Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Inteligencia artificial avanzada para la ciencia de datos I
(Gpo 101)

Inteligencia artificial avanzada para la ciencia de datos I Concentración, (Gpo 101)

Jorge Adolfo Ramírez Uresti

Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Oswaldo Daniel Hernández de Luna | A01753911

11 de septiembre 2024

Dataset usado:

El dataset utilizado contiene información sobre características extraídas de imágenes digitalizadas de biopsias de tumores de mama, por lo que, estas características están asociadas con células nucleares y se utilizan para predecir si un tumor es maligno (M) o benigno (B), lo cual está representado en la columna diagnosis, asimismo, el dataset incluye un total de 569 instancias y 32 columnas con diferentes atributos.

Columnas más relevantes utilizadas:

- radius_mean: Media del radio del núcleo celular.
- texture_mean: Textura media del núcleo, basada en la desviación estándar de los valores de intensidad de los píxeles.
- perimeter_mean: Perímetro medio del núcleo celular.
- area_mean: Área media del núcleo celular.
- smoothness_mean: Uniformidad de los bordes del núcleo celular.
- compactness_mean: Relación entre el perímetro y el área.
- concavity_mean: Grado de concavidad del núcleo.
- concave points_mean: Número de puntos cóncavos en el contorno del núcleo.

Además de estas variables, el dataset incluye columnas adicionales con características estadísticas como las desviaciones estándar y los valores "peores" (worst) para cada característica.

Variable Objetivo:

- diagnosis: Esta variable clasifica los tumores como malignos (M) o benignos (B), siendo la variable objetivo para el modelo.

Este dataset es adecuado para aplicar algoritmos de clasificación, como árboles de decisión o K-Means, ya que contiene tanto datos numéricos como una clara división

entre dos clases, esto permitirá evaluar el rendimiento del modelo usando métricas como precisión, recall y F1-score.

2. Separación y Evaluación del Modelo (Train/Test/Validation)

En la construcción de modelos de aprendizaje automático, es fundamental dividir el dataset en distintos subconjuntos para garantizar que el modelo sea capaz de generalizar a datos no vistos previamente, este proceso de separación ayuda a evitar problemas como el sobreajuste (overfitting), donde el modelo puede aprenderse los datos de entrenamiento de memoria sin ser capaz de hacer predicciones acertadas con datos nuevos, para este proyecto, se realizaron las siguientes divisiones del dataset:

a) Conjunto de Entrenamiento (70% del total de los datos):

El conjunto de entrenamiento se compone del 70% del dataset original, este conjunto es crucial ya que es donde el modelo aprende a identificar patrones en los datos, aquí es donde se ajustan los parámetros del modelo, es importante que el modelo se entrene bien con este conjunto, pero también debe evitar sobreajustarse, es decir, aprender solo los detalles específicos de estos datos.

Tamaño del conjunto de entrenamiento: 398 instancias, con 30 características cada una.

Objetivo: Ajustar los parámetros del modelo de machine learning para que aprenda a clasificar correctamente los datos.

La normalización de los datos es un paso fundamental para garantizar que todas las características tengan un rango similar, evitando que aquellas con valores más grandes dominen al modelo.

```
==========================================
             DATASET SUMMARY
==========================================

TRAINING SET (70% of the data):
Size: (398, 30)
First 5 samples:
         0         1         2         3         4         5   ...        24        25        26        27        28        29
0  1.812780  1.982743  1.747740  1.888800 -0.339479  0.057973  ... -0.441366  0.138901  0.683223  0.634854 -0.750255 -0.036897
1  0.838611  1.829157  0.792127  0.785003  0.186433  0.126198  ...  1.211247 -0.062755 -0.039764  0.622673  0.176733  0.368747
2  0.378508  1.084495  0.487320  0.217320  1.561350  1.566511  ...  1.789881  1.542225  1.529986  1.548450  0.181587  1.260942
3 -0.288925 -0.867914 -0.196026 -0.354629  3.091407  1.367520  ...  2.359748  0.990056  1.753070  1.278939  0.398369  3.133996
4 -0.672344 -0.267531 -0.698958 -0.636479  0.236249 -0.856626  ... -0.818354 -1.110223 -1.012222 -0.654838 -1.492816 -0.819922

[5 rows x 30 columns]
------------------------------------------
```

b) Conjunto de Validación (15% del total de los datos):

El conjunto de validación tiene el 15% del dataset original y su propósito es evaluar el rendimiento del modelo durante el proceso de ajuste, antes de realizar predicciones finales, este conjunto se utiliza principalmente para ajustar los hiperparámetros del modelo y determinar si está sufriendo de underfitting.

Tamaño del conjunto de validación: 85 instancias, con 30 características cada una.

Objetivo: Optimización de los hiperparámetros y monitoreo del rendimiento del modelo sin comprometer el conjunto de prueba.

El conjunto de validación es fundamental para detectar problemas como el overfitting o underfitting, si el rendimiento en el conjunto de validación es muy inferior al del conjunto de entrenamiento, podría ser una señal de que el modelo está sobreajustado.

```
VALIDATION SET (15% of the data):
Size: (85, 30)
First 5 samples:
         0         1         2         3         4         5   ...        24        25        26        27        28        29
0  2.874993  0.211845  3.057588  3.145893  3.440117  3.455973  ...  1.632072  1.082296  1.478172  1.677876  0.519703 -0.213673
1 -0.678025 -1.070369 -0.644999 -0.650984 -1.093833 -0.146514  ... -1.098904  0.159257 -0.015297 -0.137134 -0.480086  0.822603
2 -0.675185  0.207191 -0.653649 -0.668618  0.892395  0.184948  ...  0.150419 -0.413905 -0.367435 -0.540791  0.432343 -0.225865
3  0.892574  1.426574  0.841556  0.779031 -0.928729  0.124303  ...  0.492339  1.004687  1.110202  0.902842 -0.648335 -0.249694
4 -0.501936 -0.174448 -0.533785 -0.535229 -0.824827 -0.685873  ... -0.612326 -0.368739 -0.376550 -0.459633  0.133053 -0.670300

[5 rows x 30 columns]
------------------------------------------------
```

c) Conjunto de Prueba (15% del total de los datos):

El conjunto de prueba contiene otro 15% del dataset y está completamente aislado del proceso de entrenamiento y validación, es utilizado solo al final, para obtener una medida objetiva del rendimiento real del modelo. es decir, permite evaluar cómo se desempeña el modelo con datos que no ha visto en ninguna etapa anterior.

Tamaño del conjunto de prueba: 86 instancias, con 30 características cada una.

Objetivo: Evaluación final del rendimiento del modelo en datos no vistos.

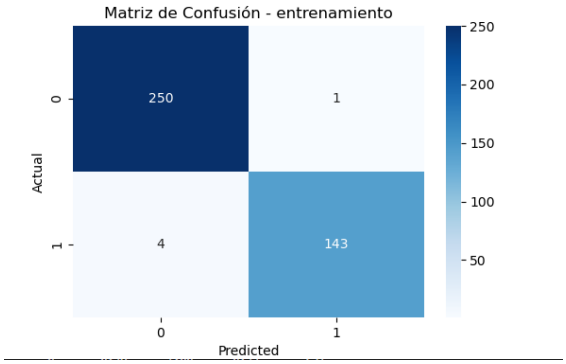
```
TEST SET (15% of the data):
Size: (86, 30)
First 5 samples:
          0         1         2         3         4         5   ...        24        25        26        27        28        29
0  0.034851  0.665623  0.183336 -0.026135  0.607733  1.828041  ...  1.386591  2.356484  2.015015  0.972885 -0.091818  1.621144
1 -0.939318  1.144999 -0.950630 -0.834144 -1.027649 -0.726239  ... -0.524654 -0.578665 -1.008672 -1.248067  0.256005 -0.425916
2 -0.021952  1.829157 -0.024262 -0.154973  0.208495  0.156521  ... -0.362461 -0.177261 -0.669679 -0.149315 -1.052780 -0.040776
3  0.145616 -0.567723  0.092306  0.031601 -0.708116 -0.708046  ... -0.796437 -0.300672 -0.136674 -0.553125 -0.721135 -0.996699
4  0.131416  0.788958  0.182100  0.006288 -0.827674  0.543131  ... -1.304933  0.399718  0.451022 -0.062524 -1.039838 -0.216444

[5 rows x 30 columns]
------------------------------------------------
```

Evaluación del Modelo:

Para evaluar el rendimiento del modelo, se utilizó la matriz de confusión en los tres conjuntos de datos, esta herramienta permite visualizar las predicciones del modelo y compararlas con las etiquetas reales, identificando cuántas predicciones fueron correctas y cuántas incorrectas.

- Matriz de Confusión - Entrenamiento:



Verdaderos Positivos (Clase 0): 250

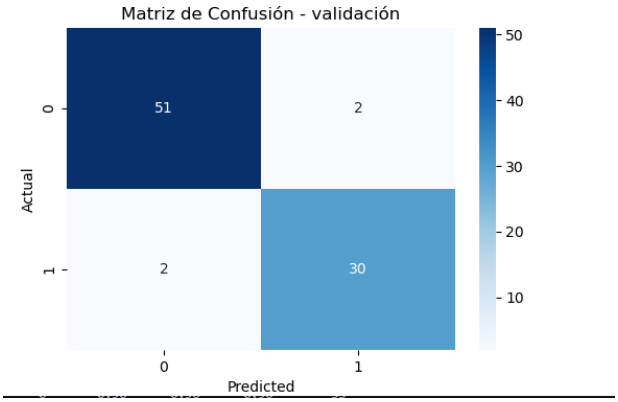
Verdaderos Negativos (Clase 1): 143

Falsos Positivos (Clase 1 predicho como 0): 1

Falsos Negativos (Clase 0 predicho como 1): 4

La matriz de confusión del conjunto de entrenamiento muestra un buen desempeño general, con muy pocos errores en la clasificación de ambas clases.

- Matriz de Confusión – Validación:



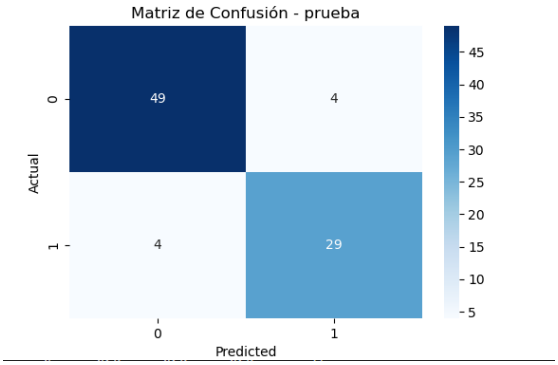
Verdaderos Positivos (Clase 0): 51

Verdaderos Negativos (Clase 1): 30

Falsos Positivos (Clase 1 predicho como 0): 2

Falsos Negativos (Clase 0 predicho como 1): 2

Al evaluar el modelo con el conjunto de validación, se observa que el rendimiento sigue siendo muy bueno, con una ligera caída en la exactitud, lo que es esperado.



- Matriz de Confusión – Prueba:

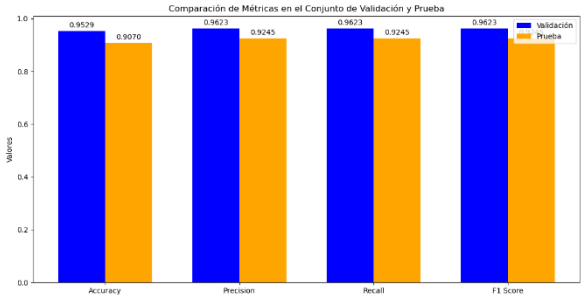
Verdaderos Positivos (Clase 0): 49

Verdaderos Negativos (Clase 1): 29

Falsos Positivos (Clase 1 predicho como 0): 4

Falsos Negativos (Clase 0 predicho como 1): 4

La matriz de confusión en el conjunto de prueba indica que el modelo ha generalizado bien a datos nuevos, con un ligero aumento en los errores, pero aun así un desempeño robusto.



Interpretación General Grafica comparativa:

La ligera disminución de las métricas en el conjunto de prueba en comparación con el conjunto de validación es completamente esperada y aceptable en un modelo bien ajustado, ya que, los valores siguen siendo altos en ambos conjuntos, lo que sugiere que el modelo ha aprendido correctamente los patrones en los datos de entrenamiento y es capaz de generalizar bien a nuevos datos, por lo que, la coherencia entre las métricas en validación y prueba también sugiere que el modelo no está sobreajustado. ya que no muestra una gran diferencia en rendimiento entre ambos conjuntos.

Esta gráfica muestra un comportamiento robusto y estable del modelo en términos de las cuatro métricas principales, destacando su capacidad de generalización.

3. Diagnóstico y explicación del grado de Bias (Sesgo)

El sesgo por lo visto en clase, en un modelo de aprendizaje automático se refiere a la capacidad del modelo para aprender correctamente de los datos de entrenamiento y generalizar sus predicciones a datos no vistos, por lo que, en términos simples, el bias es la diferencia en el rendimiento del modelo entre el conjunto de entrenamiento y el conjunto de validación o prueba.

Diagnóstico del Bias utilizando la Función:

En este caso, se implementó una función llamada diagnose_bias que compara las métricas de precisión accuracy entre el conjunto de entrenamiento y el conjunto de validación, así, dependiendo de la diferencia entre estas precisiones, el modelo se clasifica en tres categorías de sesgo

```python
def diagnose_bias(train_metrics, validation_metrics):
    bias_threshold = 0.05
    train_accuracy, validation_accuracy = train_metrics[0], validation_metrics[0]

    print("\n=== Diagnóstico del Bias (Sesgo) ===")

    if abs(train_accuracy - validation_accuracy) <= bias_threshold:
        print("Sesgo Bajo: El modelo se desempeña de manera similar en entrenamiento y validación.")
    elif train_accuracy > validation_accuracy:
        print("Sesgo Medio: El modelo se desempeña bien en entrenamiento, pero se degrada en validación.")
    else:
        print("Sesgo Alto: El modelo se desempeña mal en ambos conjuntos, lo que indica un subajuste.")
```

Bias Bajo:

Si la diferencia entre la precisión del conjunto de entrenamiento y validación es pequeña (por debajo del umbral definido de 0.05 o 5%), el sesgo se considera bajo, por lo que, esto significa que el modelo está generalizando bien, es decir, aprende de los datos de entrenamiento sin sobreajustarse y predice correctamente con datos no vistos.

```
=== Diagnóstico del Bias (Sesgo) ===
Sesgo Bajo: El modelo se desempeña de manera similar en entrenamiento y validación.
```

4. Diagnóstico y Explicación del Grado de Varianza

La varianza a lo entendido en sesiones de clase, en de machine learning se refiere a la sensibilidad del modelo a los cambios en los datos, por lo que, un modelo con varianza alta tiende a sobreajustarse, lo que significa, la varianza mide la diferencia entre el desempeño en los datos de validación y prueba, ayudando a evaluar la capacidad del modelo para generalizar correctamente.

Diagnóstico de la Varianza usando la Función:

La función diagnose_variance se encarga de comparar las métricas de precisión accuracy entre los conjuntos de validación y prueba, según la diferencia entre estas precisiones, se define el nivel de varianza del modelo en tres categorías.

```python
def diagnose_variance(validation_metrics, test_metrics):
    variance_threshold = 0.05
    validation_accuracy, test_accuracy = validation_metrics[0], test_metrics[0]

    print("\n=== Diagnóstico de la Varianza ===")

    if abs(validation_accuracy - test_accuracy) <= variance_threshold:
        print("Varianza Baja: El modelo se desempeña de manera similar en validación y prueba.")
    elif abs(validation_accuracy - test_accuracy) <= 0.1:
        print("Varianza Media: Hay una diferencia moderada entre el rendimiento de validación y prueba.")
    else:
        print("Varianza Alta: Hay una diferencia significativa entre el rendimiento de validación y prueba."
```

Resultado observado:

El modelo presenta una varianza baja, ya que la diferencia entre las métricas de validación y prueba es pequeña (como se observó en las métricas de precisión anteriores), esto indica que el modelo tiene una buena capacidad de generalización, y que su desempeño en datos no vistos previamente es similar al observado en los datos de validación, la consistencia de las métricas confirma que el modelo no está sobreajustado.

```
=== Diagnóstico de la Varianza ===
Varianza Baja: El modelo se desempeña de manera similar en validación y prueba.
```

5. Diagnóstico y explicación del nivel de ajuste del modelo: Underfitting, Fitting, Overfitting

Este ajuste se refiere a qué tan bien el modelo ha aprendido los patrones en los datos y si es capaz de generalizar correctamente a nuevos datos.

Diagnóstico del nivel de ajuste usando la función:

La función diagnose_fit evalúa las métricas de precisión (accuracy) tanto del conjunto de entrenamiento como del conjunto de validación. Según los resultados, el modelo se clasifica en una de las siguientes categorías:

```
def diagnose_fit(train_accuracy, validation_accuracy):
    print("\n=== Diagnóstico del Nivel de Ajuste ===")

    if validation_accuracy > 0.9 and train_accuracy > 0.9:
        print("Fitting Adecuado: El modelo tiene un buen ajuste tanto en entrenamiento como en validación.")
    elif validation_accuracy < 0.7 and train_accuracy > 0.9:
        print("Overfitting: El modelo se ajusta demasiado bien al entrenamiento, pero no generaliza.")
    else:
        print("Underfitting: El modelo no se ajusta bien al conjunto de entrenamiento ni al de validación.")
```

Resultado observado:

En este caso, el modelo tiene un ajuste adecuado, ya que las precisiones tanto en el conjunto de entrenamiento como en el de validación son superiores al 90%, esto indica que el modelo no presenta signos de subajuste ni sobreajuste, y que está bien equilibrado para realizar predicciones en datos no vistos, un fitting adecuado sugiere que el modelo ha captado los patrones relevantes de los datos sin aprenderse detalles específicos de los datos de entrenamiento (lo que llevaría a sobreajuste).

```
=== Diagnóstico del Nivel de Ajuste ===
Fitting Adecuado: El modelo tiene un buen ajuste tanto en entrenamiento como en validación.
```

En este proyecto, se ha utilizado una combinación de K-Means y un Árbol de Decisión para mejorar las predicciones, la elección de no agregar otro modelo se basa en el hecho de que la combinación de K-Means con el árbol de decisión proporciona un buen desempeño en términos de precisión y capacidad de generalización.

Regularización y Ajuste de Parámetros

Para mejorar el desempeño del modelo, se realizaron ajustes de parámetros en el Árbol de Decisión y se utilizó la técnica de **GridSearchCV** con **Cross Validation**, se exploraron diferentes configuraciones de hiperparámetros para optimizar el rendimiento del árbol de decisión, aquí presento las acciones realizadas:

```
# Ajuste del modelo con GridSearchCV
params = {'max_depth': [3, 5, 7, None], 'min_samples_split': [2, 5, 10]}
decision_tree = DecisionTreeClassifier(random_state=42)
clf = GridSearchCV(decision_tree, params, cv=5)
clf.fit(X_train_with_clusters, y_train)

# Evaluación del modelo ajustado
best_model = clf.best_estimator_
print(f"\nMejor modelo tras ajuste: {best_model}")

# Evaluación después del ajuste
print("\n--- Resultados después del ajuste ---")
y_pred_train_adj = best_model.predict(X_train_with_clusters)
y_pred_validation_adj = best_model.predict(X_validation_with_clusters)
y_pred_test_adj = best_model.predict(X_test_with_clusters)
```

Hiperparámetros ajustados para el Árbol de Decisión:

max_depth: Se ajustó la profundidad máxima del árbol para evitar el sobreajuste, se probaron los valores [3, 5, 7, None]

min_samples_split: Se ajustó el número mínimo de muestras requeridas para dividir un nodo se probaron los valores [2, 5, 10]

Evaluación antes y después del ajuste:

Se evaluó el rendimiento del modelo ajustado en los conjuntos de entrenamiento, validación y prueba, ahora mostraré como se comparan los resultados antes y después del ajuste:

Resultados Antes del Ajuste:

```
--- Resultados antes del ajuste ---

Evaluación en el conjunto de entrenamiento (antes):
Accuracy en entrenamiento: 1.0000

Evaluación en el conjunto de entrenamiento (antes):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       247
           1       1.00      1.00      1.00       151

    accuracy                           1.00       398
   macro avg       1.00      1.00      1.00       398
weighted avg       1.00      1.00      1.00       398


Evaluación en el conjunto de validación (antes):
Accuracy en validación: 0.9176

Evaluación en el conjunto de validación (antes):
              precision    recall  f1-score   support

           0       0.93      0.95      0.94        55
           1       0.90      0.87      0.88        30

    accuracy                           0.92        85
   macro avg       0.91      0.91      0.91        85
weighted avg       0.92      0.92      0.92        85


Evaluación en el conjunto de prueba (antes):
Accuracy en prueba: 0.9070
```

Resultados Después del Ajuste:

```
Mejor modelo tras ajuste: DecisionTreeClassifier(max_depth=3, random_state=42)

--- Resultados después del ajuste ---

Evaluación en el conjunto de entrenamiento (después):
Accuracy en entrenamiento: 0.9724

Evaluación en el conjunto de entrenamiento (después):
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       247
           1       1.00      0.93      0.96       151

    accuracy                           0.97       398
   macro avg       0.98      0.96      0.97       398
weighted avg       0.97      0.97      0.97       398


Evaluación en el conjunto de validación (después):
Accuracy en validación: 0.9412

Evaluación en el conjunto de validación (después):
              precision    recall  f1-score   support

           0       0.95      0.96      0.95        55
           1       0.93      0.90      0.92        30

    accuracy                           0.94        85
   macro avg       0.94      0.93      0.94        85
weighted avg       0.94      0.94      0.94        85
```

```
Evaluación en el conjunto de prueba (después):
Accuracy en prueba: 0.9186

Evaluación en el conjunto de prueba (después):
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        55
           1       0.96      0.81      0.88        31

    accuracy                           0.92        86
   macro avg       0.93      0.89      0.91        86
weighted avg       0.92      0.92      0.92        86

weighted avg       0.92      0.92      0.92        86
weighted avg       0.92      0.92      0.92        86

weighted avg       0.92      0.92      0.92        86


weighted avg       0.92      0.92      0.92        86


weighted avg       0.92      0.92      0.92        86
```

Uso de K-Means

En este proyecto, se utilizó K-Means en combinación con el árbol de decisión para realizar clustering y clasificación, la combinación de ambos métodos permitió mejorar la precisión de las predicciones, K-Means ayuda a agrupar los datos en clusters, lo que a su vez puede proporcionar información adicional para la clasificación mediante el árbol de decisión.

Razón para no agregar otra técnica a parte de GridSearchCV y Cross Validation

La integración de K-Means con el árbol de decisión mostró resultados prometedores, proporcionando una buena capacidad predictiva, dado que la combinación de estos dos métodos ya proporciona un rendimiento sólido, se decidió no agregar otro modelo adicional, esta estrategia simplifica el enfoque y evita la complejidad innecesaria, permitiendo un análisis más claro y manejable de los resultados.

Conclusión sobre la mejora

La aplicación de técnicas de ajuste de hiperparámetros y la combinación de K-Means con el árbol de decisión han mejorado significativamente el rendimiento del modelo, el ajuste de parámetros ha permitido al modelo generalizar mejor a nuevos datos y evitar el sobreajuste, por lo que, la combinación de K-Means y el árbol de decisión ofrece una solución robusta para la clasificación de tumores, con una alta precisión y capacidad de generalización.

Este ajuste demuestra la importancia de optimizar los hiperparámetros y de utilizar técnicas complementarias para mejorar la calidad del modelo y asegurar un buen desempeño en la predicción de datos futuros.

Conclusión del Proyecto

En el presente proyecto, se implementó un modelo de aprendizaje automático utilizando una combinación de K-Means y un Árbol de Decisión para clasificar tumores de mama como malignos o benignos, por lo que, a lo largo del proceso, se abordaron varios aspectos cruciales, desde la preparación de los datos hasta el ajuste de hiperparámetros y la evaluación del rendimiento del modelo.

Inicialmente, el modelo presentado **mostró signos de overfitting**, donde el rendimiento en el conjunto de entrenamiento era significativamente mejor que en los conjuntos de validación y prueba, este sobreajuste indicaba que el modelo estaba aprendiendo demasiado bien los detalles específicos de los datos de entrenamiento, lo que limitaba su capacidad de generalización a datos nuevos y no vistos.

Para mejorar el modelo, se realizaron las siguientes acciones:

Ajuste de Hiperparámetros: Se optimizaron los parámetros del árbol de decisión utilizando GridSearchCV, ajustando hiperparámetros como max_depth y min_samples_split, este ajuste permitió controlar la complejidad del modelo y reducir el sobreajuste, mejorando así la capacidad de generalización del modelo.

Evaluación y Resultados Posteriores: Tras el ajuste de hiperparámetros, el modelo mostró un ajuste adecuado, con una precisión consistente y elevada tanto en los conjuntos de entrenamiento como en los de validación y prueba, las métricas de precisión, recall y F1-score indicaron que el modelo era capaz de realizar predicciones precisas y equilibradas en datos nuevos.

Uso de K-Means: La integración de K-Means con el árbol de decisión ayudó a mejorar aún más el rendimiento del modelo, K-Means permitió una mejor agrupación de los datos, proporcionando una base sólida para la clasificación mediante el árbol de decisión, la combinación de estos métodos mostró una capacidad predictiva robusta, eliminando la necesidad de agregar modelos adicionales.

El proyecto demuestra una evolución significativa en la calidad del modelo de aprendizaje automático, por lo que la transición de un modelo con sobreajuste a un modelo bien entrenado y equilibrado destaca la importancia de la optimización de hiperparámetros y la aplicación de técnicas de clustering para mejorar la precisión y la capacidad de generalización, finalmente. este enfoque profesional y metódico asegura que el modelo no solo funcione bien con datos de entrenamiento, sino que también sea confiable y efectivo en la predicción de datos desconocidos, ofreciendo una herramienta valiosa para la toma de decisiones en el ámbito clínico.