



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Aplicación de Gestión de  
Quirófanos mediante  
Inteligencia Computacional**



Presentado por Jesús García Armario  
en Universidad de Burgos — 3 de julio de 2023  
Tutores: Bruno Baruque Zanón, Daniel Urda  
Muñoz y Raúl Marticorena Sánchez







UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



D. Bruno Baruque Zanón y D. Daniel Urda Muñoz, profesores del departamento de Digitalización, área de Ciencia de la Computación e Inteligencia Artificial y D. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Jesús García Armario, con DNI 47338433-V, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado **Aplicación de Gestión de Quirófanos mediante Inteligencia Computacional**.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 3 de julio de 2023

Vº. Bº. del Tutor:

Vº. Bº. del Tutor:

D. Bruno Baruque Zanón

D. Daniel Urda Muñoz

Vº. Bº. del Tutor:

D. Raúl Marticorena  
Sánchez





## Resumen

**Introducción:** La planificación y gestión del tiempo quirúrgico constituye más del 60 % del esfuerzo económico de un hospital. A lo largo de la literatura se han propuesto múltiples enfoques para optimizar este problema, donde la estimación de la *duración* de las intervenciones constituye uno de los desafíos más importantes en la actualidad. Proponemos en este trabajo una solución que aborde ambas perspectivas (predicción y optimización) aplicando algoritmos de inteligencia computacional e integrando la solución en una API que sirva como proveedora de servicios a una aplicación web integrada en la nube de Amazon (AWS).

**Material y método:** Se parte de un dataset de más de 24000 pacientes intervenidos entre 2016 y 2022. El lenguaje seleccionado para la implementación de los algoritmos ha sido Python, apoyado en librerías nativas y dedicadas (*Scikit-learn*, *Pandas* o *DEAP* entre otras). Para el diseño de datos, empleamos el lenguaje MySQL con el motor InnoDB, desplegándolo en Amazon RDS. Para desarrollar la API y la aplicación web empleamos el framework *Flask* junto al lenguaje HTML, plantillas de estilo CSS de Bootstrap 5 y Javascript. Los espacios de trabajo fueron integrados en un *Docker*, almacenado en Amazon ECR y desplegados en Amazon ECS .

**Resultados:** Alcanzamos valores de RMSE en la línea de los conseguidos por otros autores (40-50min sobre un total de 480 minutos de planificación). Conseguimos alcanzar la convergencia en la fase de optimización tras combinar un enfoque basado en heurísticas y algoritmos bio-inspirados. Los modelos y algoritmos se integraron correctamente en una API funcionante, con tiempos de respuesta aceptables e integradas sobre una aplicación web con un interfaz comprensible para el usuario.

**Conclusiones y discusión:** Hemos logrado ofrecer una solución útil y eficaz para la tarea de gestionar eficientemente los quirófanos. Sin embargo, existen restricciones y variables no exploradas provistas en la literatura que podrían mejorar el rendimiento de este proyecto de cara a desarrollos futuros.

## Descriptores

Gestión hospitalaria, Planificación quirúrgica, Predicción de tiempos, Estimación de duración de intervenciones, Inteligencia Computacional, Aprendizaje Supervisado, Aprendizaje Máquina, Red neuronal, Árbol de regresión, Optimización, Metaheurísticas, Algoritmo

## II

Genético, AWS, AWS ECR, Flask, MySQL, Bootstrap, Docker, Container, API, Computación en la nube.



## Abstract

**Introduction:** Surgery time and OR scheduling represent more than 60 % of the costs inside a hospital. Many solutions have been provided along specialized publications. Besides, predicting surgery time durations has become the most challenging issue regarding this topic. We propose a solution within both perspectives (prediction and optimization) applying machine learning algorithms and developing our solution inside an API as a service provider for a web application deployed on AWS.

**Material and methods:** We began composing a dataset containing data from 24000 patients who received surgery between 2016 and 2022. We chose Python to implement our algorithms, supported by native and dedicated (*Scikit-learn*, *Pandas*, *DEAP*...) libraries. We used MySQL and InnoDB to design our database model and it was deployed inside Amazon RDS. Finally, we used *Flask Framework*, HTML, Bootstrap 5 CSS stylesheets and javascript to build our API and web application. These workspaces were integrated on a Docker container, stored inside Amazon ECR and deployed on Amazon ECS.

**Results:** We achieved RMSE values similar to those reviewed inside the bibliography (40-50 minutes for a total planification of 480 minutes). During optimization stage, we managed to obtain convergence after developing a heuristic and bio-inspired approach. Also, these models and algorithms were successfully integrated to build a functional API, obtaining good response times and being able to communicate efficiently with a web application having an user friendly GUI.

**Summary and discussion:** After our efforts, we managed to develop an useful and efficient approach. However, there are still many restrictions and attributes not reviewed in this project that may provide a powerful improvement after future explorations.

## Keywords

Health Management, Surgery Scheduling, Time prediction, Surgical time prediction, Artificial Intelligence, Supervised Learning, Machine Learning, Artificial Neural Network, Regression tree, Optimization, Metaheuristic, Genetic Algorithm, AWS, AWS ECR, Flask, MySQL, Bootstrap, Docker, Container, API, Cloud Computing.

---

# Índice general

---

Índice general	iv
Índice de figuras	vi
Índice de tablas	vii
Introducción	1
Objetivos del proyecto	3
2.1. Definición de Objetivos . . . . .	3
2.2. Consideraciones y Marco de Trabajo . . . . .	5
Conceptos teóricos	7
3.1. Aprendizaje Supervisado . . . . .	7
3.2. Optimización en Planificación . . . . .	16
Técnicas y herramientas	21
4.1. Metodología de Desarrollo . . . . .	21
4.2. Elementos de Programación . . . . .	23
4.3. Integración y Despliegue . . . . .	26
Aspectos relevantes del desarrollo del proyecto	29
5.1. Obtención y Preprocesamiento de Datos . . . . .	29
5.2. Aprendizaje Supervisado: Predicción . . . . .	36
5.3. Optimización: Planificación y Gestión . . . . .	39
5.4. Integración y Despliegue . . . . .	41
Trabajos relacionados	45

<i>ÍNDICE GENERAL</i>	v
6.1. Machine Learning y Predicción del Tiempo Quirúrgico . . .	46
6.2. Optimización en Planificación . . . . .	49
<b>Conclusiones y Líneas de trabajo futuras</b>	<b>53</b>
<b>Bibliografía</b>	<b>55</b>

---

## Índice de figuras

---

2.1. Subdivisión del Proyecto . . . . .	3
2.2. Marco de Trabajo del Proyecto . . . . .	5
3.1. Ejemplo de Árbol de Decisiones . . . . .	10
3.2. Ejemplo de Random Forest . . . . .	12
3.3. Funcionamiento del Perceptrón. [26] . . . . .	14
3.4. Perceptrón Multicapa [26] . . . . .	14
3.5. Representación discreta y continua del tiempo. Fuente: [15] . . . . .	18
4.1. Funcionamiento de Scrum. Fuente:[48] . . . . .	21
4.2. Características y Posibilidades de GitHub. Fuente: [28] . . . . .	22
4.3. Librerías más usadas en Data Science . . . . .	23
4.4. Ejemplo de fichero Dockerfile para construcción de un contenedor . . . . .	27
4.5. Panel de control de Amazon RCS . . . . .	28
5.1. Número de Intervenciones Registradas por Especialidad . . . . .	30
5.2. Variables del Conjunto de Datos Base . . . . .	30
5.3. Variables del Conjunto de Datos Ampliado . . . . .	31
5.4. Aspecto de la Base de Datos antes de la edición. . . . .	32
5.5. Aspecto de los datos tras la limpieza y edición. Previo a codificación . . . . .	32
5.6. Media de duraciones de intervenciones por Especialidad . . . . .	33
5.7. Matriz de Correlación de Spearman . . . . .	34
5.8. Sección del Árbol de Regresión Construido . . . . .	37
5.9. Representación de rendimiento del árbol de regresión . . . . .	38
5.10. Evolución de la función de evaluación en el modelo seleccionado . . . . .	40
5.11. Ejemplo de comunicación con API para planificación (GET) . . . . .	42
5.12. Visualización del Panel de Gestión de Usuarios . . . . .	43
5.13. Visualización de un listado de predicciones . . . . .	43
5.14. Visualización de un ejemplo de planificación . . . . .	44
6.1. Técnicas de Optimización. Fuente [22] . . . . .	51

---

# Índice de tablas

---

3.1. Métricas de Quirófano . . . . .	16
3.2. Eficiencia y Calidad . . . . .	17
4.1. Resumen de IDEs . . . . .	24
4.2. Librerías Python Empleadas . . . . .	26
5.1. Comparativa de algoritmos ML . . . . .	38
5.2. Resultados con N=100, 3 Quirófanos y 5 días . . . . .	40



---

# Introducción

---

Durante muchos años, la comunidad científica ha tratado de elaborar modelos predictivos que ayuden en tareas de **organización industrial**.

Los hospitales producen **servicios**, y desde hace tiempo han empezado a tomar decisiones estratégicas en los servicios que proveen, debido en gran parte al incremento en la oferta de productos sanitarios y nuevos tratamientos disponibles, así como al entorno competitivo que nos rodea. Por este motivo, las clínicas, como cualquier otra empresa, buscan medidas para reducir costes e impulsar sus demandas financieras [31].

Si hablamos de **sanidad privada**, las dos terceras partes de los ingresos de un hospital dependen de las intervenciones quirúrgicas, las cuales constituyen aproximadamente un 40 % del gasto hospitalario [44].

Los quirófanos forman parte de un entorno muy complejo, conformado por **múltiples capas**, y condicionado por un gran número de *interacciones sociales, impredecibilidad y baja tolerancia a fallos* [47].

Por otro lado, las irregularidades en el flujo de trabajo dentro del área quirúrgica resultan en perjuicios sobre la **salud mental de los profesionales**, a menudo motivados por factores como: *amenaza de demandas, alta presión para realizar tareas difíciles, conflicto de intereses...*[55].

Por este motivo, incrementar la productividad en los quirófanos tendría una gran influencia en el rendimiento financiero de los mismos, sin dejar de lado algunas consideraciones éticas (*reducción de listas de espera, priorización, equidad en reparto de recursos...*), incrementando la calidad de los servicios y la satisfacción de los pacientes de forma directamente proporcional a las mejoras conseguidas.

El gasto asociado a los quirófanos es uno de los más elevados de cada hospital, por lo que la optimización del proceso resulta crucial de cara a una gestión eficiente de los recursos. No obstante, el diseño de una planificación quirúrgica resulta muy difícil, debido a una enorme incertidumbre respecto a la duración de las cirugías [10].

Dentro de los problemas de optimización, si analizamos la literatura encontramos que la planificación quirúrgica ocupa un área especial dentro de esta disciplina. En función del criterio a mejorar, los investigadores han propuesto diferentes soluciones una vez identificados y analizados los problemas [22].

Podemos contribuir a la investigación desde diversos puntos de vista, adoptando múltiples estrategias. Por ello, derivado de disciplinas como la minería o el análisis de datos, se presentan estrategias novedosas que permitan contar con iniciativas prometedoras de cara a lograr la **eficiencia** [50].

Nuestro proyecto tratará de aunar **ambos conflictos** en el área de gestión de espacios quirúrgicos mediante un proceso de *análisis exploratorio* y posterior *implementación* de un producto software capaz de **predecir la duración estimada** en base a un conjunto de características aplicables a los pacientes y, en base a esa información, **optimizar** el reparto de los actos quirúrgicos en función de los recursos disponibles y la prioridad asignada.



---

# Objetivos del proyecto

---

## 2.1. Definición de Objetivos

El objetivo **principal** del proyecto consiste en la elaboración de un *sistema de gestión de quirófanos* que permita al usuario la introducción de un conjunto de pacientes listos para intervenir, junto a una recopilación de los recursos y restricciones disponibles, para generar **automáticamente** una propuesta de planificación en base a los tiempos predichos y los recursos disponibles.

Específicamente, podemos subdividir los objetivos en función de una de las tres áreas principales a implementar, tal y como podemos consultar en la figura 2.1:

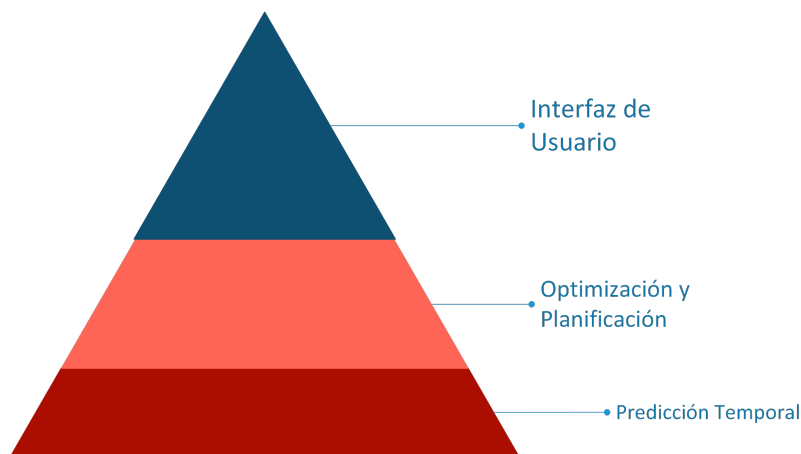


Figura 2.1: Subdivisión del Proyecto

**1. Objetivos del Proyecto de Software**

- a) Predecir el tiempo estimado de un acto quirúrgico.
- b) Planificar quirófanos en función del tiempo y las restricciones.
- c) Recibir, almacenar y validar los datos.
- d) Devolver una salida al cliente.

**2. Objetivos para Predecir la Duración**

- a) Obtener una predicción del tiempo quirúrgico estimado en base a un conjunto de variables fijas.
- b) Calcular la predicción en un tiempo de ejecución asumible por el usuario.
- c) Obtener un error de predicción igual o inferior a los calculados en otros estudios y registrados en la literatura.
- d) Integrarla con la entrada del objetivo 3.

**3. Objetivos para Optimizar la Planificación**

- a) Generar un plan quirúrgico capaz de priorizar pacientes.
- b) Obtener una planificación que no sobrepase el límite de tiempo establecido.
- c) Crear una propuesta que minimice la aparición de huecos vacíos.
- d) Generar la propuesta en tiempo asumible.
- e) Integrarla con la salida del objetivo 2.

**4. Objetivos de la API**

- a) Fácil de usar para los usuarios.
- b) Informar al cliente de las transacciones y errores.
- c) Capaz de guiar al usuario en caso de error y corregirlo.
- d) Acceso y respuesta en tiempo razonable.
- e) Ofrecer la opción de predecir, planificar o combinar ambas funciones.
- f) Ser compatible con formatos estándares de entrada y salida de información: CSV, JSON...

## 2.2. Consideraciones y Marco de Trabajo

Una vez efectuada la definición **formal** de las metas que deseamos alcanzar durante el desarrollo del proyecto, esbozaremos la estructura de los pasos a seguir para lograr cumplir con los términos.

En definitiva, nos encontramos ante un proyecto cimentado sobre dos áreas principales: **aprendizaje automático** y **optimización**, las cuales conforman una capa extensa de *backend*, tras los procesos intermedios de análisis, exploración y selección.

Sobre esta capa se implementará otra de servicios, *frontend*, encargada de la interacción con los clientes y planteada a modo de API para poder ofrecer la **funcionalidad** requerida y su fácil adaptación a la interfaz deseada.

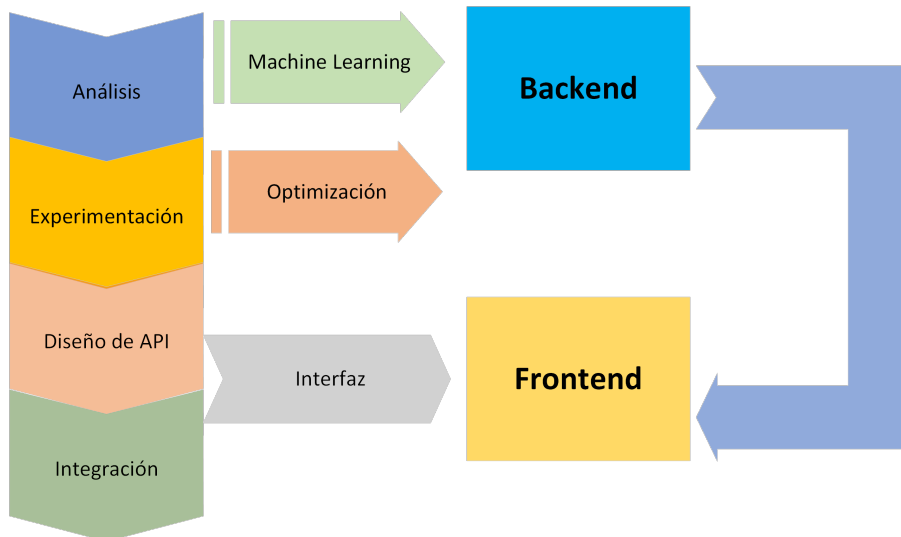


Figura 2.2: Marco de Trabajo del Proyecto



---

## Conceptos teóricos

---

En el siguiente proyecto se hace uso de algunas técnicas de **aprendizaje automático** y **optimización** que conviene describir de cara a la mejor comprensión de los pasos seguidos hacia su resolución.

Tal y como se ha comentado, se parte de un **modelo predictivo**, basado en técnicas de *aprendizaje supervisado* para estimar la duración de una intervención quirúrgica en base a los datos introducidos por el usuario.

Posteriormente, partiremos de la salida del modelo para realizar una **programación** de tipo *parallel scheduling*<sup>1</sup>, usando una aproximación con restricciones que definen un problema **MIP**<sup>2</sup> [30] y resultan en un problema *NP-Hard*.

### 3.1. Aprendizaje Supervisado

El aprendizaje supervisado es un enfoque de inteligencia artificial en el cual un algoritmo es entrenado a partir de datos de entrada para producir una **determinada salida**.

Este modelo es entrenado hasta que es capaz de detectar los patrones y relaciones entre los datos de entrada y las *etiquetas* de salida, de forma que pueda otorgar resultados precisos cuando se le presenten datos *nuevos* [43].

Reflejaremos en las siguientes subsecciones los diferentes algoritmos empleados para el proceso predictivo.

---

<sup>1</sup>Programar  $n$  trabajos en  $m$  máquinas paralelas [57], se asemeja a programar  $n$  intervenciones en  $m$  quirófanos simultáneos

<sup>2</sup>*Mixed Integer Programming*: Problemas de programación dinámica en la que algunas de sus variables deben ser enteras [46]

## Regresión Lineal

Existen diferentes tipos de algoritmos de regresión, siendo cada uno de ellos el más adecuado en función del problema a resolver.

El mecanismo de regresión lineal más **simple** es aquel definido para dibujar una línea a través de un grafo que muestra dependencia lineal entre ambas variables y su ecuación es [4]:

$$y = mx + b \quad (3.1)$$

Donde  $m$  representa a la pendiente de la recta,  $b$  a la ordenada en el origen,  $y$  es la variable dependiente (característica de salida) y  $x$  la independiente (característica de entrada).

Para modelos de regresión lineal múltiple, que es el que manejaremos en nuestro trabajo, usamos la ecuación siguiente [4]:

$$y = m_1x_1 + m_2x_2 + \dots + m_nx_n + b \quad (3.2)$$

## Método de Mínimos Cuadrados

Para obtener los parámetros de la ecuación, usaremos el **método de mínimos cuadrados**, el cual se encarga de buscar la curva que mejor se ajusta a un conjunto de puntos minimizando la suma cuadrática de los residuos a los puntos de la curva [54].

Operando [54]:

$$ss_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 \quad (3.3)$$

$$ss_{xy} = \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \quad (3.4)$$

De este modo, podemos estimar la *pendiente* como:

$$b = \frac{ss_{xy}}{ss_{xx}} \quad (3.5)$$

Y la *ordenada en el origen* en términos de  $b$ :

$$a = \bar{y} - b\bar{x} \quad (3.6)$$

## K-Nearest Neighbors (KNN)

Se trata de una de las formas más simples de aplicar un algoritmo de aprendizaje supervisado, con utilidad para tareas tanto de **regresión** como de **clasificación**.

Se considera un algoritmo *no paramétrico* en tanto que no existen preconcepciones acerca de los datos subyacentes [4].

La base de su funcionamiento es la estimación de proximidad mediante el cálculo de **distancias**, en especial la **distancia euclídea** [20]:

$$d(p, q) = \sqrt{(p - q)^2} \quad (3.7)$$

Podemos describir el algoritmo en los siguientes pasos [2]:

1. Determinar el número de vecinos más cercanos, conocido como el parámetro  $K$ .
2. Usar la función de **distancia** para calcular la distancia entre cada instancia del conjunto a predecir y todos los del entrenamiento.
3. Ordenar la distancia de menor a mayor y elegir tantos elementos del conjunto de entrenamiento como  $K$ .
4. Obtener la etiqueta o el valor numérico de los K-Vecinos.
5. Devolver la *media* (regresión) o la *moda* (clasificación).

## Árbol de Decisión

Se trata de otro tipo de aprendizaje supervisado que, al igual que KNN, puede ser usado para ambos tipos de problema.

Actualmente, es una de las herramientas **más usadas** para la toma de decisiones [38] en todo el mundo.

Para cumplir con esta tarea, se dibuja un árbol con diferentes *ramas* y *hojas*, de forma que se incluyan *todos los valores* de una situación particular [38], tal y como podemos ver en la figura 3.1:

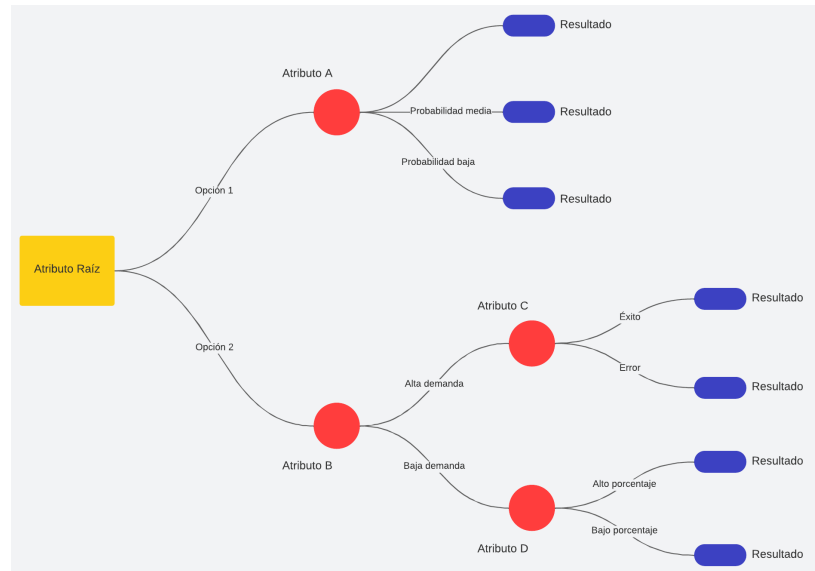


Figura 3.1: Ejemplo de Árbol de Decisiones

Un árbol divide los datos en *subárboles* de forma progresiva, desde la raíz, pasando por los nodos internos o de decisión y hasta llegar a los resultados, nodos hoja o **terminales**.

Existen varios algoritmos para la construcción de estos árboles, algunos de los más conocidos son ID3, C4.5, C5.0 y CART. De ellos, quizás el más famoso y del que derivan los demás, sea ID3, con un método de construcción **descendente y voraz** con atributos categóricos alcanzando la mayor *ganancia de información*.

Conviene hacer referencia al método CART, cuya implementación optimizada por la librería *scikit-learn* es la que hemos empleado en el proyecto. Este método es compatible con árboles de clasificación y regresión, aceptando tanto atributos como variables de decisión numéricas, creando árboles binarios con la mayor ganancia de información en cada nodo [4].

### Selección de Atributos

Para elegir cuál de los  $N$  atributos del conjunto de datos debe ser colocado en cada uno de los nodos se establecen unos criterios de *selección*.

Los más usados son:



1. **Entropía:** Medida de la pureza e incertidumbre. Cuanto mayor sea, menor será su pureza. El cálculo es [4]:

$$E(S) = \sum_{i=1}^c -P_i \log_2 P_i \quad (3.8)$$

2. **Ganancia de Información:** Al construir un árbol, debemos seleccionar los atributos que alcancen la *mayor* ganancia de información y menor entropía:

$$GI(Y, X) = E(Y) - E(Y, X) \quad (3.9)$$

3. **Índice Gini:** Al contrario que la GI, este índice favorece la existencia de conjuntos de *gran tamaño* y se calcula como la diferencia entre la unidad y la suma cuadrada de probabilidades para cada clase.

$$Gini = 1 - \sum_{i=1}^c (P_i)^2 \quad (3.10)$$

Debemos tener en consideración que la implementación del paquete *scikit-learn* toma como medida de selección *predeterminada* el **índice Gini** [42].

## Random Forest

Este algoritmo permite la resolución tanto de problemas de clasificación como de regresión, siendo descrito por primera vez por *Tin Kam Ho en 1995* [25], cuya **esencia** radica en la construcción de múltiples árboles dentro de un subespacio aleatorio del conjunto de variables de entrada.

Un modelo *Random Forest* consiste en una colección de clasificadores con estructura *en árbol*:  $h(x, \Theta_k)$ ,  $k = 1, \dots$  donde el conjunto de  $\Theta_k$  son vectores aleatorios independientes y con distribución **idéntica** y cada árbol emite un *único voto* hacia la clase más popular de la entrada  $x$  [8].

Aunque la definición previa hace referencia al modelo clásico, podemos hacer extrapolación hacia **árboles de regresión** y obtener predicciones en dicho ámbito [4].

Podemos ver un ejemplo de construcción de este modelo en la figura 3.2

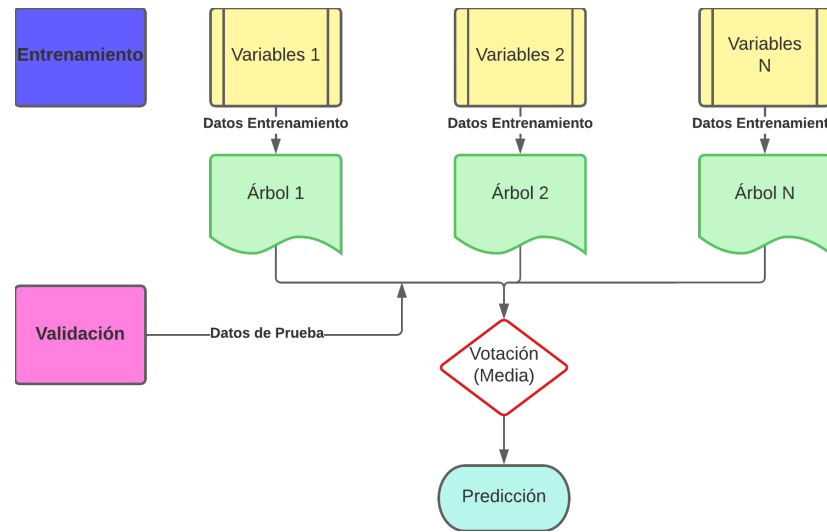


Figura 3.2: Ejemplo de Random Forest

Algunos problemas recientes, como los **diagnósticos y procedimientos médicos**, tienen demasiadas variables de entrada, aunque cada una de ellas tan sólo contiene una pequeña cantidad de información. En estos casos, un árbol de decisión o regresión único ofrecerá resultados tan sólo un poco mejores que la selección aleatoria. Sin embargo, la *combinación* de árboles con un crecimiento usando características aleatorias puede **mejorar la precisión** [8].

Generalmente, existen **dos técnicas** para combinar varios árboles en uno solo [4]:

1. **Bagging**: En este método, propuesto por Leo Breiman [7], se crean múltiples versiones de un predictor para alcanzar un único modelo agregado. Mediante el *promedio* o la *moda*, se produce la salida. Cada una de las versiones se forman realizando muestras aleatorias en tiempo de ejecución del conjunto de datos de entrenamiento y usándolas como un nuevo conjunto.
2. **Boosting**: Al contrario que en la metodología anterior, los modelos son construidos de forma *secuencial*, de manera similar al proceso de aprendizaje en el *descenso de gradiente*, donde los pesos se centran en las instancias con predicciones incorrectas, buscando impulsar las predicciones de casos *desafiantes*. Por tanto, las predicciones no provienen de una votación o promedio igualitario, sino *ponderado*.

## Redes Neuronales Artificiales

Las **redes neuronales artificiales** son sistemas complejos basados en modelos matemáticos inspirados en la función, estructura y el procesamiento de la información del cerebro humano y el sistema nervioso central [26].

De esta forma, una red es concebido como un sistema capaz de *aprender* a predecir salidas tras realizar numerosas iteraciones.

Durante el proceso de aprendizaje, la salida de cada una de las neuronas será tomada como la entrada de la siguiente, tras pasar por una *función umbral*. Para obtener la mejor *precisión*, se aplica un algoritmo de entrenamiento con **propagación hacia atrás**.

Este proceso de aprendizaje se ha convertido en una valiosa herramienta para encontrar relaciones lineales y no lineales *complejas*, de forma similar a una **caja negra**, y con múltiples aplicaciones en ingeniería, biología, matemáticas y medicina [37].

### Perceptrón

El concepto de perceptrón fue derivado del modelo descrito por McCulloch y Pitts [34], los cuales simularon los principios de las células nerviosas en un modelo matemático, comprobando que una única neurona es **capaz de realizar funciones lógicas**.

$$\sum w_i x_i \geq b \quad (3.11)$$

Este modelo consiste en *dos capas*. La primera, conocida como **capa de entrada**, recibe los estímulos y los transmite hacia la última capa, conocida como **capa de salida**. En ella, todos los estímulos de entrada son multiplicados por sus pesos respectivos y sumados junto al *sesgo*. Finalmente, la función de activación es llamada con estos parámetros

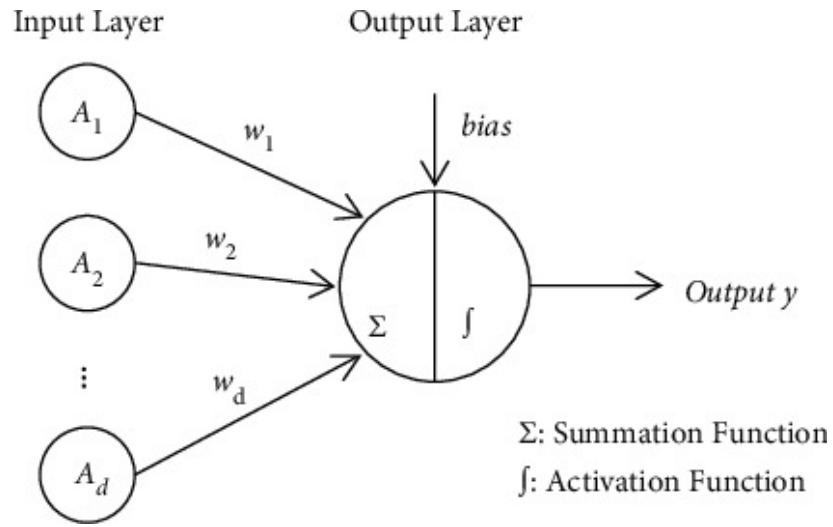


Figura 3.3: Funcionamiento del Perceptrón. [26]

### Perceptrón Multicapa

Para una mejora en la resolución de *problemas no lineales*, Hetch-Nielsen propuso una variante del perceptrón, añadiendo capas adicionales de neuronas entre la **entrada** y la **salida** [24].

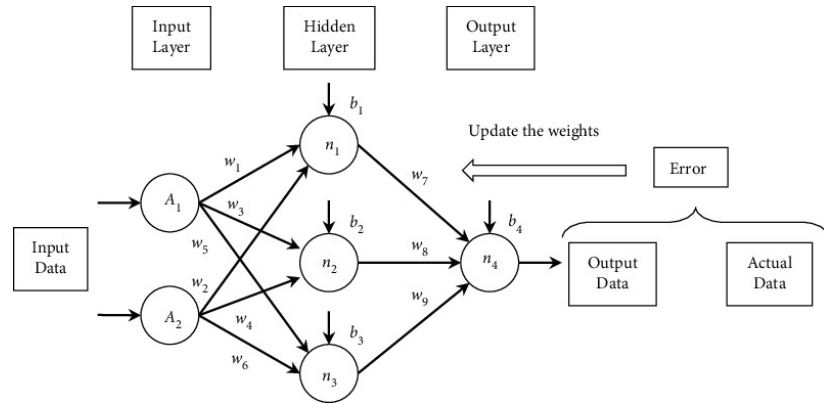


Figura 3.4: Perceptrón Multicapa [26]

En este modelo, diferenciamos dos componentes principales: *neuronas* y *conexiones*. Las neuronas serán las unidades de procesamiento, y cada conexión tendrá un peso y/o un sesgo correspondiente. Además, las capas ocultas tratarán de ser independientes del entorno, motivo por el cual reciben el nombre de *capas ocultas*.

Al igual que en el perceptrón, el entrenamiento se realiza mediante un proceso de **retropropagación**, realizando correcciones a los pesos de los nodos para minimizar el error de la salida, dado por [23]:

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n) \quad (3.12)$$

Y usando un **descenso de gradiente** obtenemos el cambio en cada peso [23]:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \quad (3.13)$$

Métrica	Descripción
<i>Hora de Comienzo</i>	Retraso o anticipo en el comienzo de la actividad quirúrgica.
<i>Tiempo de Recambio</i>	Tiempo entre cada uno de los pacientes.
<i>Uso del bloque</i>	Proporción del tiempo asignado que se dedica a la realización de la actividad.
<i>Tasa de Cancelaciones</i>	Las suspensiones o cancelaciones suponen tiempo de actividad desaprovechado.

Tabla 3.1: Métricas de Quirófano

## 3.2. Optimización en Planificación

### Métricas de Rendimiento

Antes de desarrollar cualquier medida que tenga como objetivo alcanzar una mejora de **rendimiento en la planificación quirúrgica (OR performance)**, debemos definir qué entendemos por rendimiento y, en su caso, *cómo medirlo*.

Algunas de las más importantes son las referidas en la tabla 3.1.

No obstante, las métricas usadas para cuantificar el rendimiento de las medidas de optimización en quirófano son muy diversas a lo largo de la literatura. Muchos artículos focalizan la eficiencia en aspectos meramente cualitativos, mientras que otros se centran en mejorar las métricas más objetivas, buscando la minimización del malgasto de recursos y la correcta gestión del tiempo [50].

Además, debemos tener en consideración que, mientras que estas medidas de rendimiento personalizan la estructura del problema, también limitan su tamaño, pues cuanto mayor es el número de los criterios de evaluación, la estructura será de mayor tamaño y complicación [22].

En nuestro caso, optaremos por **maximizar** la tasa de utilización del sitio quirúrgico y **minimizar** los tiempos de espera, teniendo en cuenta las restricciones en cuanto a la ponderación de su prioridad, que definiremos formalmente en apartados posteriores de la memoria.

Eficiencia	Calidad
Maximizar utilización y reducción de retrasos y esperas	Calidad en cuidados
	Seguridad del paciente
	<i>Buen hacer</i> profesional.

Tabla 3.2: Eficiencia y Calidad

Podemos resumir, por tanto, que un correcto rendimiento debe satisfacer requisitos de **eficiencia** y **calidad** [49].

El objetivo de nuestro proyecto será **minimizar** una función de coste para asignar  $N$  casos quirúrgicos a un conjunto de *quirófanos disponibles* considerando como **restricción** el tiempo disponible en cada sala.

Este enfoque es similar al propuesto en [14], donde se modela el problema según un criterio de **MIP**<sup>3</sup> y **planificación de máquinas en paralelo**.

De forma preliminar a explicar el proceso de diseño e implementación de la solución, conviene reseñar algunos conceptos teóricos en referencia a los *patrones* y *metaheurísticas* empleadas.

## Programación Lineal Entera Mixta (MIP)

La programación **matemática**, en especial la que incluye restricciones basadas en enteros junto a otras (*MILP*), es una de las más usadas para modelar problemas de planificación debido a su rigurosidad, flexibilidad y extensibilidad [15], pudiendo ser usadas por procesos **simples** que incluyen una única etapa y unidad, hasta procesos complejos de varias etapas y objetivos.

La naturaleza de los problemas de planificación es inherentemente **combinatoria**, dado el gran número de decisiones *discretas* a tener en cuenta (localización, equipamiento, tiempo...), lo que incrementa el tiempo de resolución **exponencialmente** a medida que aumenta el tamaño del problema (*NP-Completo* [18]).

## Formulación Matemática

Para formular un **modelo matemático de planificación**, el mayor problema a resolver es *cómo representar el tiempo*.

<sup>3</sup>**MIP**: Programación Lineal Entera Mixta.

En concreto, contamos con dos horizontes: *modelos continuos* y *modelos discretos*:

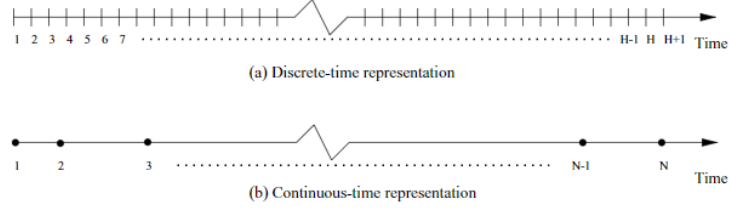


Figura 3.5: Representación discreta y continua del tiempo. Fuente: [15]

Dado que dividiremos el tiempo quirúrgico disponible en **ventanas**, nos centraremos en formular el problema utilizando el **modelo discreto**. Según éste, las variables de decisión  $W_{ijt}$  se introducen para representar que una tarea  $i$  puede o no ser procesada en la unidad  $j$  durante el intervalo de tiempo  $t$ :

$$\sum_{i \in I_j} \leq 1, \forall j \in J, t \in T \quad (3.14)$$

Por otro lado, si una tarea comienza a ejecutarse en una unidad dentro de ese intervalo de tiempo, **ninguna otra** puede ejecutarse en la misma unidad hasta que ha sido completada [15]:

$$\sum_{i' \in I_j} \sum_{t'=t}^{t+\alpha_{ij}-1} W_{i'jt'} - 1 \leq M(1 - W_{ijt}), \forall j \in J, i \in I_j, t \in T \quad (3.15)$$

## Heurísticas

Tal y como definió Coen [29]: “ *El rol de la ingeniería es: Haz lo que creas que representa la mejor práctica en el momento de la decisión. La definición del método de ingeniería depende de la heurística... Toda la ingeniería es heurística.* ”

Dado a que el problema estudiado puede ser **similar** al paradigma de la mochila y es idéntico a la planificación paralela, hemos implementado *dos heurísticas* modificando el enfoque de [30] y ajustándolo a nuestras restricciones.



La primera de ellas se basa en el **mayor tiempo de procesado** (LPT<sup>4</sup>) y la segunda en la relación con la **fecha de vencimiento más temprana** (ratio LPT/EDD<sup>5</sup>).

---

**Algoritmo 1** Heurística LPT
 

---

```

 $Q \leftarrow$  Set de cirugías sin programar
 $f_k^d \leftarrow$  Suma de los tiempos quirúrgicos que han sido programados en el
quirófano  $k$  el día  $d$ .
 $U \leftarrow null$ 
Ordena los elementos de  $Q$  por prioridad.
Ordena los elementos de  $Q$  en orden creciente de duración.
 $d^* \leftarrow 1$ 
while  $d^* \leq d$  do
   $flag \leftarrow True$ 
  while  $flag$  do
     $q \leftarrow Q.pop$ 
     $k \leftarrow \min_{f_{k,t+q,t}^{d=d^*}}$ 
    if  $Q.end$  then
       $flag \leftarrow False$ 
    end if
    if  $k == null$  then
       $Q.append \leftarrow q$ 
      continue
    end if
     $U[d][k].append \leftarrow q$ 
  end while
   $d^* = d^* + 1$ 
end while
return  $U$ 

```

---

## Algoritmo Genético

Un algoritmo genético es una **metaheurística** inspirada en los procesos naturales de selección, evolución y herencia. En ellos, se combina la *supervivencia del más adaptado* sobre las estructuras, así como con un intercambio de información estructurado y aleatorizado, formando un **algoritmo de búsqueda** inspirado en las mecánicas de evolución *humana* [19].

---

<sup>4</sup>**LPT**: Longest Processing Time

<sup>5</sup>**EDD**: Earliest Due Date

**Algoritmo 2** Heurística EDD

---

$Q \leftarrow$  Set de cirugías sin programar  
 $f_k^d \leftarrow$  Suma de los tiempos quirúrgicos que han sido programados en el quirófano  $k$  el día  $d$ .  
 $P \leftarrow Q.prioridades$   
 $U \leftarrow null$   
 Ordena los elementos de  $Q$  en orden **decreciente** de  $t_i/p_i$   
 Ejecutar el bucle *While* de 1.  
**return**  $U$

---

Tras un proceso de **codificación** de los individuos en *cromosomas*, obtendremos un *genotipo* generado, en principio, de forma aleatoria. Se aplicarán operadores de **selección**, **evaluación**, **cruce** y **mutación** a dicha población hasta que se satisfaga un *criterio de parada*, devolviendo al mejor individuo como solución.

Según los autores [19, 30], el mayor desafío para resolver un problema con este enfoque será el diseño de los cromosomas y la función de evaluación.

**Algoritmo 3** Estructura AG

---

$P \leftarrow iniciaPoblación$   
 $mejor \leftarrow fitness(P)$   
 $N = 0$   
**while**  $N \leq ngen$  **do**  
      $padre1, padre2 \leftarrow seleccion(P)$   
     **if**  $probcx \geq rand$  **then**  
          $hijo1, hijo2 \leftarrow cruce(padre1, padre2)$   
     **end if**  
     **if**  $probmur \geq rand$  **then**  
          $ind \leftarrow mutar(ind)$   
     **end if**  
      $P \leftarrow actualizaPoblación$   
      $mejor \leftarrow fitness(P)$   
**end while**  
**return**  $mejor$

---

---

## Técnicas y herramientas

---

### 4.1. Metodología de Desarrollo

Como metodología de desarrollo de nuestro proyecto, hemos tomado un enfoque **ágil**, inspirado en Scrum [41].

Para ello, se han dividido las diferentes tareas en Sprints, coincidiendo la finalización de cada Sprint con una **reunión** de los tutores y el desarrollador, finalizando en la confección de un **entregable** (*release*).

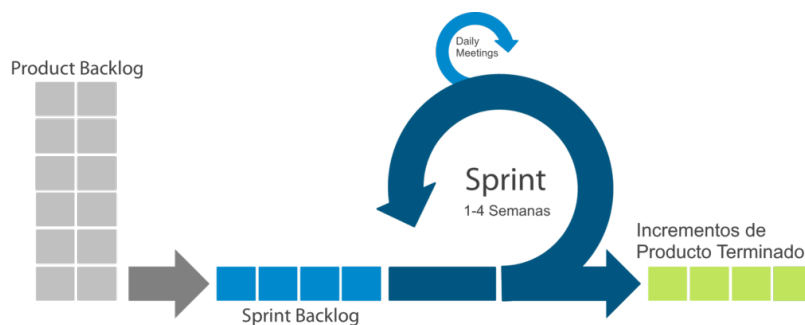


Figura 4.1: Funcionamiento de Scrum. Fuente:[48]

### Plataformas de Apoyo al Desarrollo

#### Repositorio y Git

El código fuente, así como la **documentación** (incluyendo la Memoria y los Anexos) se encuentran en un *repositorio Github* público, accesible desde: **Repositorio: GestorQuirófanos**



## 4.2. Elementos de Programación

### Lenguajes Utilizados

Como lenguaje de **programación**, hemos usado Python[52]. Python es considerado por muchos como uno de los lenguajes más útiles para la programación orientada al *machine learning* y *ciencia de datos*.

Python es un lenguaje de programación **interpretado y orientado a objetos**, cuya sintaxis hace énfasis en la legibilidad y depuración del código, ofreciendo la potencia y flexibilidad de los lenguajes compilados con una curva de aprendizaje *suave*. [1]

Además, podemos encontrar que varios manuales dedicados a estas materias [53] se enfocan directamente en la implementación de estos algoritmos usando este lenguaje, lo que sumado a su sencillez de aprendizaje, su carácter *open source* y la existencia de múltiples librerías y entornos de desarrollo dedicados a cada una de las fases del proceso de **Ciencia de Datos**, han hecho que nos decantemos por su uso durante el desarrollo de este proyecto.

La versión empleada para las pruebas y la ejecución en el equipo de desarrollo fue *Python 3.9.13*, instalado a partir de la distribución **Anaconda**, que es una distribución **libre y abierta** de los lenguajes R y Python enfocada principalmente a tareas de *ciencia de datos* y *aprendizaje automático*.

Por otro lado, el interfaz desarrollado ha sido una *aplicación web*, motivo por el cual se han añadido plantillas de estilo *CSS*, scripts en *JavaScript* y lenguaje de marcado *HTML*.



Figura 4.3: Librerías más usadas en Data Science

Para la redacción de la **memoria**, empleamos *L<sup>A</sup>T<sub>E</sub>X*, dada su *portabilidad* y la *facilidad* para ajustarse a los requerimientos de entrega.

## Entornos de Desarrollo Integrado (IDEs)

Para la redacción del código fuente e integración continua con el repositorio, se empleó el IDE **Visual Studio Code**, dado que contiene una gran variedad de **plugins** que facilitan tanto la redacción como la ejecución de código Python, así como la edición de HTML, JavaScript y CSS de cara al interfaz web.

Por otro lado, para la redacción de la memoria en  $\text{\LaTeX}$ , empleamos el IDE **Overleaf**, que se trata de una herramienta de publicación y redacción colaborativa **en línea**, enfocada a la edición y publicación de documentos científicos, de manera que la revisión de la memoria pudiese realizarse en *tiempo real* por parte del autor y los tutores. Además, nos permite integrar el desarrollo con **nuestro repositorio**, así como *enlazar la bibliografía* con un gestor bibliográfico.

Por último, se ha contado con el apoyo de **Mendeley Reference Manager** para almacenar las referencias y artículos empleados como apoyo bibliográfico en nuestro trabajo, permitiéndose el acceso a sus recursos gracias a la licencia proporcionada a los estudiantes de la Universidad de Burgos (*modalidad de acceso institucional*).

Tal y como podemos comprobar en la tabla 4.1, hemos variado de un entorno de desarrollo a otro en función de las características del código fuente, en función de las funcionalidades que queramos aprovechar de los mismos de cara al apoyo de nuestra redacción.

Lenguaje	IDE
Python	Visual Studio Code
HTML	Visual Studio Code
CSS	Visual Studio Code
JavaScript	Visual Studio Code
MySQL	MySQL Workbench
	phpMyAdmin
LaTeX	Overleaf
BibTeX	Mendeley Reference Manager

Tabla 4.1: Resumen de IDEs

## Librerías

Una **librería** no es más que un conjunto de *archivos de código* empleados para el desarrollo de software. En función de su finalidad, existen librerías para gran variedad de funcionalidades: cálculo matricial, manejo de ficheros, protocolos de comunicaciones, representación gráfica, algoritmia...

Tal y como hemos reseñado en apartados anteriores, uno de los aspectos más positivos de Python para este proyecto es la disponibilidad de *librerías específicas* para las labores de inteligencia computacional y análisis de datos.

De entre todas las usadas, que se resumen en la tabla 4.2, conviene **destacar**:

1. **Scikit-Learn**: Esta librería [42] es una de las gratuitas para Python. Cuenta con la implementación de varios algoritmos de clasificación, regresión, clustering y reducción de la dimensionalidad, convirtiéndose en una *herramienta básica* para programar sistemas de modelado y análisis de datos.
2. **DEAP**: Se trata de un *framework* para realizar labores de computación evolutiva [16], proveyendo a los desarrolladores del *pegamento esencial* para construir algoritmos evolutivos para sistemas sofisticados.
3. **Flask**: Framework de Python, encargado de realizar labores de *controlador* entre la lógica de negocio y la vista en una aplicación web. Se basa en el motor de renderizado de plantillas *Jinja2* y en la especificación WSGI de Werkzeug.
4. **Pandas**: Librería repleta de utilidades para realizar labores de pre-procesamiento, análisis y minería de datos. Su uso ha sido esencial a la hora de importar y exportar los conjuntos de datos empleados en las labores de explotación y experimentación de los modelos.
5. **Matplotlib**: Contiene funciones dedicadas a la representación de datos. Muy útil para evaluar el rendimiento de los modelos de inteligencia artificial y mostrar de forma empírica sus resultados en la documentación.

Librerías	SK-Learn	Pandas	NumPy	Matplotlib	DEAP	Flask
Preprocesado		X	X	X		
Análisis de datos	X	X	X	X		
Representación de datos	X		X	X		
Entrenamiento de modelos	X		X			
Evaluación de modelos	X	X	X	X		
Optimización			X		X	
Evaluación de optimización	X		X	X	X	
Métricas y Cálculo			X			
Integración API						X

Tabla 4.2: Librerías Python Empleadas

## 4.3. Integración y Despliegue

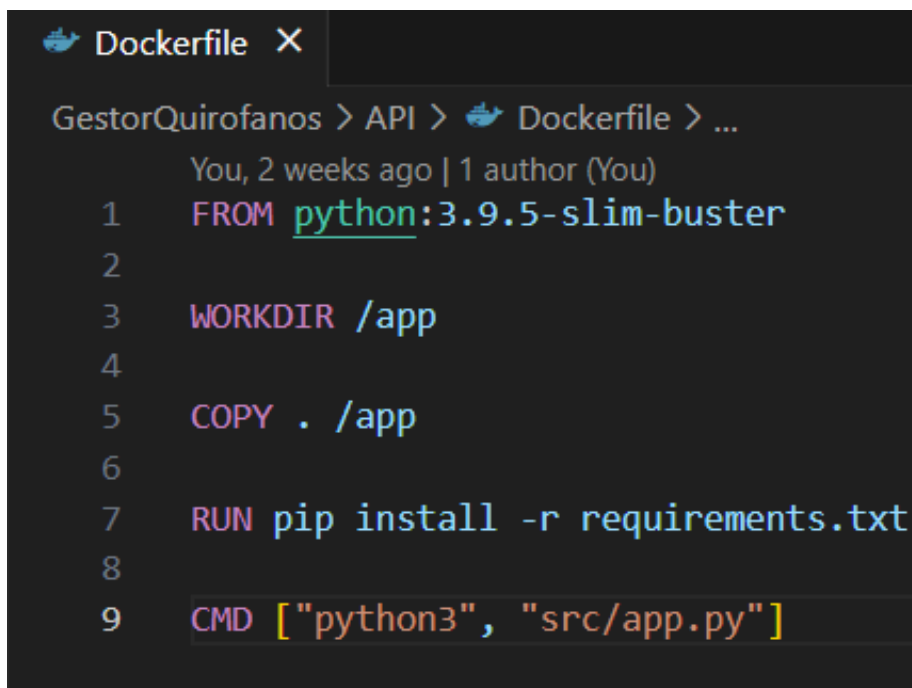
### Docker

Un contenedor, o *docker*, nos permite encapsular la lógica de una aplicación, incluyendo su código, librerías y dependencias funcionales, para su ejecución en un entorno virtualizado, sin necesidad de que los usuarios deban preparar su entorno de ejecución o desarrollo para ejecutar las funcionalidades del sistema.

Este tipo de solución se encuentra ampliamente extendida para el desarrollo y portabilidad de soluciones ML y aplicaciones distribuidas, sobretodo, desde el auge de las tecnologías de *computación en la nube*, ofreciendo una infraestructura idónea para el **despliegue** y el acceso remoto a estas funcionalidades.

Para su construcción, debemos instalar el cliente **Docker Desktop** y confeccionar un fichero *Dockerfile* que especifique a la herramienta las dependencias y la estructura del contenedor:





```
Dockerfile X
GestorQuirofanos > API > Dockerfile > ...
You, 2 weeks ago | 1 author (You)
1 FROM python:3.9.5-slim-buster
2
3 WORKDIR /app
4
5 COPY . /app
6
7 RUN pip install -r requirements.txt
8
9 CMD ["python3", "src/app.py"]
```

Figura 4.4: Ejemplo de fichero Dockerfile para construcción de un contenedor

## Amazon Web Services

Tal y como hemos señalado en apartados anteriores, el objetivo final de este proyecto será obtener una *interfaz* operativa que integre las funcionalidades y requerimientos desarrollados durante el transcurso del trabajo.

Para permitir el acceso *remoto* a las funcionalidades del sistema, hemos optado por desplegar los contenedores y la base de datos haciendo uso del paquete de *Cloud Computing* ofrecido por Amazon Web Services.

En concreto, han sido necesarias las siguientes *dependencias* para el despliegue:

- **Amazon RCS:** Para mantener el esquema de la base de datos relacional y los contenidos persistentes. Usamos el lenguaje MySQL y el motor InnoDB.
- **Amazon ECR:** Almacenamiento de los contenedores desarrollados a partir de Docker, para su posterior despliegue.
- **Amazon ECS:** Creación de un clúster y despliegue de cada uno de los contenedores a modo de *servicios y tareas*.

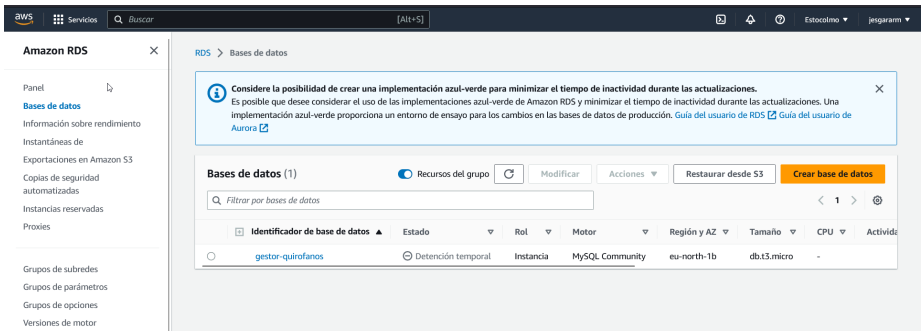


Figura 4.5: Panel de control de Amazon RCS

---

# Aspectos relevantes del desarrollo del proyecto

---

## 5.1. Obtención y Preprocesamiento de Datos

### Fuente de Datos

Se obtuvieron datos relativos a intervenciones quirúrgicas realizadas en el **Hospital Universitario Virgen del Rocío** de Sevilla, comprendidas entre las fechas *1 de Enero de 2016* y *1 de Noviembre de 2022*, con adecuada anonimización de los datos que garanticen su uso en el ámbito universitario.

Se obtuvieron datos relativos a las siguientes **especialidades quirúrgicas**:

1. Cirugía Plástica y Reparadora.
2. Cirugía Oral y Maxilofacial
3. Neurocirugía.
4. Cirugía General y del Aparato Digestivo.
5. Cirugía Ortopédica y Traumatología.
6. Otorrinolaringología.
7. Cirugía Torácica

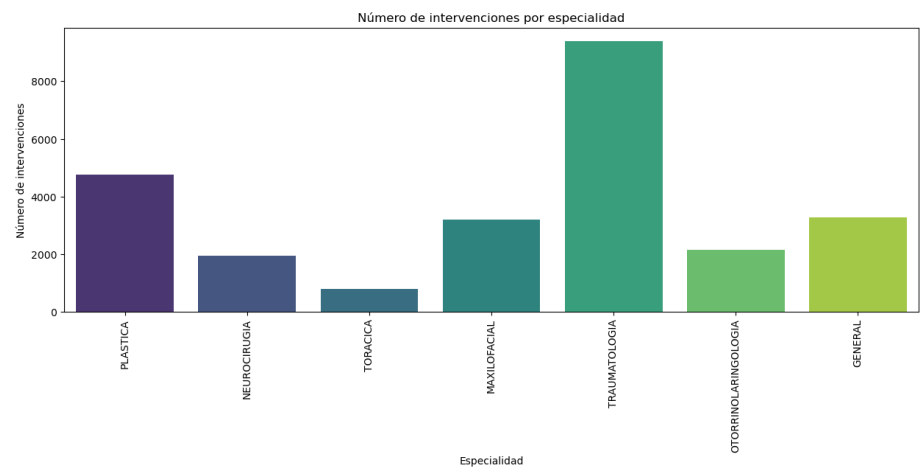


Figura 5.1: Número de Intervenciones Registradas por Especialidad

En total, se consideraron **25492 individuos** tras la realización de labores de adecuación e integración de las fuentes de datos.

Variable	Tipo
NHC	Entero
INTERVENCIÓN	Real
TIPO	Entero
TURNO	Entero
CARÁCTER ECONÓMICO	Entero
PONDERACIÓN	Entero
DURACIÓN	Entero
ESPECIALIDAD_GENERAL	Entero
ESPECIALIDAD_MAXILOFACIAL	Entero
ESPECIALIDAD_NEUROCIRUGIA	Entero
ESPECIALIDAD_OTORRINOLARINGOLOGIA	Entero
ESPECIALIDAD_PLASTICA	Entero
ESPECIALIDAD_TORACICA	Entero
ESPECIALIDAD_TRAUMATOLOGIA	Entero

Figura 5.2: Variables del Conjunto de Datos Base

Por otro lado, contamos con otro dataset, de menor tamaño y limitado a una única especialidad (*Cirugía Plástica y Reparadora*), que **amplía** el número de variables.

Variable	Tipo
NHC	Entero
INTERVENCIÓN	Real
TIPO	Entero
TURNO	Entero
CARÁCTER ECONÓMICO	Entero
PONDERACIÓN	Entero
DURACIÓN	Entero
Usuario (Sexo)	Entero
Asa	Entero
procedimiento (Con garantía  Sin garantía)	Entero
Código facultativo responsable	Entero
Código diagnóstico	object
Código especialidad	Entero
Código procedimiento	Real
Código prioridad asistencial	Entero
Código tipo de anestesia	Entero
Código tipo de cirugía	Entero
EDAD	Entero

Figura 5.3: Variables del Conjunto de Datos Ampliado

Conviene destacar la *relevancia* y el *desafío* que supone trabajar con **datos reales** a la hora de implementar modelos predictivos. Algunas de las características que han sido tomadas en consideración a lo largo del trabajo han sido:

- Presencia de atributos *missing* o inválidos (valores nulos, incongruencia de tipos...).
- Imprecisión de los *instrumentos de medida* (errores humanos en la recogida de datos, registros inapropiados...) que generan valores atípicos (**outliers**).
- Baja *validez interna* (valores de **error** sobre el conjunto de prueba más elevados) compensada por una elevada *validez interna* (métricas **estables** en extrapolación poblacional).

Tras obtener los **listados** del repositorio de datos relativos a intervenciones quirúrgicas en el hospital, se realizó una labor de *adecuación* de los mismos a nuestro entorno de trabajo. [35].

Figura 5.4: Aspecto de la Base de Datos antes de la edición.

Figura 5.5: Aspecto de los datos tras la limpieza y edición. Previo a codificación

Una vez obtuvimos nuestro dataset listo para la realización de labores de **minería**, procedimos a efectuar un *análisis preliminar*, donde cabe destacar que la *media* de duración de cada intervención **no varía** de forma estadísticamente significativa en función de la especialidad (*ANOVA-Test*<sup>6</sup>).

<sup>6</sup>Análisis de la varianza. Test estadístico realizado para valorar si existen diferencias significativas en las medias de los valores entre varios grupos

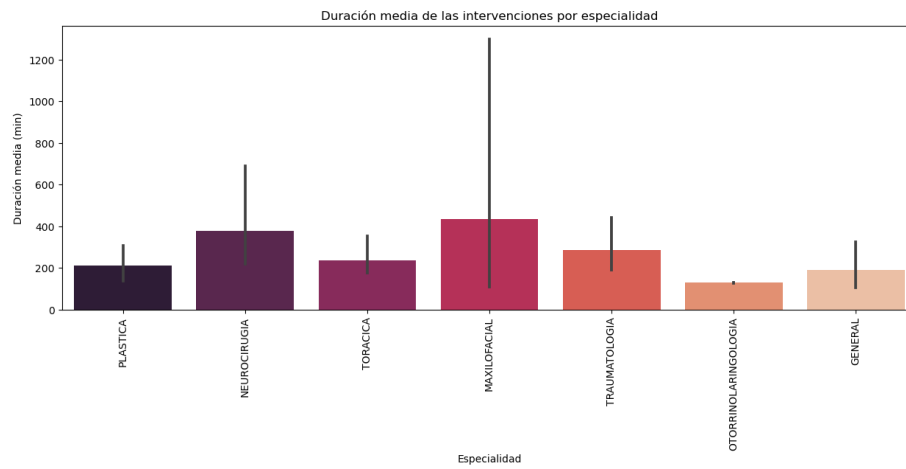


Figura 5.6: Media de duraciones de intervenciones por Especialidad

Por otro lado, efectuamos un estudio de la **correlación** entre algunas de las variables registradas, centrándonos en los valores devueltos por el *coeficiente de correlación de Spearman* [40]. Tras esta evaluación, encontramos que los valores referentes al **tipo, código de intervención y ponderación (*prioridad*)** son aquellos que, a priori, se encuentran más relacionados con la **duración**.

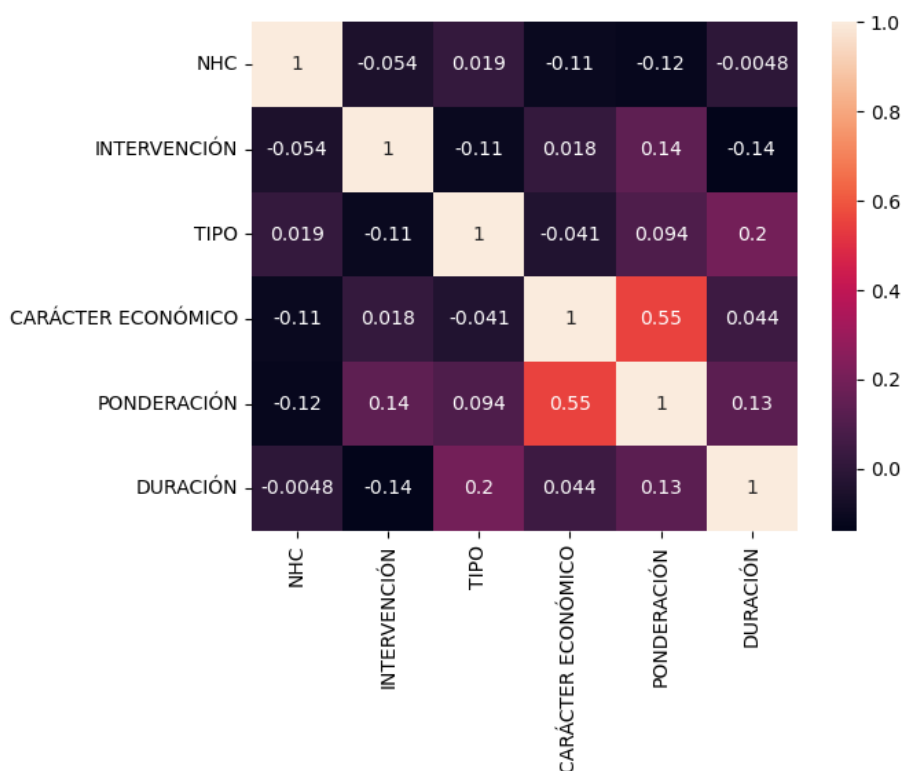


Figura 5.7: Matriz de Correlación de Spearman

## Preprocesamiento de Datos

La mayor parte de los algoritmos de aprendizaje supervisado, especialmente aquellos que emplearemos en nuestro estudio, requieren contar con un conjunto de variables **numéricas** como conjunto de *características*:

1. Para las variables **binarias**, se realizó una codificación a nivel de bit (0,1).
2. Para las variables **ordinales**, se asignó una escala *creciente* en función de la categoría.
3. Para las variables **nominales**, usamos la técnica de *One Hot Encoding*<sup>7</sup>, al tratarse de una de las técnicas más recomendadas para el manejo de

<sup>7</sup>Técnica que consiste en la división de una variable categórica en cada uno de sus posibles valores, realizando una asignación binaria en función del mismo a la instancia de entrenamiento.



variables categóricas en el entrenamiento de modelos de aprendizaje automático[45].

El seguimiento y proceso tanto de los procesos de **codificación** y **procesado**, así como de exportación posterior, se encuentra detallado en los Jupyter Notebooks:

- *./Datos/PreprocesadoDatos.ipynb*
- *./Datos/codificacionDatos.ipynb*

Por último, de entre todas las estrategias [13] existentes para lidiar con los valores **ausentes** durante la fase de procesamiento, optamos por la **eliminación** de instancias, dado que el carácter arbitrario de las labores de codificación y la *escasez* de variables correlacionadas dificultan la aplicación de métodos basados en *sustitución o interpolación*.

## 5.2. Aprendizaje Supervisado: Predicción

En este apartado se describe el procedimiento de exploración, análisis e implementación de diferentes modelos predictivos, basados en el paradigma de *aprendizaje supervisado* con el objetivo de obtener la predicción más **fidedigna** posible, de forma que pudiese ser empleada como entrada del modelo de optimización posterior.

### Dificultades Iniciales

Tras comenzar la implementación de varios modelos con el **dataset principal**, observamos que las métricas de error devolvían unos valores *desproporcionados*, en torno a miles de minutos de error medio, muy diferentes otorgados por otros estudios [51].

Esta situación nos hizo sospechar de la presencia de numerosos *outliers* en nuestro conjunto de datos inicial.

Este problema fue solucionado haciendo uso del IQR<sup>8</sup> [5].

Tras la aplicación de esta medida, obtuvimos valores de error (*reducción a menos de un 10 % de los previos*) mucho más coherentes y concordantes de la literatura, constituyendo un buen punto de partida para nuestro ensayo.

### Explotación de los Modelos

Los modelos de aprendizaje han sido aplicados siguiendo las bases teóricas especificadas en apartados anteriores y basándonos en la implementación de la librería *scikit-learn* [1], ejecutándose en ambos conjuntos de datos.

Para el ajuste de los hiperparámetros, empleamos un modelo de validación cruzada de tipo *GridSearch*<sup>9</sup>.

Comenzamos aplicando un modelo simple de **regresión múltiple**, donde encontramos diferencias entre ambos conjuntos, disminuyendo el RMSE un 20 % e incrementando un 50 % el índice  $R^2$ .

No obstante, tras aplicar un **árbol de regresión**, obtuvimos una *reducción de la raíz del error cuadrático medio* ligeramente superior al 20 %, con escasas diferencias entre ambos datasets.

---

<sup>8</sup>Técnica de selección de datos del conjunto de entrenamiento que escoge aquellas instancias con valores de la variable dependiente comprendidos entre 1.5 veces por debajo y por encima de los cuartiles primero y tercero, respectivamente.

<sup>9</sup>Técnica de validación cruzada incluida en *scikit-learn*, consistente en extraer los mejores valores y combinaciones introducidos en la cuadrícula de parámetros.

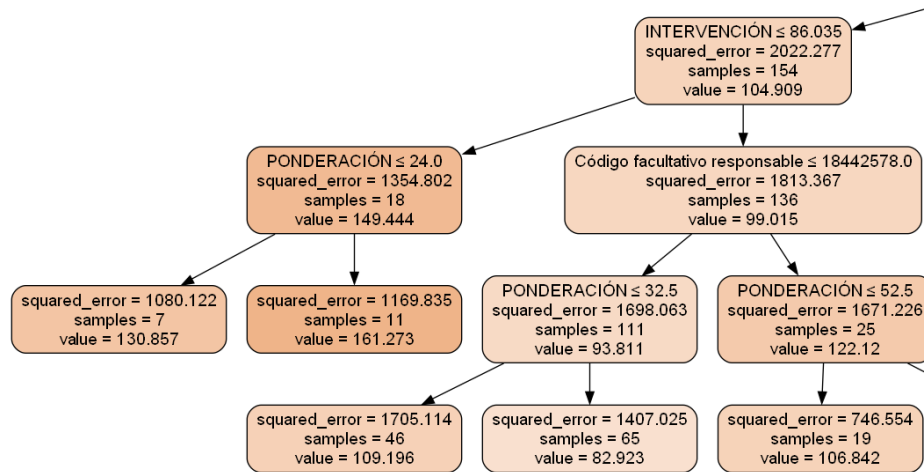


Figura 5.8: Sección del Árbol de Regresión Construido

La ejecución de KNN<sup>10</sup> y MLP<sup>11</sup> ofrecieron métricas de error son *similares* a las otorgadas por el algoritmo de regresión lineal, con un tiempo de ejecución **sustancialmente mayor**.

Los resultados ofrecidos por el *ensemble* de **random forest** fueron muy similares a los devueltos por el árbol de regresión, tanto en métricas de error como en tiempo de ejecución sobre el conjunto de validación.

Por último, tratamos de convertir el planteamiento del problema en uno de *clasificación*, dividiendo la duración en intervalos, pero las medidas de validez (*precisión*, *AUC*...) y la escasez de ejemplos en la literatura que empleasen este enfoque nos hicieron centrarnos en el enfoque inicial.

## Comparación y selección del modelo

Tras el entrenamiento y validación de los algoritmos de aprendizaje, comparamos las métricas calculadas y seleccionamos aquel modelo que mejor se ajusta a los **objetivos** planteados en el proyecto.

<sup>10</sup>K-Vecinos más cercanos, con K=14 y distancia de Manhattan ( $\sum_{i=1}^n |p_i - q_i|$ )

<sup>11</sup>Perceptrón Multicapa

<i>Modelos</i>	<i>MSE</i>	<i>RMSE</i>	<i>MAE</i>
Regresión Lineal	3461.43	58.83	43.27
Árbol de Regresión	2143.10	46.29	<b>33.48</b>
KNN	3084.32	55.54	<b>39.82</b>
Random Forest	2364.55	48.63	<b>34.75</b>
Perceptrón Multicapa	2804.00	52.96	40.46

Tabla 5.1: Comparativa de algoritmos ML

En base a estos resultados<sup>12</sup>, decidimos seleccionar el **árbol de regresión**, ya entrenado y guardado<sup>13</sup> para su uso como servicio por los clientes.

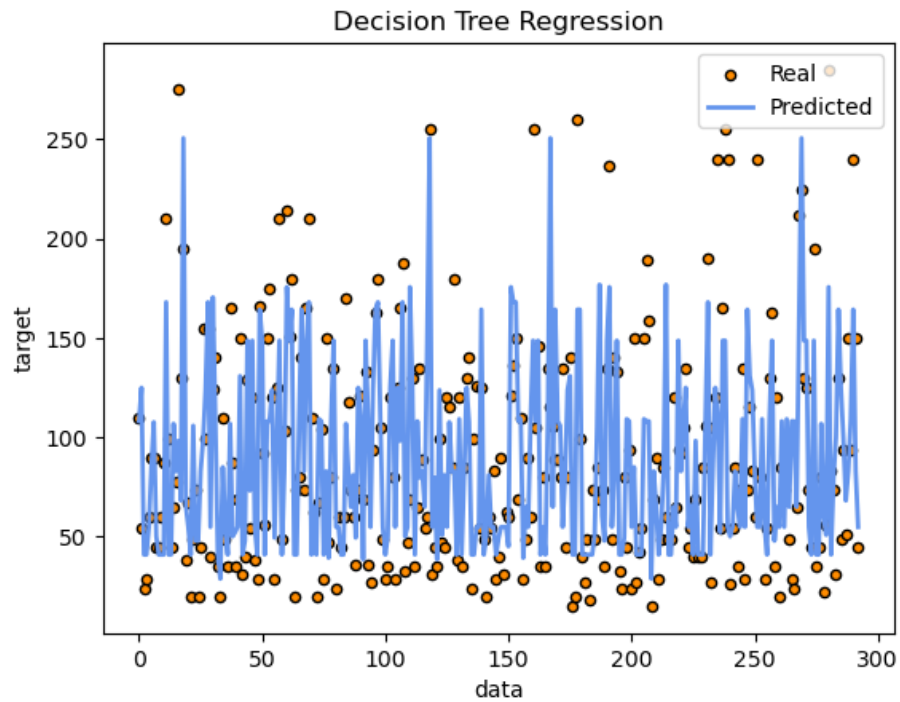


Figura 5.9: Representación de rendimiento del árbol de regresión

<sup>12</sup>Para más detalles, consultar los Jupyter Notebooks incluidos en la carpeta *Experimentación/Modelos* del repositorio.

<sup>13</sup>Las librerías *pickle* y *joblib* permiten serializar y deserializar objetos.

### 5.3. Optimización: Planificación y Gestión

Una vez seleccionado y entrenado el algoritmo de predicción, pasamos a diseñar y evaluar diferentes alternativas para optimizar el sistema de planificación quirúrgica.

Establecimos dos objetivos principales en este apartado:

1. **Maximizar** la prioridad de los casos asignados a las salas de operaciones.
2. **Minimizar** la aparición de huecos libres.

Modelamos el problema a través de la siguiente *función de fitness*:

$$\min \sum_{i=1}^N \sum_{j=1}^N \frac{L_{ij}}{P_{ij}} \quad (5.16)$$

Donde  $L$  representa los huecos libres en el quirófano  $i$  el día  $j$  y  $P$  la suma de prioridades de los casos asignados, debiendo cumplirse que:

$$L_i \geq 0 \quad (5.17)$$

Tal y como pudimos comprobar en la sección 3.2, se adaptaron en primer lugar dos heurísticas [30]. En el enfoque original, se consideraba una fecha límite para priorizar los casos en dos grupos, ordenándolos por tiempo en una cola. Nosotros dividimos los valores de prioridad en deciles y, una vez allí, reordenamos por tiempo antes de incluirlos en la cola.

Posteriormente, codificamos un **algoritmo genético**, siguiendo una disposición cromosómica *vectorial*, usando separadores para diferenciar quirófanos y días y generando individuos **válidos** para la población inicial. A la función de *evaluación* se le añadió una comprobación de *validez* y *distancia* [11].

Para la *selección*, empleamos el **torneo con elitismo**, como mecanismo de *cruce*, adaptamos el concepto de **cruce ordenado en dos puntos** al diseño de nuestro individuo [30] y para la mutación el **intercambio en un punto**.

Cabe destacar que introducimos en la población dos individuos generados a partir de las heurísticas anteriores, con el objetivo de **acelerar** la convergencia.

Por último, probamos una implementación del algoritmo *PSO*<sup>14</sup>, aunque sus valores de validez **no alcanzaron** las cotas mínimas de desempeño, probablemente por una inadaptación de nuestro modelaje a la metaheurística.

Modelos	Fitness	Tiempo
<i>LPT</i>	0,0043	0,001s
<i>LPT/EDD</i>	0,0049	0,001s
<i>Genético Random</i>	0,0025	20,520s
<i>Genético + Heurísticas</i>	0,0020	17,520s

Tabla 5.2: Resultados con N=100, 3 Quirófanos y 5 días

Como podemos comprobar, el algoritmo que mejor responde a nuestros objetivos, siguiendo un **tiempo de ejecución razonable** se basa en la combinación de las heurísticas con el enfoque genético, motivo por el cual ha sido el **seleccionado** para el desarrollo del producto.

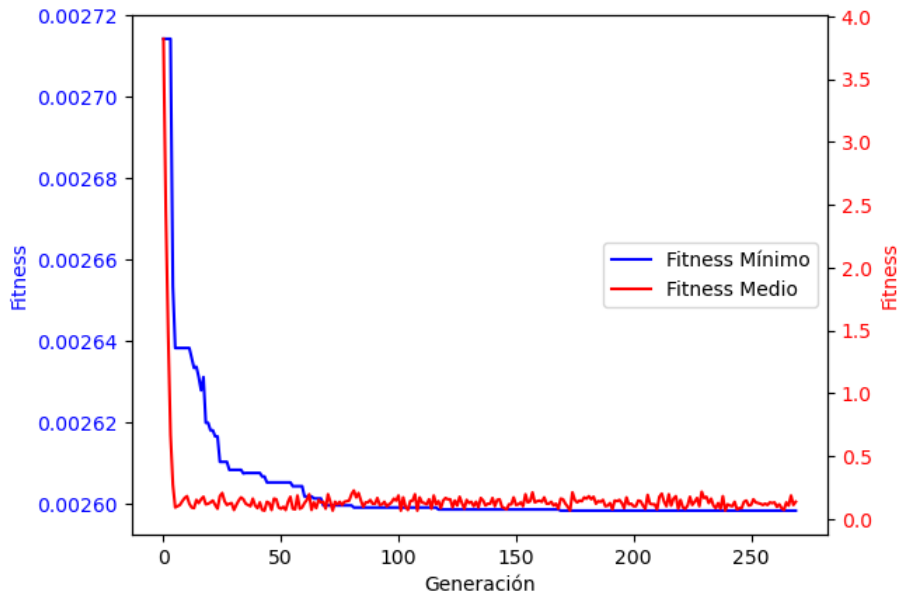


Figura 5.10: Evolución de la función de evaluación en el modelo seleccionado

<sup>14</sup>Optimización de enjambre de partículas: Algoritmo bio-inspirado iterativo, evocando el comportamiento de las partículas en la naturaleza.

## 5.4. Integración y Despliegue

Una vez explorados y seleccionados los algoritmos base para la consecución de los **objetivos** planteados en nuestro trabajo, decidimos integrar el desarrollo en una API.

Para su desarrollo, empleamos el framework *Flask* [21] para Python, encapsulando los siguientes **servicios** para los clientes:

1. Recepción y validación de listados en formato CSV o JSON.
2. Predicción de duración prevista de intervenciones.
3. Planificación quirúrgica en base al número de quirófanos y días introducidos por el usuario.
4. Entrega de resultados en formato JSON.

Uno de los principales problemas más frecuentes para el despliegue de herramientas de *Machine Learning* o cualquier otro tipo de solución computacional consiste en adaptar los equipos/servidores de los clientes al entorno empleado para su desarrollo (*lenguajes, librerías, OS...*).

Como solución a este paradigma, una vez desarrollada y lista para su explotación, planteamos para su **despliegue** recurrir a un *Docker* [36] que contenga tanto los recursos del sistema operativo como los lenguajes y librerías necesarios para poder proveer del servicio desarrollado, bien de forma local o remota, garantizando su adecuada **portabilidad y usabilidad**.

Así, una vez desplegado, podemos comprobar cómo recibe y devuelve de forma *satisfactoria* las peticiones de los clientes:

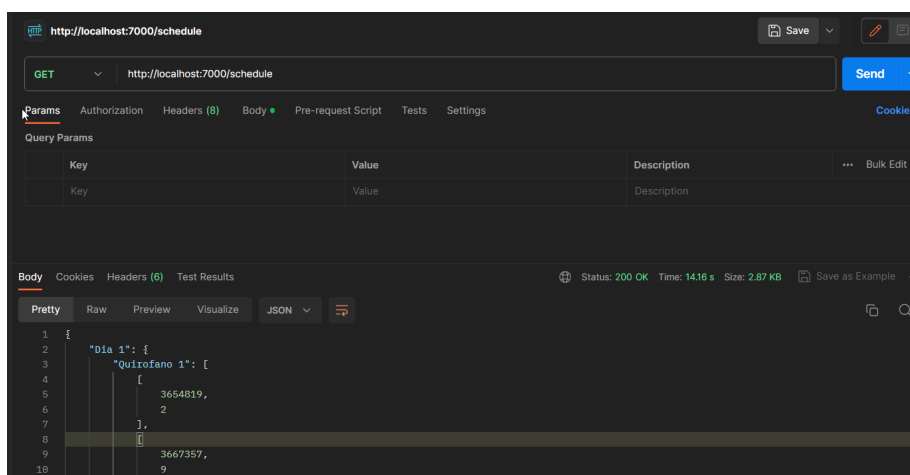


Figura 5.11: Ejemplo de comunicación con API para planificación (GET)

## Interfaz Web

De forma paralela al desarrollo de la API, permitiendo su uso y adaptación a cualquier interfaz por parte de los clientes, decidimos ejemplificar su funcionamiento elaborando una aplicación web que aproveche los servicios implementados y desplegados con anterioridad.

De nuevo, mediante el *framework* *Flask* y algunas de sus dependencias (*Flask-Login*, *Flask-MySQL*, *Flask-WTF*...), el motor de renderizado de plantillas *Jinja2*, el lenguaje de marcado HTML y un modelo de base de datos relacional MySQL, implementamos esta **solución** siguiendo el paradigma *Modelo-Vista-Controlador*.

En cuanto a las definiciones de **estilo** de las plantillas estáticas, se tomaron las definiciones *CSS* publicadas y ofrecidas por ***Bootstrap 5***, de código abierto.

En primer lugar, desarrollamos un sistema de **gestión de usuarios**, distinguiendo el rol de *administrador* y cediéndole los permisos para la creación, modificación y eliminación de nuevas cuentas, y permitiendo ejecutar las funciones de manera sencilla y *transparente* a través del interfaz:



Gestor Quirófanos

Gestión de Usuarios

Predicciones

Planificaciones

Jesús García Armario

</

Figura 5.12: Visualización del Panel de Gestión de Usuarios

En segundo lugar, ofrecemos la *persistencia* de las solicitudes (predicción o planificación), permitiendo a los clientes la creación, visualización o eliminación de las mismas.

Dentro del apartado de **creación de planificaciones**, será el *usuario* quien deberá indicar el número de quirófanos, el tiempo de recambio entre cada paciente y los días a programar, como parámetros que se envían al algoritmo distribuido.

Durante este nivel del desarrollo, destaca el *formato* de cara a la visualización de los datos calculados en los terminales de los clientes, tanto los listados de pacientes junto a la **duración**, como las planificaciones y su línea temporal.

Panel de Predicciones			
Bienvenido Jesús García Armario			
Añadir Predicción Volver			
DURACION	ESPECIALIDAD	ID	PRIORIDAD
107.25	NEUROCIRUGIA	2381056	0
73.73076923076923	PLASTICA	1466811	23
162.64462809917356	GENERAL	1882304	232
219.5	TRAUMATOLOGIA	1514204	20
149.01123595505618	TRAUMATOLOGIA	1675268	0
234.44444444444446	NEUROCIRUGIA	3667357	9

Figura 5.13: Visualización de un listado de predicciones



Figura 5.14: Visualización de un ejemplo de planificación

---

## Trabajos relacionados

---

A la hora de revisar la literatura, hemos encontrado un número considerable de trabajos que tratan nuestro planteamiento, aunque siempre dividiendo la estructura en los dos interrogantes iniciales: *predicción* y *planificación*.

Ha resultado especialmente útil la revisión de *Martínez et al., 2021* [32], donde encontramos las tendencias de diferentes técnicas clásicas de ML, sirviéndonos como punto de comparación con los resultados obtenidos en nuestro enfoque.

Por otra parte, hemos encontrado alternativas que tratan de ir un paso más allá e integrar la predicción continua dentro del propio quirófano, permitiendo a los gestores tomar decisiones en tiempo real, haciendo uso de redes neuronales [27].

Desde el punto de vista de la **gestión**, hemos encontrado enfoques muy diversos, donde cabe destacar la aportación de *Lin et al., 2020* [30], cuya implementación del algoritmo genético ha sido base para nuestra resolución.

Destacan en este campo las modelizaciones que tienen en cuenta diversos recursos que no han sido considerados en este estudio: personal de enfermería [12], situaciones de emergencia [39], disponibilidad de camas en UCI/Recuperación [10] o todos los recursos humanos y materiales disponibles [3].

Se deja en los siguientes apartados una revisión del estado del arte actual, así como la evolución y las tendencias actuales, tanto en investigación como en operatividad dentro del día a día hospitalario.

## 6.1. Machine Learning y Predicción del Tiempo Quirúrgico

### La tarea de predecir

Planificar quirófanos consiste en asignar bloques de tiempo a cada uno de los pacientes que forman parte de la cartera de servicios de un hospital, de manera que se lleven a cabo las intervenciones maximizando el uso de recursos y minimizando las pérdidas.

Para ello, se han empleado múltiples herramientas cuantitativas que sirven como apoyo la toma de decisiones de los gestores a lo largo de la literatura, aunque a día de hoy prevalece la **estimación basada en la experiencia** [6] frente a la exploración de nuevas iniciativas basadas en el aprendizaje máquina.

A pesar de su importancia, predecir la duración de una intervención no es fácil, dada la enorme variabilidad de situaciones que pueden suceder en el día a día del hospital.

El código ligado al procedimiento ha sido identificado como el factor más significativo a la hora de predecir la duración de una intervención [51], aunque éstos pueden presentar mucha variabilidad en función de la complejidad de la misma.

Actualmente, son muchos los hospitales que estiman la duración de una intervención recurriendo a las medias históricas para los mismos grupos diagnósticos a la hora de planificar las cirugías [56], aunque estas aproximaciones han resultado ser ineficaces y se trasladan en un uso inadecuado de los recursos hospitalarios cuando se trasladan a medidas de planificación [51].

### Evolución del estado del arte

Si tratamos de *minimizar el error humano* a la hora de estimar la duración de una intervención quirúrgica determinada, existen varios estudios que aplican diversos algoritmos **predictivos** a este problema, y comparan su rendimiento entre ellos y respecto al cálculo aplicando exclusivamente la experiencia del gestor.

Durante las últimas décadas, se han empleado gran cantidad de técnicas estadísticas y de aprendizaje máquina, desde regresión lineal, a técnicas de comparación de medias, aproximaciones bayesianas, redes neuronales y

métodos basados en árboles. Sin embargo, pese a la mejora creciente en la precisión, muchos de estos modelos siguen otorgando errores de predicción especialmente altos, debido en gran parte a la escasez de variables disponibles para su análisis en los conjuntos de datos [51].

Es por ello que se recomienda recopilar y añadir datos que afecten a las características individuales de los pacientes, así como a la mayor parte del personal implicado (cirujanos, anestelistas...), pues pueden influir en la duración final del acto quirúrgico [56].

## Ejemplos de Aplicación

A lo largo de la literatura, son varios los estudios centrados en analizar conjuntos de datos individuales y explorar diferentes modelos predictivos que atañen la duración de una determinada intervención quirúrgica.

Por ejemplo, en un estudio reciente [33] basado en la población pediátrica, llegaron a la conclusión de que los algoritmos basados en **árboles** para la predicción basada en variables categóricas (sexo, raza, edad, diagnóstico, riesgo anestésico...) obtienen menor error predictivo que el uso de las técnicas convencionales de predicción basadas en la experiencia.

Por otro lado, en otro estudio que exploró hasta nueve especialidades quirúrgicas [32] e incorporó variables como el identificador del cirujano responsable y su experiencia, comparó cuatro modelos de ML (Regresión Lineal, SVM, Árboles de Regresión y Bagged Trees), obteniendo **mejores métricas** que los estudios previos en todos los modelos, concluyendo que los mejores resultados se obtuvieron con Bagged Trees.

## Consideraciones Finales

Actualmente, el método más común empleado en los sistemas sanitarios se basa en **asumir** las decisiones de los cirujanos, las cuales se basan en su *experiencia previa* con intervenciones similares. No obstante, los profesionales tienden a **evitar riesgos** y presentan grandes limitaciones en su capacidad para estimar la duración, resultando en *sobreestimación* o *infraestimación* en torno al 40 % de los casos [26].

Hasta el momento, podemos afirmar que los algoritmos de ML son capaces de predecir la duración de los quirófanos, aunque no hay ninguna solución definitiva, y el rendimiento de los modelos depende, fundamentalmente, del conjunto de datos que sirva como objeto de entrenamiento, así como al servicio de aplicación y sus características intrínsecas.

En nuestro proyecto, construiremos un conjunto de datos de entre los disponibles y trataremos de explorar las diferentes alternativas propuestas en el estado del arte para predecir, con el menor índice de error, la duración de cada intervención de cara a planificar con el mejor aprovechamiento de recursos posible.

## 6.2. Optimización en Planificación

Tal y como hemos reseñado en apartados anteriores, planificar quirófanos consiste en asignar recursos hospitalarios a casos quirúrgicos individuales en función del tiempo estimado para completar la cirugía, por lo que muchos autores han comparado su resolución como una extensión de un problema de Job Shop, proponiendo modelos de programación lineal para su resolución [44].

No obstante, los autores difieren en cada uno de los estudios, y parecen aplicar las técnicas de optimización que más encajan con la modelización de su problema, existiendo pocas revisiones en la literatura referentes a la planificación y programación de quirófanos [22].

### Características de los Pacientes

La mayor parte de los estudios existentes en la literatura dividen sus herramientas de planificación en función de la **electividad** de los pacientes: Electivos (*programados*) y No Electivos (*urgentes, emergentes, hospitalizados...*).

Debido a la incertidumbre del segundo grupo, éstos no forman parte de antemano de las planificaciones de los gestores, aunque su aparición imprevista genera que se les dé una prioridad máxima a su llegada y se desplace la de los otros grupos en el proceso [9].

Aunque existen estudios que tienen en cuenta a ambos grupos y desarrollan medidas de planificación, no existen estudios robustos en la literatura que **combinen** ambos casos, centrándose en uno de ambos, en función de las características de cada hospital [22].

Sin embargo, los investigadores tienden a centrar sus esfuerzos en desarrollar medidas de optimización considerando principalmente el primer grupo de pacientes, replanificando si es necesario en el caso de que lleguen casos urgentes o emergentes [39], teniendo en cuenta que la evaluación de cualquiera de los grupos debe reflejar tanto los tiempos de espera de los pacientes, así como el efecto de la sobrecarga de trabajo en los trabajadores y recursos del entorno hospitalario [22].

En nuestro trabajo, y siguiendo la estela de la mayor parte de los autores, nos hemos centrado en las medidas de optimización de pacientes **electivos**, dada la complejidad y la dificultad de obtención de datos en las fuentes consultadas que nos permitan asumir la incertidumbre de diseñar un

modelo que tenga en consideración a ambos grupos y ofrezca un rendimiento aceptable.

## Alcance de la Planificación

Si tenemos en consideración la bibliografía disponible hasta el momento, encontramos cómo la tarea de planificación llega a incluir, en algunos estudios, el uso de áreas como la Unidad de Reanimación Postanestésica o la de Cuidados Intensivos, como parte del proceso, pudiendo mejorar el rendimiento desde el punto de vista estratégico a largo plazo [22].

No obstante, la mayor parte de los autores no tienen en cuenta estas características y la mayor parte de los avances realizados en la materia se centran en el uso exclusivo del área quirúrgica, dejando para investigaciones futuras la inclusión del resto de facilidades hospitalarias en el cómputo final.

Por otro lado, los recursos del quirófano suelen ser reconocidos como una unidad, aunque algunos estudios como [12] tuvieron en consideración la **dotación de personal de enfermería** como parte de los recursos a optimizar en el quirófano, no llegando a obtener conclusiones relevantes sobre su importancia en el modelo, siendo las diferencias de rendimiento no significativas respecto a su inclusión tradicional en el bloque.

Es por estos motivos por los que hemos decidido centrar nuestros esfuerzos en la elaboración de un modelo de gestión centrado en la **sala de operaciones** y considerando los recursos disponibles como un único elemento indivisible, tanto en las tareas de predicción como en las de planificación.

## Técnicas de Optimización

Las medidas computacionales empleadas en gran parte de los estudios tienen como objetivo la minimización del tiempo de espera de los pacientes y el coste hospitalario, junto a la maximización del tiempo de utilización de la sala quirúrgica.

Podemos encontrar gran número de aproximaciones, dado que el problema a modelar es NP-Complejo (ILP). Desde metaheurísticas específicas para un entorno concreto, hasta alternativas genéricas como algoritmos genéticos, recocido simulado, colonias de hormigas, *particle swarm*, programación dinámica...

En una revisión reciente [22], la mayor parte de las aproximaciones fueron por *simulación basada en la experiencia* y usando modelos de programación entera mixta (MILP).



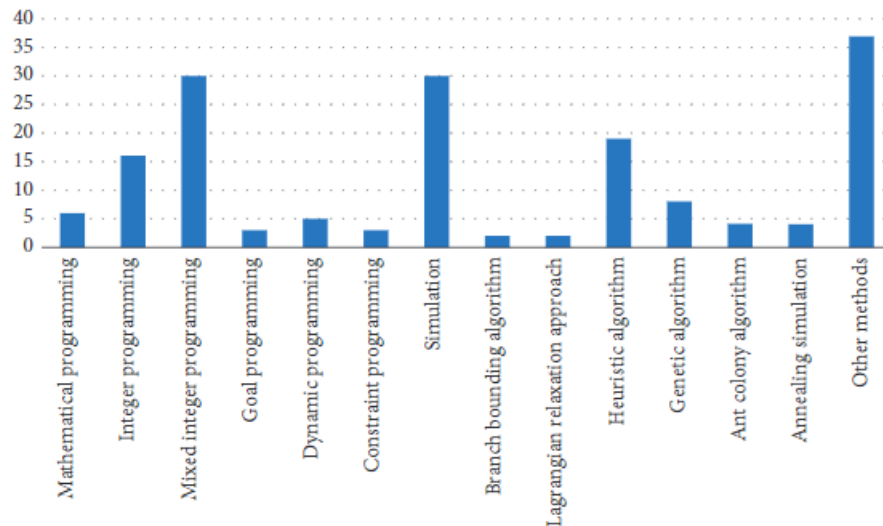


Figura 6.1: Técnicas de Optimización. Fuente [22]

## Aproximación a la Modelización

En función del foco a optimizar, encontramos en la literatura gran cantidad de aproximaciones a su modelización.

Podemos considerarlo como un problema de Job Shop Múltiple, o incluso como una evolución del problema de la mochila y procesamiento paralelo [30], aunque en todos, la finalidad será la misma: **asignar un bloque de tiempo, en una sala determinada, a un paciente dado** para la realización de una intervención quirúrgica.

Dependerá, por tanto, de las restricciones y los objetivos que marquemos durante la resolución, así como de la estrategia a implementar.

## Consideraciones Finales

Como podemos comprobar, una planificación óptima de los quirófanos constituye un rol de gran importancia en los hospitales, debido a la enorme cantidad de recursos que consumen.

A su vez, existen gran cantidad de variables a tener en cuenta, lo cual complica [30] la optimización, pudiendo llegar a incluir tantas variables (*disponibilidad de cirujanos, equipamiento, anestesistas, camas hospitalarias...*) como se desee.

En nuestro caso, nos centraremos en **asignar pacientes a huecos de quirófano**, asumiendo la disponibilidad del resto de factores, debido a que la literatura se centra en la resolución de este problema y a su dificultad de implementación.

Una vez formulado el problema con nuestras restricciones, probaremos diferentes modelizaciones y algoritmos de planificación durante el transcurso del proyecto, eligiendo finalmente el que nos ofrezca mejor rendimiento de cara a la implementación final de la herramienta.

---

## Conclusiones y Líneas de trabajo futuras

---

En este trabajo hemos tratado de ofrecer una herramienta a los gestores sanitarios para mejorar su desempeño en la planificación de pacientes en listas de espera quirúrgicas.

Para ello, hemos modelado un sistema de **predicción** y otro de **optimización**, basados en Inteligencia Computacional, así como integrado conocimientos de diversas ramas de la Ingeniería, adquiridos a lo largo de la titulación a través de las distintas asignaturas, además de recopilar nueva información específica y necesaria para la ejecución del proyecto.

Tras revisar los *resultados* y compararlos con otros estudios similares en la bibliografía, podemos concluir que:

- Los modelos de aprendizaje basados en *árboles* parecen ser los más adecuados para abordar el problema de predecir la duración de una determinada intervención quirúrgica, en base a las variables consideradas.
- Las medidas de error obtenidas **disminuyen** al incrementar el número de variables, aunque los valores devueltos constituyen un modelo *válido* y en *consonancia* con lo esperable tras revisar la literatura.
- La aplicación de un algoritmo genético ofrece resultados **muy buenos** y en **tiempo asumible**, comportándose mucho mejor tras la combinación con heurísticas contrastadas y empleando *restricciones blandas* que faciliten la evolución estocástica hacia la convergencia.

- La API desarrollada cumple con los objetivos planteados al principio del proyecto y los mecanismos de *integración* permiten su uso y distribución de forma fácil y asumible para los clientes.

Por otra parte, existen gran número de factores que no han sido explotados en el desarrollo del proyecto y sirven como piedra angular para posibles implementaciones futuras:

- Integración continua con SGBD hospitalarias y consultas automatizadas para obtener las variables de interés.
- Ampliación del *dataset* de entrenamiento, incluyendo factores subjetivos que pudiesen influir en la predicción (*código del facultativo, duración prevista...*).
- Desarrollo de un modelo de predicción, ajustado a cada especialidad, para predecir los intervalos *vacíos* entre cada caso quirúrgico (*limpieza, preparación...*).
- Integración en sistemas de gestión hospitalaria de RRHH para disponer del personal necesario en cada sala de operaciones que garantice el rendimiento óptimo propuesto por el algoritmo.
- Optimización paralela y secuencial, considerando el *circuito completo* de estancia hospitalaria, los recursos disponibles y los costes asociados.

Además, una vez desarrollada este *doble servicio*, se ha implementado un ejemplo de interfaz, a modo de *aplicación web*, siguiendo así la tendencia de numerosos sistemas informáticos de los distintos servicios sanitarios de este país, permitiendo su ejecución en un servidor **interno**, de acceso **restringido y distribuido** e **integrable** con el resto de herramientas disponibles en los servicios informáticos de cada región.

En definitiva, hemos desarrollado una **utilidad** que cumple con los objetivos propuestos al inicio del trabajo, siendo capaz de ofrecer un amplio abanico de posibilidades sobre las que trabajar hasta obtener un sistema completo, robusto y fiable que pueda ser introducido en el futuro dentro de nuestro Sistema Sanitario.

---

## Bibliografía

---

- [1] Scikit-Learn, Herramienta básica para el Data Science en Python, 2021.
- [2] Abedalmuhdi Almomany, Walaa R. Ayyad, and Amin Jarrah. Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study. *Journal of King Saud University - Computer and Information Sciences*, 34(6):3815–3827, 6 2022.
- [3] Roberto Baretto, Thierry Garaix, and Xiaolan Xie. A branch-and-price-and-cut algorithm for operating room scheduling under human resource constraints. *Computers & Operations Research*, 152:106136, 4 2023.
- [4] Hoss Belyadi and Alireza Haghighat. Supervised learning. *Machine Learning Guide for Oil and Gas Using Python*, pages 169–295, 1 2021.
- [5] Harika Bonthu. Detecting and Treating Outliers, 5 2021.
- [6] Sally Brailsford and Jan Vissers. OR in healthcare: A European perspective. *European Journal of Operational Research*, 212(2):223–234, 7 2011.
- [7] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 8 1996.
- [8] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.
- [9] Tugba Cayirli and Emre Veral. Outpatient scheduling in health care: a review of literature. *Production and Operations Management*, 12(4):519–549, 1 2009.

- [10] Batuhan Çelik, Serhat Gul, and Melih Çelik. A stochastic programming approach to surgery scheduling under parallel processing principle. *Omega*, 115:102799, 2 2023.
- [11] Carlos A. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 1 2002.
- [12] C. Di Martinelly, P. Baptiste, and M.Y. Maknoon. An assessment of the integration of nurse timetable changes with operating room planning and scheduling. *International Journal of Production Research*, 52(24):7239–7250, 12 2014.
- [13] Tlamele Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8(1):140, 10 2021.
- [14] H. Fei, C. Chu, N. Meskens, and A. Artiba. Solving surgical cases assignment problem by a branch-and-price approach. *International Journal of Production Economics*, 112(1):96–108, 3 2008.
- [15] Christodoulos A. Floudas and Xiaoxia Lin. Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. *Annals of Operations Research*, 139(1):131–162, 10 2005.
- [16] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary Algorithms Made Easy. *Journal of Machine Learning Research*, 13(70):2171–2175, 2012.
- [17] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., USA, 1999.
- [18] Michael R Garey and David S Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.
- [19] David Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, United States, 2<sup>o</sup> edition, 1 1989.
- [20] M. Greenacre. Correspondence Analysis. *International Encyclopedia of Education, Third Edition*, pages 103–111, 1 2009.

- [21] Miguel Grinberg. *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.", 2018.
- [22] Ş. Gür and T. Eren. Application of Operational Research Techniques in Operating Room Scheduling Problems: Literature Overview. *Journal of Healthcare Engineering*, 2018, 2018.
- [23] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Delhi, 2 edition, 1998.
- [24] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Longman Publishing Co., Inc., 1989.
- [25] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282, 1995.
- [26] Li Huang, Xiaomin Chen, Wenzhi Liu, Po-Chou Shih, and Jiaxin Bao. Automatic Surgery and Anesthesia Emergence Duration Prediction Using Artificial Neural Networks. *Journal of Healthcare Engineering*, 2022:1–17, 4 2022.
- [27] York Jiao, Bing Xue, Chenyang Lu, Michael S. Avidan, and Thomas Kannampallil. Continuous real-time prediction of surgical case duration using a modular artificial neural network. *British Journal of Anaesthesia*, 128(5):829–837, 5 2022.
- [28] Jamie Jones. *Getting Going and Best Practices Guide*, 2018.
- [29] Billy Vaughn Koen. Toward a Definition of the Engineering Method. *Engineering Education*, 75(3):150–155, 12 1984.
- [30] Yang-Kuei Lin and Yin-Yi Chou. A hybrid genetic algorithm for operating room scheduling. *Health Care Management Science*, 23(2):249–263, 6 2020.
- [31] A. Maleki, H. Hosseini-saz, and M. Jasemi. A comparative analysis of the efficient operating room scheduling models using robust optimization and upper partial moment. *Healthcare Analytics*, 3, 2023.
- [32] Oscar Martinez, Carol Martinez, Carlos A. Parra, Saul Rugeles, and Daniel R. Suarez. Machine learning for surgical time prediction. *Computer Methods and Programs in Biomedicine*, 208:106220, 9 2021.

- [33] Neal Master, Zhengyuan Zhou, Daniel Miller, David Scheinker, · Nicholas Bambos, · Peter Glynn, Nicholas Bambos, and Peter Glynn. Improving predictions of pediatric surgical durations with supervised learning. *International Journal of Data Science and Analytics*, 4:35–52, 2017.
- [34] W S McCulloh and W Pitts. A logical calculus of the ideas immanent in neural nets. *Bull Math. Biophys*, 5:133–137, 1943.
- [35] Wes McKinney and others. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56, 2010.
- [36] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.
- [37] Hossein Moayedi, Mansour Mosallanezhad, Ahmad Safuan A. Rashid, Wan Amizah Wan Jusoh, and Mohammed Abdullahi Muazu. A systematic review and meta-analysis of artificial neural network application in geotechnical engineering: theory and applications. *Neural Computing and Applications*, 32(2):495–518, 1 2020.
- [38] Arundhati Navada, Aamir Nizam Ansari, Siddharth Patil, and Balwant A. Sonkamble. Overview of use of decision tree algorithms in machine learning. In *2011 IEEE Control and System Graduate Research Colloquium*, pages 37–42. IEEE, 6 2011.
- [39] Issam Nouaouri, J-C. Nicolas, and Daniel Jolly. Operating room scheduling under unexpected events : the case of a disaster. 2011.
- [40] Ellis Batten Page. Ordered Hypotheses for Multiple Treatments: A Significance Test for Linear Ranks. *Journal of the American Statistical Association*, 58(301):216–230, 3 1963.
- [41] Marta Palacio. *Scrum Master*. Iubaris Info 4 Media, 3.07 edition, 2 2022.
- [42] F Pedregosa, G Varoquaux, and A Gramfort. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [43] David Petersson. Supervised Learning.



- [44] Dinh Nguyen Pham and Andreas Klinkert. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 185(3):1011–1025, 3 2008.
- [45] Kedar Potdar, Taher S., and Chinmay D. A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications*, 175(4):7–9, 10 2017.
- [46] A. Richards and J. How. Mixed-integer programming for control. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2676–2683. IEEE.
- [47] David H. Rothstein and Mehul V. Raval. Operating room efficiency. *Seminars in Pediatric Surgery*, 27(2):79–85, 4 2018.
- [48] Javier Sáez Hurtado. Cómo funciona la Metodología Scrum: qué es y cómo utilizarla, 12 2021.
- [49] Birgithe E Sandbaek, Berit I Helgheim, Odd I Larsen, and Sigurd Fasting. Impact of changed management policies on operating room efficiency. *BMC Health Services Research*, 14(1):224, 12 2014.
- [50] A.M. Schouten, S.M. Flipse, K.E. van Nieuwenhuizen, F.W. Jansen, A.C. van der Eijk, and J.J. van den Dobbelsteen. Operating Room Performance Optimization Metrics: a Systematic Review. *Journal of Medical Systems*, 47(1), 2023.
- [51] Zahra ShahabiKargar, Sankalp Khanna, Norm Good, Abdul Sattar, James Lind, and John O’Dwyer. Predicting Procedure Duration to Improve Scheduling of Elective Surgery. pages 998–1009. 2014.
- [52] Guido Van Rossum and Fred L Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.
- [53] Jake VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. O’Reilly Media, Inc., 1st edition, 2016.
- [54] Eric W Weisstein. Least Squares Fitting.
- [55] Ana Wheelock, Amna Suliman, Rupert Wharton, E. D. Babu, Louise Hull, Charles Vincent, Nick Sevdalis, and Sonal Arora. The Impact of Operating Room Distractions on Stress, Workload, and Teamwork. *Annals of Surgery*, 261(6):1079–1084, 6 2015.

- [56] Ian H. Wright, Charles Kooperberg, Barbara A. Bonar, and Gerard Bashein. Statistical Modeling to Predict Elective Surgery Time. *Anesthesiology*, 85(6):1235–1245, 12 1996.
- [57] Wenxun Xing and Jiawei Zhang. Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics*, 103(1-3):259–269, 7 2000.