eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Computación

Optimización de parámetros de carrera en la Fórmula 1 utilizando algoritmos genéticos

---

Aitor Malo Delgado

Febrero 2023

Director UPV/EHU

Ibai Gurrutxaga Goikoetxea

Resumen

Este proyecto se basa en la búsqueda de la estrategia de carrera óptima para un circuito y un piloto en particular dentro de un simulador de Fórmula 1. Se recogen datos del simulador y se definen los factores que influyen al definir una estrategia de carrera. Con esta información se utilizan técnicas de aprendizaje supervisado para analizar y predecir el comportamiento del coche en cualquier escenario y se implementa un optimizador genético a la hora de encontrar la combinación de parámetros que complete la carrera en el menor tiempo posible. Los resultados son satisfactorios tras ser evaluados introduciendo los parámetros de las soluciones en el simulador y comparando las estrategias obtenidas con las utilizadas en la realidad.

Índice

1. Introducción

La Fórmula 1 es la categoría reina dentro del automovilismo deportivo internacional. El Campeonato Mundial de Fórmula 1, es un torneo de carreras de monoplazas donde 10 escuderías compiten por ganar en los circuitos que forman parte de la competición.

Actualmente, cada coche de la parrilla dispone de hasta 300 sensores que generan millones de puntos de datos mientras están en la pista y se envían en tiempo real al Pit Wall, donde los ingenieros de pista de los equipos analizan la información para entender y controlar el comportamiento del monoplaza en cada instante. Un monoplaza envía alrededor de 3GB de información a los 100 km recorridos.

Visualizar, analizar y entender correctamente todos los datos que proporcionan los sensores supone una gran ventaja a la hora de tomar decisiones correctas durante las carreras, diseñar buenas estrategias y mejorar el rendimiento entre el coche y el piloto. El análisis de los datos es imprescindible en la Fórmula 1 moderna y técnicas como el aprendizaje automático o la Inteligencia Artificial juegan un papel muy importante en los equipos, que cuentan con científicos de datos y personal especializado en la materia dentro de sus plantillas.

En este proyecto se utiliza un simulador de gestión de carreras de Fórmula 1 para recoger datos telemétricos sobre un monoplaza y analizar cómo actúa en distintas situaciones. Para un piloto y circuito en concreto, se realizan simulaciones modificando los parámetros que influyen al definir la estrategia de carrera y se recoge la información proporcionada para analizarla y predecir la estrategia de carrera óptima en base a unas restricciones.

Se dispone de información sobre 2035 vueltas y se emplean algoritmos de aprendizaje automático para predecir el comportamiento del coche en todas las situaciones posibles que pueda haber durante la carrera.

Debido a que son bastantes los parámetros que definen una estrategia de carrera, tienen varios valores posibles y pueden variar en cada vuelta, existen muchas combinaciones que provocan que el espacio de búsqueda para encontrar la estrategia óptima sea masivo. Esto plantea un problema de optimización que en este trabajo se resuelve mediante la implementación de un algoritmo genético, donde los individuos son las estrategias de carrera y el objetivo es minimizar el tiempo necesario para finalizar una carrera completa.

El proyecto se divide en la fase de recogida de datos, fase de análisis de los datos y fase de planteamiento e implementación del optimizador genético. Durante el desarrollo se realizan comparativas entre los algoritmos de regresión considerados para las predicciones y se estudian diferentes posibilidades y variaciones que existen a la hora de llevar a cabo la programación del algoritmo genético.

1.1 Objetivos del proyecto

El objetivo del proyecto es encontrar una configuración de parámetros óptima para minimizar lo máximo posible el tiempo de carrera dentro de un simulador de Fórmula 1. Para ello se han definido tres principales tareas a cumplir.

1.2 Tareas

Se distinguen tres bloques principales de tareas durante el proyecto, desde la recogida de los datos del simulador hasta la implementación del optimizador genético. Los tres bloques son consecutivos y a continuación se detallan las subtareas que los definen.

1.2.1 Recogida de datos

Utilizando uno de los simuladores de gestión de carreras analizados, realizar simulaciones con un mismo piloto y circuito utilizando distintas configuraciones y parámetros de ajuste en el coche. Con diferentes combinaciones de carga de gasolina, tipo de neumáticos e indicaciones al piloto, recoger en una tabla la información que proporciona el coche al final de cada vuelta, como el tiempo marcado, la carga de combustible y el estado de los neumáticos.

1.2.2 Análisis de datos

Con toda la información recogida, procesar los datos e implementar regresores que permitan predecir el tiempo por vuelta, el desgaste de los neumáticos y el consumo de gasolina del monoplaza en cada vuelta para poder construir la función objetivo.

1.2.3 Optimizador genético

Plantear y desarrollar un problema de optimización utilizando un algoritmo genético que minimice el tiempo necesario para completar una carrera y encuentre la mejor estrategia. El espacio de búsqueda de soluciones posibles es inmenso por lo que hay que evaluar los resultados obtenidos variando los parámetros de ajuste del optimizador para obtener las mejores soluciones posibles.

2. Fundamentos de F1 Manager 22

En este capítulo se explican los conceptos básicos del simulador seleccionado para el desarrollo y se describen los parámetros relacionados con la estrategia de carrera que han sido considerados para la resolución. Al final del capítulo se muestra un breve análisis y comparativa sobre los tres simuladores que fueron considerados como posibles opciones a utilizar.

2.1 F1 Manager 22

F1 Manager 22 es el videojuego oficial de simulación de gestión de carreras del campeonato de Fórmula 1. Es la primera entrega de la serie y fue desarrollado y distribuido por Frontier Developments. Es compatible con Microsoft Windows, PS4, PS5, Xbox One y Xbox Series X/S y fue lanzado oficialmente el 30 de agosto de 2022.

El videojuego ofrece la posibilidad de disputar el Campeonato Mundial de Fórmula 1 desde la perspectiva del jefe de un equipo seleccionado. Durante los Grandes Premios, el simulador tiene la opción de disponer del punto de vista de los ingenieros de pista y permite que el jugador pueda observar y analizar el comportamiento del coche durante las sesiones de entrenamiento y clasificación, teniendo acceso a una gran cantidad de datos y posibilidades en la configuración de parámetros y simulaciones. El jugador puede modificar los parámetros de carrera y diseñar la estrategia a seguir durante la misma.

2.2 Estrategia de carrera en F1 Manager 22

La estrategia de carrera es uno de los conceptos más importantes de la Fórmula 1 actual y también lo es dentro de F1 Manager 22. La toma de decisiones correctas antes y durante la carrera es imprescindible para completar las 57 vueltas en un tiempo competitivo y en este proyecto se tienen en cuenta los siguientes parámetros a la hora de definir una estrategia de carrera.

2.2.1 Selección de neumáticos iniciales

Existen tres tipos de neumáticos para condiciones de seco en la pista y dos tipos de neumáticos para condiciones de lluvia. Los neumáticos de lluvia no ofrecen mucho juego para las estrategias ya que dependen en gran parte de la cantidad de agua que hay en el asfalto, por lo que no están incluidos en el análisis.

Según su rendimiento y durabilidad, los neumáticos de seco se clasifican en tres tipos como se detalla en la tabla 2.2.

Tipo de compuesto	Durabilidad	Rendimiento
Blando	Baja	Alto
Medio	Media	Medio
Duro	Alta	Bajo

Tabla 2.1. Comparativa de rendimiento y durabilidad de los compuestos de seco.

Todos los monoplazas eligen uno de los tres compuestos con los que comenzar la carrera. Existe la posibilidad de alargar la durabilidad de los neumáticos sacrificando su rendimiento, pero ninguno de los compuestos aguanta una carrera completa y hay que parar en boxes una o varias veces para cambiarlos.

El rendimiento y durabilidad de los tipos de neumático varía según el circuito y las condiciones del mismo, pero siempre se cumplen las reglas reflejadas en la tabla. Asimismo las diferencias de tiempos entre los compuestos es variable en función del circuito y condiciones de la pista.

2.2.2 Gasolina

Actualmente los coches de Fórmula 1 no repostan gasolina durante la carrera, todos los pilotos comienzan la carrera con 110 Kg de combustible en el depósito. La cantidad de gasolina influye en el peso del coche y por tanto en los tiempos de vuelta, ya que un coche más ligero es más rápido que un coche con sobrepeso de combustible.

2.2.3 Indicaciones al piloto

Como ingenieros de pista, el simulador ofrece la posibilidad de controlar la forma en la que el piloto gestiona los mapas del motor y trata los neumáticos. Las indicaciones al piloto se dividen en indicaciones de mezcla de motor y en indicaciones de ritmo. Pueden cambiarse en cada vuelta de la carrera.

Indicaciones de motor

Existen varias mezclas de motor posibles que el piloto puede seleccionar cada vuelta desde el volante. La gestión del combustible es muy importante ya que hay que terminar la carrera con algo de combustible en el depósito pero sin sobrepeso de gasolina. Se consideran las mezclas detalladas a continuación y se caracterizan por la tabla 2.3.

Mezcla	Consumo de combustible	Tiempo por vuelta
Conservar (1)	Bajo	Alto
Equilibrado (2)	Medio	Medio
Apretar (3)	Alto	Bajo

Tabla 2.2. Comparativa de consumo y tiempo por vuelta de las indicaciones de motor.

Elegir una mezcla alta durante muchas vueltas provoca que el coche no termine la carrera por falta de gasolina y hacerlo con una mezcla conservadora provoca que el coche acabe con sobrepeso y haya completado la carrera más lento de lo que debería. No hay una mezcla adecuada para todas las situaciones, según el momento conviene utilizar una mezcla u otra.

Indicaciones de ritmo

Existen varias formas en las que el piloto puede tratar los neumáticos. Una conducción más agresiva provoca mayor desgaste en las gomas y menores tiempos por vuelta, mientras que una conducción relajada permite estirar el neumático durante la carrera pero imprime un ritmo de carrera más lento. Las tres indicaciones de ritmo que se le pueden asignar al piloto aparecen en la tabla 2.4.

Indicación de ritmo	Desgaste de neumático	Tiempo por vuelta
Conservar (1)	Bajo	Alto
Estándar (2)	Medio	Medio
Atacar (3)	Alto	Bajo

Tabla 2.3. Comparativa de desgaste y tiempo por vuelta de las indicaciones de ritmo.

Al igual que ocurre con las mezclas del motor, no es viable apretar durante toda la carrera ni mantener un ritmo conservador en todas las vueltas, ya que se puede perder mucho tiempo en ambos casos. Hay que saber cuál es la indicación de ritmo óptimo para cada vuelta en función de la situación de carrera y el estado del coche.

2.3.4 Paradas en boxes

Todos los monoplazas deben parar en boxes mínimo una vez para cambiar el tipo de neumáticos con el que comienzan la carrera. En el circuito de *Sakhir*, una parada en boxes supone 25 segundos de media desde que el coche entra en el *pit lane* hasta que vuelve a la pista, asumiendo que los mecánicos cambian las ruedas en un tiempo estimado de 3 segundos.

Las estrategias de una parada no tienen porqué ser la mejor opción y hay que decidir cuántas paradas realizar durante la carrera, en qué vueltas hacerlo y sobre todo qué tipo de neumáticos montar en cada una para terminar la carrera en el menor tiempo posible.

2.3 Otros simuladores analizados

La elección del simulador es una decisión importante para conseguir el objetivo del proyecto y existen varias opciones que pueden considerarse para ser seleccionadas. A continuación se muestra una breve descripción de cada simulador y una tabla comparativa con las ventajas y desventajas de cada uno.

F1 Manager 22: Es un simulador de gestión de carreras de Fórmula 1 que ofrece la posibilidad de actuar como ingeniero de pista y tomar decisiones sobre los parámetros que influyen en las estrategias de carrera. Este simulador se basa en la Fórmula 1 actual y es similar en los resultados, escenarios y comportamiento de los parámetros.

MotorSport Manager: Es un simulador de gestión de carreras de coches o motos. No es un simulador específico para la categoría de Fórmula 1 pero existe la posibilidad de introducir coches de Fórmula 1. El funcionamiento y las posibilidades son similares a las de F1 Manager 22.

F1 2020 + SimHub: F1 2020 es un simulador de carreras de la temporada 2020 de Fórmula 1 y SimHub es un software que permite recoger datos automáticamente de F1 2020. El concepto de F1 2020 es diferente a F1 Manager 22 y MotorSport Manager porque se centra en las carreras desde la perspectiva de los pilotos y no desde la gestión de parámetros. Los tiempos por vuelta, circuitos y comportamiento son similares a los de la temporada 2020 de Fórmula 1, al igual que ocurre con F1 Manager 22.

Simulador	**Ventajas**	**Desventajas**
F1 Manager 22	● Es un simulador oficial lanzado en 2022 por lo que es similar a la categoría de Fórmula 1 actual. ● Ofrece el punto de vista de ingeniero de pista con muchas posibilidades, parámetros y opciones de configuración.	● La recogida de los datos es manual. ● Ofrece tantos parámetros e información durante las simulaciones que no es viable recoger todos de forma manual.

	● La velocidad de simulación se puede multiplicar hasta x16.	
Motorsport Manager	● Ofrece el punto de vista de ingeniero de pista con muchas posibilidades, parámetros y opciones de configuración. ● La velocidad de simulación se puede multiplicar hasta x16.	● La recogida de los datos es manual. ● La información proporcionada no es realista ya que los tiempos por vuelta distan mucho de los tiempos reales de Fórmula 1 y el simulador no incluye los circuitos reales.
F1 2020 + SimHub	● Es un simulador oficial lanzado en 2020 por lo que es similar a la categoría de Fórmula 1 actual. ● La recogida de los datos es más rápida que en los anteriores simuladores porque hay una parte automática.	● La velocidad de simulación es a tiempo real. ● Dispone de menos información, menos parámetros y menos posibilidades en la configuración.

Tabla 2.4. Ventajas y desventajas de los tres simuladores analizados.

3. Fundamentos técnicos teóricos

En este capítulo se explican los principios básicos teóricos sobre las técnicas y disciplinas que se han seguido e implementado para resolver el problema, tanto para el análisis de los datos como para la optimización de la función objetivo.

3.1 Aprendizaje Supervisado

El aprendizaje supervisado es un método de análisis de datos que se utiliza para predecir el valor de una función con precisión a partir de unos datos de entrada. Se define por el uso de un conjunto de datos etiquetados que se utilizan para entrenar el modelo. Pertenece a la ciencia del aprendizaje automático y se puede dividir en dos métodos cuando se trata de minería de datos:

Clasificación: Los algoritmos de clasificación se utilizan para asignar una clase discreta a un conjunto de datos de entrada.

Regresión: Este método predice la salida correspondiente a unas variables predictoras con valores numéricos reales que pueden ser discretos o continuos.

Existen diferentes algoritmos para el aprendizaje supervisado en función de si es un problema de clasificación o regresión, pero en este proyecto las predicciones se realizan en base a valores numéricos continuos por lo que sólo se tienen en cuenta algoritmos regresores.

3.1.1 Regresión Lineal Múltiple

Es un método de análisis predictivo basado en un modelo matemático que se utiliza para conocer la relación de dependencia entre dos o más variables independientes y una variable dependiente. El modelo cuenta con varias variables regresoras que se multiplican por un coeficiente estimado para establecer su relación funcional lineal con la variable a predecir. La fórmula matemática que utiliza es la siguiente:

$$Y_i = \beta_0 + \sum_{i=1}^{n} \beta_i \chi_i + \varepsilon_i$$

Y_i : La variable dependiente a predecir. Su resultado depende del sumatorio del producto de los coeficientes de regresión por su variable independiente correspondiente sumado a β_0 y al error de perturbación. β_0 es el coeficiente que determina el punto de intersección con el eje de ordenadas.

β_i, $\forall i \in 0 \leq i \leq n$: Son los coeficientes de regresión que se estiman utilizando el método de mínimos cuadrados. Este método busca la función continua que mejor se aproxime a los datos

minimizando el error cuadrático entre las observaciones y la recta de regresión, para obtener el mejor ajuste y lograr más precisión en las predicciones. La función de error cuadrático a minimizar para calcular los coeficientes es la siguiente:

$$\mathbf{S}(\beta_0, \beta_i ..., \beta_n) = \sum_{i=1}^{n} \varepsilon_i^2$$

χ_i: Las variables independientes o explicativas que influyen para predecir Y_i.

n: Número de variables independientes del modelo.

ε_i: Es el error de perturbación aleatoria y se calcula obteniendo la diferencia entre el valor real y el estimado. Indica la bondad de ajuste realizado para cada punto y cumple con las siguientes hipótesis del modelo de regresión lineal clásico:

Media cero: $\boldsymbol{E}(\varepsilon_i) = 0$

Homocedasticidad: $\boldsymbol{V}ar(\varepsilon_i) = \sigma^2 < \infty, \forall i$

Incorrelación: $\boldsymbol{C}ov(\varepsilon_i, \varepsilon_j) = 0, \forall i \neq j$

Normalidad en las perturbaciones: $\varepsilon \sim \boldsymbol{N}(0, \sigma^2)$

3.1.2 Support Vector Regressor

Es un algoritmo enfocado en resolver problemas de regresión perteneciente a la familia de los *Support Vector Machines*. Se basa en construir un conjunto de hiperplanos separados en un espacio de dimensionalidad alto para predecir valores con precisión.

Matemáticamente, los *SVR* minimizan una función en base a unas restricciones, acorde con el siguiente esquema:

Función a minimizar: $\frac{1}{2}||\omega||^2$

Restricción: $\left|y_i - (\omega \bullet x_i) - b\right| \leq \varepsilon$

ω: Son los vectores de soporte y están formados por los puntos que conforman las líneas paralelas a los hiperplanos que definen las áreas de regresión.

y_i: Es el valor objetivo de la clase a predecir de la iésima muestra de entrenamiento.

x_i: Es un vector formado por las variables predictoras que caracterizan la iésima muestra de entrenamiento.

$(\omega \bullet x_i) + b$: Es la predicción para la muestra x_i. La variable b es una constante única para todo el modelo.

ε: Este valor define un margen de tolerancia donde no se penalizan los errores en las predicciones. A mayor valor de ε, mayor error se admite para la solución.

Los *SVR* definen una zona en torno a los hiperplanos donde se ignoran los errores. Cuando se trata de una regresión que se realiza con más dimensiones, existen funciones de kernel no lineales para aplicar la transformación de asignar puntos de menor dimensión a otros de mayor. Existen también funciones kernel lineales, pero en este proyecto se utiliza un kernel no lineal llamado función de base radial.

El valor de esta función depende de la distancia normalmente euclidiana entre el valor de entrada y un punto fijo. Se utilizan para construir aproximaciones de funciones a través de una suma de funciones de base radial.

3.1.3 Multi-layer Perceptron

El perceptrón multicapa es una red neuronal artificial que se utiliza para resolver problemas que no son linealmente separables. Las neuronas pueden utilizar funciones de activación no lineales y se emplea backpropagation para el cálculo del gradiente de la función de pérdida y ajuste de la red. Su arquitectura se basa en una capa de entrada, una o varias capas ocultas y una capa de salida. Los perceptrones multicapa permiten establecer regiones de decisión más complejas que las de dos semiplanos.

Arquitectura de la red

Capa de entrada: Está formada por neuronas que corresponden con las características de cada individuo X de entrenamiento. Se compone de *c* neuronas, siendo *c* el número de características que determinan cada individuo. En estos nodos no se produce procesamiento interno.

Capas ocultas: Están formadas por unidades de red no observables que aplican una función a la suma ponderada de las entradas por los pesos y mapean el resultado en un margen que normalmente está comprendido entre 0 y 1 o entre -1 y 1. Algunas de las funciones de activación más utilizadas son ReLU y tangente hiperbólica.

Capa de salida: Esta capa proporciona el resultado final de todo el procesamiento de datos que realiza la red neuronal. Utiliza funciones de activación para relacionar la suma ponderada de las

entradas con los valores de la capa correcta. Algunas de las funciones más utilizadas son la tangente hiperbólica y la sigmoide.

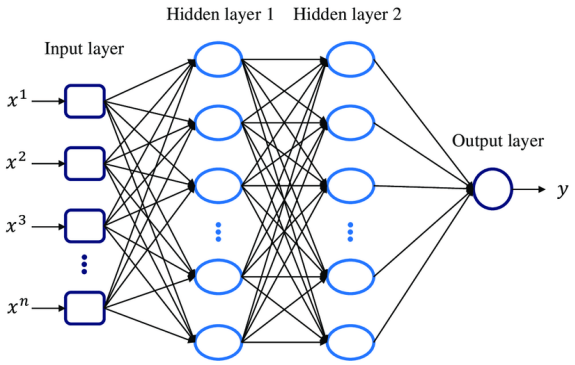


Imagen 3.1 Arquitectura de un perceptrón multicapa formado por dos capas ocultas.

El esquema matemático que siguen las neuronas en las capas ocultas y en la capa de salida para su activación es el siguiente:

$$z = \sum_{i=1}^{n} X_i W_i + b_i$$

$$a = F(z)$$

X_i: Es el vector de características que representa al individuo iésimo del conjunto de entrenamiento.

W_i: W es la matriz de pesos que se multiplica por la matriz X de individuos. W_i es el vector de pesos que corresponde con el vector X_i y con el que se hace el producto escalar.

b_i: Es un sesgo que corresponde a la diferencia entre el valor medio predecido por el modelo y el valor medio real. Representa el error sistemático del sistema y sirve para evitar el underfitting y hacer predicciones con exactitud.

z : Resultado del sumatorio entre el producto de los pesos por las características más el valor *bias*.

$F(z)$: Función de activación de la neurona. Se aplica una de las funciones posibles al valor de z para obtener un valor mapeado en un margen continuo.

a: Resultado de aplicar la función de activación a z.

La alimentación de la red neuronal es hacia delante por lo que el flujo tiene una dirección desde la capa de entrada hasta la capa de salida. El output de cada neurona corresponde con el input de las neuronas de la siguiente capa y el output de la capa de salida es el valor final predecido.

Aprendizaje de la red

Completado el flujo de propagación de la red, el siguiente paso es evaluar la salida predecida contra la salida esperada. Para ello, existen funciones de coste como el error cuadrático medio o la entropía cruzada binaria. En función del valor de la función de coste, el modelo ajusta los parámetros de pesos y bias para reducir el error y acercarse a la salida esperada.

Para realizar este ajuste las salidas de error se propagan desde la capa de salida hacia las neuronas de la primera capa oculta. Básicamente se aplica la regla de la cadena para derivar las funciones de coste y medir la sensibilidad del valor de salida con respecto al cambio de valor de la entrada y así conocer la dirección de la función de coste.

A través de esta propagación para atrás de los errores, se resuelven cuáles son los pesos que tienen errores y se ajustan para que el error sea el mínimo posible. Esta técnica se conoce como backpropagation y descenso del gradiente. Se aplica iterativamente un número limitado de veces.

La toma de decisiones para asignar el valor de los parámetros de la red es labor del programador y no existe un modelo que resuelva todos los problemas de forma óptima, cada modelo debe adaptarse al planteamiento del problema.

3.1.4 Decision Tree Regressor

Es un algoritmo que utiliza un árbol de decisión como modelo predictivo de regresión. La manera en la que opera es a través de la respuesta a ciertas preguntas relacionadas con las variables predictoras del conjunto de entrenamiento. Cada pregunta corresponde con un nivel del árbol y cada respuesta desemboca en el siguiente nivel del árbol hasta llegar al resultado predecido, las hojas del árbol.

Existen diferentes algoritmos para la construcción de los árboles de decisión y suelen trabajar de manera *top-down,* escogiendo en cada nivel del árbol la variable que mejor divide el conjunto de elementos.

Se utilizan distintos criterios para medir la importancia e influencia de las variables predictoras con respecto a la variable de clase, como el método de ganancia de información que calcula la entropía para medir la incertidumbre en las clasificaciones y hacer una división más precisa.

Las preguntas que definen los niveles de cada árbol son del tipo ¿$X_i \leq c$?, siendo X_i la variable iésima predictora y c un valor de corte que hay que estimar de tal manera que se minimice la función de coste y las predicciones sean lo más precisas posibles. La función de coste que hay que minimizar define el error en las predicciones y se define de la siguiente forma:

Función de coste: $\sum_{i=1}^{n} (y_i - \widehat{y_i})^2$

y_i: El valor iésimo real de la clase del individuo a predecir.

$\widehat{y_i}$: El valor de la clase predecido para el individuo iésimo.

n: Número total de individuos del conjunto de entrenamiento.

En la imagen 3.1 se muestra un ejemplo de la estructura de un árbol de decisión regresor simple. Suponemos que el modelo consta de dos variables predictoras X_1, X_2 y una variable de clase continua a predecir Y. También sabemos que la variable X_2 ofrece más información sobre la clase que X_1.

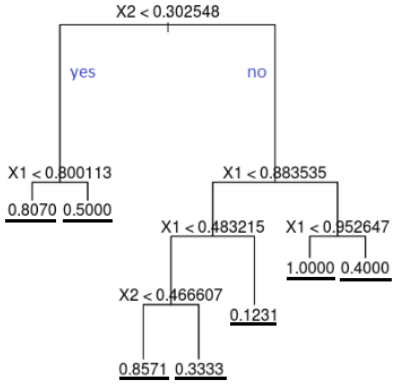


Imagen 3.2 Estructura de un árbol de decisión de regresión simple.

Las hojas del árbol son los valores que están subrayados y son los posibles resultados de las predicciones. Los valores de corte se actualizan iterativamente hasta encontrar los mejores resultados o hasta alcanzar el criterio de parada del algoritmo. A través de este árbol el espacio de las variables es dividido en hiper-rectángulos como se muestra en la imagen 3.2.

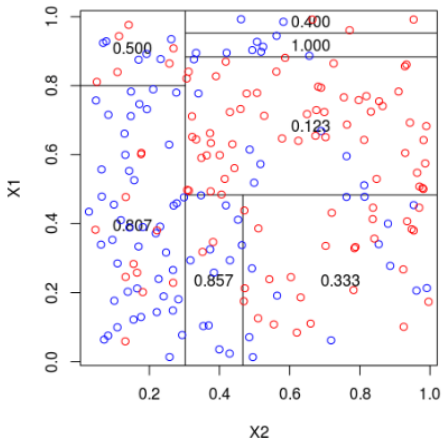


Imagen 3.3 Hiper-rectángulos de regresión formados por el espacio de X1 y X2.

Todas las observaciones que quedan dentro del mismo hiper-rectángulo tienen el mismo valor predecido, por lo que distintos individuos pueden seguir el mismo camino por el árbol hasta llegar a la misma hoja. Los posibles valores estimados corresponden con las hojas del árbol. Existen diferentes criterios para representar la función de coste a minimizar, así como para elegir el valor de corte para cada nodo.

3.1.5 K Nearest Neighbors

El algoritmo de *k* vecinos más cercanos es un clasificador y regresor de aprendizaje supervisado que utiliza la proximidad entre los individuos como criterio para realizar las predicciones. El objetivo es identificar los vecinos más cercanos del conjunto de entrenamiento para el individuo cuyo valor de clase queremos predecir. Después se hace una media entre los valores de clase de estos vecinos más cercanos y se obtiene el valor de la predicción del individuo analizado.

Para calcular la distancia que hay entre los individuos existen diferentes métodos pero en este proyecto se utiliza la distancia euclidiana. Asimismo, existe la posibilidad de ponderar las distancias y asignar más peso a los vecinos que más cerca se encuentran a la hora de obtener el resultado de la predicción. De esta manera la predicción está más influenciada por los vecinos que más cerca tiene.

3.2 Algoritmos genéticos

Son métodos metaheurísticos inspirados en la evolución biológica de Darwin como estrategia para resolver problemas de búsqueda y optimización. Estos algoritmos hacen evolucionar una población de individuos sometiéndose a acciones aleatorias semejantes a las que actúan en la evolución biológica, tales como la selección natural, mutaciones y recombinaciones genéticas. Los algoritmos genéticos se componen de los conceptos detallados a continuación.

3.2.1 Codificación Numérica

Tras parametrizar el problema en una serie de variables, cada individuo de la población se codifica en forma de cromosoma. Cada cromosoma está formado por un número limitado de genes que contienen la información acerca de la solución que representa el individuo.

Existen varios criterios para representar los genes y dependen de la naturaleza y tipo del problema. En este proyecto los individuos se representan mediante una codificación numérica, que es un tipo de codificación que utiliza una cadena de números enteros para representar las soluciones. Existen otros tipos de codificación como la codificación binaria, por valor directo o en forma de árbol.

3.2.2 Parámetros

Para el estudio de los algoritmos genéticos hay que tener en cuenta una serie de parámetros que definen sus características y criterios evolutivos. Estos parámetros son el tamaño de la población y las probabilidades de cruce y mutación.

Tamaño de la población

Este parámetro indica el número de cromosomas que componen una población para una generación en concreto. Cuando esta medida es insuficiente, el algoritmo genético tiene pocas posibilidades de reproducirse y la búsqueda de soluciones es escasa y poco óptima. Sin embargo, cuando la población es excesiva el algoritmo es muy lento computacionalmente y no tiene porqué dar mejores resultados. De hecho hay estudios que demuestran que hay un límite a partir del cuál es ineficiente elevar el tamaño de la población puesto que no se consigue una mayor velocidad para resolver el problema.

Probabilidad de cruce

Este valor indica la probabilidad con la que se producen cruces entre los cromosomas padres. En caso de que la probabilidad sea nula, los hijos de la siguiente generación son copias exactas de los padres. En caso de que haya probabilidad de cruce no nula, los hijos se forman a partir de los cromosomas de los padres. Cuando la probabilidad de cruce es del 100% los hijos se crean totalmente por cruce.

Probabilidad de mutación

La probabilidad de mutación indica la frecuencia con la que los genes de un cromosoma son mutados. Cuando no hay mutación, los descendientes son los mismos que había tras el cruce o reproducción de los padres. En caso de que se produzcan mutaciones, ciertos genes del cromosoma descendiente son modificados.

Al igual que ocurre en la evolución biológica, no conviene que la probabilidad de mutación sea muy alta porque la población puede degenerar rápidamente. Sin embargo, encontrar una probabilidad de mutación adecuada puede ser beneficioso en términos evolutivos porque contribuye a la diversidad genética de la especie y puede generar individuos que se adapten mejor a las condiciones. Normalmente esta probabilidad suele ser menor al 1%.

3.2.3 Esquema básico evolutivo

Para aplicar los algoritmos genéticos al campo de la resolución de problemas, hay que seguir una serie de pasos. Estos pasos básicos son los descritos a continuación.

3.2.3.1 Inicialización

En primer lugar hay que crear una población inicial formada por un número determinado de individuos o cromosomas. Lo aconsejable es que los individuos de la población sean generados de forma aleatoria para garantizar diversidad en las soluciones y evitar que la búsqueda de una solución óptima se vea limitada por las posibilidades de los individuos.

3.2.3.2 Evaluación

Para el correcto funcionamiento del proceso evolutivo durante las generaciones, debe existir un método que indique si los individuos de la población representan o no buenas soluciones de cara al problema. De esto se encarga la función de evaluación, que establece una medida numérica de la bondad de cada cromosoma. Esta función de *fitness* se considera como la probabilidad que tiene ese cromosoma de sobrevivir y reproducirse para generar nuevos individuos.

3.2.3.3 Operadores de cruce y mutación

Los individuos que forman la población se reproducen a través de los operadores de cruce y mutación para generar nuevos descendientes.

Cruce

Una vez realizada la selección de los cromosomas se procede a realizar la reproducción o cruce entre los cromosomas padres. Concretamente los cruces consisten en el intercambio de material genético entre dos cromosomas, con el objetivo de crear un descendiente que mejore la aptitud de los padres. El descendiente contiene información de los dos cromosomas que han sido cruzados.

Existen diferentes tipos de cruces y dependen de la codificación de los individuos y de la naturaleza del problema. Existen cruces por puntos, uniformes y aritméticos. Sin embargo, dependen de la codificación de los individuos y de la naturaleza del problema. Los cruces utilizados en este proyecto son específicos para la resolución y se explican en el apartado de Desarrollo de la Solución.

Mutación

Tras el cruce de los padres para generar nuevos descendientes, puede ocurrir que alguno de los genes de los hijos varíe de forma aleatoria. Esto imita el comportamiento que se da en la naturaleza y se utiliza para garantizar que ningún punto del espacio de búsqueda tenga una probabilidad nula de ser examinado. La mutación más usual es el reemplazo aleatorio de genes y en este proyecto es la utilizada para la implementación.

3.2.3.4 Selección

Una vez generada la descendencia a través de los operadores de cruce y mutación entre los individuos de una población, es necesario seleccionar a los individuos más capacitados entre las dos generaciones (progenitores y descendientes) para crear una descendencia más facultada que defina la siguiente generación. Existen varias maneras de realizar la selección entre los individuos pero en este proyecto se llevan a cabo los métodos explicados a continuación.

Selección por rueda de ruleta

Para este tipo de selección cada cromosoma tiene una parte de la ruleta en función de su *fitness*. Los cromosomas con mayor adaptación tienen una parte mayor en la ruleta y es más probable que sean seleccionados. Se hace girar la ruleta y se selecciona el cromosoma en el que se para la ruleta. En caso de que las probabilidades de los individuos difieran mucho y haya pocos individuos con buen *fitness*, este método de selección no es el más adecuado ya que se reduce la diversidad genética.

Selección elitista

En ciertas ocasiones puede suceder que se pierda el cromosoma con mejor adaptación. El tipo de selección elitista copia el mejor individuo o los *n* mejores individuos en la población de la nueva generación. El elitismo puede mejorar el funcionamiento de los algoritmos genéticos al evitar que se pierda la mejor solución.

3.3 Métricas de evaluación

Para evaluar los resultados de los modelos de regresión se utiliza una validación cruzada de 10 iteraciones en el dataset completo y se aplican las métricas mencionadas a continuación para medir la precisión en las capas de test de cada iteración.

10-cross fold validation: Es un procedimiento de evaluación de resultados de un análisis que mide la independencia de los resultados con respecto a las particiones del conjunto de entrenamiento y conjunto de test, para definir la validez y solidez del modelo. En este caso se divide el dataset en 10 capas y durante 10 iteraciones se utilizan 9 capas para el conjunto de entrenamiento y la capa de test restante para la validación.

R Squared: Esta medida estadística representa la proporción de la varianza para una variable dependiente que se explica por la varianza de unas variables independientes. Estudia cómo una variación de las variables independientes influye en la variación de la variable dependiente y viceversa. El valor oscila entre 0 y 1. A mayor valor, mayor porcentaje de varianza en la variable dependiente se explica por la varianza de las variables independientes, por lo que el modelo tendrá mejor precisión. El esquema matemático es el siguiente:

$$R^2 = 1 - \frac{\sum_{I=1}^{N} (\widehat{Y_i} - \bar{Y})^2}{\sum_{I=1}^{N} (Y_i - \bar{Y})^2}$$

El numerador es la variación inexplicable y el denominador la variación total. A menor proporción de variación inexplicable por variación total mayor es el valor de R Squared**.**

Mean Absolute Error (MAE): Representa la media de los errores absolutos de todas las predicciones realizadas por el modelo. La fórmula es la siguiente:

$$MAE = \frac{1}{N} \sum_{I=1}^{N} \left| \widehat{Y_i} - Y_i \right|$$

A mayor valor de MAE, mayor es el error cometido y por tanto menos fiable es el modelo.

Mean Squared Error (MSE): Representa la media de los errores elevados al cuadrado de todas las predicciones realizadas. La fórmula matemática sigue el siguiente esquema:

$$MSE = \frac{1}{N} \sum_{I=1}^{N} \widehat{(Y_i} - Y_i)^2$$

Al igual que ocurre con MAE, a menor valor de MSE mayor precisión tiene el modelo.

Root Mean Squared Error (RMSE): Es la raíz cuadrada del valor de Mean Squared Error.

$$RMSE = \sqrt{MSE}$$

Mean Absolute Percentage Error (MAPE): Es el porcentaje medio de error absoluto cometido en las estimaciones del modelo. Esta medida considera la proporción del error en función de los valores reales de la clase a predecir. Sigue la siguiente fórmula matemática:

$$MAPE = \frac{100}{N} \sum_{I=1}^{N} \left| \frac{\widehat{Y_i} - Y_i}{Y_i} \right|$$

4. Desarrollo de la solución

En este capítulo se detalla el proceso de desarrollo que se ha llevado a cabo para completar las tareas y lograr el objetivo principal. El simulador es necesario para la función objetivo, pero no es viable ejecutar carreras completas con todas las configuraciones posibles y extraer el tiempo que tarda cada configuración, debido a su lentitud. Es por ello que se ejecutan varias simulaciones en concreto para extraer datos y entrenar los modelos de regresión necesarios para construir la función objetivo, y así poder estimar el tiempo total de una carrera completa de forma precisa en base a cualquier combinación de parámetros.

La implementación del código ha sido utilizando el entorno de programación *Google Colab* y el lenguaje *python* con las librerías *numpy*, *pandas*, *seaborn*, *sklearn* y *matplotlib*.

4.1 Entorno de programación

Para la implementación se ha utilizado *Google Colab*, que es un entorno de programación que permite ejecutar código arbitrario de python desde el navegador. Se trata de un servicio de documentos ejecutables que permiten escribir, ejecutar y compartir código en Python dentro de *Google Drive*. Los documentos son cuadernos formados por celdas que contienen bloques de código, texto o imágenes.

El código se ejecuta en máquinas virtuales asignadas a cada cuenta de *Google* y se eliminan cuando están un tiempo inactivas. No es necesaria ninguna configuración ni instalación previa para poder utilizar los cuadernos y están enfocados para problemas de inteligencia artificial y aprendizaje automático. Existen planes de pago para utilizar GPUs más potentes pero en este proyecto se utiliza el *Colab* sin premium y las ejecuciones son en los servidores y GPUs por defecto.

La versión de Python es la número 3 y a continuación se mencionan las librerías incluidas en la implementación.

numpy: Es una librería especializada en el cálculo numérico de grandes volúmenes y que da soporte para crear vectores y matrices multidimensionales. En el proyecto se utiliza sobre todo para la creación y manejo de vectores y para seleccionar elementos aleatorios de los vectores.

pandas: Ofrece estructuras de datos flexibles que facilitan la manipulación y el tratamiento de los datos. Agrupa en una misma librería las funcionalidades de cargar datos, modelar, analizar y manipular. En este trabajo se utiliza para leer el archivo *.xlsx* donde se encuentran los datos recogidos del simulador.

matplotlib: Sirve para la generación de gráficos en dos dimensiones a partir de datos contenidos en listas o vectores.

seaborn: Es una librería de visualización de datos desarrollada sobre matplotlib pero que ofrece gráficas más atractivas.

sklearn: Scikit-learn es una librería enfocada al aprendizaje automático. Incluye algoritmos de clasificación y regresión. En este proyecto esta librería se utiliza para implementar los algoritmos explicados en el capítulo 3.

4.2 Modelos de regresión

Los algoritmos de regresión sirven para poder implementar la función objetivo. Estos métodos estiman el tiempo por vuelta, el consumo de combustible y el desgaste de los neumáticos en base a unas variables de entrada. Con estos predictores se puede estimar el tiempo total de una carrera completa para una configuración determinada. Antes de su implementación es necesario recoger y almacenar los datos de las simulaciones ejecutadas.

4.2.1 Recogida de los datos

La primera fase se basa en recoger los datos necesarios para el análisis. Realizando simulaciones variando los parámetros del coche, se almacena en una tabla la información que proporciona el monoplaza al final de cada vuelta simulada.

Los parámetros variables que se han considerado para las simulaciones son: Tipo de neumáticos, carga de gasolina e indicaciones al piloto. Se han realizado para 4 cargas iniciales diferentes de gasolina, 3 tipos de neumáticos y 9 combinaciones posibles referentes a las indicaciones de ritmo y combustible. En total se han llevado a cabo 108 simulaciones.

La información que se recoge sobre cada vuelta es el **estado y desgaste** de los neumáticos, la **carga de gasolina y consumo**, **tipo de neumáticos**, **indicaciones al piloto** y el **tiempo marcado**. El tipo de neumáticos y las indicaciones al piloto son constantes en todas las vueltas de una misma simulación. Todos estos valores se recogen al final de cada giro.

Los parámetros constantes son el piloto, coche, circuito y condiciones del circuito. Concretamente se utiliza el piloto *Charles Leclerc* de la escudería *Ferrari* en el circuito de *Sakhir, Bahréin.* El simulador ofrece la posibilidad de acelerar el tiempo de las simulaciones hasta 16 veces más rápido de lo normal pero la extracción de los datos se realiza manualmente.

La manera de representar las indicaciones al piloto en el desglose es la siguiente:

E-E: Combustible Equilibrado (2) - Ritmo Estándar (2)
A-A: Combustible Apretar (3) - Ritmo Ataque(3)
C-C: Combustible Conservar (1) - Ritmo Conservar (1)
E-A: Combustible Equilibrado (2) - Ritmo Ataque(3)
A-E: Combustible Apretar (3) - Ritmo Estándar (2)

C-E: Combustible Conservar (1) - Ritmo Estándar (2)
E-C: Combustible Equilibrado (2) - Ritmo Conservar (1)
A-C: Combustible Apretar (3) - Ritmo Conservar (1)
C-A: Combustible Conservar (1) - Ritmo Ataque(3)

Las tablas 4.1, 4.2, 4.3 y 4.4 muestran un desglose del número de vueltas recogidas en cada simulación. Cada simulación depende de la carga de gasolina inicial, tipo de neumático e indicaciones al piloto con las que se simulan las vueltas. Las tablas se dividen según la carga de gasolina inicial utilizada para las simulaciones y cada una de las celdas corresponde con una simulación diferente.

	E-E	A-A	C-C	E-A	A-E	C-E	E-C	A-C	C-A	**TOTAL**
Blandos	28	13	34	15	24	32	34	32	18	**230**
Medios	34	18	34	22	34	34	35	34	24	**269**
Duros	34	22	34	26	34	34	34	34	30	**282**
TOTAL	**96**	**53**	**102**	**63**	**92**	**100**	**103**	**100**	**72**	**781**

Tabla 4.1 Desglose del número de vueltas según el neumático y las indicaciones al piloto con una carga de gasolina de 110 Kg.

	E-E	A-A	C-C	E-A	A-E	C-E	E-C	A-C	C-A	**TOTAL**
Blandos	22	12	22	13	22	22	22	22	16	**173**
Medios	31	18	31	19	30	31	31	30	23	**244**
Duros	35	21	35	25	36	35	36	36	29	**288**
TOTAL	**88**	**51**	**88**	**57**	**88**	**88**	**89**	**88**	**68**	**705**

Tabla 4.2 Desglose del número de vueltas según el neumático y las indicaciones al piloto con una carga de gasolina del 100% de la durabilidad estimada de cada neumático.

	E-E	A-A	C-C	E-A	A-E	C-E	E-C	A-C	C-A	**TOTAL**
Blandos	10	10	10	10	10	10	10	10	10	**90**
Medios	15	15	15	15	15	15	15	15	15	**135**
Duros	18	18	18	18	18	18	18	18	18	**162**
TOTAL	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**43**	**387**

Tabla 4.3 Desglose del número de vueltas según el neumático y las indicaciones al piloto con una carga de gasolina del 50% de la durabilidad estimada de cada neumático.

	E-E	A-A	C-C	E-A	A-E	C-E	E-C	A-C	C-A	**TOTAL**
Blandos	4	4	4	4	4	4	4	4	4	**36**
Medios	6	6	6	6	6	6	6	6	6	**54**
Duros	8	8	8	8	8	8	8	8	8	**72**
TOTAL	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**18**	**162**

Tabla 4.4 Desglose del número de vueltas según el neumático y las indicaciones al piloto con una carga de gasolina del 25% de la durabilidad estimada de cada neumático.

DESGLOSE TOTAL

	E-E	A-A	C-C	E-A	A-E	C-E	E-C	A-C	C-A	**TOTAL**
Blandos	64	39	70	42	60	68	70	68	48	**529**
Medios	86	57	86	62	85	86	87	85	68	**702**
Duros	95	69	95	77	96	95	96	96	85	**804**
TOTAL	**245**	**165**	**251**	**181**	**241**	**249**	**253**	**249**	**201**	**2035**

Tabla 4.5 Desglose total del número de vueltas según el neumático y las indicaciones al piloto con todas las cargas de gasolina probadas.

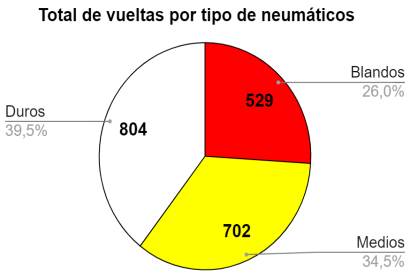


Imagen 4.1 Desglose del número total de vueltas según el tipo de neumáticos.

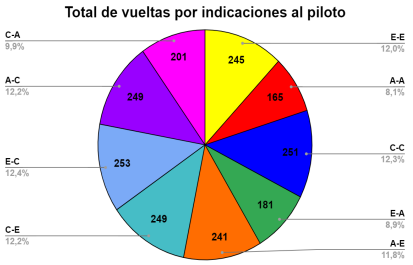


Imagen 4.2 Desglose del número total de vueltas según las indicaciones al piloto.

Como reflejan las imágenes 4.1 y 4.2, el mayor número de vueltas recogidas es con neumáticos duros y con indicaciones al piloto bajas. Esto es porque con neumáticos blandos e indicaciones al piloto altas se acaba la gasolina y / o se devastan los neumáticos antes de completar todas las vueltas del plan de simulación.

4.2.2 Análisis de los datos

Con toda la información de las vueltas recogida en una tabla, el siguiente paso es analizar los datos para predecir el comportamiento del coche en todos los escenarios posibles de la carrera e implementar una función objetivo que estime el tiempo total de una carrera completa para una configuración determinada de parámetros. Para ello se implementan tres regresores de aprendizaje supervisado que predicen el **tiempo por vuelta**, **consumo de combustible** y **desgaste de los neumáticos** de cada vuelta.

Para implementar las funciones de **calcularTiempo()**, **calcularConsumo()** *y* **calcularDesgaste()** se han probado varios algoritmos y se han acabado utilizando los modelos con mejores resultados obtenidos para cada caso. Los algoritmos regresores utilizados en el análisis son: Regresión lineal múltiple, Support Vector Regressor, Multilayer Perceptron, Decision Tree Regressor y K-Nearest Neighbors.

Se dispone de un dataset de 2035 filas y 7 columnas para el entrenamiento y evaluación de los algoritmos. Las filas hacen referencia a las vueltas y las columnas a las características que definen cada vuelta. Los atributos de las columnas son: **Tipo de neumáticos, Estado de los neumáticos**, **Carga de gasolina**, **Indicación de ritmo**, **Indicación de motor**, **Desgaste del neumático**, **Consumo de gasolina** y **Tiempo por vuelta**. Todos estos valores se marcan al final de cada vuelta.

Tipo de neumáticos	Estado de los neumáticos (%)	Gasolina (Kg)	Ritmo	Motor	Desgaste (%)	Consumo (Kg)	Tiempo (s)
1	92.0	42.0	2	2	4.0	1.8	95.993
1	88.0	40.2	2	2	4.0	1.8	95.592
1	83.0	38.3	2	2	5.0	1.9	95.248
1	79.0	36.5	2	2	4.0	1.8	95.798
1	75.0	34.6	2	2	4.0	1.9	95.833

Imagen 4.3: Ejemplo de una muestra del dataset.

Los tres regresores utilizan las mismas variables independientes que influyen para sus predicciones y son las primeras 5 columnas de la imagen 4.3. Las tres últimas columnas son las variables dependientes de los regresores de *calcularDesgaste()*, *calcularConsumo()* y *calcularTiempo()* respectivamente.

Tipo de neumáticos: Es el tipo de compuesto con el que se ha realizado la vuelta. Hay 3 valores posibles que son 1, 2 o 3. (Blandos = 1, Medios = 2, Duros = 3)

Estado de los neumáticos: Es el estado de los neumáticos al completar la vuelta y es un valor entero entre 0 y 100. (Medido en %)

Gasolina (Kg): Carga de gasolina del monoplaza al terminar la vuelta. Es un valor entero entre 0 y 110. (Medido en Kg)

Ritmo: La indicación de ritmo que ha seguido el piloto en la vuelta analizada. Existen los valores posibles de 1, 2 y 3. (Conservar = 1, Estándar = 2, Atacar = 3)

Motor: La indicación de motor que ha seguido el piloto en la vuelta analizada. Existen los valores posibles de 1, 2 y 3. (Conservar = 1, Equilibrado = 2, Apretar = 3)

A continuación se muestra un análisis de los algoritmos utilizados para el estudio con diferentes configuraciones y para cada uno de los regresores. Para cada prueba se aplica un 10-cross fold validation y se extraen las medias de las métricas de cada capa junto con las desviaciones típicas.

calcularTiempo()

Esta función predice el tiempo necesario para completar una vuelta en función de los parámetros de entrada o variables independientes.

Regresión lineal múltiple

R squared	MAE	MSE	RMSE	MAPE
0.821 ± 0.110	0.356 ± 0.019	0.189 ± 0.022	0.434 ± 0.025	$0.36\% \pm 1.9 \cdot 10^{-4}$

Tabla 4.6 Tabla de métricas del algoritmo Regresión Lineal Múltiple tras la validación 10-cross fold validation.

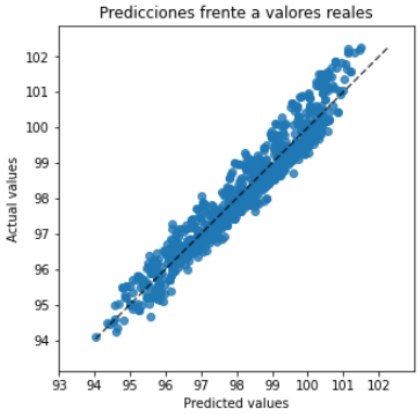


Imagen 4.4 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de tiempo utilizando Regresión Lineal Múltiple.

Support Vector Regressor

Parámetros

Kernel: Función para transformar los puntos de una dimensión menor a otra mayor. Se ha probado el kernel de base radial, lineal y polinomial.

C: Margen de penalización definido para las predicciones. A mayor valor de C menor margen de penalización por lo que el modelo es más restrictivo. Se han probado los valores 1, 3, 5, 10, 15.

epsilon: Define el área del hiperplano donde las predicciones son correctas. Este valor se mantiene constante y es 0,001.

degree: El grado utilizado para el kernel polinomial. Se ignora en los demás kernel. Se han probado los valores 3 y 5.

Kernel Radial Basis function

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.560 ± 0.279	0.593 ± 0.177	0.550 ± 0.291	0.714 ± 0.201	$0.60\% \pm 1.8 \cdot 10^{-3}$
3	0.840 ± 0.109	0.351 ± 0.108	0.194 ± 0.109	0.424 ± 0.120	$0.35\% \pm 1.1 \cdot 10^{-3}$
5	0.920 ± 0.061	0.232 ± 0.070	0.092 ± 0.052	0.293 ± 0.080	$0.23\% \pm 7.3 \cdot 10^{-4}$
10	0.946 ± 0.042	0.181 ± 0.052	0.061 ± 0.034	0.239 ± 0.064	$0.18\% \pm 5.4 \cdot 10^{-4}$
15	0.955 ± 0.031	0.165 ± 0.044	0.051 ± 0.029	0.220 ± 0.058	$0.16\% \pm 4.6 \cdot 10^{-4}$
20	0.958 ± 0.029	0.158 ± 0.040	0.048 ± 0.027	0.212 ± 0.055	$0.16\% \pm 4.2 \cdot 10^{-4}$

Tabla 4.7 Tabla de métricas del kernel de base radial en función del valor del margen de penalización.

Linear Kernel

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.794 ± 0.121	0.330 ± 0.033	0.219 ± 0.051	0.465 ± 0.051	$0.33\% \pm 3.3 \cdot 10^{-4}$
3	0.794 ± 0.121	0.330 ± 0.033	0.218 ± 0.051	0.464 ± 0.051	$0.33\% \pm 3.3 \cdot 10^{-4}$
5	0.802 ± 0.115	0.333 ± 0.033	0.211 ± 0.047	0.457 ± 0.048	$0.33\% \pm 3.3 \cdot 10^{-4}$
10	0.815 ± 0.111	0.366 ± 0.027	0.201 ± 0.029	0.447 ± 0.031	$0.37\% \pm 2.7 \cdot 10^{-4}$
15	-	-	-	-	-
20	-	-	-	-	-

Tabla 4.8 Tabla de métricas del kernel lineal en función del valor del margen de penalización.

El tiempo de cómputo requerido para una configuración de kernel lineal con C = 15 o C = 20 es elevado y no es viable utilizarlo para la función. Además los resultados no son lo suficientemente buenos como para poder considerar su uso.

Polynomial Kernel (Degree 1)

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.401 ± 0.353	0.687 ± 0.166	0.732 ± 0.345	0.834 ± 0.191	0.70% ± $1.7 \cdot 10^{-3}$
3	0.631 ± 0.209	0.531 ± 0.114	0.442 ± 0.178	0.652 ± 0.129	0.54% ± $1.2 \cdot 10^{-3}$
5	0.734 ± 0.147	0.437 ± 0.081	0.309 ± 0.091	0.549 ± 0.081	0.44% ± $8.6 \cdot 10^{-4}$
10	0.782 ± 0.130	0.362 ± 0.049	0.236 ± 0.043	0.484 ± 0.045	0.36% ± $5.2 \cdot 10^{-4}$
15	0.790 ± 0.125	0.346 ± 0.043	0.225 ± 0.043	0.472 ± 0.045	0.35% ± $4.5 \cdot 10^{-4}$
20	0.792 ± 0.124	0.338 ± 0.039	0.222 ± 0.046	0.468 ± 0.047	0.34% ± $4.0 \cdot 10^{-4}$

Tabla 4.9 Tabla de métricas del kernel polinomial de grado 1 en función del valor del margen de penalización.

(Degree 2)

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.305 ± 0.428	0.701 ± 0.173	0.805 ± 0.329	0.880 ± 0.173	0.71% ± $1.8 \cdot 10^{-3}$
3	0.588 ± 0.256	0.501 ± 0.122	0.458 ± 0.170	0.665 ± 0.123	0.50% ± $1.2 \cdot 10^{-3}$
5	0.651 ± 0.216	0.455 ± 0.108	0.384 ± 0.142	0.609 ± 0.113	0.46% ± $1.1 \cdot 10^{-3}$
10	0.699 ± 0.182	0.424 ± 0.094	0.330 ± 0.118	0.565 ± 0.103	0.43% ± $9.9 \cdot 10^{-4}$
15	0.714 ± 0.170	0.418 ± 0.090	0.315 ± 0.110	0.553 ± 0.098	0.42% ± $9.5 \cdot 10^{-4}$
20	0.719 ± 0.166	0.415 ± 0.088	0.310 ± 0.108	0.548 ± 0.096	0.42% ± $9.3 \cdot 10^{-4}$

Tabla 4.10 Tabla de métricas del kernel polinomial de grado 2 en función del valor del margen de penalización.

(Degree 3)

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.281 ± 0.494	0.677 ± 0.192	0.805 ± 0.353	0.879 ± 0.178	0.69% ± $2.0 \cdot 10^{-3}$
3	0.375± 0.440	0.626 ± 0.176	0.686 ± 0.292	0.812 ± 0.163	0.63% ± $1.8 \cdot 10^{-3}$
5	0.399 ± 0.427	0.615 ± 0.173	0.655 ± 0.281	0.793 ± 0.161	0.62% ± $1.8 \cdot 10^{-3}$

10	0.426 ± 0.405	0.603 ± 0.166	0.624 ± 0.265	0.774 ± 0.156	$0.61\% \pm 1.7 \cdot 10^{-3}$
15	0.439 ± 0.391	0.598 ± 0.162	0.610 ± 0.258	0.766 ± 0.154	$0.61\% \pm 1.7 \cdot 10^{-3}$
20	0.449 ± 0.382	0.594 ± 0.162	0.600 ± 0.255	0.759 ± 0.153	$0.60\% \pm 1.7 \cdot 10^{-3}$

Tabla 4.11 Tabla de métricas del kernel polinomial de grado 3 en función del valor del margen de penalización.

Los mejores resultados los ofrece el kernel de base radial. Los valores convergen para C = 20 y no merece la pena utilizar valores más altos porque requiere más tiempo de cómputo y los resultados no mejoran lo suficiente.

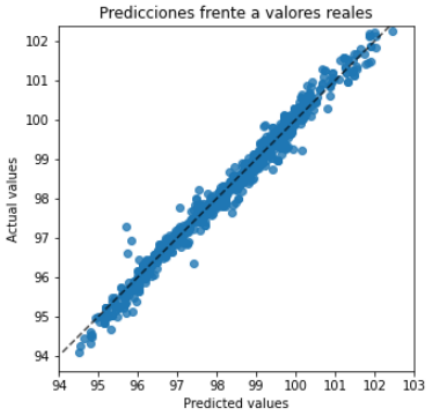


Imagen 4.5 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de tiempo utilizando Support Vector Regression con kernel de base radial y C=20.

Multilayer Perceptron

Iteraciones : 3000

Alpha : Es la fuerza del término de regularización para los ajustes. Se utiliza el valor por defecto que es de 0.0001

Solver : Es el método para solucionar el ajuste de pesos. Se utiliza *adam* que es un algoritmo que utiliza un learning rate adaptativo. Es el que mejor solución ha dado de los tres probados y es muy eficiente.

N: Número de neuronas de la capa oculta. Se hacen pruebas con 25, 50, 100, 150 y 200.

Activación: Función de activación de las neuronas. Se hacen pruebas con *ReLu*, *tanh* y *sigmoide.*

Para diferentes funciones de activación y número de neuronas en las capas ocultas, las métricas obtenidas son las mostradas en la tabla 4.12.

Activación	N	R squared	MAE	MSE	RMSE	MAPE
ReLu	25	0.452 ± 0.478	0.554 ± 0.187	0.539 ± 0.391	0.695 ± 0.234	0.56% ± 1.9· 10^{-3}
	50	0.198 ± 1.516	0.503 ± 0.317	0.500 ± 0.630	0.622 ± 0.335	0.50% ± 3.1· 10^{-3}
	100	0.723 ± 0.294	0.382 ± 0.107	0.249 ± 0.125	0.483 ± 0.125	0.38% ± 10^{-3}
	150	0.538 ± 0.380	0.511 ± 0.186	0.454 ± 0.269	0.645 ± 0.194	0.52% ± 1.9· 10^{-3}
	200	0.717 ± 0.253	0.389 ± 0.101	0.257 ± 0.111	0.494 ± 0.112	0.39% ± 10^{-3}
Tanh	25	-1.710 ± 2.503	1.429 ± 0.434	2.953 ± 1.414	1.667 ± 0.415	1.4% ± 4.4· 10^{-3}
	50	-1.714 ± 2.478	1.432 ± 0.432	2.964 ± 1.412	1.671 ± 0.413	1.4% ± 4.4· 10^{-3}
	100	-1.696 ± 2.386	1.433 ± 0.430	2.979 ± 1.413	1.676 ± 0.411	1.4% ± 4.4· 10^{-3}
	150	0.877 ± 0.095	0.291 ± 0.100	0.153 ± 0.115	0.369 ± 0.129	10^{-3}
	200	0.887 ± 0.082	0.269 ± 0.063	0.129 ± 0.078	0.347 ± 0.095	0.27% ± 6.7· 10^{-4}
Logistic	25	-1.700 ± 2.472	1.428 ± 0.431	2.947 ± 1.405	1.666 ± 0.412	1.4% ± 4.4· 10^{-3}
	50	-1.624 ± 2.542	1.318 ± 0.550	2.713 ± 1.644	1.547 ± 0.564	1.3% ± 5.6· 10^{-3}
	100	-0.271 ± 1.025	1.031 ± 0.654	2.103 ± 1.879	1.251 ± 0.733	1.05% ± 6.8· 10^{-3}
	150	0.919 ± 0.061	0.219 ± 0.050	0.089 ± 0.053	0.288 ± 0.079	0.22% ± 5.3· 10^{-4}
	200	0.913 ± 0.071	0.227 ± 0.059	0.099 ± 0.071	0.299 ± 0.098	0.23% ± 6.1· 10^{-4}

Tabla 4.12 Tabla de métricas del perceptrón multicapa en relación con el número de neuronas en la capa oculta y la función de activación.

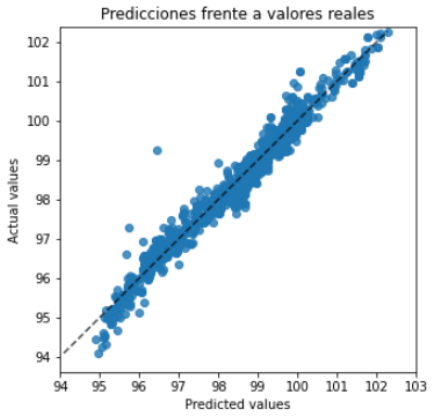


Imagen 4.6 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de tiempo utilizando el perceptrón multicapa con 150 neuronas en la capa oculta y función de activación logística.

Decision Tree Regressor

Parámetros

Criterio: Es la función que mide la calidad de las divisiones para las predicciones. Existen varias opciones como *squared_error*, *absolute_error*, *poisson* o *friedman_mse*.

Splitter: La estrategia utilizada para elegir los puntos de corte de las divisiones en cada nodo. El valor utilizado es *best* para que elija siempre la mejor división.

Criterio	R squared	MAE	MSE	RMSE	MAPE
squared_error	0.457 ± 0.387	0.620 ± 0.184	0.593 ± 0.329	0.744 ± 0.196	0.63% ± 1.9· 10^{-3}
absolute_error	0.372 ± 0.483	0.653 ± 0.224	0.652 ± 0.402	0.776 ± 0.223	0.66% ± 2.3· 10^{-3}
poisson	0.268 ± 0.630	0.739 ± 0.265	0.879 ± 0.673	0.888 ± 0.301	0.75% ± 2.7· 10^{-3}
friedman_mse	0.463 ± 0.397	0.610 ± 0.186	0.577 ± 0.328	0.734 ± 0.193	0.62% ± 1.9· 10^{-3}

Tabla 4.13 Tabla de métricas del árbol de decisión de regresión en función del criterio utilizado para las divisiones.

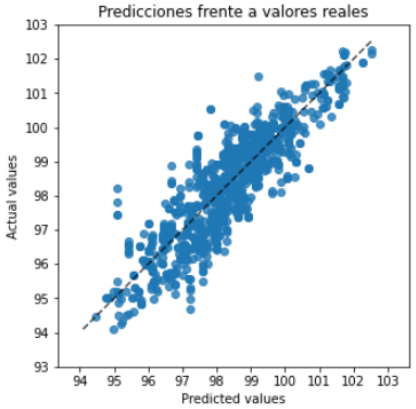


Imagen 4.7 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de tiempo utilizando Decision Tree Regressor utilizando friedman_mse como criterio.

K Nearest Neighbors

Parámetros

K : Número de vecinos considerados, se utilizan los valores 1, 3, 5, 10, 15 y 30.

Weights: Función de pesos utilizada para calcular la importancia de cada vecino. En este caso el valor para este parámetro que mejores resultados ofrece es *distance* que equivale a que los pesos son inversos a las distancias, por lo que el vecino más cercano es el que más peso tiene para las predicciones de los valores de clase.

K	R squared	MAE	MSE	RMSE	MAPE
1	0.334 ± 0.487	0.713 ± 0.179	0.745 ± 0.355	0.838 ± 0.205	$0.72\% \pm 1.8 \cdot 10^{-3}$
3	0.427 ± 0.477	0.641 ± 0.218	0.647 ± 0.401	0.767 ± 0.242	$0.65\% \pm 2.2 \cdot 10^{-3}$
5	0.433 ± 0.463	0.642 ± 0.227	0.653 ± 0.412	0.769 ± 0.246	$0.65\% \pm 2.3 \cdot 10^{-3}$
10	0.422 ± 0.461	0.660 ± 0.242	0.689 ± 0.440	0.787 ± 0.262	$0.67\% \pm 2.5 \cdot 10^{-3}$
15	0.415 ± 0.463	0.667 ± 0.248	0.707 ± 0.454	0.796 ± 0.271	$0.68\% \pm 2.5 \cdot 10^{-3}$
30	0.400 ± 0.460	0.679 ± 0.246	0.731 ± 0.454	0.810 ± 0.272	$0.69\% \pm 2.5 \cdot 10^{-3}$

Tabla 4.14 Tabla de métricas del algoritmo de los vecinos más cercanos en función del número de vecinos considerado.

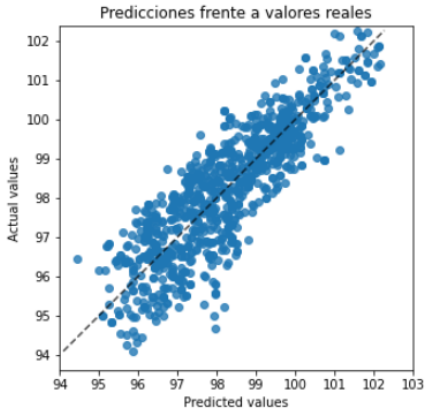


Imagen 4.8 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de tiempo utilizando 5-NN.

calcularConsumo()

Esta función predice el consumo de combustible en kilogramos de cada vuelta en función de los parámetros o variables independientes.

Regresión lineal múltiple

R squared	MAE	MSE	RMSE	MAPE
0.926 ± 0.014	0.033 ± 0.001	$0.001 \pm 2.2 \cdot 10^{-4}$	0.041 ± 0.002	1.8% ± 0.001

Tabla 4.15 Tabla de métricas del algoritmo Regresión Lineal Múltiple tras la validación 10-cross fold validation.

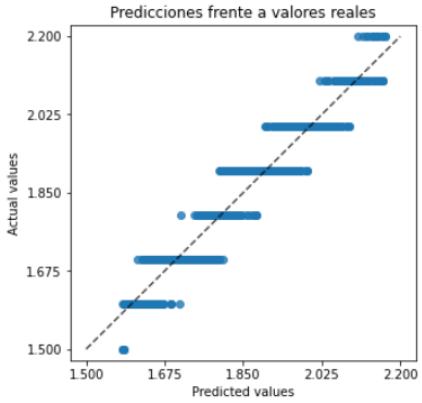


Imagen 4.9 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de consumo de combustible utilizando Regresión Lineal Múltiple.

Support Vector Regressor

Tras probar con los tres tipos de kernel, en la tabla 4.16 se recogen los resultados del kernel de base radial porque es el que mejores resultados ha obtenido.

Kernel Radial Basis function

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.875 ± 0.021	0.043 ± 0.003	$0.002 \pm 6 \cdot 10^{-4}$	0.054 ± 0.005	2.3% ± 0.001
3	0.909 ± 0.014	0.036 ± 0.002	$0.002 \pm 3.1 \cdot 10^{-4}$	0.046 ± 0.003	1.9% ± 0.001
5	0.913 ± 0.014	0.035 ± 0.002	$0.002 \pm 2.8 \cdot 10^{-4}$	0.045 ± 0.003	1.9% ± 0.001
10	0.918 ± 0.015	0.034 ± 0.002	$0.001 \pm 2 \cdot 10^{-4}$	0.043 ± 0.002	1.8% ± 0.001
15	0.919 ± 0.015	0.033 ± 0.002	$0.001 \pm 2.3 \cdot 10^{-4}$	0.043 ± 0.002	1.8% ± 0.001
20	0.919 ± 0.016	0.033 ± 0.002	$0.001 \pm 2.6 \cdot 10^{-4}$	0.043 ± 0.002	1.8% ± 0.001

Tabla 4.16 Tabla de métricas del kernel de base radial en función del valor del margen de penalización.

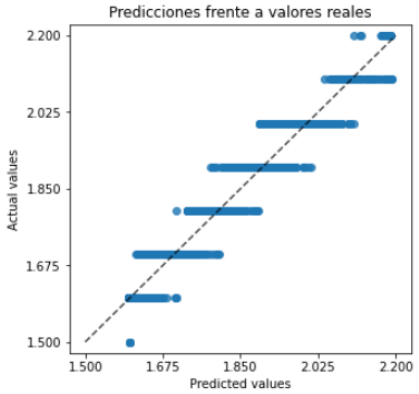


Imagen 4.10 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de consumo de combustible utilizando Support Vector Regression con el kernel de base radial y C=15.

Multilayer Perceptron

Activación	N	R squared	MAE	MSE	RMSE	MAPE
ReLu	25	0.681 ± 0.144	0.060 ± 0.013	0.007 ± 0.003	0.083 ± 0.019	3.2% ± 0.007
	50	0.541 ± 0.382	0.078 ± 0.030	0.010 ± 0.007	0.096 ± 0.032	4.2% ± 0.017
	100	0.625 ± 0.180	0.073 ± 0.019	0.008 ± 0.004	0.091 ± 0.020	4% ± 0.010
	150	0.764 ± 0.121	0.058 ± 0.012	0.005 ± 0.002	0.071 ± 0.015	3.1% ± 0.007
	200	0.811 ± 0.071	0.052 ± 0.008	0.004 ± 0.001	0.065 ± 0.009	2.8% ± 0.004
Tanh	25	0.186 ± 0.364	0.136 ± 0.016	0.027 ± 0.006	0.165 ± 0.018	7.4% ± 0.008
	50	0.708 ± 0.216	0.062 ± 0.020	0.006 ± 0.004	0.078 ± 0.022	3.4% ± 0.011
	100	0.862 ± 0.047	0.043 ± 0.005	$0.003 \pm 9.6 \cdot 10^{-4}$	0.056 ± 0.008	2.3% ± 0.003
	150	0.850 ± 0.083	0.045 ± 0.010	0.003 ± 0.001	0.057 ± 0.012	2.4% ± 0.006
	200	0.841 ± 0.062	0.047 ± 0.008	0.003 ± 0.001	0.059 ± 0.009	2.5% ± 0.004
Logistic	25	0.175 ± 0.148	0.118 ± 0.008	0.019 ± 0.002	0.139 ± 0.009	6.4% ± 0.004
	50	0.747 ± 0.049	0.061 ± 0.007	0.006 ± 0.001	0.077 ± 0.008	3.3% ± 0.004
	100	0.845 ± 0.033	0.048 ± 0.007	0.003 ± 0.001	0.060 ± 0.009	2.6% ± 0.003
	150	0.888 ± 0.021	0.040 ± 0.003	$0.002 \pm 4.9 \cdot 10^{-4}$	0.051 ± 0.004	2.1% ± 0.001
	200	0.884 ± 0.019	0.041 ± 0.004	$0.002 \pm 5.2 \cdot 10^{-4}$	0.052 ± 0.004	2.2% ± 0.002

Tabla 4.17 Tabla de métricas del perceptrón multicapa en relación con el número de neuronas en la capa oculta y la función de activación.

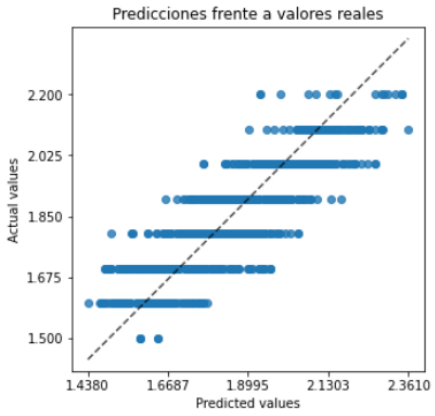


Imagen 4.11 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de consumo de combustible utilizando el perceptrón multicapa con 150 neuronas en la capa oculta y función de activación logística.

Decision Tree Regressor

Criterio	R squared	MAE	MSE	RMSE	MAPE
squared_error	0.854 ± 0.027	0.033 ± 0.004	$0.003 \pm 3.9 \cdot 10^{-4}$	0.058 ± 0.003	1.8% ± 0.002
absolute_error	0.847 ± 0.029	0.033 ± 0.004	$0.003 \pm 4.7 \cdot 10^{-4}$	0.059 ± 0.004	$1.8\% \pm 2.5 \cdot 10^{-3}$
poisson	0.857 ± 0.032	0.032 ± 0.005	$0.003 \pm 4.7 \cdot 10^{-4}$	0.057 ± 0.004	1.7% ± 0.002
friedman_mse	0.856 ± 0.027	0.033 ± 0.003	$0.003 \pm 3.7 \cdot 10^{-4}$	0.058 ± 0.003	1.8% ± 0.002

Tabla 4.18 Tabla de métricas del árbol de decisión de regresión en función del criterio utilizado para las divisiones.

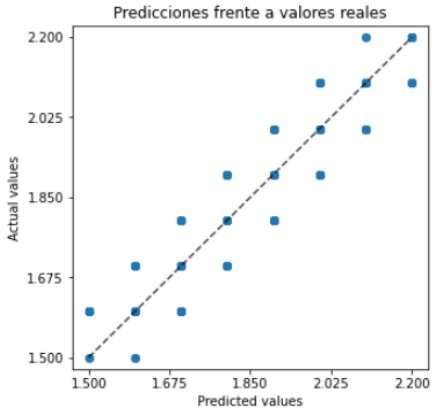


Imagen 4.12 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de consumo de combustible utilizando el Decision Tree Regressor con el criterio poisson.

K Nearest Neighbors

K	R squared	MAE	MSE	RMSE	MAPE
1	0.433 ± 0.266	0.149 ± 0.016	0.033 ± 0.005	0.183 ± 0.015	8.1% ± 0.008
3	0.027 ± 0.260	0.127 ± 0.014	0.024 ± 0.004	0.154 ± 0.016	6.9% ± 0.008
5	0.033 ± 0.242	0.126 ± 0.014	0.022 ± 0.004	0.149 ± 0.015	6.9% ± 0.007
10	0.042 ± 0.195	0.128 ± 0.011	0.022 ± 0.003	0.149 ± 0.012	7% ± 0.006
15	0.060 ± 0.174	0.128 ± 0.010	0.022 ± 0.003	0.148 ± 0.011	7% ± 0.005
30	0.066 ± 0.145	0.128 ± 0.008	0.022 ± 0.002	0.148 ± 0.008	7% ± 0.004

Tabla 4.19 Tabla de métricas del algoritmo de los vecinos más cercanos en función del número de vecinos considerado.

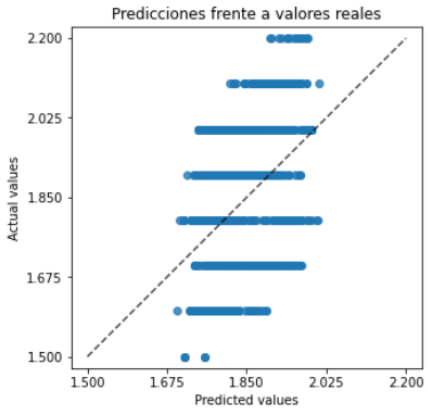


Imagen 4.12 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de consumo de combustible utilizando 30-NN.

calcularDesgaste()

Esta función predice el porcentaje de desgaste de los neumáticos de cada vuelta en función de los parámetros o variables independientes.

Regresión lineal múltiple

R squared	MAE	MSE	RMSE	MAPE
0.690 ± 0.071	0.524 ± 0.076	0.455 ± 0.188	0.663 ± 0.125	20% ± 0.031

Tabla 4.20 Tabla de métricas del algoritmo Regresión Lineal Múltiple tras la validación 10-cross fold validation.

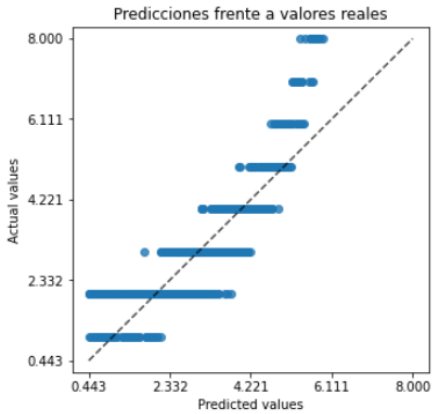


Imagen 4.13 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de desgaste de neumáticos utilizando Regresión Lineal Múltiple.

Support Vector Regressor

Al igual que en *calcularConsumo()*, en la tabla 4.21 se muestran las métricas del kernel que mejores resultados ha obtenido tras las pruebas.

Kernel Radial Basis function

C	*R squared*	*MAE*	*MSE*	*RMSE*	*MAPE*
1	0.107 ± 0.360	0.890 ± 0.342	1.618 ± 1.428	1.169 ± 0.499	31.5% ± 0.061
3	0.373 ± 0.263	0.741 ± 0.290	1.151 ± 1.059	0.982 ± 0.432	25.9% ± 0.043
5	0.497 ± 0.213	0.668 ± 0.257	0.930 ± 0.882	0.880 ± 0.393	23.3% ± 0.032
10	0.620 ± 0.141	0.587 ± 0.204	0.686 ± 0.627	0.764 ± 0.320	20.9% ± 0.028
15	0.650 ± 0.115	0.566 ± 0.177	0.614 ± 0.530	0.729 ± 0.286	20.3% ± 0.028
20	0.668 ± 0.096	0.553 ± 0.157	0.570 ± 0.463	0.709 ± 0.259	20% ± 0.027

Tabla 4.21 Tabla de métricas del kernel de base radial en función del valor del margen de penalización.

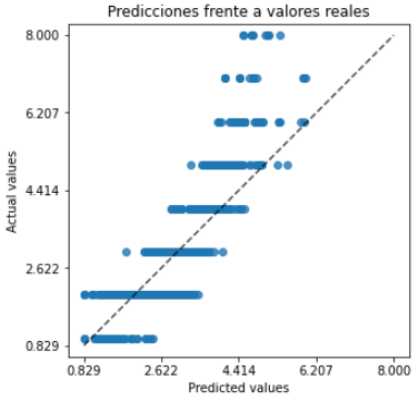


Imagen 4.14 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de desgaste de neumáticos utilizando Support Vector Regression con kernel de base radial y C=20.

Multilayer Perceptron

Activación	**N**	R squared	MAE	MSE	RMSE	MAPE
ReLu	25	0.748 ± 0.045	0.475 ± 0.072	0.363 ± 0.108	0.596 ± 0.084	18.3% ± 0.022
	50	0.727 ± 0.084	0.501 ± 0.063	0.384 ± 0.105	0.614 ± 0.084	19.2% ± 0.030
	100	0.767 ± 0.054	0.462 ± 0.077	0.338 ± 0.134	0.572 ± 0.104	17.6% ± 0.022
	150	0.745 ± 0.041	0.483 ± 0.075	0.381 ± 0.167	0.605 ± 0.120	18.3% ± 0.025
	200	0.723 ± 0.090	0.506 ± 0.079	0.400 ± 0.193	0.619 ± 0.129	19.4% ± 0.032
	25	0.779 ± 0.059	0.450 ± 0.082	0.325 ± 0.165	0.557 ± 0.121	17.4% ± 0.030

Tanh	50	0.748 ± 0.045	0.475 ± 0.072	0.363 ± 0.108	0.596 ± 0.084	18.3% ± 0.022
	100	0.762 ± 0.059	0.468 ± 0.078	0.346 ± 0.135	0.579 ± 0.106	17.6% ± 0.025
	150	0.716 ± 0.077	0.515 ± 0.099	0.427 ± 0.207	0.637 ± 0.146	19.5% ± 0.027
	200	0.709 ± 0.057	0.522 ± 0.098	0.439 ± 0.207	0.647 ± 0.142	19.5% ± 0.026
Logistic	25	0.786 ± 0.048	0.443 ± 0.069	0.311 ± 0.119	0.549 ± 0.095	17.3% ± 0.023
	50	0.746 ± 0.078	0.480 ± 0.069	0.358 ± 0.117	0.592 ± 0.089	18.4% ± 0.028
	100	0.761 ± 0.030	0.472 ± 0.088	0.360 ± 0.149	0.588 ± 0.116	18.1% ± 0.021
	150	0.747 ± 0.047	0.484 ± 0.090	0.378 ± 0.169	0.602 ± 0.123	18.3% ± 0.019
	200	0.732 ± 0.053	0.497 ± 0.083	0.395 ± 0.157	0.618 ± 0.112	18.9% ± 0.022

Tabla 4.22 Tabla de métricas del perceptrón multicapa en relación con el número de neuronas en la capa oculta y la función de activación.

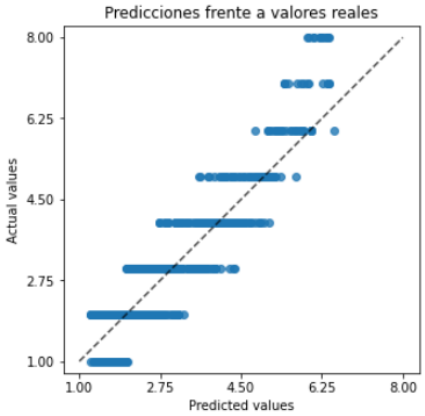


Imagen 4.15 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de desgaste de neumáticos utilizando el perceptrón multicapa con 25 neuronas en la capa oculta y función de activación logística.

Decision Tree Regressor

Criterio	R squared	MAE	MSE	RMSE	MAPE
squared_error	0.665 ± 0.134	0.414 ± 0.052	0.442 ± 0.082	0.662 ± 0.060	16.4% ± 0.028
absolute_error	0.606 ± 0.134	0.472 ± 0.071	0.529 ± 0.112	0.723 ± 0.076	19.1% ± 0.032
poisson	0.690 ± 0.122	0.396 ± 0.067	0.415 ± 0.107	0.639 ± 0.074	16.2% ± 0.030
friedman_mse	0.665 ± 0.135	0.413 ± 0.056	0.441 ± 0.080	0.661 ± 0.059	16.4% ± 0.028

Tabla 4.23 Tabla de métricas del árbol de decisión de regresión en función del criterio utilizado para las divisiones.

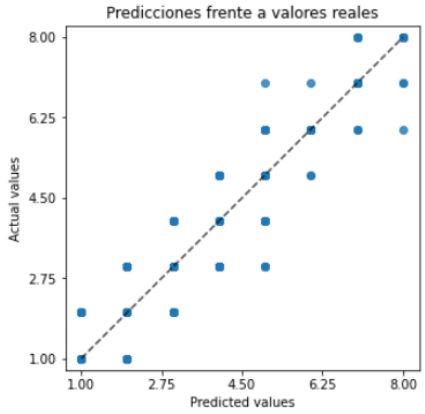


Imagen 4.16 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de desgaste de neumáticos utilizando Decision Tree Regressor con el criterio poisson.

K Nearest Neighbors

K	R squared	MAE	MSE	RMSE	MAPE
1	0.011 ± 0.527	0.897 ± 0.404	1.887 ± 1.433	1.302 ± 0.437	32.7% ± 0.100
3	0.022 ± 0.472	0.898 ± 0.384	1.629 ± 1.442	1.187 ± 0.467	33.1% ± 0.102
5	0.015 ± 0.488	0.916 ± 0.388	1.648 ± 1.445	1.192 ± 0.476	34.3% ± 0.112
10	0.004 ± 0.458	0.939 ± 0.376	1.703 ± 1.464	1.209 ± 0.489	35.5% ± 0.118
15	0.004 ± 0.430	0.952 ± 0.366	1.727 ± 1.451	1.220 ± 0.487	36.1% ± 0.119
30	0.001 ± 0.392	0.957 ± 0.347	1.735 ± 1.439	1.224 ± 0.485	36.6% ± 0.116

Tabla 4.24 Tabla de métricas del algoritmo de los vecinos más cercanos en función del número de vecinos considerado.

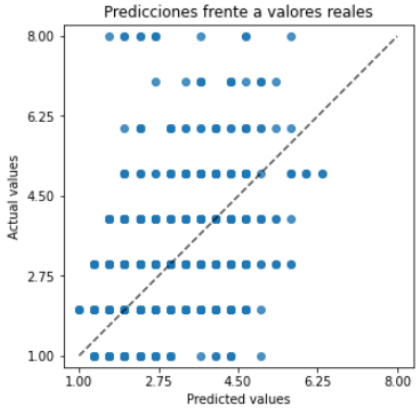


Imagen 4.17 Predicciones de las capas de test tras el 10-cross fold validation frente a los valores reales de desgaste de neumáticos utilizando 3-NN.

Conclusión

Teniendo en cuenta las configuraciones con mejores resultados de cada algoritmo, en las tablas 4.25, 4.26 y 4.27 se muestran las métricas correspondientes para cada regresor analizado.

calcularTiempo()

Método	R Square	MAE	MSE	RMSE	MAPE
Regresión lineal	0.821 ± 0.110	0.356 ± 0.019	0.189 ± 0.022	0.434 ± 0.025	$0.36\% \pm 1.9 \cdot 10^{-4}$
SVR	0.958 ± 0.029	0.158 ± 0.040	0.048 ± 0.027	0.212 ± 0.055	$0.16\% \pm 4.2 \cdot 10^{-4}$
MLP	0.723 ± 0.294	0.382 ± 0.107	0.249 ± 0.125	0.483 ± 0.125	0.38% ± 0.001
Decision Tree Regressor	0.458 ± 0.391	0.609 ± 0.187	0.589 ± 0.334	0.742 ± 0.196	0.62% ± 0.001
KNN	0.407 ± 0.465	0.660 ± 0.229	0.690 ± 0.418	0.793 ± 0.247	0.67% ± 0.002

Tabla 4.25 Tabla de métricas con las mejores configuraciones de cada algoritmo utilizado para el regresor calcularTiempo().

El algoritmo elegido para la función *calcularTiempo()* es Support Vector Regressor con los siguientes parámetros:

Kernel: Radial Basis Function.
C: 20
Epsilon: 0.001
Los demás están por defecto.

calcularConsumo()

Método	R Square	MAE	MSE	RMSE	MAPE
Regresión lineal	0.926 ± 0.014	0.033 ± 0.001	$0.001 \pm 2.2 \cdot 10^{-4}$	0.041 ± 0.002	1.8% ± 0.001
SVR	0.919 ± 0.015	0.033 ± 0.002	$0.001 \pm 2.3 \cdot 10^{-4}$	0.043 ± 0.002	1.8% ± 0.001
MLP	0.888 ± 0.021	0.040 ± 0.003	$0.002 \pm 4.9 \cdot 10^{-4}$	0.051 ± 0.004	2.1% ± 0.001

Decision Tree Regressor	0.857 ± 0.032	0.032 ± 0.005	$0.003 \pm 4.7 \cdot 10^{-4}$	0.057 ± 0.004	1.7% ± 0.002
KNN	0.066 ± 0.145	0.128 ± 0.008	0.022 ± 0.002	0.148 ± 0.008	7% ± 0.004

Tabla 4.26 Tabla de métricas con las mejores configuraciones de cada algoritmo utilizado para el regresor calcularConsumo().

El algoritmo elegido para la función *calcularConsumo()* es Support Vector Regressor con los siguientes parámetros:

Kernel: Radial Basis Function.
C: 15
Epsilon: 0.001
Los demás están por defecto.

calcularDesgaste()

Método	R Square	MAE	MSE	RMSE	MAPE
Regresión lineal	0.690 ± 0.071	0.524 ± 0.076	0.455 ± 0.188	0.663 ± 0.125	20% ± 0.031
SVR	0.668 ± 0.096	0.553 ± 0.157	0.570 ± 0.463	0.709 ± 0.259	20% ± 0.027
MLP	0.786 ± 0.048	0.443 ± 0.069	0.311 ± 0.119	0.549 ± 0.095	17.3% ± 0.023
Decision Tree Regressor	0.690 ± 0.122	0.396 ± 0.067	0.415 ± 0.107	0.639 ± 0.074	16.2% ± 0.030
KNN	0.022 ± 0.472	0.898 ± 0.384	1.629 ± 1.442	1.187 ± 0.467	33.1% ± 0.102

Tabla 4.27 Tabla de métricas con las mejores configuraciones de cada algoritmo utilizado para el regresor calcularDesgaste().

El algoritmo elegido para la función *calcularDesgaste()* es Multi Layer Perceptron con los siguientes parámetros:

Neuronas en la capa oculta: 25
Función de activación: Logística
Los demás están por defecto.

4.3 Optimizador genético

El siguiente paso es realizar una búsqueda dentro del espacio de soluciones para encontrar una combinación de parámetros que minimice lo máximo posible el valor de la función objetivo. El espacio de soluciones se compone de $[57 + 57^2 + 57^3] * [3^{116} + 3^{117} + 3^{118}]$ posibilidades diferentes, por lo que se plantea un algoritmo genético para la búsqueda y optimización de los parámetros de carrera.

4.3.1 Representación de las soluciones

Cada solución es una configuración posible de parámetros de carrera. La representación de los individuos se ha realizado mediante una codificación numérica para los genes y la longitud de los cromosomas es variable en función del número de paradas en boxes. Un ejemplo de un individuo codificado:

[3, 2, 23, 45, 2, 1, 2, 2, 3, 1, 2, 3, 1, 1, 2, 1, 1, 2, 3, 1, 2, 1, 2, 3, 3, 1, 2, 1, 3, 2, 3, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 3, 2, 3, 2, 1, 2, 2, 2, 2, 3, 2, 3, 1, 2, 2, 3, 1, 3, 3, 2, 3, 1, 1, 3, 1, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2]

Neumáticos iniciales: El primer gen representa los neumáticos iniciales de la carrera y en este caso significa que la elección ha sido utilizar neumáticos duros.

Número de paradas en boxes: El segundo gen es el número de paradas en boxes durante la carrera y su valor está limitado entre 1 y 3.

Vueltas de parada en boxes: Es un conjunto de genes de longitud variable que indica las vueltas en las que se hacen las paradas en boxes. Su longitud depende del número de paradas y en este caso se elige parar en la vuelta 23 y 45.

Neumáticos a montar en cada parada: Conjunto de genes de la misma longitud que el número de paradas que indica el neumático a montar en cada parada. En este caso se monta neumático medio en la primera parada (vuelta 23) y neumático blando en la segunda parada (vuelta 45).

Indicaciones de ritmo: Conjunto de genes que tienen la misma longitud que el número de vueltas de la carrera (57) y representa las indicaciones de ritmo al piloto por cada vuelta.

Indicaciones de motor: Lo mismo que las indicaciones de ritmo pero representan las indicaciones de motor al piloto.

Cada individuo agrupa todos los genes en sólo un vector. El cromosoma del ejemplo representa una estrategia de dos paradas, por lo que la longitud del cromosoma es de 120 genes. Las longitudes posibles de los individuos son de 118, 120 y 122 genes.

4.3.2 Función objetivo

La función objetivo determina el tiempo total estimado que tarda un individuo en terminar una carrera completa. Haciendo uso de los regresores, se calcula el tiempo estimado de cada vuelta y se actualiza la carga de gasolina y el estado de los neumáticos en base al consumo y desgaste predecidos. Este proceso se repite iterativamente con los valores actualizados y por cada una de las vueltas de la carrera para obtener el tiempo total estimado.

Cada parada en boxes supone una penalización de 25 segundos al tiempo total de la carrera, que es el tiempo estimado de parada que ofrece el simulador para este circuito, además del cambio de neumáticos correspondiente con el que afrontar las siguientes vueltas hasta la próxima parada o hasta el final de la carrera.

Todos los individuos comienzan con un estado de los neumáticos del 100% independientemente de su tipo y con una carga de gasolina de 110 Kg, aunque estos valores no están codificados en su representación.

Los parámetros de la función corresponden con los genes de los individuos comentados en el apartado 4.3.1. Se utiliza un proceso que identifica los genes de los cromosomas y los convierte en formato de parámetros de carrera para que la función objetivo haga su trabajo.

Existen dos opciones posibles en la salida de la función objetivo al evaluar a los individuos:

Estrategia válida: La salida de la función ofrece un valor que se encuentra entre los 5000 y los 8000 segundos. Cuando los parámetros son válidos y se ha completado la carrera se considera una estrategia válida.

Estrategia fallida: En caso de que se devasten los neumáticos antes de terminar la carrera, se agote el combustible, se haga más de una parada en boxes en la misma vuelta o no se utilicen dos tipos diferentes de neumáticos, la función objetivo devuelve un valor de 100000 porque la configuración de parámetros no es válida.

4.3.3 Evaluación

Evaluación

El criterio para evaluar a los individuos y medir su adaptación al entorno es utilizando la función objetivo. Como es un problema de minimización, el *fitness* de cada cromosoma se representa de la siguiente manera:

$$fitness = \frac{\frac{1}{y}}{\sum_{i=1}^{n} \frac{1}{y_i}}$$

y: El valor de la función objetivo para el cromosoma analizado.
y_i: El valor de la función objetivo del individuo iésimo de la población.
n: Número total de individuos que forman la población.

Tras ser evaluados, los individuos que tienen un menor valor en la función objetivo son los que más valor *fitness* tienen y por tanto cuentan con más posibilidades de reproducirse y mantenerse vivos durante el proceso evolutivo.

4.3.4 Operadores de cruce y mutación

Cruce

Cada individuo seleccionado para reproducirse se cruza con otro individuo de la población y generan dos descendientes. Se utilizan tres operadores de cruce diferentes para las pruebas en el estudio del optimizador genético.

Cruce uniforme

En cada gen se escoge aleatoriamente el gen de uno de los padres para formar parte del primer descendiente y el gen del progenitor no seleccionado corresponde al segundo descendiente. Esto se hace con todos los genes y se generan dos individuos que comparten diferente información genética sobre los mismos progenitores.

Ejemplo

Padre: [1,**2,15,45,2,2**,1,1,2,…,1,2,3,3,3,2,...,2,2,1] (longitud 120)
Madre: [3,**1,24,1**,3,3,1,…,1,1,1,1,1,1,...,3,3,3] (longitud 118)

Descendiente 1: [3,**2,15,45,2,2**,1,3,2,...,1,2,1,3,1,2,...,3,2,3] (longitud 120)
Descendiente 2: [1,**1,24,1,**3,1,1,...,1,1,3,1,3,1,...,2,3,1] (longitud 118)

Cruce por un punto

Se selecciona un punto aleatorio común a los dos cromosomas progenitores. Desde el primer gen de uno de los progenitores hasta el punto selecionado, la información genética corresponde al primer descendiente. Desde el punto hasta el último gen del segundo progenitor, también va al primer descendiente. Todo el material genético que no corresponde al primer descendiente define al segundo descendiente.

Ejemplo

Padre: [1,**2,15,45,2,2,**1,1,2,…,1,2,3,3,3,2,…,2,2,1]
Madre: [3, **1,24,1,** 3,3,1,…,1,1,1,1,1,1,…,3,3,3]

Descendiente 1: [1,**2,15,45,2,2,**1,1,2,…,1,1,1,1,1,1,…,3,3,3]
Descendiente 2: [3,**1,24,1,**3,3,1,…,1,2,3,3,3,2,…,2,2,1]

Cruce por dos puntos

Este tipo de cruce sigue exactamente la misma lógica que el cruce por un punto con la diferencia de que se seleccionan dos puntos aleatorios en vez de uno. El funcionamiento es similar como se muestra en el ejemplo.

Ejemplo

Padre: [1,**2,15,45,2,2,**1,1,2,…,1,2,3,3,3,2,…,2,2,1]
Madre: [3, **1,24,1,** 3,3,1,…,1,1,1,1,1,1,…,3,3,3]

Descendiente 1: [1,**2,15,45,2,2,**1,1,2,…,1,1,1,1,1,2,…,2,2,1]
Descendiente 2: [3,**1,24,1,**3,3,1,…,1,2,3,3,3,1,…,3,3,3]

Los genes indicados en negrita se juntan y actúan como si fueran sólo uno, mientras que el resto de genes actúan de manera independiente. Esto es porque el número de vueltas, vueltas de parada y neumáticos a montar en cada parada son genes que están relacionados y actúan en conjunto. Los genes subrayados son los puntos de corte de los cromosomas.

Mutación

Existe una posible mutación en los genes de los individuos y se basa en cambiar aleatoriamente el valor de uno de ellos. El único gen que no puede ser cambiado en las mutaciones es el respectivo al número de paradas en boxes.

4.3.5 Selección

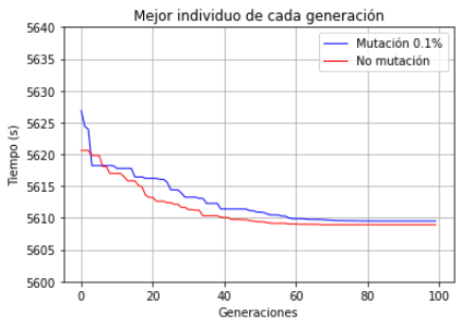
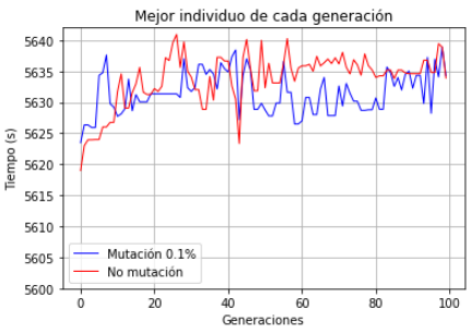
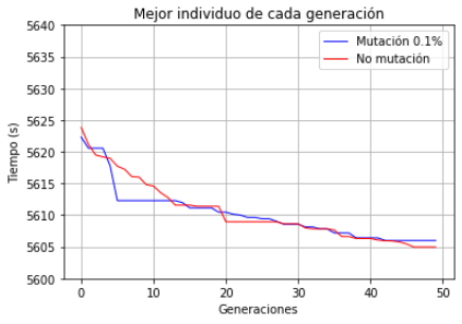
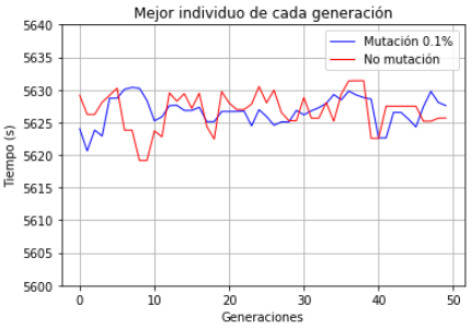
Para seleccionar a los individuos de una población y definir la población de la siguiente generación, se hacen pruebas utilizando el método de la ruleta y un elitismo del 10%. Cuando se producen todos los cruces y mutaciones de cada generación, la población de tamaño *n* se convierte en una población de 2*n* y se seleccionan *n* individuos para formar la nueva población, ya que su tamaño es constante durante todo el proceso evolutivo.

4.3.6 Experimentación

Para atacar el problema de optimización se analiza el comportamiento de los parámetros que influyen en el proceso evolutivo del algoritmo para saber qué valores tienen mejor rendimiento y conseguir una configuración que encuentre las mejores soluciones posibles.

Se han ejecutado 12 pruebas iniciales para evaluar el rendimiento del optimizador en función de su operador de cruce, método de selección y tasa de mutación. La población inicial se genera de manera aleatoria en todas las pruebas y el tamaño es de 1000 individuos.

Debido al coste computacional y tiempo requerido para las ejecuciones, primero se han realizado 4 pruebas iniciales de 100 generaciones con el operador de cruce uniforme para conocer la influencia de las mutaciones y del método de selección. Habiendo obtenido la máxima información sobre los métodos de selección y el factor mutación durante más generaciones, se ejecutan pruebas de 50 generaciones para comparar el funcionamiento de los operadores de cruce y poder construir un optimizador con la mejor configuración posible para la prueba final.

<table>
<tr><th></th><th>Elitismo 10%</th><th>Ruleta</th></tr>
<tr><td rowspan="2">Cruce uniforme</td><td> 
</td><td> 
</td></tr>
<tr><td>Mejor tiempo: 1:33:29
(a)</td><td>Mejor tiempo: 1:33:38
(b)</td></tr>
<tr><td rowspan="2">Cruce por un punto</td><td> 
</td><td> 
</td></tr>
<tr><td>Mejor tiempo: 1:33:25
(c)</td><td>Mejor tiempo: 1:33:39
(d)</td></tr>
</table>

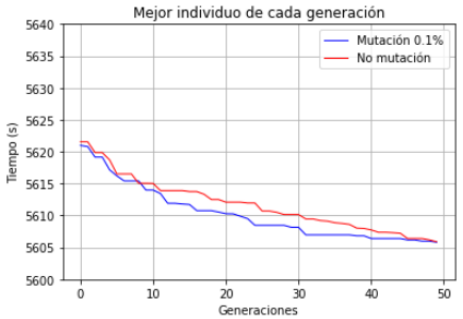
Cruce por dos puntos		
	Mejor tiempo: 1:33:26 (e)	**Mejor tiempo:** 1:33:38 (f)

Tabla 4.28 Comparativa de rendimiento del algoritmo en función de los parámetros del proceso evolutivo.

Para el operador de cruce uniforme, el método de la ruleta parece combinar mejor con las mutaciones que el elitismo, aunque el elitismo ofrece mejores resultados que el mecanismo de la ruleta. El mejor individuo evaluado pertenece a la prueba del elitismo sin mutación. Parece que el mecanismo de la ruleta no converge en 100 generaciones y con el elitismo no mejoran mucho los resultados a partir de la generación 60.

Para el operador de cruce por uno y dos puntos, el elitismo ofrece mejores resultados que el método de la ruleta y el factor mutación mejora los resultados sólo en el caso de emplear el método de la ruleta y cruce por dos puntos, aunque con el elitismo la diferencia no es muy elevada. El método de la ruleta no converge en ninguno de los casos y con el elitismo parece que hacen falta más generaciones para converger pero la tendencia es de bajar los tiempos. En ambos tipos de cruce el mejor individuo ha sido en la prueba de elitismo sin mutación.

Por último, los operadores de cruce por puntos convergen más rápido que el operador de cruce uniforme, ya que para menos generaciones los resultados son mejores en todos los casos. La diferencia en este caso no es muy notable pero el operador de cruce por un punto ha obtenido mejores resultados que el operador de dos puntos. Los dos mejores resultados de las pruebas los ofrecen el operador de cruce por un punto sin mutación y el operador de cruce por dos puntos con mutación, respectivamente y ambos con selección elitista.

Con esta información se realizan dos pruebas finales para intentar llegar a la solución óptima global. Estas pruebas utilizan los parámetros de los dos mejores resultados obtenidos en las pruebas iniciales pero se ejecutan para 100 generaciones.

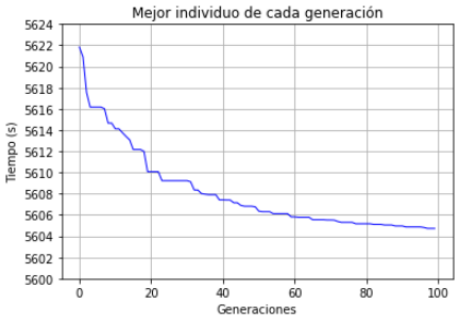
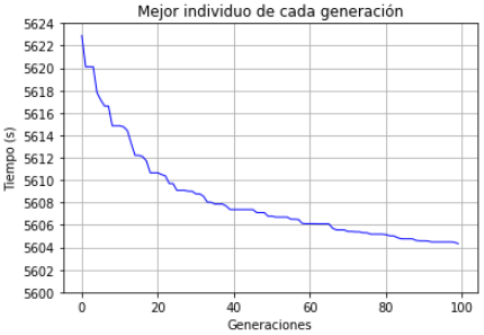
Cruce por un punto sin mutación y con elitismo	Cruce por dos puntos con mutación y con elitismo
	
Mejor tiempo: 1:33:24 (g)	**Mejor tiempo:** 1:33:24 (h)

Tabla 4.29 Comparativa de rendimiento de las dos pruebas finales.

4.3.7 Resultados

Tras ejecutar las pruebas finales, se observa que la mejora con respecto a las pruebas iniciales no es muy notable. A partir de la generación 60 el algoritmo genético converge y los mejores individuos de cada generación se diferencian por milésimas de segundo. A continuación se muestran las tres mejores estrategias obtenidas por el algoritmo genético.

Estrategia A
Tiempo total de carrera estimado: 01:33:24
Gomas iniciales: 1 (Blandos)
Número de paradas: 2
Vueltas de parada: [18, 39]
Neumáticos a montar: [2, 1] (Medios, Blandos)
Ritmos: [3, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 2, 3, 3, 2, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 3, 2, 2, 2, 3, 3, 3, 3, 3, 3, 1, 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3]
Mezclas: [3, 3, 3, 3, 2, 3, 2, 3, 3, 3, 2, 3, 2, 1, 3, 3, 3, 2, 1, 3, 3, 3, 2, 3, 2, 2, 2, 3, 2, 2, 1, 3, 2, 3, 3, 2, 3, 2, 3, 2, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3, 2, 1, 2, 3, 3]

Estrategia B
Tiempo total de carrera estimado: 01:33:24
Gomas iniciales: 2 (Medios)
Número de paradas: 2
Vueltas de parada: [23, 39]
Neumáticos a montar: [1, 1] (Blandos, Blandos)
Ritmos: [2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 1, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 2, 2, 2, 1, 2, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3]
Mezclas: [2, 3, 3, 3, 3, 1, 2, 1, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 1, 3, 3, 3, 2, 2, 1, 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 3, 3]

Estrategia C
Tiempo total de carrera estimado: 01:33:25
Gomas iniciales: 1 (Blandos)
Número de paradas: 2
Vueltas de parada: [20, 40]
Neumáticos a montar: [1, 2] (Blandos, Medios)
Ritmos: [2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 3, 3, 3, 3, 2, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 3, 3, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 2, 3, 2, 3, 3, 3, 3]
Mezclas: [3, 3, 3, 3, 2, 2, 3, 3, 1, 3, 3, 3, 2, 2, 3, 3, 3, 2, 3, 3, 2, 2, 1, 3, 1, 3, 2, 2, 1, 1, 1, 2, 2, 3, 3, 3, 1, 3, 3, 3, 2, 1, 1, 3, 2, 2, 1, 1, 2, 3, 2, 2, 1, 1, 3, 1, 3]

Las tres estrategias tienen en común que se realizan dos paradas en boxes y se utilizan dos compuestos blandos y un compuesto de neumáticos medios. El simulador muestra un informe pre carrera con las mejores estrategias previstas, tal y como se muestra en la imagen 4.18.

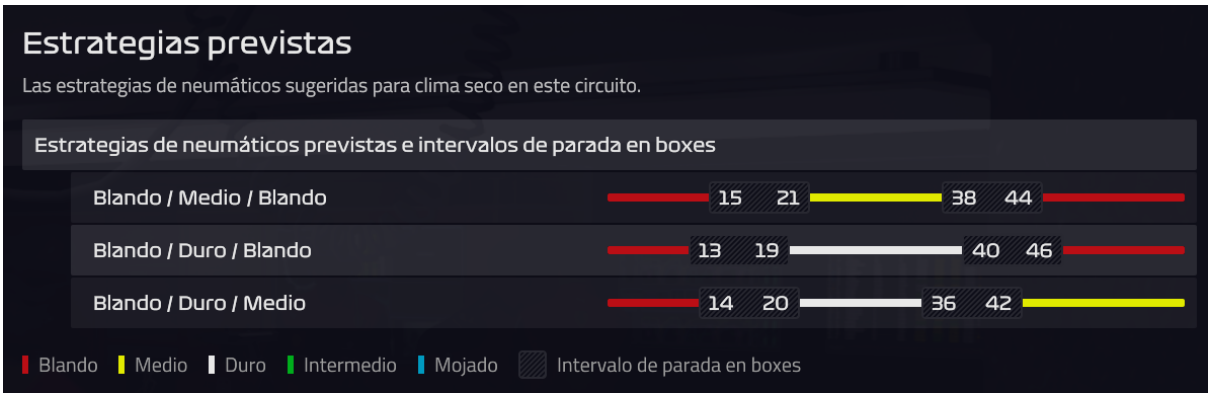


Imagen 4.18 Estrategias de carrera previstas sugeridas por el simulador.

Se observa que la estrategia más rápida sugerida por el simulador encaja con la Estrategia A mostrada anteriormente, ya que tanto los tipos de neumáticos como el orden de uso y ventanas de parada coinciden con la imagen.

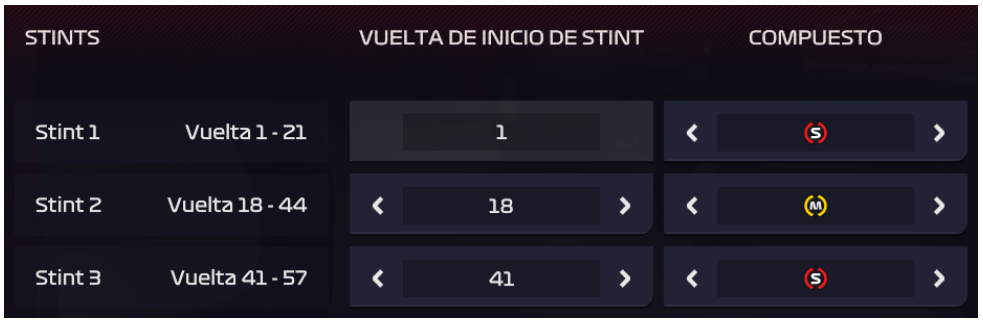


Imagen 4.19 Configuración de la información sobre las paradas en boxes y los compuestos a utilizar en la estrategia A.

El simulador también ofrece una estimación del tiempo total de la carrera según los parámetros de carrera que selecciones, como se muestra en las imágenes 4.20, 4.21 y 4.22

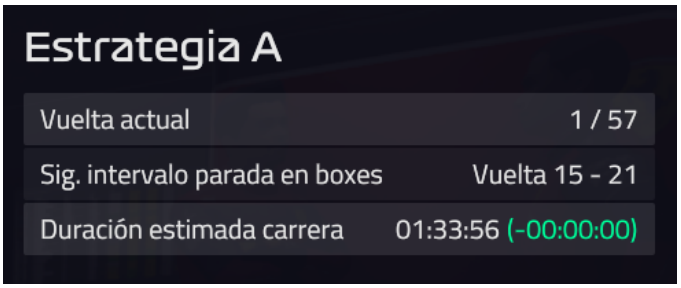


Imagen 4.20 Duración estimada de carrera ofrecida por el simulador seleccionando los parámetros de carrera respectivos a la Estrategia A calculada por el algoritmo genético.

Imagen 4.21 Duración estimada de carrera ofrecida por el simulador seleccionando los parámetros de carrera respectivos a la Estrategia B calculada por el algoritmo genético.

Imagen 4.22 Duración estimada de carrera ofrecida por el simulador seleccionando los parámetros de carrera respectivos a la Estrategia C calculada por el algoritmo genético.

La media de la desviación en las estimaciones del simulador con respecto a las calculadas por el algoritmo genético es de 34 segundos aproximadamente. En el caso de las estrategias B y C, aparece un tiempo delta positivo de 15 y 8 segundos, que significa que las estrategias B y C planteadas por el algoritmo genético son 15 y 8 segundos más rápidas que la segunda y tercera estrategia más rápidas por defecto calculadas por el simulador. Estas estrategias aparecen en las imágenes 4.23 y 4.24 y coinciden con la información mostrada en la imagen 4.18.

Imagen 4.23 Segunda estrategia más rápida por defecto ofrecida por el simulador.

Imagen 4.24 Tercera estrategia más rápida por defecto ofrecida por el simulador.

Para realizar una comparativa adecuada con un escenario real, únicamente se dispone del GP de Bahréin de 2022, ya que son los mismos monoplazas, pilotos y normativa. En este Gran Premio salió el coche de seguridad 2 veces, lo que influye en el tiempo total de la carrera y en las estrategias de los pilotos, que normalmente aprovechan para parar otra vez y montar neumáticos nuevos. La carrera se completó en 1:37:33 y en la página oficial de la Fórmula 1 publicaron las estrategias de parada en boxes esperadas junto con las ventanas de parada, como se muestra en la imagen 4.21.

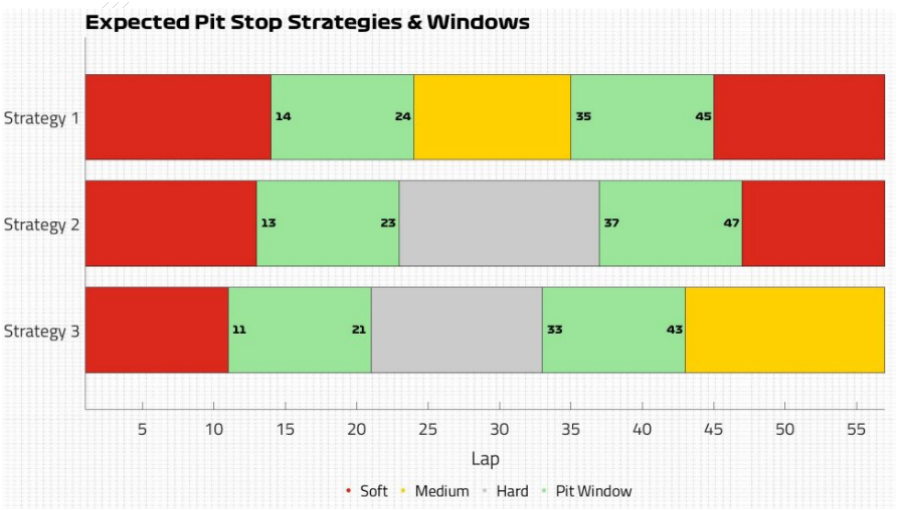


Imagen 4.21 Estrategias de parada en boxes esperadas para el GP de Bahréin de 2022.

La imagen 4.21 muestra cómo la estrategia esperada más rápida se compone de dos paradas y un orden de neumáticos: Blandos, Medios, Blandos. Además, la ventana de parada de la primera parada (para montar neumáticos medios) es entre la vuelta 14 y la 24 y la ventana de parada de la segunda (para montar neumáticos blandos) es entre la vuelta 35 y la 45.

5. Conclusiones y líneas futuras

Tras la recogida de los datos del simulador se implementan varios regresores para predecir el comportamiento del coche y estimar el tiempo total de una carrera, debido a la inviabilidad de utilizar el simulador para ejecutar carreras con todas las combinaciones de parámetros posibles. Haciendo uso de estos regresores, se diseña un algoritmo genético para la búsqueda y optimización de soluciones dentro del espacio de posibilidades variando los parámetros que lo definen y ajustando el modelo para obtener la estrategia que complete la carrera en el menor tiempo posible.

Los regresores obtienen resultados satisfactorios y predicen de forma precisa el tiempo por vuelta, el consumo de combustible y el desgaste de los neumáticos. Estos algoritmos son capaces de estimar el tiempo total de las carreras más rápido de lo que lo hace el simulador y con resultados similares, por lo que permiten plantear un optimizador genético que obtenga soluciones realistas y prometedoras. Además, la segunda y tercera estrategia más rápidas planteadas por el simulador son más lentas que la segunda y tercera estrategias más rápidas calculadas por el algoritmo genético.

Los resultados del algoritmo genético varían según los operadores de cruce, criterios de selección y tasa mutativa, pero utilizando la configuración que ofrece el mejor rendimiento en las pruebas realizadas, se consiguen resultados satisfactorios ya que se asemejan al simulador y a la realidad. Las configuraciones con mejor rendimiento han sido utilizando la selección elitista y los operadores de cruce por puntos, ya que el cruce uniforme no obtiene tan buenos resultados y tarda más en converger, mientras que el método de la ruleta para la selección es peor que el elitismo en todos los casos. La diferencia entre utilizar mutación y no utilizarla no es muy significativa en los resultados para ninguna de las configuraciones probadas.

Como posibles líneas futuras, se podría destacar la consideración de más parámetros de carrera como las condiciones meteorológicas, la temperatura y el estado del asfalto o las indicaciones al piloto del ERS. También sería interesante aplicar el análisis a todos los circuitos del calendario e incluso a todos los pilotos de la parrilla.

En la Fórmula 1 las estrategias no son estáticas debido a que durante las carreras existe la posibilidad de que ocurran sucesos aleatorios como la salida del coche de seguridad virtual o real, banderas rojas y amarillas, accidentes, tráfico en pista o cambios meteorológicos. Sería interesante plantear un modelo que sea capaz de cambiar las estrategias durante las carreras en función de estos fenómenos y encontrar siempre las mejores opciones para cada situación.

Por otro lado, se podrían considerar los compuestos de lluvia y utilizar más opciones en las indicaciones al piloto, ya que en el proyecto se utilizan 3 pero el simulador ofrece la posibilidad de seleccionar 5 posibles indicaciones de ritmo, motor y ERS.

Otra opción interesante sería tener en cuenta a todos los pilotos durante la carrera, para predecir en qué vuelta van a hacer las paradas o para estimar cuándo van a adelantar al piloto de delante o van a ser adelantados por el piloto de detrás.
Esto es interesante porque cada equipo cuenta con dos pilotos en la pista y se pueden combinar ambas estrategias para conseguir el mejor resultado en términos de equipo.

Por último, podría considerarse la información de los entrenamientos no sólo para analizar y predecir las estrategias de las carreras, si no también para las sesiones de clasificación. Estas últimas consideraciones están enfocadas para lograr un mejor resultado de equipo y no únicamente del piloto, por lo que es necesario, mínimo, considerar a los dos pilotos del equipo.

6. Gestión del proyecto

En este capítulo se detallan las tareas necesarias y sus dependencias para lograr el objetivo final del proyecto y se muestran las desviaciones durante el proceso de desarrollo con respecto a la planificación inicial.

6.1 Descripción

El proyecto se basa en la búsqueda de la configuración de parámetros que minimice lo máximo posible el tiempo necesario para terminar una carrera completa dentro de un simulador de Fórmula 1. Se realizan simulaciones en F1 Manager 22 y se extraen los datos necesarios para entrenar los modelos de regresión necesarios e implementar un optimizador genético. La recogida de los datos se extrae manualmente y tanto el análisis como la implementación del algoritmo genético se desarrollan a través del entorno Google Colab con el lenguaje de programación python.

6.2 Planificación de tareas y desviaciones

El desarrollo del trabajo se divide en tres bloques compuestos por tareas como se detalla a continuación.

Recogida de datos: Esta primera fase representa las tareas relacionadas con la selección del simulador a utilizar y la recogida de los datos que ofrecen las simulaciones.

B1.T1: Búsqueda de información sobre los posibles simuladores que pueden ser seleccionados para utilizar en el proyecto y toma de decisión sobre cuál emplear.

B1.T2: Análisis de las posibilidades del simulador seleccionado y definición de los objetivos y parámetros a considerar para plantear las simulaciones.

B1.T3: Ejecución de las simulaciones y recogida de los datos para su almacenamiento en tablas.

Análisis de datos: La segunda fase recoge las tareas relacionadas con el análisis de los datos recogidos y la implementación de los modelos de regresión.

B2.T1: Búsqueda de información sobre los conceptos teóricos necesarios para el análisis y selección de los algoritmos a utilizar en la implementación de los regresores.

B2.T2: Implementación de los regresores utilizando los algoritmos seleccionados en B2.T1. Aplicar distintas configuraciones en los parámetros de cada algoritmo.

B2.T3: Ejecutar los regresores y evaluar los resultados aplicando las métricas correspondientes.

B2.T4: Realizar una comparativa con los resultados de todas las configuraciones de los algoritmos utilizados para seleccionar el mejor algoritmo y configuración de parámetros para cada regresor.

Optimizador genético: El último bloque hace referencia a las tareas relacionadas con el algoritmo genético y la resolución del problema de optimización.

B3.T1: Investigación sobre los fundamentos teóricos necesarios para el planteamiento e implementación del algoritmo genético.

B3.T2: Implementación de la función objetivo y de los aspectos que definen el proceso evolutivo del algoritmo.

B3.T3: Ejecutar diferentes pruebas variando los parámetros del optimizador y recoger los resultados obtenidos.

B3.T4: Evaluar los resultados para ejecutar una prueba final con los parámetros que han ofrecido un mejor rendimiento durante las pruebas.

B3.T5: Ejecutar la prueba final y evaluar los resultados realizando una comparativa con los resultados del simulador y con un escenario real.

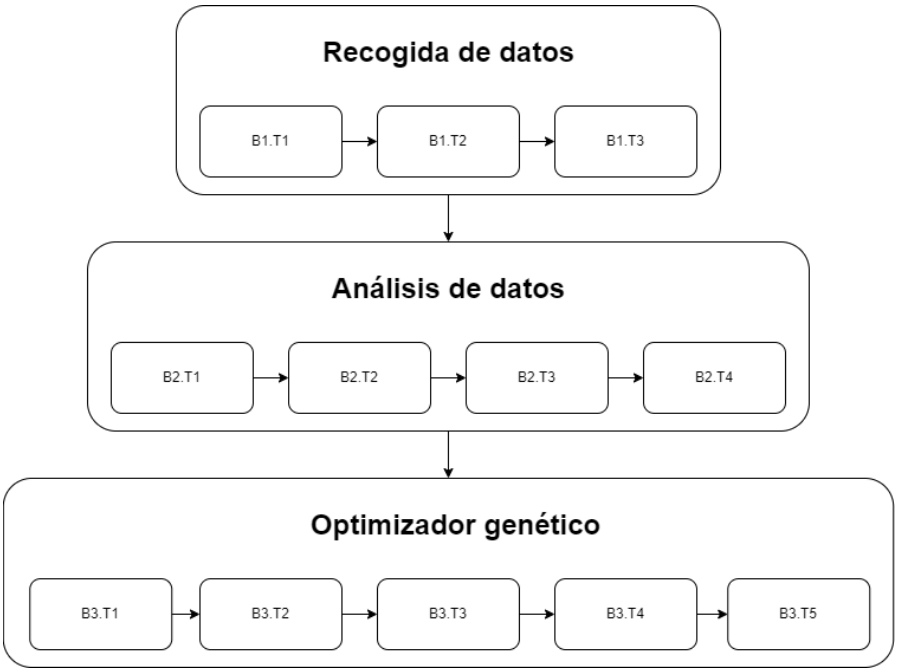


Tabla 6.1 Diagrama de dependencias entre las tareas.

Como se muestra en la tabla 6.1, las dependencias entre las tareas definidas son de fin a comienzo ya que la finalización de una tarea permite el comienzo de la siguiente y no pueden realizarse en paralelo. Ocurre lo mismo para los bloques ya que deben ser realizados de manera ordenada y secuencial.

Diagrama de Gantt - Estimación

Tarea	Fecha inicio	Fecha fin	Noviembre				Diciembre				Enero			
			Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12
B1.T1	02/11/2022	06/11/2022												
B1.T2	07/11/2022	14/11/2022												
B1.T3	15/11/2022	29/11/2022												
B2.T1	30/11/2022	07/12/2022												
B2.T2	08/12/2022	18/12/2022												
B2.T3	19/12/2022	22/12/2022												
B2.T4	23/12/2022	27/12/2022												
B3.T1	28/11/2022	30/11/2022												
B3.T2	02/01/2023	12/01/2023												
B3.T3	13/01/2023	15/01/2023												
B3.T4	16/01/2023	20/01/2023												
B3.T5	21/01/2023	23/01/2023												

Tabla 6.2 Diagrama de Gantt sobre la estimación de los marcos temporales de las tareas.

Diagrama de Gantt - Real

Tarea	Fecha inicio	Fecha fin	Noviembre				Diciembre				Enero			
			Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12
B1.T1	02/11/2022	23/11/2022												
B1.T2	24/11/2022	15/12/2022												
B1.T3	16/12/2022	25/12/2022												
B2.T1	26/12/2022	26/12/2022												
B2.T2	27/12/2022	29/12/2022												
B2.T3	29/12/2022	03/01/2023												
B2.T4	03/01/2023	05/01/2023												
B3.T1	06/01/2023	08/01/2023												
B3.T2	08/01/2023	12/01/2023												
B3.T3	12/01/2023	17/01/2023												
B3.T4	17/01/2023	19/01/2023												
B3.T5	20/01/2023	25/01/2023												

Tabla 6.3 Diagrama de Gantt sobre los marcos temporales reales llevados a cabo para completar las tareas.

Tal y como muestra la tabla 6.3, ha habido una desviación considerable en el cumplimiento de las tareas con respecto a la planificación inicial. La razón principal de la desviación fue la dificultad de definir y concretar los objetivos del proyecto, debido a la dependencia directa con los simuladores.

El simulador seleccionado es bastante complejo ya que ofrece una gran cantidad de parámetros y posibilidades, por lo que concretar qué parámetros utilizar, cómo definir una estrategia de carrera y qué simulaciones ejecutar para el estudio y resolución de este proyecto supuso más tiempo del planificado.

Una vez concretados estos aspectos, no ha habido grandes dificultades durante el desarrollo de las tareas del bloque 2 y 3, ya que la mayoría se han cumplido en un tiempo menor al estimado. Durante las últimas semanas hubo una ligera sobrecarga de trabajo debida en gran parte a los problemas para hacer ejecuciones de muchas horas desde el entorno Google Colab.

Bibliografía

[1] Formula 1 and Machine Learning
Recuperado de https://www.elucidate.ai/post/formula-1-and-machine-learning

[2] How Formula 1 Incorporates Amazon's Ai and Machine
Recuperado de https://www.jumpstartmag.com/how-formula-1-incorporates-amazons-ai-and-machine/

[3] What is Supervised Learning? | IBM
Recuperado de https://www.ibm.com/topics/supervised-learning

[4] Regresión lineal con python | Ciencia de datos
Recuperado de https://www.cienciadedatos.net/documentos/py10-regresion-lineal-python.html

[5] Regresión lineal múltiple - Documentación de IBM
Recuperado de https://www.ibm.com/docs/es/cognos-analytics/11.1.0?topic=tests-multiple-linear-regression

[6] Support Vector Machines (SVM) Algorithm Explained
Recuperado de https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[7] What is the influence of C in SVMs with linear kernel? - Cross Validated
Recuperado de https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

[8] Support Vector Regression (SVR) | JacobSoft
Recuperado de https://www.jacobsoft.com.mx/es_mx/support-vector-regression/

[9] Support Vector Regression (SVR) using linear and non-linear kernels
Recuperado de https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

[10] Support Vector Regression Clearly Explained - Joseph Rivera
Recuperado de https://www.youtube.com/watch?v=NmXBr6gfUkA&t=331

[11] Arquitectura (Perceptrón multicapa) - Documentación de IBM
Recuperado de https://www.ibm.com/docs/es/spss-statistics/saas?topic=perceptron-architecture-multilayer

[12] What is ReLU and Sigmoid activation function? - Nomidl
Recuperado de
https://www.nomidl.com/deep-learning/what-is-relu-and-sigmoid-activation-function/

[13] El backpropagation y la solución a los problemas del perceptrón multicapa
Recuperado de https://keepcoding.io/blog/backpropagation-solucion-perceptron-multicapas/#:~:text=El%20backpropagation%20b%C3%A1sicamente%20lo%20que,van%20de%20delante%20hacia%20atr%C3%A1s

[14] How Decision tree classification and regression algorithm works - ArcGIS Pro | Documentation
Recuperado de https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-decision-tree-classification-and-regression-works.htm

[15] K-Nearest Neighbors (KNN) in Python | DigitalOcean
Recuperado de https://www.digitalocean.com/community/tutorials/k-nearest-neighbors-knn-in-python

[16] Evaluation Metrics for Regression Models
Recuperado de https://www.enjoyalgorithms.com/blog/evaluation-metrics-regression-models

[17] How To Compute Standard Deviation in NumPy - Spark By Examples
Recuperado de https://sparkbyexamples.com/numpy/python-numpy-standard-deviation-function/

[18] Cross-Validation en Python
Recuperado de https://deepnote.com/@a_mas/Cross-Validation-en-Python-685fa851-b5b2-4c5b-b5fb-3dc5ae64838f

[19] Algoritmos Genéticos en 5 minutos - BitBoss
Recuperado de https://www.youtube.com/watch?v=RBrXGyo0kIw&t=158s

[20] Introduction to Optimization with Genetic Algorithm | by Ahmed Gad | Towards Data Science
Recuperado de https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b

[21] Genetic Algorithms Tutorial
Recuperado de https://www.tutorialspoint.com/genetic_algorithms/index.htm

[22] Genetic Algorithm Implementation in Python | by Ahmed Gad | Towards Data Science
Recuperado de https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6

[23] Métodos de optimización: Algoritmos genéticos.
Recuperado de https://www.tradingsys.org/metodos-de-optimizacion-algoritmos-geneticos

[24] Uniform Crossover in Genetic Algorithm - Google Scholar
Recuperado de https://www.researchgate.net/profile/Gilbert-Syswerda-3/publication/201976488_Uniform_Crossover_in_Genetic_Algorithms/links/5f84cf27299bf1b53e22ee7c/Uniform-Crossover-in-Genetic-Algorithms.pdf

[25] How to define a Fitness Function in a Genetic Algorithm? | by Vijini Mallawaarachchi | Towards Data Science
Recuperado de https://www.tradingsys.org/metodos-de-optimizacion-algoritmos-geneticos

[26] Mutation Operator - an overview | ScienceDirect Topics
Recuperado de https://www.sciencedirect.com/topics/engineering/mutation-operator

[27] Google Colab: Todo lo que necesita saber - Geekflare
Recuperado de https://geekflare.com/es/google-colab/

[28] What is NumPy? - NumPy v1.24 Manual
Recuperado de https://numpy.org/doc/stable/user/whatisnumpy.html

[29] Seaborn. Statistical Data Visualization - Seaborn 0.12.2 Documentation
Recuperado de https://seaborn.pydata.org

[30] What is Pandas in Python? Everything You Need To Know - ActiveState
Recuperado de https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/

[31] What is Matplotlib in Python? How to use it for plotting? - ActiveState
Recuperado de https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/

[32] Scikit Learn - Introduction
Recuperado de https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm

Apéndice: Enlace al código fuente del proyecto

Enlace de GitHub al código fuente y dataset utilizados en el proyecto:
AMalo99/Optimizaci-n-de-par-metros-F1 (github.com)