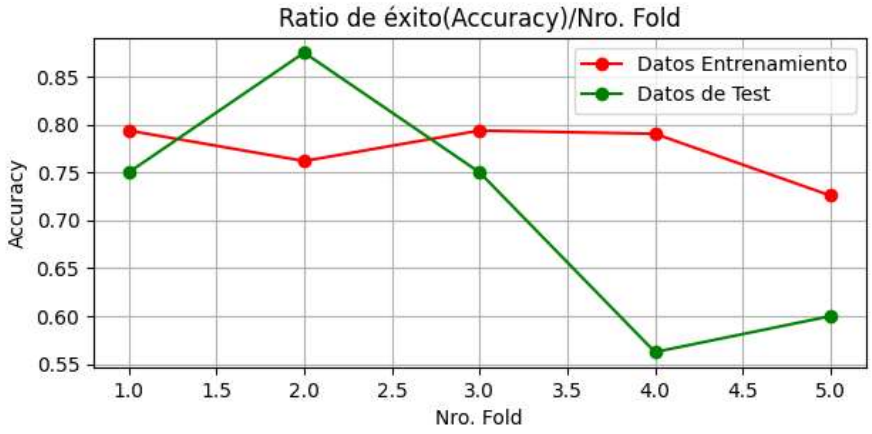
- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 70% asi que no existe un sobreajuste notable.
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el primer fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el quinto fold. Esto significa que los valores del conjunto de test están dispersos entre sí, por eso para la gráfica de los datos de test varía en accuracy según el fold que miremos, mientras que los valores del conjunto de entrenamiento están cercanos, por eso la gráfica de los datos del conjunto de entrenamiento varían muy poco en accuracy según el fold que miremos. Además, como de 5 fold, se cumple que los 4 primeros fold del conjunto de entrenamiento tienen mayor accuracy que los 4 primeros fold del conjunto de test, por lo que podemos llegar a la conclusión de que existe sobreajuste

In [14]:

```
#######################################################
# Random Forest Classifier
#######################################################

modelo = RandomForestClassifier(random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[14]:

```
array([0.79365079, 0.76190476, 0.79365079, 0.79032258, 0.72580645])
```



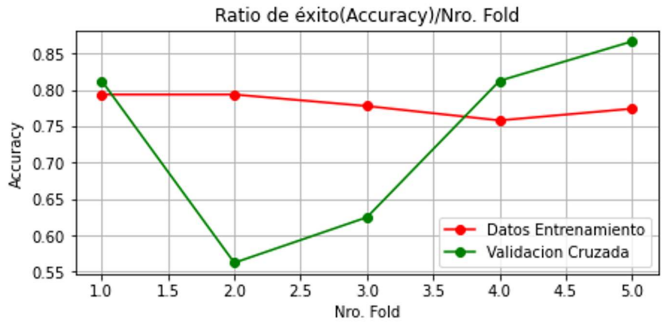
In [15]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        54
           1       0.70      0.56      0.62        25

    accuracy                           0.78        79
   macro avg       0.76      0.72      0.74        79
weighted avg       0.78      0.78      0.78        79
```

3.1 Solucion Random Forest Classifier

```
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        55
           1       0.64      0.58      0.61        24

    accuracy                           0.77        79
   macro avg       0.73      0.72      0.72        79
weighted avg       0.77      0.77      0.77        79
```



Comentario Random Forest Classifier:

- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 77% asi que no existe un sobreajuste notable.

- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el quinto fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el primer fold. Esto significa que los valores del conjunto de test están dispersos entre sí, por eso para la gráfica de los datos de test varía en accuracy según el fold que miremos, mientras que los valores del conjunto de entrenamiento están cercanos, por eso la gráfica de los datos del conjunto de entrenamiento varían muy poco en accuracy según el fold que miremos.

¿Cuál considero mejor? Viendo tanto la clasificación de Random Forest como la de Decision Tree, me decantaría por el modelo proporcionado por el Random Forest ya que nos proporciona mejor accuracy sin llegar a considerarse sobre ajuste debido a las gráficas proporcionadas

3.2 Eliminar todas las columnas con valores faltantes

In [16]:

```
df_base_res = df_base.dropna(axis=1)
df_base_res
```

Out[16]:

	nEmbarazos	funcionPediDiabe	edad	diabetes
0	6	0.627	50	1
1	1	0.351	31	0
2	8	0.672	32	1
3	1	0.167	21	0
4	0	2.288	33	1
...	...	...	...	...
763	10	0.171	63	0
764	2	0.340	27	0
765	5	0.245	30	0
766	1	0.349	47	1
767	1	0.315	23	0

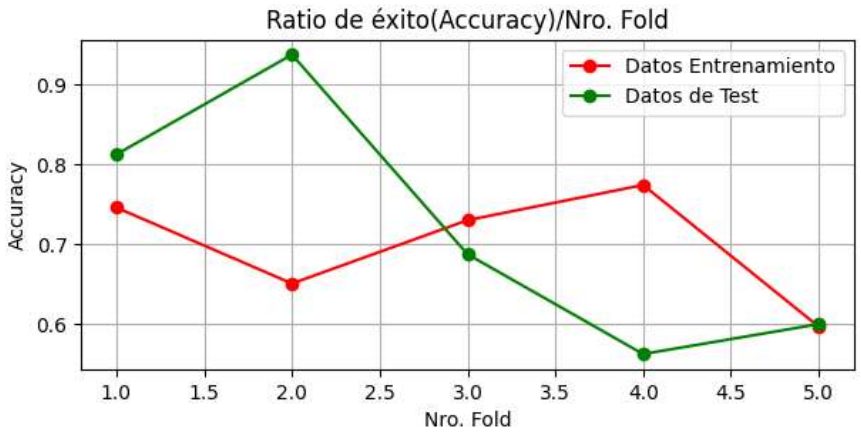
768 rows × 4 columns

In [17]:

```
#########################################################
# Decision Tree Classifier
#########################################################

modelo = DecisionTreeClassifier(random_state=0)
X_train, X_test, Y_train, Y_test = train_test_split(exog, endog, test_size=0.2, random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[17]:

```
array([0.74603175, 0.65079365, 0.73015873, 0.77419355, 0.59677419])
```



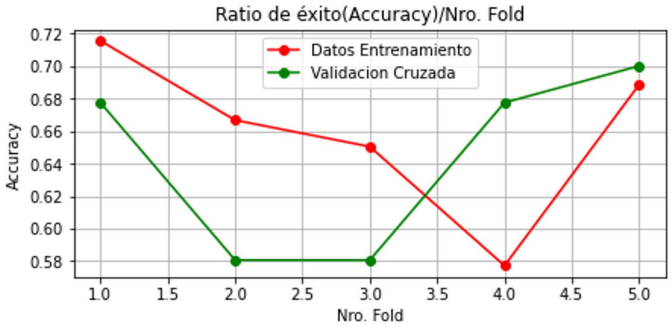
In [18]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        54
           1       0.71      0.60      0.65        25

    accuracy                           0.80        79
   macro avg       0.77      0.74      0.75        79
weighted avg       0.79      0.80      0.79        79
```

3.2 Solucion Decision Tree Classifier

```
              precision    recall  f1-score   support

           0       0.77      0.76      0.76        99
           1       0.57      0.58      0.58        55

    accuracy                           0.69       154
   macro avg       0.67      0.67      0.67       154
weighted avg       0.70      0.69      0.70       154
```



Comentario Decision Tree Classifier:

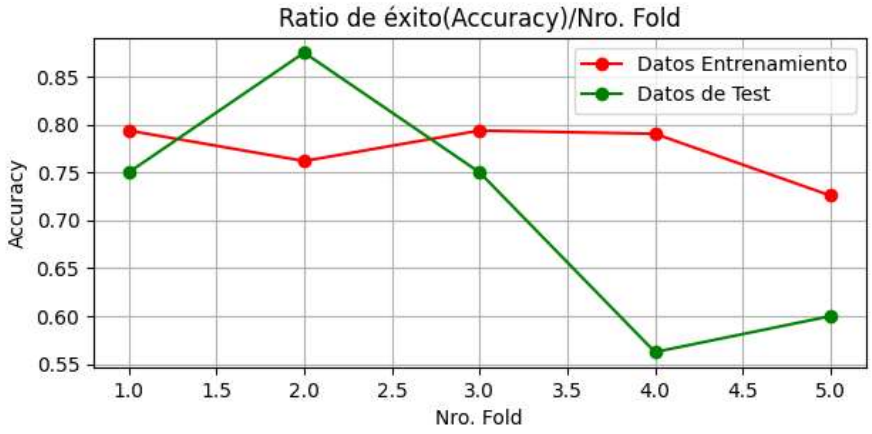
- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 69% asi que no existe un sobreajuste notable.
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el primer fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el quinto fold. Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor.

In [19]:

```
#######################################################
# Random Forest Classifier
#######################################################

modelo = RandomForestClassifier(random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[19]:

```
array([0.79365079, 0.76190476, 0.79365079, 0.79032258, 0.72580645])
```



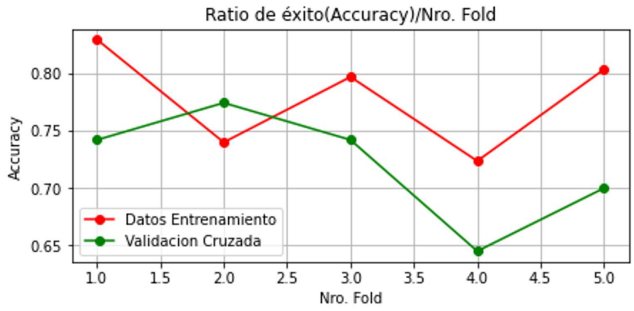
In [20]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        54
           1       0.70      0.56      0.62        25

    accuracy                           0.78        79
   macro avg       0.76      0.72      0.74        79
weighted avg       0.78      0.78      0.78        79
```

3.2 Solucion Random Forest Classifier

```
              precision    recall  f1-score   support

           0       0.74      0.87      0.80        99
           1       0.66      0.45      0.54        55

    accuracy                           0.72       154
   macro avg       0.70      0.66      0.67       154
weighted avg       0.71      0.72      0.71       154
```



Comentario Random Forest Classifier:

- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 72% asi que no existe un sobreajuste notable.

- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el primer fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el segundo fold. Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor.

¿Cuál considero mejor? Viendo tanto la clasificación de Random Forest como la de Decision Tree, me decantaría por el modelo proporcionado por el Random Forest ya que nos proporciona mejor accuracy sin llegar a considerarse sobre ajuste debido a las gráficas proporcionadas

3.3 Imputamos valores NA con la media

In [21]:

```
df_base_res = pd.DataFrame([df_base[col].replace({np.nan:np.mean(df_base[col])}) for col in df_base.columns])
df_base_res.T
```

Out[21]:

	nEmbarazos	concentracionGlucosa	presionArterialSistolica	pliegueCutaneo	insulinaSerica	IMC	funcionPediDiabe	edad	diabetes
0	6.0	148.0	72.0	35.00000	155.548223	33.6	0.627	50.0	1.0
1	1.0	85.0	66.0	29.00000	155.548223	26.6	0.351	31.0	0.0
2	8.0	183.0	64.0	29.15342	155.548223	23.3	0.672	32.0	1.0
3	1.0	89.0	66.0	23.00000	94.000000	28.1	0.167	21.0	0.0
4	0.0	137.0	40.0	35.00000	168.000000	43.1	2.288	33.0	1.0
...	...	...	...	...	...	...	...	...	...
763	10.0	101.0	76.0	48.00000	180.000000	32.9	0.171	63.0	0.0
764	2.0	122.0	70.0	27.00000	155.548223	36.8	0.340	27.0	0.0
765	5.0	121.0	72.0	23.00000	112.000000	26.2	0.245	30.0	0.0
766	1.0	126.0	60.0	29.15342	155.548223	30.1	0.349	47.0	1.0
767	1.0	93.0	70.0	31.00000	155.548223	30.4	0.315	23.0	0.0

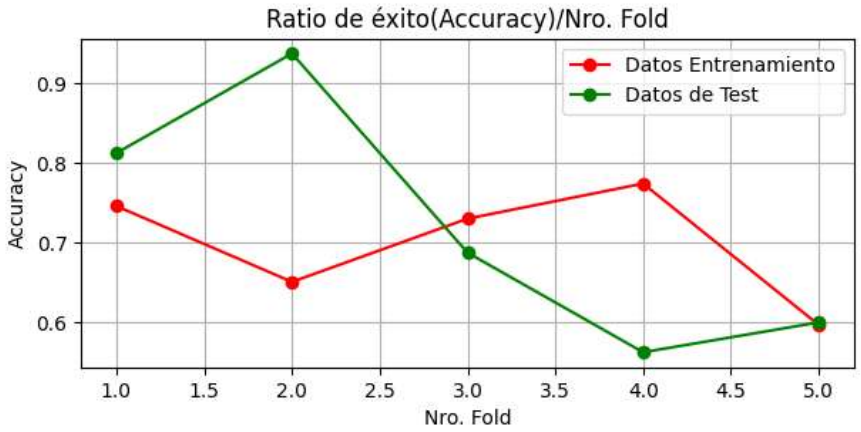
768 rows × 9 columns

In [22]:

```
#########################################################
# Decision Tree Classifier
#########################################################

modelo = DecisionTreeClassifier(random_state=0)
X_train, X_test, Y_train, Y_test = train_test_split(exog, endog, test_size=0.2, random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[22]:

```
array([0.74603175, 0.65079365, 0.73015873, 0.77419355, 0.59677419])
```



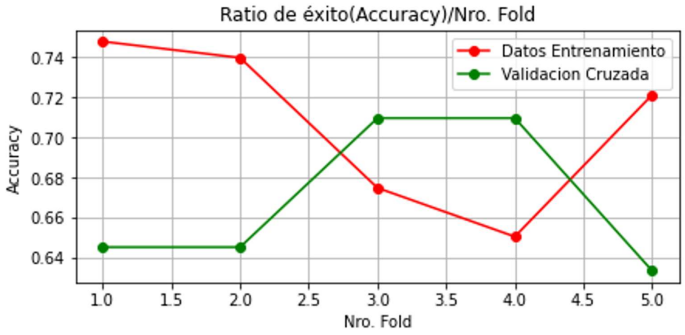
In [23]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86        54
           1       0.71      0.60      0.65        25

    accuracy                           0.80        79
   macro avg       0.77      0.74      0.75        79
weighted avg       0.79      0.80      0.79        79
```

3.3 Solucion Decision Tree Classifier

```
              precision    recall  f1-score   support

           0       0.81      0.73      0.77        99
           1       0.58      0.69      0.63        55

    accuracy                           0.71       154
   macro avg       0.70      0.71      0.70       154
weighted avg       0.73      0.71      0.72       154
```



Comentario Decision Tree Classifier:

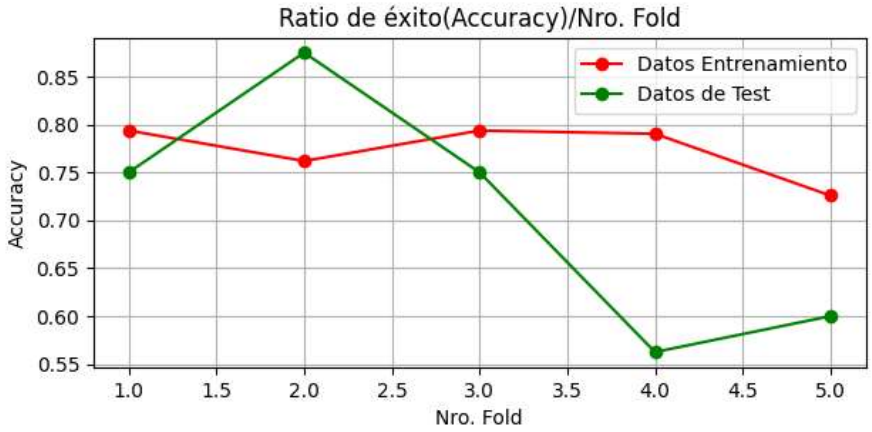
- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1**.** Además, podemos observar como el accuracy es de un 71% asi que no existe un sobreajuste notable**.**
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el primer fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el tercer y cuarto fold**.** Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor**.**

In [24]:

```
#######################################################
# Random Forest Classifier
#######################################################

modelo = RandomForestClassifier(random_state=0)
modelo.fit(X_train, Y_train)

graficar_accuracy_scores(modelo, X_train, Y_train, X_test, Y_test, nparts=5, seed=seed, jobs=-1)
```

Out[24]:

```
array([0.79365079, 0.76190476, 0.79365079, 0.79032258, 0.72580645])
```



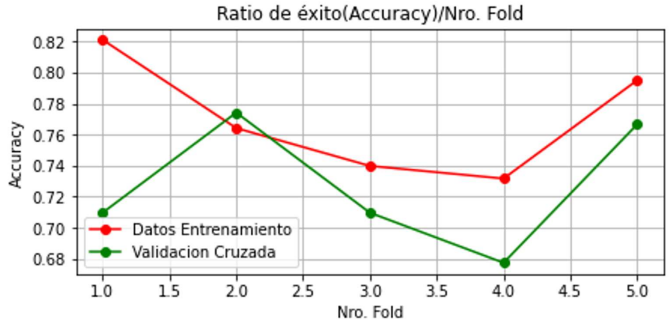
In [25]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.89      0.85        54
           1       0.70      0.56      0.62        25

    accuracy                           0.78        79
   macro avg       0.76      0.72      0.74        79
weighted avg       0.78      0.78      0.78        79
```

3.3 Solucion Random Forest Classifier

```
              precision    recall  f1-score   support

           0       0.76      0.87      0.81        99
           1       0.68      0.51      0.58        55

    accuracy                           0.74       154
   macro avg       0.72      0.69      0.70       154
weighted avg       0.73      0.74      0.73       154
```



Comentario Random Forest Classifier:

- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 74% asi que no existe un sobreajuste notable.

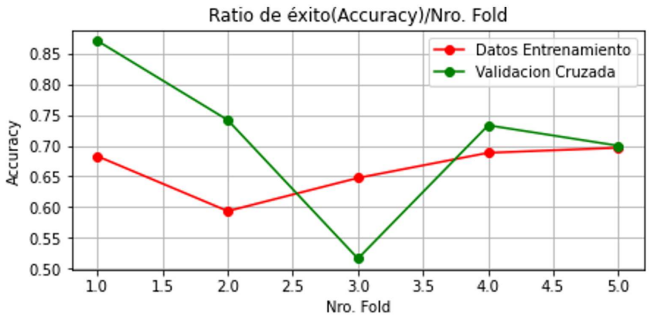
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el primer fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el segundo fold. Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor. Además, en 4 de los 5 fold, el accuracy de los datos de entrenamiento es mayor por lo que consecuentemente lleva a pensar que va a existir un sobreajuste.

¿Cuál considero mejor? Viendo tanto la clasificación de Random Forest como la de Decision Tree, me decantaría por el modelo proporcionado por el Decision Tree ya que nos proporciona menos accuracy, pero tiene considerablemente menos sobreajuste si queremos generalizar.

3.4 Imputamos valores NA con interpolacion

En este caso no he podido replicar exactamente el código ya que hay muchas formas de hacer interpolacion.

	precision	recall	f1-score	support
0	0.85	0.80	0.82	102
1	0.64	0.71	0.67	51
accuracy			0.77	153
macro avg	0.74	0.75	0.75	153
weighted avg	0.78	0.77	0.77	153



Comentario Decision Tree Classifier:

- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 77% asi que no existe un sobreajuste notable.
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el quinto fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el primer fold. Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor. Además, que el conjunto de test en su mayoría de los fold supere en accuracy al de entrenamiento nos da información de que el modelo está subajustado.

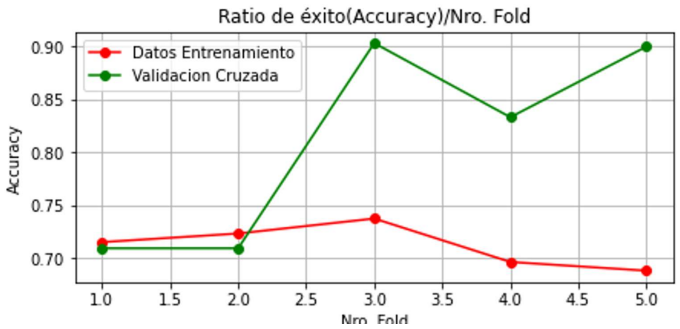
In [26]:

```
Y_test_pred = modelo.predict(X_test)
print(classification_report(Y_test, Y_test_pred))
```

	precision	recall	f1-score	support
0	0.81	0.89	0.85	54
1	0.70	0.56	0.62	25
accuracy			0.78	79
macro avg	0.76	0.72	0.74	79
weighted avg	0.78	0.78	0.78	79

3.4 Solucion Random Forest Classifier

	precision	recall	f1-score	support
0	0.82	0.91	0.87	102
1	0.78	0.61	0.68	51
accuracy			0.81	153
macro avg	0.80	0.76	0.77	153
weighted avg	0.81	0.81	0.80	153



Comentario Random Forest Classifier:

- **Classification report**: La etiqueta 0 es mejor etiquetada que la etiqueta 1 ya que la medida de precision (exactitud), f1-score y recall es mayor que la de la etiqueta 1. Además, podemos observar como el accuracy es de un 81% asi que no existe un sobreajuste notable.
- **Graficar_accuracy_scores**: Los gráficos nos indican que, para el modelo entrenado, el quinto fold nos da un mejor ratio accuracy para los datos de entrenamiento, y para los datos de validación cruzada, nos da un mejor resultado el primer fold. Esto significa que los valores del conjunto de test y de entrenamiento están dispersos entre sí, por eso ambas gráficas varían cada una de accuracy en cada valor. Además, que el conjunto de test en su mayoría de los fold supere en accuracy al de entrenamiento nos da información de que el modelo está subajustado.

¿Cuál considero mejor? Viendo tanto la clasificación de Random Forest como la de Decision Tree, me decantaría por el modelo proporcionado por el Random Forest ya que nos proporciona mayor

Comentarios Adicionales

- Alguna forma de mejorar sería implementar un sistema de correlaciones para analizar cuales son las variables mejor para pronosticar o sino un algoritmo de PCA.
- La mejor forma, a mi opinión es eliminar las columnas que tienen elementos NA ya que no es que sea el modelo con mejor accuracy, pero es el que mejor que se adapta ya que no presenta ni sobreajuste, ni subajuste.