Uso de Técnicas de Aprendizaje Automático en la Clasificación de Gliomas

Introducción al Aprendizaje Automático

Grupo 6: Micaela Floridia - Vanessa Galeano - Leandro Villanueva

##Primera parte - Análisis Exploratorio de Datos

Introducción

Para este trabajo, elegimos una temática que creemos puede tener un impacto significativo en la vida de las personas: los gliomas, un tipo de tumor cerebral. Una clasificación precisa y un diagnóstico adecuado de estos tumores son cruciales, ya que diferentes grados de glioma requieren enfoques de tratamiento distintos. Identificar correctamente si un glioma es de bajo grado (LGG), menos agresivo, o un glioblastoma multiforme (GBM), más agresivo, puede influir en las decisiones terapéuticas y, en consecuencia, mejorar las expectativas de los pacientes.

Nuestro objetivo con este análisis es explorar enfoques que mejoren la precisión en la identificación del tipo de glioma. Aplicando técnicas de aprendizaje automático aprendidas en esta materia y analizando las mutaciones genéticas clave asociadas con estos tumores, esperamos contribuir al avance en la predicción de gliomas y al diagnóstico temprano, así como al tratamiento más personalizado para los pacientes.

La relevancia de este tema nos motiva a abordarlo con dedicación y rigor. Nos entusiasma la oportunidad de aplicar lo aprendido en esta materia a un contexto que puede hacer una diferencia significativa en la vida de las personas, y estamos comprometidos a realizar un trabajo que aporte valor.

###Descripción breve del conjunto de datos

El conjunto de datos *Glioma Grading Clinical and Mutation Features* está compuesto por una combinación de características clínicas y mutaciones genéticas observadas en pacientes con glioma, es decir, pacientes ya diagnosticados. Se incluyen datos como la edad de los pacientes y el tipo de tumor, que puede ser un glioma de bajo grado (LGG), menos agresivo, o un glioblastoma multiforme (GBM), más agresivo. Las muestras también incluyen información detallada sobre mutaciones genéticas específicas encontradas en los tejidos tumorales de los pacientes, relevadas mediante biopsia. En la práctica, estos genes pueden o no estar relacionados con el tipo de tumor.

Enlace de donde se obtuvo el conjunto de datos:
archive.ics.uci.edu/dataset/759/glioma+grading+clinical+and+mutation+features+dataset

Enlace al estudio original: mdpi.com/1422-0067/23/22/14155

###Descripción del conjunto de datos

```
#from google.colab import drive
#drive.mount('/content/drive')

#Importamos las librerías que vamos a utilizar:
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import accuracy_score, precision_score,
recall_score, confusion_matrix
from sklearn.metrics import classification_report, roc_curve, auc
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression


#Leemos el dataset y visualizamos los encabezados y las primeras
filas:
#data = pd.read_csv('/content/drive/MyDrive/TCGA_InfoWithGrade.csv')
data = pd.read_csv('/content/TCGA_InfoWithGrade.csv')
data.head()
```

```
{"type":"dataframe","variable_name":"data"}
```

a - ¿Cuántas filas y columnas tiene?

```
data.shape
```

```
(839, 24)
```

El dataset tiene 839 filas y 24 columnas.

b - ¿Hay valores faltantes en columnas?

```
 data.isna().sum()
```

```
Grade                0
Gender               0
Age_at_diagnosis     0
Race                 0
IDH1                 0
TP53                 0
ATRX                 0
PTEN                 0
EGFR                 0
CIC                  0
```

```
MUC16                0
PIK3CA               0
NF1                  0
PIK3R1               0
FUBP1                0
RB1                  0
NOTCH1               0
BCOR                 0
CSMD3                0
SMARCA4              0
GRIN2A               0
IDH2                 0
FAT4                 0
PDGFRA               0
dtype: int64
```

Ninguna columna tiene valores faltantes.

c - Tipos de datos presentes:

```
data.dtypes
```

```
Grade                 int64
Gender                int64
Age_at_diagnosis    float64
Race                  int64
IDH1                  int64
TP53                  int64
ATRX                  int64
PTEN                  int64
EGFR                  int64
CIC                   int64
MUC16                 int64
PIK3CA                int64
NF1                   int64
PIK3R1                int64
FUBP1                 int64
RB1                   int64
NOTCH1                int64
BCOR                  int64
CSMD3                 int64
SMARCA4               int64
GRIN2A                int64
IDH2                  int64
FAT4                  int64
PDGFRA                int64
dtype: object
```

La primer columna, 'Grade', representa el grado del glioma, indicado con los numeros '0' o '1':

- Bajo grado (0 = LGG), agresivo.
- Glioblastoma multiforme (1 = GBM), muy agresivo.

El género ('Gender') también es binario, donde '0' representa masculino y '1' femenino.

'Age_at_diagnosis' indica la edad al momento del diagnóstico y está expresada como Float.

'Race' indica la raza o grupo étnico en una escala numérica del 0 al 3, donde:

- 0: Blanco
- 1: Asiático
- 2: Negro o Afroamericano
- 3: Indígena americano o nativo de Alaska

El resto de las columnas corresponde a los genes analizados en la biopsia, donde el valor '1' indica que el gen está mutado y el valor '0' que no lo está.

Verificamos la distribución de las clases en nuestro conjunto de datos para asegurarnos de que no exista un desbalance significativo:

```
y = data['Grade']
conteo_clases = y.value_counts()
porcentaje_clases = (y.value_counts(normalize=True) *
100).round().astype(int)
print("Conteo de clases:", conteo_clases)
print("\nPorcentaje de clases:",porcentaje_clases)
```

```
Conteo de clases: Grade
0    487
1    352
Name: count, dtype: int64

Porcentaje de clases: Grade
0    58
1    42
Name: proportion, dtype: int64
```

Encontramos que el 58% de las instancias pertenecen a la clase 0 (glioma de bajo grado - LGG) y el 42% a la clase 1 (glioblastoma multiforme - GBM). Aunque las clases no están perfectamente balanceadas, la diferencia no es extrema, por lo que decidimos continuar sin aplicar técnicas de ajuste por desbalance.

Descripción del problema / Pregunta a resolver

El objetivo de este análisis es predecir el tipo de glioma que presenta cada paciente. Para ello, utilizaremos como variable objetivo ('target') la columna 'Grade', que indica si el glioma es de bajo grado (LGG - valor 0) o glioblastoma multiforme (GBM - valor 1), tratándose de un problema de clasificación binaria. Evaluaremos el rendimiento de nuestro modelo utilizando métricas como la exactitud, precisión, exhaustividad (recall) y el F1-Score.

Las clases se interpretarán de la siguiente manera:

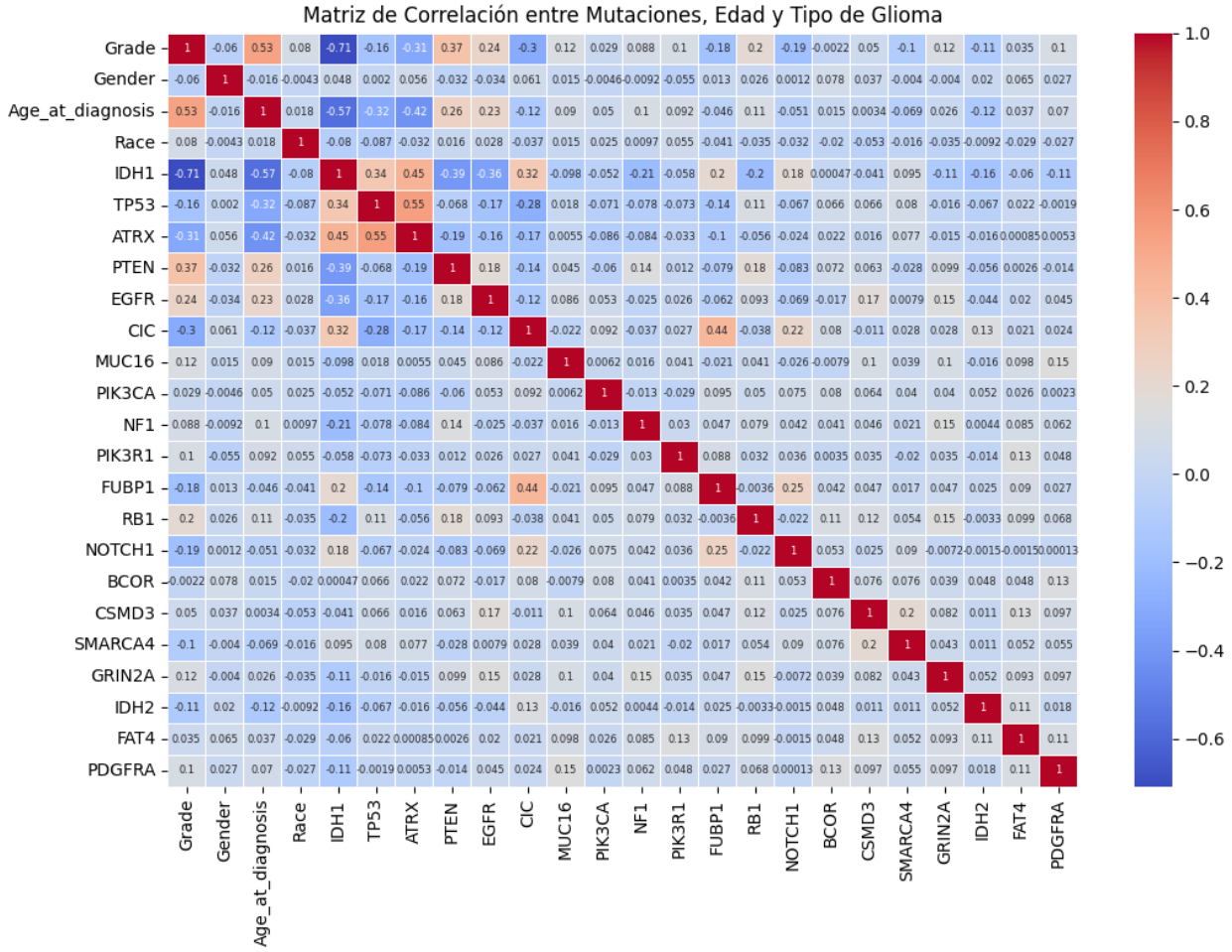
- Positivo: glioblastoma multiforme (GBM, Clase 1).
- Negativo: glioma de bajo grado (LGG, Clase 0).

Dado que los falsos negativos (glioblastomas, que son muy agresivos, clasificados erróneamente como gliomas de bajo grado, menos agresivos) pueden llevar a tratamientos incorrectos o insuficientes, decidimos priorizar la exhaustividad (recall). Es fundamental identificar correctamente los glioblastomas para asegurar que el paciente reciba un tratamiento adecuado, dado el pronóstico más severo de estos tumores. De este modo, buscamos minimizar los falsos negativos y evitar que un diagnóstico incorrecto de glioblastoma resulte en tratamientos menos efectivos, lo que podría afectar significativamente la salud y calidad de vida del paciente.

###**Elección de atributos**

Para comprender la relación entre las mutaciones genéticas y el tipo de tumor, así como para seleccionar las características más relevantes, utilizamos la matriz de correlación. Esta herramienta nos permite identificar las asociaciones entre las mutaciones y el tipo de glioma, ayudándonos a discernir qué genes tienen una influencia significativa en la clasificación del tumor. Al enfocarnos en las variables con mayor valor predictivo, podemos mejorar la precisión del análisis. Además, la matriz de correlación facilita la reducción de la dimensionalidad del conjunto de datos, optimizando la eficiencia del modelo.

```
#Generamos la matriz de correlación
correlacion = data.corr()

#Mostramos la matriz con un mapa de calor 'coolwarm'
#Le reducimos la fuente para que sea mas legible
plt.figure(figsize=(12, 8))
# Ajustamos "size" para reducir el tamaño de la fuente
sns.heatmap(correlacion, annot=True, cmap='coolwarm', linewidths=0.5,
            annot_kws={"size": 6})
plt.title('Matriz de Correlación entre Mutaciones, Edad y Tipo de
Glioma')
plt.show()
```



La matriz de correlación revela que el gen IDH1 tiene una fuerte correlación negativa con Grade (−0.71), lo que sugiere que su mutación está asociada con un pronóstico más favorable, es decir, con gliomas de bajo grado (LGG). Esto implica que la presencia de mutaciones en el gen IDH1 podría estar vinculada a una menor agresividad del tumor.

En cuanto a la edad, observamos una correlación positiva moderada con Grade (0.53). Esto indica que los pacientes mayores tienden a presentar gliomas más agresivos, como el glioblastoma multiforme (GBM). En otras palabras, a medida que aumenta la edad, la probabilidad de tener un glioma más agresivo también aumenta.

Además, algunos genes, como ATRX, PTEN y EGFR, muestran una correlación moderada con el grado del glioma, con valores cercanos a 0.30. Aunque estas correlaciones no son extremadamente fuertes, sugieren que las mutaciones en estos genes podrían estar relacionadas con la severidad del glioma. Esta información es útil para considerar estos genes en la clasificación y tratamiento de los tumores, pero su influencia debe ser evaluada en conjunto con otras variables y datos adicionales.

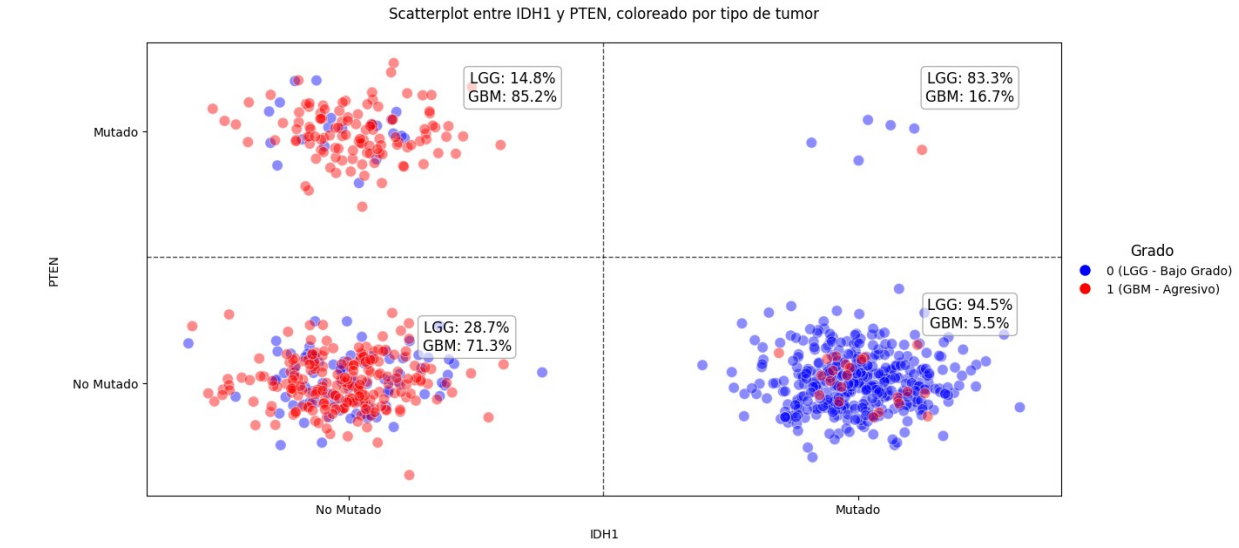
Teniendo en cuenta lo mencionado anteriormente, seleccionamos las mutaciones de los genes IDH1 y PTEN como atributos, ya que creemos que tienen un buen potencial predictivo para la clasificación del tipo de tumor. Para visualizar estas relaciones, creamos un gráfico de dispersión (scatterplot) en el que el gen IDH1 se representa en el eje de las abscisas y el gen PTEN en el eje

de las ordenadas. Además, coloreamos los puntos según el tipo de tumor. Dado que ambos atributos son binarios, para mejorar la visualización de la densidad de puntos en cada área del gráfico, añadimos un poco de ruido a los datos.

```
# Creamos la columna 'IDH1_with_jitter' para agregar ruido a cada dato
del gen IDH1:
data['IDH1_with_jitter']=data['IDH1'] +
0.1*np.random.normal(size=len(data))

# Creamos la columna 'PTEN_with_jitter' para agregar ruido a cada dato
del gen PTEN:
data['PTEN_with_jitter']=data['PTEN'] +
0.1*np.random.normal(size=len(data))

# Figura y ajuste del tamaño
fig, ax = plt.subplots(figsize=(14, 8))

# Paleta de colores
palette = {0: 'blue', 1: 'red'}

# Scatter con Jitter
sns.scatterplot(
    data=data,
    x='IDH1_with_jitter',
    y='PTEN_with_jitter',
    hue='Grade',
    palette=palette,
    s=80,
    alpha=0.45,
    hue_order=[0, 1],
    ax=ax
)

# Etiquetas (Ejes y título)
ax.set_title('Scatterplot entre IDH1 y PTEN, coloreado por tipo de
tumor', pad=20)
ax.set_xlabel('IDH1', labelpad=10)
ax.set_ylabel('PTEN', labelpad=10)

# Configuración de las etiquetas
ax.set_xticks([0, 1])
ax.set_xticklabels(['No Mutado', 'Mutado'])
ax.set_yticks([0, 1])
ax.set_yticklabels(['No Mutado', 'Mutado'])

# Líneas divisorias por región
ax.axhline(y=0.5, color='black', linestyle='--', linewidth=1,
alpha=0.7)
ax.axvline(x=0.5, color='black', linestyle='--', linewidth=1,
alpha=0.7)
```

```
# Leyenda
from matplotlib.lines import Line2D
legend_elements = [
    Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',
markersize=10, label='0 (LGG - Bajo Grado)'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='red',
markersize=10, label='1 (GBM - Agresivo)')
]
ax.legend(
    handles=legend_elements,
    title='Grado',
    title_fontsize=12,
    loc='center left',
    bbox_to_anchor=(1, 0.5),
    frameon=False
)

# Cálculo de la tabla de proporciones con índices ajustados para los
cuadrantes
tabla_proporciones = (
    data.groupby(['IDH1', 'PTEN'])['Grade']
    .value_counts(normalize=True).unstack() * 100
)
tabla_proporciones = tabla_proporciones.round(1).astype(str) + '%'

# Mapeo de nombres de filas asegurado que corresponde con los
cuadrantes
nombres_filas = {
    (0, 0): "IDH1 y PTEN No Mutados",
    (0, 1): "IDH1 No Mutado y PTEN Mutado",
    (1, 0): "IDH1 Mutado y PTEN No Mutado",
    (1, 1): "IDH1 y PTEN Mutados"
}
tabla_proporciones.index = [nombres_filas[idx] for idx in
tabla_proporciones.index]

# Coordenadas de etiquetas con porcentajes alineadas con los
cuadrantes correctos
coordenadas = [(0.35, 0.35), (0.4, 0.9), (0.9, 0.4), (0.9, 0.9)]
for (x, y), texto in zip(coordenadas, tabla_proporciones.values):
    ax.text(
        x, y, f"LGG: {texto[0]}\nGBM: {texto[1]}",
        ha='center', va='center', fontsize=12,
        transform=ax.transAxes,  # coordenadas relativas al gráfico
        bbox=dict(boxstyle="round,pad=0.3", edgecolor='gray',
facecolor='white', alpha=0.6)
    )

# Tabla inferior ajustada para coincidir con los valores correctos
```

```
tabla_ax = plt.gca().table(
    cellText=tabla_proporciones.values,
    colLabels=['LGG (Bajo Grado)', 'GBM (Agresivo)'],
    rowLabels=tabla_proporciones.index,
    cellLoc='center',
    loc='bottom',
    bbox=[0.1, -0.5, 0.9, 0.3]
)

# Fuente de la tabla
tabla_ax.auto_set_font_size(False)
tabla_ax.set_fontsize(10)

# Espacio gráfico/tabla
plt.subplots_adjust(left=0.1, right=0.85, top=0.9, bottom=0.25)

# Gráfico final
plt.show()
```



	LGG (Bajo Grado)	GBM (Agresivo)
IDH1 y PTEN No Mutados	28.7%	71.3%
IDH1 No Mutado y PTEN Mutado	14.8%	85.2%
IDH1 Mutado y PTEN No Mutado	94.5%	5.5%
IDH1 y PTEN Mutados	83.3%	16.7%

De la tabla obtenida, podemos interpretar que la principal separación en la variable objetivo (considerando los genes IDH1 y PTEN) está determinada principalmente por la mutación del gen IDH1. Por ejemplo, cuando IDH1 = 0 y PTEN = 0 (es decir, cuando ninguno de los dos genes está mutado), una proporción significativa (71%) de los tumores corresponde a glioblastoma multiforme (GBM). Esto sugiere que la ausencia de mutaciones en ambos genes podría estar asociada con una mayor probabilidad de tener un tumor más agresivo (GBM).

Por otro lado, cuando IDH1 = 1 y PTEN = 0 (es decir, cuando IDH1 está mutado y PTEN no), la proporción de gliomas de bajo grado (LGG) es muy alta (94%). Esto indica que la mutación en IDH1, sin la presencia de mutaciones en PTEN, está asociada con un pronóstico más favorable, reflejando una mayor probabilidad de tener un tumor menos agresivo (LGG).

Segunda parte - Primer modelo de Aprendizaje Automático

Introducción

En esta segunda parte del trabajo, abordaremos la construcción de los primeros modelos de Aprendizaje Automático. El objetivo es comparar diferentes modelos en función de su desempeño y extraer conclusiones que nos permitan seleccionar el más adecuado para el problema planteado.

Dado que tanto los atributos seleccionados como la variable objetivo son binarios y no hay valores faltantes en los datos, no fue necesario aplicar técnicas adicionales de preprocesamiento.

Modelo de referencia (Benchmark)

Nuestro enfoque comenzará con un modelo de referencia (benchmark) que generará predicciones manteniendo la distribución de clases del conjunto de entrenamiento. Este modelo servirá como punto de comparación para evaluar la efectividad de modelos más complejos.

```
#definición de X e y
X = data.drop('Grade', axis=1)
y = data['Grade']  # Extraemos la variable objetivo

#division del dataset en conjunto de entrenamiento [70%] y prueba
[30%]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

#modelo dummy
dummy_clf = DummyClassifier(strategy="stratified")

#entrenamiento del modelo
dummy_clf.fit(X_train, y_train)

#predicicones
y_pred = dummy_clf.predict(X_test)

#evaluacion
accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred, zero_division=1)
precision = precision_score(y_test, y_pred, zero_division=1)
report = classification_report(y_test, y_pred, zero_division=1)
```

```
#resultados
print(f"Exactitud (Accuracy): {round(accuracy,2)}")
print(f"Exhaustividad (Recall): {round(recall,2)}")
print(f"Precisión (Precision): {round(precision,2)}")
print("Reporte de clasificación:\n", report)
```

```
Exactitud (Accuracy): 0.5
Exhaustividad (Recall): 0.33
Precisión (Precision): 0.37
Reporte de clasificación:
               precision    recall  f1-score   support

           0       0.57      0.61      0.59       150
           1       0.37      0.33      0.35       102

    accuracy                           0.50       252
   macro avg       0.47      0.47      0.47       252
weighted avg       0.49      0.50      0.50       252
```

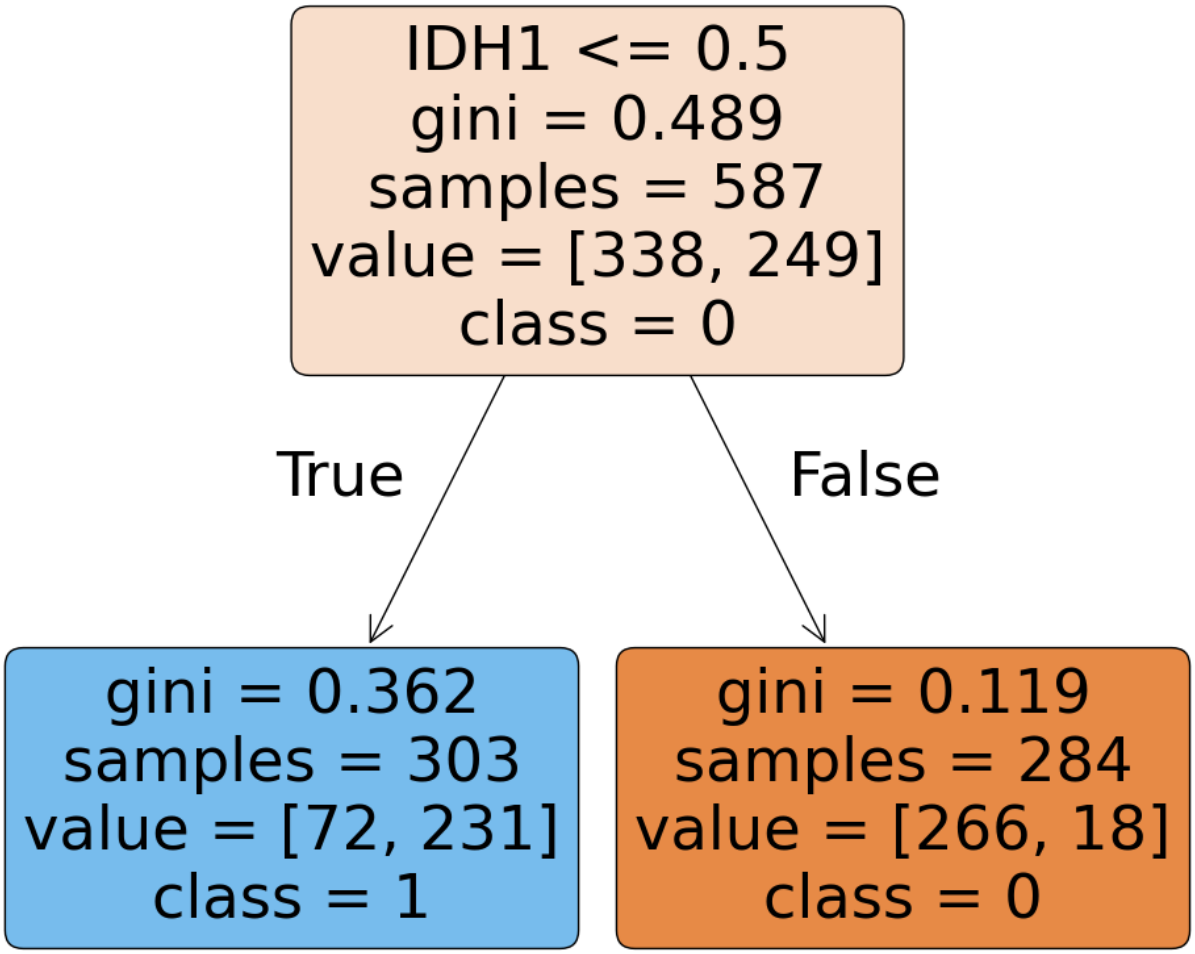
Este modelo de referencia mantiene la distribución original de las clases y logra una exactitud de 0.52, con una precisión de 0.41 y un recall de 0.43 para la Clase 1 (GBM). Considerando estas métricas, su capacidad para detectar tumores agresivos resulta insuficiente como modelo definitivo en un contexto médico. Sin embargo, proporciona un punto de referencia útil para comparar con modelos más complejos en las próximas etapas.

Entrenamiento de un primer modelo de ML

Para continuar, entrenaremos un árbol de decisión utilizando la mutación del gen IDH1 como único atributo, ya que hemos observado previamente que podría tener un buen potencial predictivo para la clasificación del tipo de tumor. Este modelo será utilizado también como benchmark.

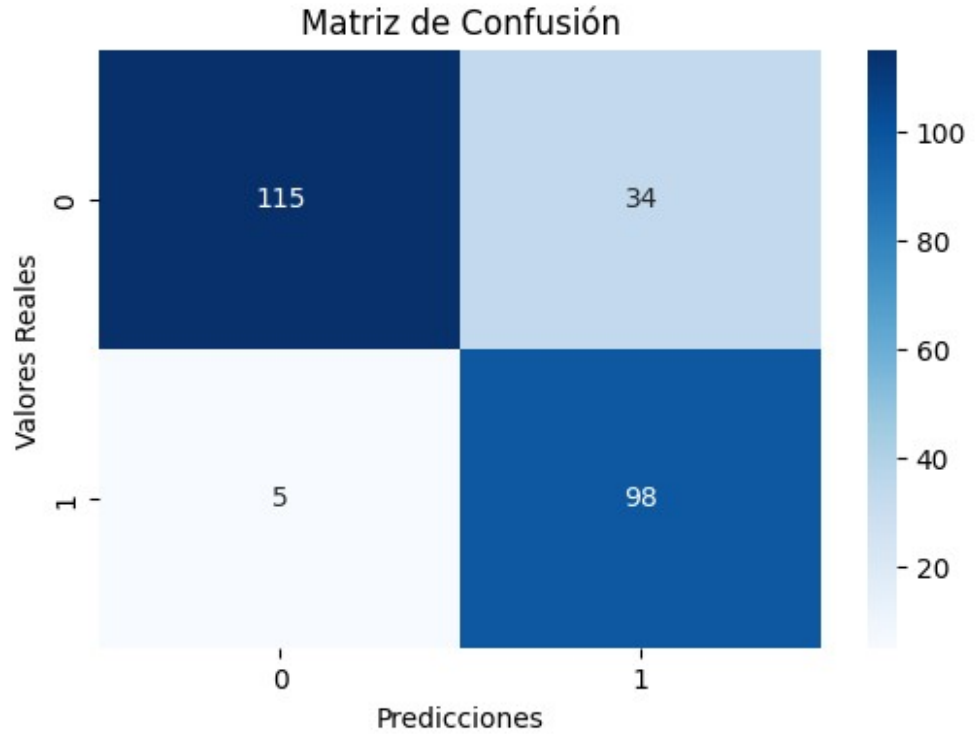
```
#separamos atributo a utilizar y target, ambos compatibles con el
análisis exploratorio previo
X1 = data[['IDH1']]
y = data['Grade']

#División del dataset en conjunto de entrenamiento [70%] y prueba
[30%]
X_train1, X_test1, y_train, y_test1 = train_test_split(X1, y,
test_size=0.3, random_state=100)

#creamos y entrenamos el modelo de Árbol de Decisión
tree_model1 = DecisionTreeClassifier(max_depth=1, random_state=100)
tree_model1.fit(X_train1, y_train)
y_pred1 = tree_model1.predict(X_test1)
```

```
#visualizamos el árbol

plt.figure(figsize=(12,12))
plot_tree(tree_model1, feature_names=X1.columns, class_names=[str(cls)
for cls in tree_model1.classes_],
          filled=True, impurity=True, rounded=True)
plt.show()
```




```
#creamos y visualizamos la matriz de confusión y las métricas de
evaluación
conf_matrix1 = confusion_matrix(y_test1, y_pred1)
plt.figure(figsize=(6, 4))
```

```
sns.heatmap(conf_matrix1, annot=True, fmt='d', cmap='Blues')
plt.title('Matriz de Confusión')
plt.xlabel('Predicciones')
plt.ylabel('Valores Reales')
plt.show()

accuracy1=accuracy_score(y_test1, y_pred1)
recall1=recall_score(y_test1, y_pred1)
precision1=precision_score(y_test1, y_pred1)

print("Exactitud (accuracy) del modelo:", round(accuracy1, 2))
print("Exhaustividad (recall) del modelo:", round(recall1, 2))
print("Precisión del modelo:", round(precision1, 2))
print("\nReporte de clasificación:")
print(classification_report(y_test1, y_pred1))
```



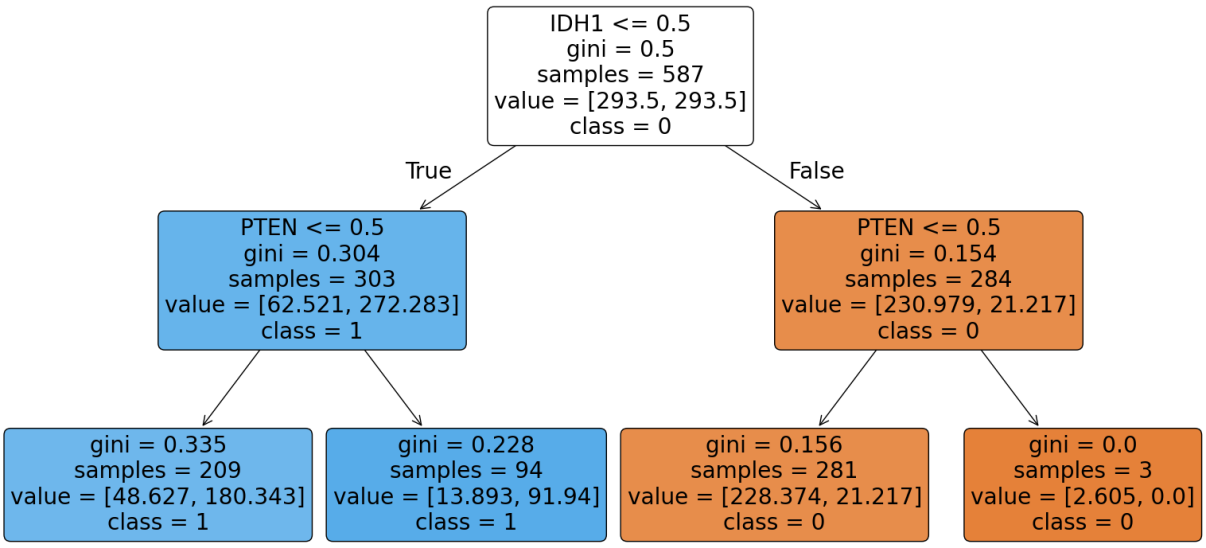

```
Exactitud (accuracy) del modelo: 0.85
Exhaustividad (recall) del modelo: 0.95
Precisión del modelo: 0.74

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.96      0.77      0.86       149
           1       0.74      0.95      0.83       103
```

```
    accuracy                           0.85       252
   macro avg       0.85      0.86      0.84       252
weighted avg       0.87      0.85      0.85       252
```

Con este modelo inicial sencillo se obtuvo un recall alto (95%), lo cual indica que la presencia o ausencia de la mutación en el gen IDH1 es efectiva para identificar la mayoría de los casos de GBM. Sin embargo, la precisión del modelo es moderada (74%), lo que sugiere la existencia de varios falsos positivos (casos identificados como GBM que en realidad no lo son). Aunque logramos nuestro objetivo de minimizar los falsos negativos, la presencia de falsos positivos podría llevar a tratamientos innecesarios. Por esta razón, aunque ya hemos cumplido con lo requerido en la consigna, exploraremos otros modelos similares en busca de un mejor equilibrio entre recall y precisión, con el objetivo de reducir los falsos positivos sin perder la capacidad de identificar casos críticos.

Agregamos al árbol de decisión otro atributo: la mutación del gen PTEN, dado que observamos previamente en la matriz de correlación que también podría tener un buen poder predictivo.

```
#Separamos atributo a utilizar en este nuevo modelo:
X2 = data[['IDH1','PTEN']]

#Dividimos el dataset en conjunto de entrenamiento [70%] y prueba
[30%]
X_train2, X_test2, y_train, y_test2 = train_test_split(X2, y,
test_size=0.3, random_state=100)

#Creamos y entrenamos el modelo de Árbol de Decisión
tree_model2 = DecisionTreeClassifier(max_depth=2,
class_weight='balanced', random_state=100)
tree_model2.fit(X_train2, y_train)
y_pred2 = tree_model2.predict(X_test2)

#Visualizamos el árbol

plt.figure(figsize=(20,10))
plot_tree(tree_model2, feature_names=X2.columns, class_names=[str(cls)
for cls in tree_model2.classes_],
          filled=True, impurity=True, rounded=True)
plt.show()

accuracy2=accuracy_score(y_test2, y_pred2)
recall2=recall_score(y_test2, y_pred2)
precision2=precision_score(y_test2, y_pred2)

print("Exactitud (accuracy) del modelo:", round(accuracy2, 2))
print("Exhaustividad (recall) del modelo:", round(recall2, 2))
print("Precisión del modelo:", round(precision2, 2))
print("\nReporte de clasificación:")
print(classification_report(y_test2, y_pred2))
```

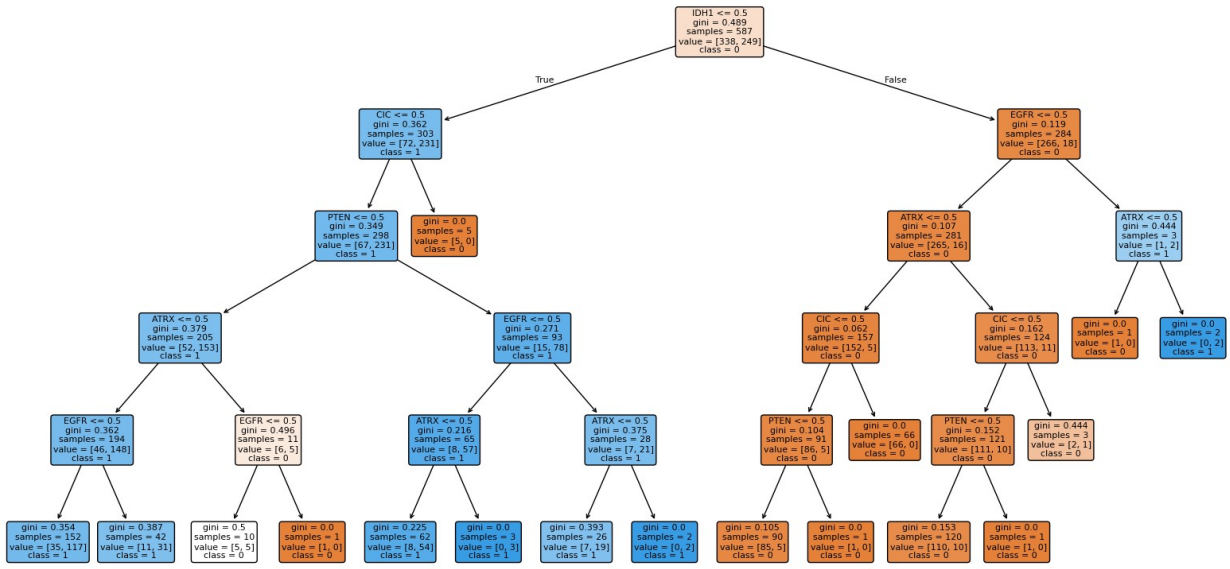



```
Exactitud (accuracy) del modelo: 0.85
Exhaustividad (recall) del modelo: 0.95
Precisión del modelo: 0.74

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.96      0.77      0.86       149
           1       0.74      0.95      0.83       103

    accuracy                           0.85       252
   macro avg       0.85      0.86      0.84       252
weighted avg       0.87      0.85      0.85       252
```

Observamos que, al incluir este atributo, las tres métricas se mantuvieron sin cambios significativos en comparación con el modelo basado únicamente en la mutación del gen IDH1, lo que indica que no mejoró el desempeño del modelo. Luego, probamos añadiendo, una a una, las mutaciones de los otros tres genes con mayor correlación con el target en la matriz (ATRX, EGFR y CIC). Sin embargo, comprobamos que las métricas tampoco mejoraron. Por este motivo, en el próximo modelo decidimos probar con todos estos atributos combinados.

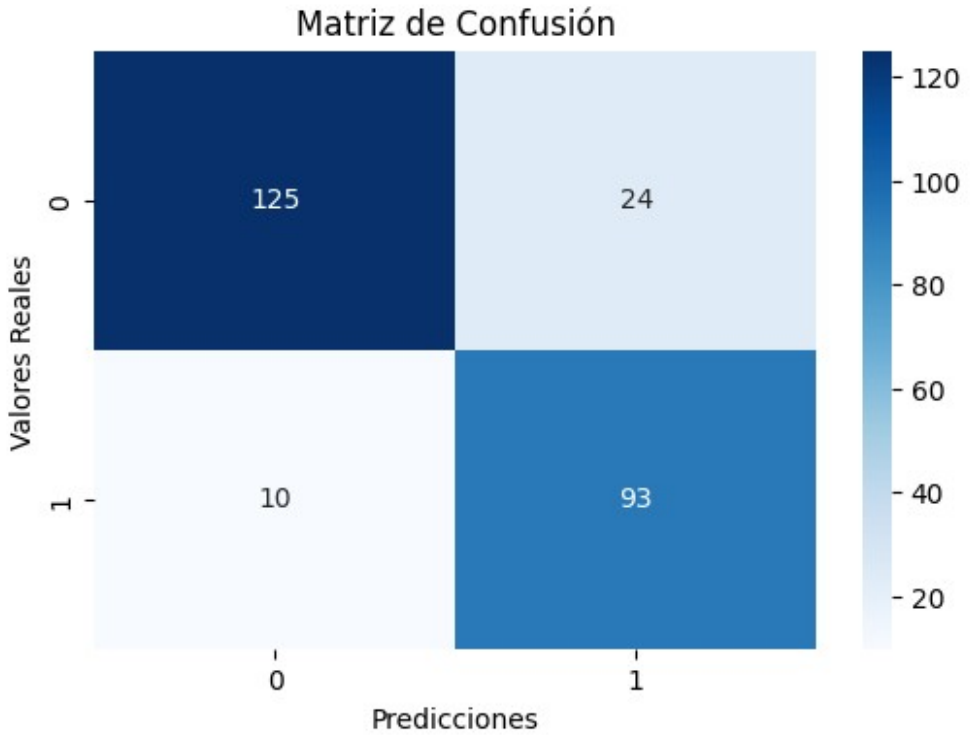
```
#separamos atributo a utilizar en este nuevo modelo:
X3 = data[['IDH1','PTEN','ATRX','EGFR','CIC']]

#dividimos el dataset en conjunto de entrenamiento [70%] y prueba
[30%]
X_train3, X_test3, y_train, y_test3 = train_test_split(X3, y,
test_size=0.3, random_state=100)
```

```
#creamos y entrenamos el modelo de Árbol de Decisión
tree_model3 = DecisionTreeClassifier(max_depth=10, random_state=100)
tree_model3.fit(X_train3, y_train)
y_pred3 = tree_model3.predict(X_test3)

#visualizamos el árbol

plt.figure(figsize=(20,10))
plot_tree(tree_model3, feature_names=X3.columns, class_names=[str(cls)
for cls in tree_model3.classes_],
          filled=True, impurity=True, rounded=True)
plt.show()
```




```
#Creamos y visualizamos la matriz de confusión y las métricas de
evaluación:
conf_matrix3 = confusion_matrix(y_test3, y_pred3)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix3, annot=True, fmt='d', cmap='Blues')
plt.title('Matriz de Confusión')
plt.xlabel('Predicciones')
plt.ylabel('Valores Reales')
plt.show()

accuracy3=accuracy_score(y_test3, y_pred3)
recall3=recall_score(y_test3, y_pred3)
precision3=precision_score(y_test3, y_pred3)

print("Exactitud (accuracy) del modelo:", round(accuracy3, 2))
print("Exhaustividad (recall) del modelo:", round(recall3, 2))
print("Precisión del modelo:", round(precision3, 2))
```

```
print("\nReporte de clasificación:")
print(classification_report(y_test3, y_pred3))
```

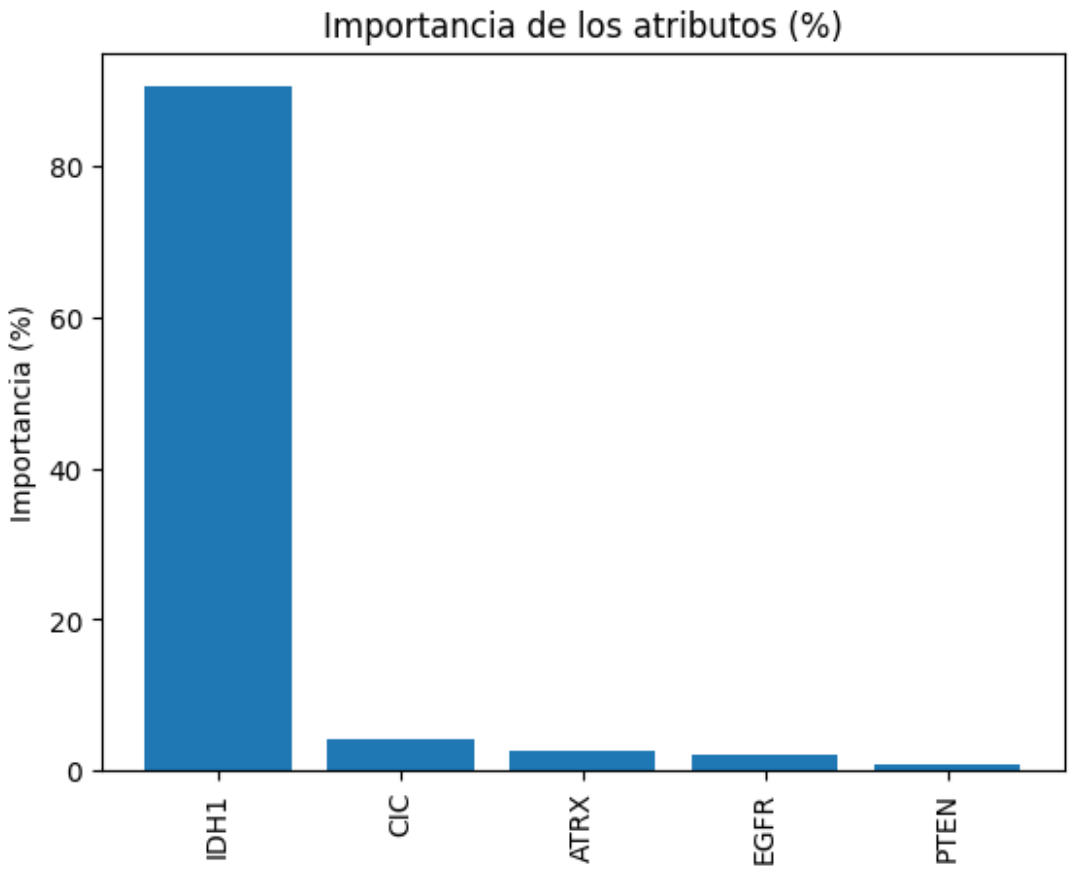



```
Exactitud (accuracy) del modelo: 0.87
Exhaustividad (recall) del modelo: 0.9
Precisión del modelo: 0.79

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.93      0.84      0.88       149
           1       0.79      0.90      0.85       103

    accuracy                           0.87       252
   macro avg       0.86      0.87      0.86       252
weighted avg       0.87      0.87      0.87       252
```

Al agregar los nuevos atributos, se observa una ligera mejora en la precisión (79%) y la exactitud (87%), aunque a costa de una leve disminución en el recall, que baja a 90%. Dado que en el contexto médico es crucial minimizar los falsos negativos para evitar no detectar tumores agresivos (GBM), el modelo basado exclusivamente en la mutación del gen IDH1 sigue siendo la opción más adecuada. Aunque la inclusión de más atributos mejora ligeramente algunas métricas, no ofrece un beneficio significativo para cumplir con nuestro principal objetivo clínico: detectar con alta sensibilidad los casos de GBM.

```python
# Gráfico de barras con la importancia de cada atributo
importances = tree_model3.feature_importances_
indices = np.argsort(importances)[::-1]

# Calcular los porcentajes de importancia de cada atributo
porcentajes = 100 * importances / importances.sum()

# Ordenar los porcentajes en el mismo orden que los índices
porcentajes = porcentajes[indices]

feature_importance_df = pd.DataFrame({
    "Atributo": np.array(X3.columns)[indices],
    "Importancia (%)": porcentajes.round(2)
})

# Mostrar la tabla
print(feature_importance_df)

plt.figure()
plt.title("Importancia de los atributos (%)")
plt.bar(range(X3.shape[1]), porcentajes, align="center")
plt.xticks(range(X3.shape[1]), np.array(X3.columns)[indices],
rotation=90)
plt.ylabel("Importancia (%)")
plt.show()
```

```
  Atributo  Importancia (%)
0     IDH1            90.46
1      CIC             4.11
2     ATRX             2.61
3     EGFR             2.12
4     PTEN             0.70
```



Al graficar la importancia de los atributos, verificamos, tal como habíamos observado en los modelos, que la mutación del gen IDH1 es el principal determinante del tipo de tumor. Por lo tanto, en principio, no parecería justificarse la inclusión de otros atributos.

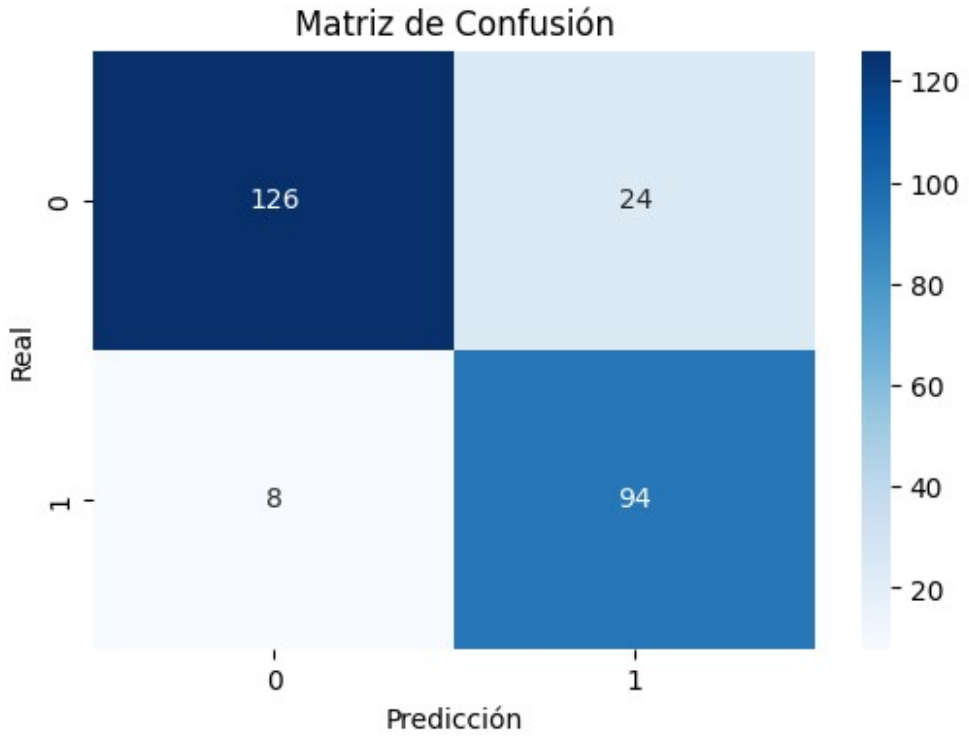
Entrenamiento de un segundo modelo de ML

Probamos cambiando el método y entrenamos un modelo de Random Forest. En este caso, agregaremos las mutaciones de todos los genes para aprovechar la capacidad de este método de capturar la variabilidad en múltiples atributos.

```
# Definimos los genes utilizados
genes = [
    "IDH1", "TP53", "ATRX", "PTEN", "EGFR", "CIC", "MUC16", "PIK3CA",
    "NF1", "PIK3R1", "FUBP1", "RB1", "NOTCH1", "BCOR", "CSMD3",
    "SMARCA4", "GRIN2A", "IDH2", "FAT4", "PDGFRA"
]

# Definimos X e y
X = data[genes]
y = data["Grade"]

# División en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.3, random_state=42)

# Definimos el modelo Random Forest
rf = RandomForestClassifier(n_estimators=100, max_depth=10,
random_state=42)

# Cross-validation con 'accuracy'
cv_accuracy = cross_val_score(rf, X_train, y_train, cv=5,
scoring='accuracy')
print('Cross-validation accuracy scores:', cv_accuracy.round(2))
print('Promedio (accuracy): ', cv_accuracy.mean().round(2))
print('Std (accuracy): ', cv_accuracy.std().round(2))

# Cross-validation con 'recall'
cv_recall = cross_val_score(rf, X_train, y_train, cv=5,
scoring='recall')
print('Cross-validation recall scores:', cv_recall.round(2))
print('Promedio (recall): ', cv_recall.mean().round(2))
print('Std (recall): ', cv_recall.std().round(2))

# Cross-validation con 'precision'
cv_precision = cross_val_score(rf, X_train, y_train, cv=5,
scoring='precision')
print('Cross-validation precision scores:', cv_precision.round(2))
print('Promedio (precision): ', cv_precision.mean().round(2))
print('Std (precision): ', cv_precision.std().round(2))

# Entrenamiento del modelo y predicciones
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

# Matriz de confusión
conf_matrix = confusion_matrix(y_test, y_pred)

# Visualizamos la matriz de confusión
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión')
plt.show()

Cross-validation accuracy scores: [0.86 0.86 0.85 0.88 0.85]
Promedio (accuracy):  0.86
Std (accuracy):  0.01
Cross-validation recall scores: [0.92 0.94 0.9  0.9  0.92]
Promedio (recall):  0.92
Std (recall):  0.01
Cross-validation precision scores: [0.79 0.78 0.78 0.83 0.77]
```

```
Promedio (precision):  0.79
Std (precision):  0.02
```



El modelo de Random Forest muestra una ligera mejora en la exactitud (86%) y la precisión (79%), con desviaciones estándar de 0.01 y 0.02 respectivamente, en comparación con el árbol de decisión. Esto sugiere un mayor número de predicciones correctas en general, así como una reducción en los falsos positivos.

Además, aunque se observa una ligera variabilidad entre los folds, la baja desviación estándar indica un rendimiento consistente, lo que refleja una buena capacidad del modelo para generalizar a datos no vistos.

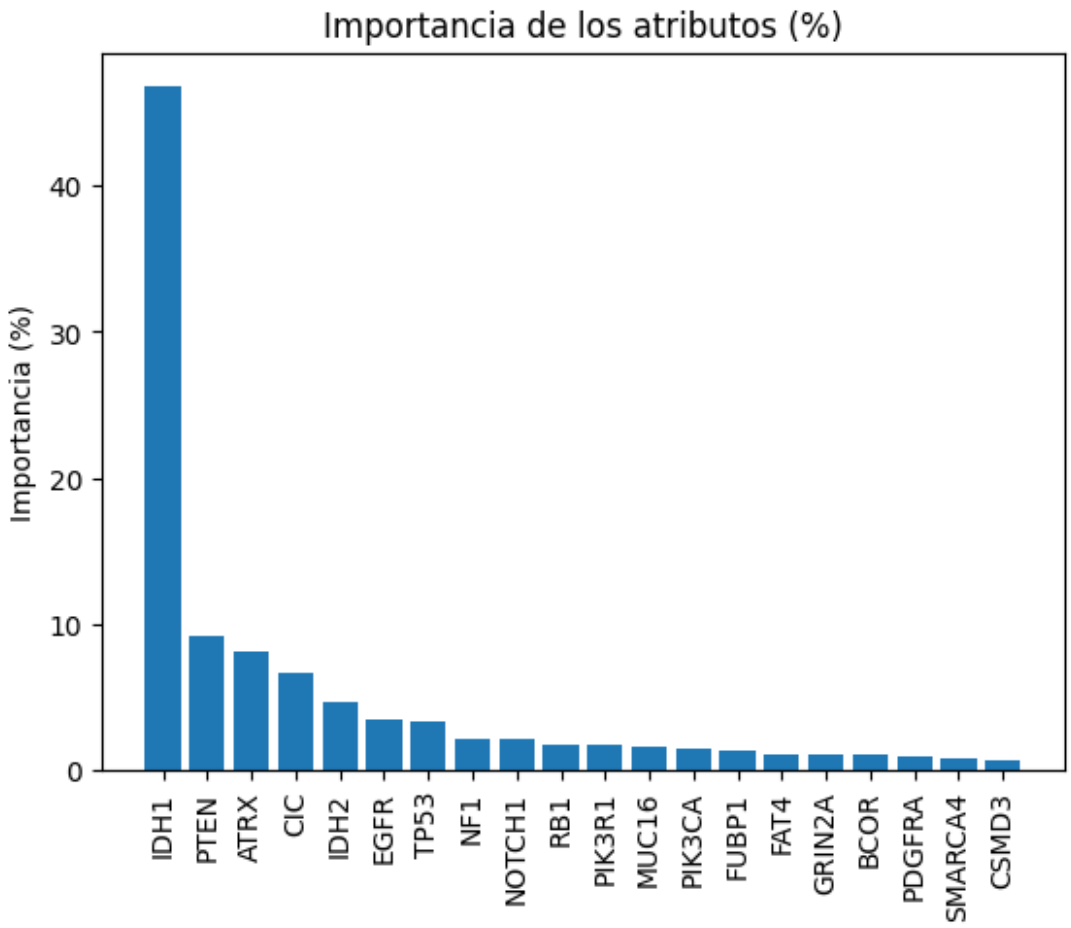
Sin embargo, el modelo de árbol de decisión utilizando el atributo IDH1 mantiene un recall más alto (95%), lo cual es crucial dado que el principal objetivo es minimizar los falsos negativos y no omitir casos de glioblastoma multiforme (GBM).

Por lo tanto, aunque el modelo de Random Forest ofrece una predicción más balanceada, el modelo basado en el árbol de decisión de profundidad 1 sigue siendo preferible debido a su superior capacidad para detectar casos críticos. De todos modos, se continuará explorando alternativas que combinen ambos enfoques para lograr un mejor equilibrio entre precisión y recall.

```
# Obtener la importancia de cada atributo en el modelo Random Forest
importances = rf.feature_importances_
indices = np.argsort(importances)[::-1]  # Ordenar los índices según la importancia de mayor a menor
```

```python
# Calcular los porcentajes de importancia de cada atributo
porcentajes_rf = 100 * importances / importances.sum()  # Convertir la importancia en porcentaje

feature_importance_df = pd.DataFrame({
    "Atributo": np.array(genes)[indices],  # Usar 'genes' para mostrar el nombre de los atributos
    "Importancia (%)": porcentajes_rf[indices].round(2)  # Mostrar los porcentajes de importancia redondeados a 2 decimales
})

# Mostrar la tabla con los atributos y su importancia
print(feature_importance_df)

# Crear el gráfico de barras para visualizar la importancia de los atributos
plt.figure()
plt.title("Importancia de los atributos (%)")
plt.bar(range(len(genes)), porcentajes_rf[indices], align="center")  # Crear las barras con los porcentajes de importancia
plt.xticks(range(len(genes)), np.array(genes)[indices], rotation=90)
# Etiquetas de los atributos en el eje x
plt.ylabel("Importancia (%)")  # Etiqueta para el eje y
plt.show()  # Mostrar el gráfico
```

	Atributo	Importancia (%)
0	IDH1	46.75
1	PTEN	9.24
2	ATRX	8.11
3	CIC	6.59
4	IDH2	4.70
5	EGFR	3.45
6	TP53	3.30
7	NF1	2.09
8	NOTCH1	2.09
9	RB1	1.75
10	PIK3R1	1.72
11	MUC16	1.60
12	PIK3CA	1.51
13	FUBP1	1.37
14	FAT4	1.07
15	GRIN2A	1.04
16	BCOR	1.04
17	PDGFRA	1.02
18	SMARCA4	0.83
19	CSMD3	0.74



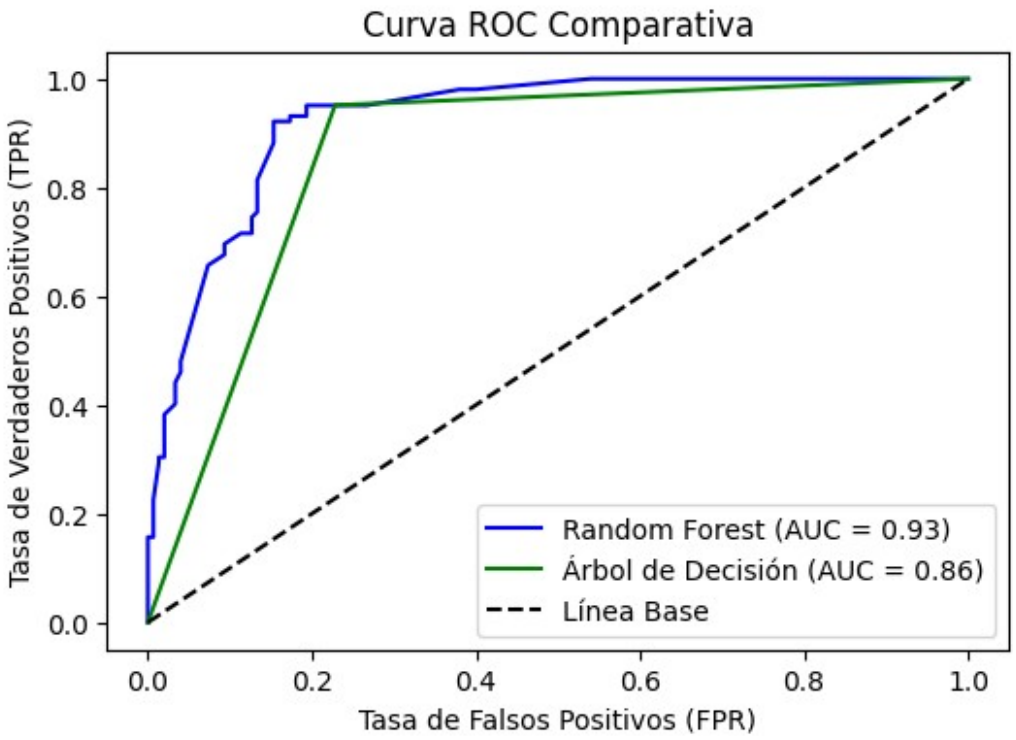
Al graficar la importancia de los atributos para este modelo, corroboramos lo concluido con el árbol de decisión respecto a la relevancia del gen IDH1.

Analisis y comparativa de los modelos

Para finalizar nuestro análisis y comparar en profundidad ambos modelos (el árbol de decisión con la mutación del gen IDH1 como único atributo y el modelo Random Forest), graficaremos sus respectivas curvas ROC y evaluaremos el área bajo cada una de ellas.

```
rf_probs = rf.predict_proba(X_test)[:, 1]

tree_probs = tree_model1.predict_proba(X_test1)[:, 1]

#curva roc -> random forest
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)
rf_auc = auc(rf_fpr, rf_tpr)

#curva roc -> arbol
tree_fpr, tree_tpr, _ = roc_curve(y_test1, tree_probs)
tree_auc = auc(tree_fpr, tree_tpr)
```

```
#grafica de curvas
plt.figure(figsize=(6, 4))
plt.plot(rf_fpr, rf_tpr, label=f'Random Forest (AUC = {rf_auc:.2f})',
color='blue')
plt.plot(tree_fpr, tree_tpr, label=f'Árbol de Decisión (AUC =
{tree_auc:.2f})', color='green')

#linea diagonal
plt.plot([0, 1], [0, 1], linestyle='--', color='black', label='Línea
Base')


plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva ROC Comparativa')
plt.legend(loc='lower right')
plt.show()
```



Al comparar los modelos mediante sus respectivas curvas ROC, observamos que el modelo de Random Forest presenta un AUC mayor (0.93) en comparación con el Árbol de Decisión (0.86). Esto indica que el Random Forest tiene un mejor rendimiento global, ya que es más efectivo para discriminar entre las clases en diferentes umbrales.

Sin embargo, dado el contexto clínico, en el cual es crucial minimizar los falsos negativos, el Árbol de Decisión sigue siendo la opción preferible. Diagnosticar incorrectamente un tumor agresivo (GBM) como de bajo grado (LGG) podría comprometer el tratamiento adecuado del

paciente. Por esta razón, priorizamos el modelo con mayor recall, que en este caso es el Árbol de Decisión, a pesar de que el Random Forest mostró un desempeño más equilibrado en general.

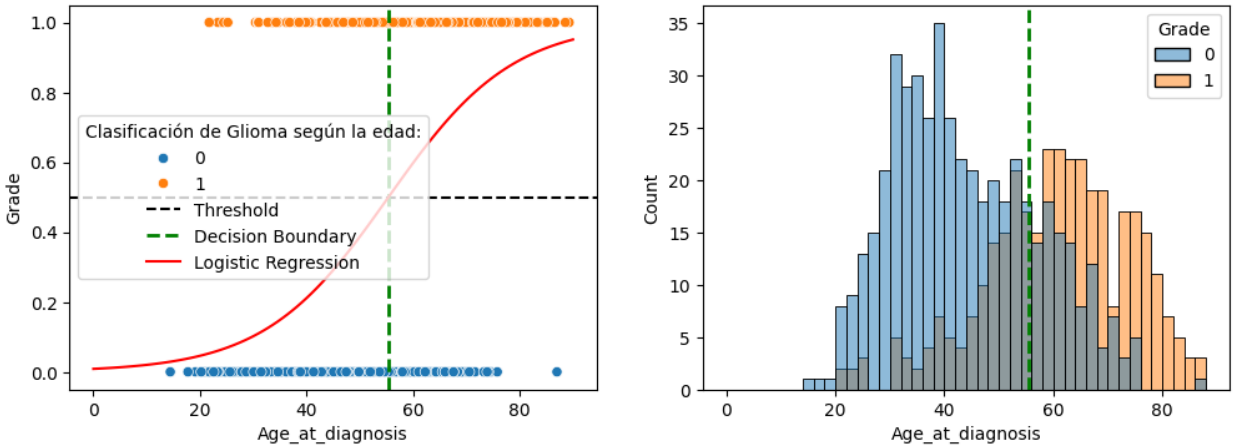
Para finalizar nuestro trabajo, también probamos un modelo de Regresión Logística, utilizando el único atributo continuo disponible: la edad del paciente al momento del diagnóstico.

```
# Definimos X e y
X = data[['Age_at_diagnosis']]  # Usamos solo la columna
'Age_at_diagnosis' como predictor
y = data['Grade']

# División en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Definimos el modelo de regresión logística
log_reg = LogisticRegression(random_state=42)

# Entrenamos el modelo
log_reg.fit(X_train, y_train)

# Hacemos predicciones en el conjunto de prueba
y_pred = log_reg.predict(X_test)

# Calculamos y mostramos la precisión
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', round(accuracy, 2))

#Calculamos y mostramos el recall
recall = recall_score(y_test, y_pred, zero_division=1)
print('Recall:', round(recall, 2))

#Calculamos y mostramos la presición
precision = precision_score(y_test, y_pred, zero_division=1)
print('Precision:', round(precision, 2))

# Mostramos el reporte de clasificación
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.73
Recall: 0.64
Precision: 0.67
              precision    recall  f1-score   support

           0       0.76      0.79      0.77       150
           1       0.67      0.64      0.65       102

    accuracy                           0.73       252
   macro avg       0.72      0.71      0.71       252
weighted avg       0.72      0.73      0.72       252
```

```
fig, axs = plt.subplots(1, 2, figsize=(12, 4))

sns.scatterplot(x='Age_at_diagnosis', y='Grade', hue='Grade',
                data=data.drop(23), ax=axs[0])

x_values = np.linspace(0, 90, 100)
y_values = log_reg.predict_proba(x_values.reshape(-1, 1))[:, 1]

axs[0].axhline(0.5, color='k', linestyle='--', label='Threshold')
axs[0].axvline(-log_reg.intercept_/log_reg.coef_,
            color='g', linestyle='--', label='Decision Boundary',
lw=2)

axs[0].plot(x_values, y_values, color='r', label='Logistic
Regression')
axs[0].legend(title='Clasificación de Glioma según la edad:')


sns.histplot(x='Age_at_diagnosis', hue='Grade', bins=np.arange(0, 90,
2),
                data=data, ax=axs[1])

axs[1].axvline(-log_reg.intercept_/log_reg.coef_, color='g',
linestyle='--',lw=2,  label='Decision Boundary')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493:
UserWarning: X does not have valid feature names, but
LogisticRegression was fitted with feature names
  warnings.warn(

<matplotlib.lines.Line2D at 0x7b6fdd19c310>
```



Las métricas obtenidas con este modelo fueron significativamente menores que las de los modelos basados en árboles de decisión y Random Forest, por lo que decidimos descartarlo.

Próximos pasos

Para continuar con nuestro trabajo, nos gustaría repetir el análisis excluyendo la mutación del gen IDH1. Aunque el Árbol de Decisión basado únicamente en IDH1 es el modelo que mejor se alinea con nuestros objetivos, entendemos que esta mutación podría no estar siempre disponible en el contexto clínico. Por ello, evaluaremos si las mutaciones en otros genes pueden predecir el grado de glioma por sí solas. Este enfoque nos permitirá comprender mejor la influencia de otras mutaciones en la clasificación y pronóstico, ampliando las opciones diagnósticas cuando IDH1 no esté disponible.

Conclusiones de la segunda parte

En este trabajo, hemos explorado diferentes modelos de machine learning para la clasificación de gliomas, comenzando con un árbol de decisión sencillo y avanzando hacia modelos más complejos como el Random Forest. También evaluamos la Regresión Logística. A lo largo del proceso, aprendimos también la importancia de equilibrar precisión y recall, especialmente en estos contextos clínicos donde entendemos que los falsos negativos representan un riesgo crítico.

Además, el uso de validación cruzada nos permitió evaluar la robustez y la capacidad de generalización de un modelos, asegurando que los resultados obtenidos no dependieran exclusivamente de una única partición del conjunto de todo el conjunto de datos.

Estos análisis nos brindaron una mayor confianza en las decisiones sobre qué modelo utilizar.

Finalmente, podemos decir que la experiencia resaltó la importancia de adaptar los modelos a las necesidades específicas de un problema. En este caso, algunos modelos utilizados que fueron más complejos, nos ofrecieron predicciones más balanceadas, pero priorizamos aquellos que maximizaran el recall para minimizar los riesgos clínicos.

Estasegunda parte práctica nos dejó como aprendizaje clave la necesidad de encontrar un balance que resulte adecuado entre la complejidad de un modelo y su rendimiento, manteniendo siempre el foco en el contexto y el objetivo final del análisis.