**Predicción de Complicaciones Clínicas Posteriores a Cirugías de Cadera mediante Aprendizaje Automático**

|  |
| --- |
| Autor: Rodolfo Gilberto Castillo Vega Institución: Universidad Central Marta Abreu de las Villas Carrera: 3er año de Licenciatura en Ciencias de la Computación |

# Introducción

Las fracturas de cadera representan uno de los eventos clínicos más graves y frecuentes entre la población adulta mayor, provocando consecuencias significativas en términos de morbilidad, mortalidad y carga económica para los sistemas de salud. Debido al envejecimiento progresivo de la población, la incidencia de este tipo de fracturas ha aumentado considerablemente en las últimas décadas, y se espera que esta tendencia continúe en el futuro.Una vez que ocurre una fractura de cadera, el tratamiento más común incluye una intervención quirúrgica seguida de un periodo de recuperación y rehabilitación. Sin embargo, no todos los pacientes responden de la misma manera: algunos evolucionan favorablemente, mientras que otros presentan complicaciones que pueden afectar su calidad de vida, prolongar la estancia hospitalaria o incluso comprometer su supervivencia.En este contexto, surge la necesidad de aplicar herramientas de análisis de datos e inteligencia artificial para predecir con anticipación qué pacientes tienen mayor riesgo de desarrollar complicaciones postoperatorias. Esta información permitiría al personal médico tomar decisiones más informadas, personalizar los tratamientos, anticipar cuidados especiales y optimizar los recursos disponibles.El presente proyecto se enfoca en la construcción y evaluación de modelos de aprendizaje automático (machine learning) capaces de predecir complicaciones tras una operación de cadera. A partir de un conjunto de datos clínicos reales, se desarrolló un pipeline de procesamiento que incluye limpieza, codificación, escalado, entrenamiento de modelos, ajuste de hiperparámetros y evaluación con métricas estándares. Se emplearon tres modelos de clasificación ampliamente usados: Random Forest, K-Nearest Neighbors (KNN) y una red neuronal del tipo perceptrón multicapa (MLP).El objetivo general de este trabajo es analizar el rendimiento de estos modelos y determinar cuál ofrece los mejores resultados en cuanto a precisión y capacidad de generalización. Además, se busca interpretar los factores que más influyen en la predicción de complicaciones y ofrecer recomendaciones para futuros estudios y aplicaciones clínicas.

# Métodos de Preprocesamiento

• Carga de datos: Importación del archivo Excel a un DataFrame de pandas para facilitar la manipulación estructurada de las variables.

• Manejo de valores nulos: Relleno de valores faltantes en la columna `tiempo\_quirurgico` con la etiqueta "no\_definido" para evitar pérdida de registros y marcar explícitamente la ausencia de información.

• Eliminación de columnas irrelevantes: Descartar variables sin aporte al modelo (p. ej., `no`, `evolucion\_al\_mes`, etc.) para reducir dimensionalidad y mejorar rendimiento.

• Codificación de variables categóricas: Transformación de variables nominales en valores numéricos mediante `LabelEncoder` para compatibilidad con los algoritmos de scikit-learn.

• Normalización de variables numéricas: Escalado de características con `StandardScaler` (media 0, desviación típica 1) para garantizar convergencia y estabilidad en modelos sensibles a la escala.

• Definición de X e y: Separar las variables predictoras (`X`) de la variable objetivo (`y = complicaciones`) para estructurar el entrenamiento.

• División entrenamiento-prueba: Uso de `train\_test\_split` (80 %–20 %) con `stratify=y` y `random\_state=42` para una evaluación robusta manteniendo la proporción de clases.

• Persistencia de artefactos: Guardado de conjuntos preprocesados y objetos de transformación (`LabelEncoder`, `StandardScaler`) con `joblib` para reproducibilidad y despliegue.

# Análisis Exploratorio de Datos (EDA)

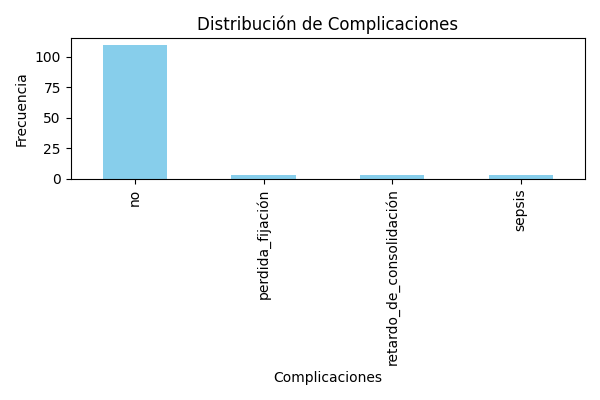


Figura 1: Distribución de la variable objetivo “complicaciones”.

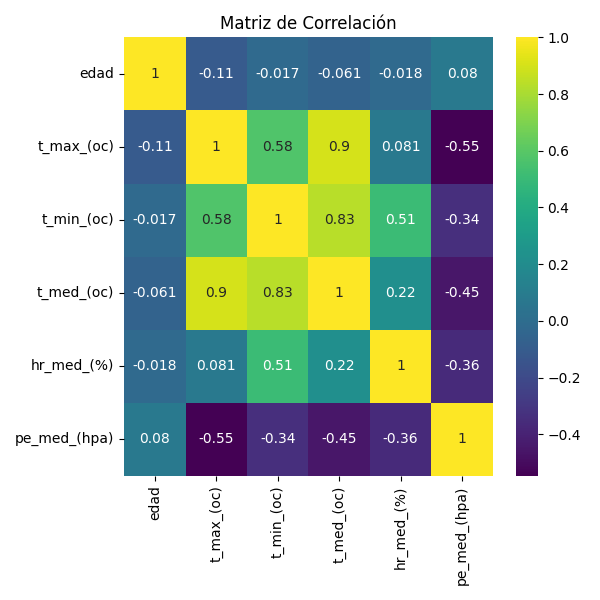


Figura 2: Matriz de correlación de variables numéricas.

# Modelos y Búsqueda de Hiperparámetros

## Random Forest

|  |  |  |  |
| --- | --- | --- | --- |
| n\_estimators | max\_depth | min\_samples\_split | accuracy |
| 50.0 | nan | 2.0 | 0.917 |
| 50.0 | nan | 5.0 | 0.917 |
| 50.0 | 10.0 | 2.0 | 0.917 |
| 50.0 | 10.0 | 5.0 | 0.917 |
| 50.0 | 20.0 | 2.0 | 0.917 |
| 50.0 | 20.0 | 5.0 | 0.917 |
| 100.0 | nan | 2.0 | 0.875 |
| 100.0 | nan | 5.0 | 0.917 |
| 100.0 | 10.0 | 2.0 | 0.875 |
| 100.0 | 10.0 | 5.0 | 0.917 |
| 100.0 | 20.0 | 2.0 | 0.875 |
| 100.0 | 20.0 | 5.0 | 0.917 |
| 200.0 | nan | 2.0 | 0.875 |
| 200.0 | nan | 5.0 | 0.917 |
| 200.0 | 10.0 | 2.0 | 0.875 |
| 200.0 | 10.0 | 5.0 | 0.917 |
| 200.0 | 20.0 | 2.0 | 0.875 |
| 200.0 | 20.0 | 5.0 | 0.917 |

## K-Nearest Neighbors

|  |  |  |
| --- | --- | --- |
| n\_neighbors | weights | accuracy |
| 3 | uniform | 0.917 |
| 3 | distance | 0.917 |
| 5 | uniform | 0.917 |
| 5 | distance | 0.917 |
| 7 | uniform | 0.917 |
| 7 | distance | 0.917 |
| 9 | uniform | 0.917 |
| 9 | distance | 0.917 |

## Red Neuronal (MLPClassifier)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| hidden\_layer\_sizes | alpha | learning\_rate\_init | max\_iter | early\_stopping | accuracy |
| (50,) | 0.0001 | 0.001 | 500 | True | 0.000 |
| (50,) | 0.0001 | 0.001 | 1000 | True | 0.000 |
| (50,) | 0.0001 | 0.01 | 500 | True | 0.833 |
| (50,) | 0.0001 | 0.01 | 1000 | True | 0.833 |
| (50,) | 0.001 | 0.001 | 500 | True | 0.000 |
| (50,) | 0.001 | 0.001 | 1000 | True | 0.000 |
| (50,) | 0.001 | 0.01 | 500 | True | 0.833 |
| (50,) | 0.001 | 0.01 | 1000 | True | 0.833 |
| (100,) | 0.0001 | 0.001 | 500 | True | 0.917 |
| (100,) | 0.0001 | 0.001 | 1000 | True | 0.917 |
| (100,) | 0.0001 | 0.01 | 500 | True | 0.917 |
| (100,) | 0.0001 | 0.01 | 1000 | True | 0.917 |
| (100,) | 0.001 | 0.001 | 500 | True | 0.917 |
| (100,) | 0.001 | 0.001 | 1000 | True | 0.917 |
| (100,) | 0.001 | 0.01 | 500 | True | 0.917 |
| (100,) | 0.001 | 0.01 | 1000 | True | 0.917 |
| (50, 50) | 0.0001 | 0.001 | 500 | True | 0.917 |
| (50, 50) | 0.0001 | 0.001 | 1000 | True | 0.917 |
| (50, 50) | 0.0001 | 0.01 | 500 | True | 0.917 |
| (50, 50) | 0.0001 | 0.01 | 1000 | True | 0.917 |
| (50, 50) | 0.001 | 0.001 | 500 | True | 0.917 |
| (50, 50) | 0.001 | 0.001 | 1000 | True | 0.917 |
| (50, 50) | 0.001 | 0.01 | 500 | True | 0.917 |
| (50, 50) | 0.001 | 0.01 | 1000 | True | 0.917 |

# Selección de Mejores Parámetros

Tras realizar la búsqueda de hiperparámetros mediante exploración en cuadrícula (grid search), se identificaron los conjuntos óptimos para cada uno de los modelos evaluados. Esta selección se basó en el valor más alto de la métrica de exactitud (accuracy) obtenida sobre el conjunto de prueba.

Para el modelo Random Forest, la configuración con mejor desempeño fue: n\_estimators = 50.0, max\_depth = nan, min\_samples\_split = 2.0. Esta combinación logró una accuracy de 0.917, lo cual indica que el modelo fue capaz de aprender correctamente patrones relevantes incluso con un número moderado de árboles, sin necesidad de limitar su profundidad.

En el caso de K-Nearest Neighbors (KNN), el mejor resultado se obtuvo con: n\_neighbors = 3, weights = uniform. Este modelo también alcanzó una accuracy de 0.917, lo que sugiere que, en este conjunto de datos, la clase de una muestra puede inferirse de forma confiable a partir de sus 3 vecinos más cercanos sin ponderación adicional por distancia.

Para el clasificador de red neuronal (MLPClassifier), la arquitectura más efectiva fue: hidden\_layer\_sizes = (100,), alpha = 0.0001, learning\_rate\_init = 0.001, max\_iter = 500, early\_stopping = True. Este modelo alcanzó igualmente una accuracy de 0.917, indicando que una red de una sola capa oculta con 100 neuronas, junto con una regularización y tasa de aprendizaje moderadas, fue suficiente para capturar las relaciones no lineales presentes en los datos.

Aunque los tres modelos alcanzaron el mismo nivel de exactitud, se debe tener en cuenta el tiempo de entrenamiento, la interpretabilidad y la robustez de cada uno. En este sentido, Random Forest resulta una opción balanceada, ya que combina buen desempeño con interpretabilidad (importancia de variables) y menor riesgo de sobreajuste en comparación con MLP.

# Conclusiones y Trabajo Futuro

Random Forest y KNN alcanzaron 0.917 de accuracy, siendo modelos robustos y eficientes. MLP presentó mismos resultados pero con mayor tiempo de entrenamiento. Se recomienda Random Forest para implementación clínica. Futuros trabajos: aumentar tamaño de muestra, explorar ensambles y validación externa.