Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

Desarrollo de modelos de aprendizaje automático para la estimación del Brix en el proceso de evaporación de una azucarera

Realizado por **Patricia Baz Domínguez**

✠✠✠

Universidad de Valladolid

15 de septiembre de 2023

Tutores: Diego García Álvarez

Carlos Javier Alonso González

Agradecimientos

A pesar de que un TFM va firmado por una única persona, son muchas las que ayudan a que salga adelante. Cada una de las personas que nombro a continuación me ha ayudado a su manera, aportando lo que estaba a su alcance en cada momento; han sido mi motor en la realización de este trabajo, tanto a nivel académico como psicológico.

En primer lugar, me gustaría dar las gracias a mis tutores, D. Carlos Javier Alonso González y D. Diego García Álvarez, por conducirme tan amablemente en este trabajo, transmitirme su pasión por el tema y dedicar tardes enteras a *debuggear* modelos en VSCode conmigo. También a los profesores D. Aníbal Bregón Bregón y D. Belarmino Pulido Junquera, por recibirme de una forma tan acogedora en el GSI y ofrecerme su apoyo incondicional durante todo el proyecto. En esta línea, todos los miembros del GSI han contribuído en la consecución de este trabajo; en especial, mi compañera de laboratorio y amiga Paula, quien me acompaña desde hace ya 6 años tanto dentro de las aulas como fuera y me anima a seguir cuando ya no puedo más. Te voy a echar de menos en las oficinas, Paula.

En segundo lugar, me gustaría llevar este agradecimiento hasta Segovia acordándome de mis tutores del TFG de Ingeniería Informática de Servicios y Aplicaciones, D. Miguel Ángel Martínez Prieto y D. Jorge Silvestre Vilches, ya que fueron quienes me animaron y me recomendaron el Máster en Ingeniería Informática y, sobre todo, quienes me hicieron decantarme por la informática y los datos como vocación en mi vida. También fueron ellos quienes me informaron acerca de la beca del Consejo Social asignada al presente trabajo.

Quiero agradecer también al resto de personal de la Universidad de Valladolid (profesores del máster, personal administrativo, conserjes, personal de limpieza, etc.) por transmitir su conocimiento y valores desde la cercanía y la generosidad.

Por otro lado, quería destacar la ayuda de Alejandro Merino Gómez, facilitador del conjunto de datos usado en el proyecto y experto que me explicó con detalle el proceso de obtención del azúcar y me guió en el estudio de los *outliers*. En esta línea, cabe mencionar también a Desirée Arias Requejo, quien aportó ideas para la creación del modelo LSTM ED a futuro con predicción de secuencia mediante uno de sus *papers*.

Además, quiero mencionar a mi familia y a mis amigos del pueblo, del grado, del máster y del programa SICUE, que siempre tratan de sacar lo mejor de mí y me aconsejan en todas y cada una de las decisiones que he de tomar en busca de mi felicidad. Muchas gracias por estar a mi lado en uno de los años más difíciles de mi vida.

Resumen

La concentración de azúcar se mide en grados Brix, los cuales se obtienen midiendo la concentración de materia seca en el líquido que sale de la etapa de evaporación del proceso de obtención de azúcar en la industria azucarera. Actualmente existen métodos de medida del Brix que consisten en estudios de laboratorio (lentos) o sensores en tiempo real (costosos), pero éstos son limitados y pueden presentar problemas de ensuciamiento o cristalización. También existen métodos de estimación, que son sensores software basados en modelos de Aprendizaje Automático o *Machine Learning* (ML) o estadísticos que permiten predecir el Brix a la salida de la etapa de evaporación en base a otras medidas del proceso. Sin embargo, son pocos los modelos probados hasta el momento: el mejor de estos sensores *soft* ha logrado un Error Cuadrático Medio o *Mean Absolute Error* (MSE) de 0,45.

Para solucionar este problema, este proyecto propone el uso de modelos de ML no probados hasta el momento y que permitan estimar el Brix a la salida de la etapa de evaporación en base a otras medidas del proceso. Se estudia la aplicación de las Redes Neuronales Recurrentes o *Recurrent Neural Networks* (RNNs) a este problema, en concreto, de las redes *Long Short-Term Memory* (LSTM), que captan dependencias a largo plazo en los datos. También se prueba la estructura Codificador-Decodificador o *Encoder-Decoder* (ED) y las predicciones a futuro. Todo ello se implementa con Pytorch, una librería de Aprendizaje Profundo o *Deep Learning* (DL) de Python, y siguiendo la metodología *Cross-Industry Standard Process for Data Mining* (CRISP-DM).

El resultado consta de un total de 7 nuevos sensores, de los cuales 4 superan el valor de la métrica de partida. El mejor resultado se obtiene con un modelo Regresión de Vectores de Soporte o *Support Vector Regression* (SVR), siendo 0,198 el MSE conseguido sobre los datos de *test*. Aún en los modelos de los últimos puestos se observa la capacidad de generalizar y de seguir el patrón de los datos.

Se concluye que los datos de las variables medidas en la industria azucarera no siguen una tendencia que pueda ser aprovechada por la complejidad interna de las redes LSTM, por lo que los modelos básicos siguen siendo tendencia. La mejora en la estimación del Brix mediante el uso de estos modelos permitiría incrementar la productividad y la rentabilidad al facilitar la toma de decisiones de manera inmediata, así como detectar anomalías o desviaciones en el proceso de evaporación. De esta manera, se podrían adoptar rápidamente medidas correctivas para evitar pérdidas en la producción.

Descriptores

Brix, evaporación, azucarera, ML, SVR, DL, LSTM.

Abstract

The sugar concentration is measured in Brix degrees, which are obtained by measuring the concentration of dry matter in the liquid exiting the evaporation stage of the sugar production process in the sugar industry. Currently, there are methods for measuring Brix that involve laboratory studies (slow) or real-time sensors (costly), but these methods have limitations and can encounter issues with fouling or crystallization. There are also estimation methods, which are software sensors based on ML or statistical models that allow for predicting Brix at the output of the evaporation stage based on other process measurements. However, few models have been tested so far, with the best-performing soft sensor achieving a MSE of 0,45.

To address this issue, this project proposes the use of untested ML models that can estimate Brix at the output of the evaporation stage based on other process measurements. The application of RNNs to this problem is investigated, specifically LSTM networks, which capture long-term dependencies in the data. Additionally, the use of ED architecture and future predictions is explored. All of this is implemented using Pytorch, a Python DL library, and following the CRISP-DM methodology.

The outcome consists of a total of 7 new sensors, with 4 of them surpassing the baseline metric. The best result is achieved with a SVR model, with an MSE of 0,198 on the test data. Even in the models at the bottom of the ranking, there is an observed ability to generalize and follow the data patterns.

In conclusion, the data from the variables measured in the sugar industry does not exhibit a trend that can be effectively captured by the internal complexity of LSTM networks, thus basic models remain effective. Improving Brix estimation through the use of these models would enhance productivity and profitability by enabling immediate decision-making and the detection of anomalies or deviations in the evaporation process. This would allow for swift corrective measures to prevent production losses.

Keywords

Brix, evaporation, sugar industry, ML, SVR, DL, LSTM.

Índice general

Índice de figuras

Índice de tablas

1: Introducción

El proceso de obtención de azúcar en la industria azucarera consta de varias fases. Una de ellas es conocida como el proceso de evaporación. En esta etapa se introduce el jugo de caña o remolacha resultante de las fases anteriores, el cual es una mezcla de agua y sacarosa, en una serie de evaporadores con el objetivo de aumentar la concentración de azúcar en el líquido. El jarabe resultante de este proceso pasa al proceso de cristalización, por lo que es clave controlar la concentración de azúcar en el jugo azucarado a la salida de la evaporación. Por una parte, si la concentración de azúcar está por encima de lo deseado, puede cristalizar antes de tiempo y bloquear los conductos de la etapa de evaporación. Por otra parte, si la concentración de azúcar está por debajo de lo deseado, puede que no cristalice en la fase preparada para ello, que es la etapa de cristalización.

Para determinar esa cantidad es habitual medir la concentración de materia seca o *dry substance content*. La unidad de medida más utilizada para ello es el denominado grado Brix (ºBx). Hay sensores físicos que miden el Brix, pero son caros; además, pueden perder precisión en la medición debido al ensuciamiento y la cristalización. Por ese motivo, puede ser útil complementar las medidas físicas con medidas estimadas por sensores software capaces de predecir la cantidad de materia seca en función de otras medidas del proceso.

Para realizar la estimación del Brix, en este proyecto, se propone el uso de una RNN, en concreto, las redes LSTM. Su funcionamiento interno usa datos actuales y pasados de otras medidas del proceso de evaporación para predecir la materia seca de azúcar a la salida, en lugar de sólo las medidas actuales que proponen otros métodos. Además, para comparar el rendimiento de este tipo de algoritmos con otros más sencillos aplicados a este mismo problema, se pretende crear modelos de predicción basados en redes más simples.

Por una parte, este trabajo está estrechamente relacionado con la asignatura "DL y sus Aplicaciones" del Máster en Ingeniería Informática de la Universidad de Valladolid, donde se estudian las RNNs LSTM. Por otra parte, continúa con una línea de investigación del Grupo de Sistemas Inteligentes (GSI) relativa al diseño de sensores software, detección y diagnóstico de fallos y estimación de parámetros, que ha dado lugar a diferentes publicaciones indexadas por parte de los integrantes del grupo. Además, ha sido una de las líneas de investigación de diferentes proyectos financiados.

1.1. Motivación

En esta sección se lleva a cabo un planteamiento del problema que motiva la realización del trabajo.

Actualmente, el deseo de todos los profesionales de la industria azucarera es que el jugo azucarado llegue al proceso de cristalización con el Brix necesario y suficiente, por lo que es clave controlar con precisión la concentración de azúcar del jarabe que sale de la evaporación. Los valores del Brix deben moverse en un margen estrecho, para garantizar que cristaliza en la siguiente etapa (cristalización) y no durante la evaporación. Los limites de este rango dependen del tipo de instalación de la fábrica azucarera. Es decir, lo ideal es poseer datos del Brix a la salida del proceso de evaporación para conseguir los propósitos predefinidos y que esta medición no sea costosa. Aunque existen sensores físicos, estos tienen un elevado coste y pueden perder precisión en la medición debido al ensuciamiento y la cristalización. En consecuencia, surge la necesidad de generar sensores software para disponer de una segunda medida del Brix cuando el sensor físico no opera en óptimas condiciones. La propuesta de este proyecto es estudiar el avance actual de los sensores software en el cálculo del Brix a la salida del proceso de evaporación con el fin de explorar la viabilidad de una Red Neuronal o *Neural Network* (NN) en este ámbito tras observar resultados de otros métodos analíticos y estadísticos. Por un lado, están la estimación indirecta a partir de la presión de vapor y la temperatura del jugo, el uso de NNs, la aplicación de Análisis de Componentes Principales o *Principal Component Analysis* (PCA) para reducir el coste de entrenamiento de la red y el método estadístico de regresión multivariante basado en componentes principales Mínimos Cuadrados Parciales o *Partial Least Squares* (PLS) [18]. Por otro lado, se utilizan algoritmos de ML como SVR y Bosque Aleatorio o *Random Forest* (RF) [44].

1.2. Estructura de la memoria

El presente documento se encuentra dividido en 7 capítulos y 2 apéndices, que tratan los diversos objetivos planteados en el proyecto, junto con algunas consideraciones ligadas a sus procesos de desarrollo y gestión del proyecto.

- Capítulo 2: Enumera y organiza los objetivos del trabajo, así como las restricciones a las que está sujeto.
- Capítulo 3: Ahonda en el proceso de gestión del proyecto, por tanto, incluye la metodología utilizada durante su desarrollo (Sección 3.1), su planificación temporal (Sección 3.2) y el balance o comparación entre lo estimado y la realidad final (Sección 3.3).
- Capítulo 4: Lista las técnicas (Sección 4.1), las tecnologías (Sección 4.2) y las herramientas (Sección 4.3) utilizadas para el desarrollo del proyecto.

- Capítulo 5: Contextualiza el proyecto dentro de la industria azucarera, repasando así el entorno de negocio mediante la Sección 5.1 y la Sección 5.2.
- Capítulo 6: Incluye el estado del arte del proyecto, distinguiendo entre sensores de medida (Sección 6.1) y sensores de estimación (Sección 6.2) del Brix.
- Capítulo 7: Expone la propuesta del proyecto (Sección 7.3), desde el resumen ejecutivo del *dataset* (Sección 7.1) hasta la interpretación de los resultados obtenidos (Sección 7.4), pasando además por la explicación del proceso de transformación de los datos (Sección 7.2).
- Capítulo 8: Analiza el resultado global del proyecto, estudiando el grado de consecución de los objetivos y sacando conclusiones acerca del trabajo y del desempeño personal durante la realización de este Trabajo Fin de Máster (TFM) (Sección 8.1). Termina con un análisis de posibles trabajos futuros (Sección 8.2).
- Bibliografía: Recoge con detalle la lista de fuentes consultadas.
- Apéndice A: Lista las siglas y los acrónimos que aparecen a lo largo de la memoria.
- Apéndice B: Presenta el material que se entrega junto con la memoria.

Se advierte que, durante toda la memoria, los nombres de archivos y los comandos en Python se muestran con una fuente `monoespaciada`, mientras que los extranjerismos y términos técnicos se escriben en *cursiva*.

2: Objetivos del proyecto

En este capítulo se explica de forma detallada cuáles son los objetivos que se persiguen con la realización del proyecto.

El presente proyecto se enmarca en la problemática asociada a la estimación del contenido materia seca mediante sensores software en el proceso de evaporación de la industria azucarera. En esta línea de trabajo se identifica un objetivo fundamental del proyecto:

OBJ-01: Desarrollar un sensor software mediante técnicas de ML para la estimación del Brix en la industria azucarera que mejore los resultados obtenidos por los sensores software ya desarrollados por el grupo.

Para obtener este objetivo, se plantean varios subobjetivos:

OBJ-01.01: Revisar el estado del arte de los sensores software existentes, con el fin de identificar aquellos que proporcionen mejores resultados.

OBJ-01.02: Desarrollar modelos con técnicas de ML para la estimación del Brix utilizando los datos de los sensores disponibles. Dada su relevancia para la predicción de series temporales, se hará hincapié en las redes LSTM.

OBJ-01.03: Comparar y evaluar el rendimiento del modelo propuesto frente a los obtenidos por sensores software desarrollados por el grupo atendiendo a distintas métricas.

El alcance y la duración del proyecto están limitados por tres razones. La primera se comenta en la Subsección 3.2.1 y es la carga de trabajo que especifican las asignaturas en las que se enmarca, esto es, "TFM", asignatura de 6 Sistema Europeo de Transferencia y Acumulación de Créditos o *European Credit Transfer System* (ECTS), e "Investigación, Desarrollo e Innovación o *Research and Development* (I+D+i) en Informática", asignatura

de 6 ECTS, de los cuales sólo 3 ECTS corresponden a este proyecto. Aún así, el convenio de prácticas curriculares de la segunda asignatura estipula que se debe aprovechar un mínimo de 190 horas. Cabe mencionar que este trabajo está asociado a una beca de colaboración con departamentos del Consejo Social de la Universidad de Valladolid, siendo el cómputo total de horas de la beca concedida de 210. Dado que 210 es mayor que 150 y que 190, se parte de la obligación de aprovechar un mínimo de 210 horas de trabajo efectivo en el proyecto. La segunda son los recursos que proporciona el grupo, esto es, un ordenador de 16 *Gigabyte* (GB) de memoria Memoria de Acceso Aleatorio o *Random Access Memory* (RAM) y una máquina virtual de 8 GB de memoria RAM. La tercera es el volumen de datos disponibles. Como se menciona en la Sección 7.1, se dispone de un total de 92998 registros, lo que equivale a un estudio de aproximadamente 11 días de producción de azúcar. Al trabajar con modelos de DL, un mayor número de datos favorece la obtención de experiencia.

3: Plan de proyecto

Este apéndice presenta el plan de proyecto elaborado para la realización del trabajo. En primer lugar, se explica la metodología de trabajo (Sección 3.1) y, a continuación, la planificación (Sección 3.2) y el balance (Sección 3.3) del proyecto.

3.1. Metodología de trabajo

El presente proyecto se ha realizado siguiendo los principios de CRISP-DM, una metodología de minería de datos que se utiliza comúnmente en el campo del análisis de datos y la ciencia de datos. Fue desarrollada por un consorcio de empresas y organizaciones en 1996 y, desde entonces, ha sido ampliamente utilizada.

La metodología CRISP-DM consta de seis fases principales [25], que se pueden observar en la Figura 3.1:

1) Comprensión del negocio: Se define el problema de negocio y se determinan los objetivos del proyecto. Se identifican las metas y se establece el alcance del proyecto. La explicación de esta fase se distribuye entre el Capítulo 4 y el Capítulo 5 del presente documento.

2) Comprensión de los datos: Se recopilan y analizan los datos disponibles. Se determina la calidad de los datos y se identifican los posibles problemas que puedan afectar la precisión y la integridad de los datos. La explicación de esta fase se realiza en la Sección 7.1 del Capítulo 6 del presente documento.

3) Preparación de los datos: Se limpian, transforman y preparan los datos para su uso en el análisis. Se seleccionan los atributos relevantes y se eliminan los datos duplicados o innecesarios. La explicación de esta fase se realiza en la Sección 7.2 del Capítulo 6 del presente documento.

4) Modelado: Se aplican técnicas de modelado para construir un modelo predictivo. Se seleccionan los algoritmos de aprendizaje automático adecuados y se ajustan los

parámetros del modelo para obtener los mejores resultados. La explicación de esta fase se realiza en la Sección 7.3 del Capítulo 6 del presente documento.

5) Evaluación: Se evalúa la precisión del modelo y se determina si cumple con los objetivos del proyecto. Se realizan pruebas para verificar la calidad del modelo y se ajusta si es necesario. La explicación de esta fase se realiza en la Sección 7.4 del Capítulo 6 del presente documento.

6) Despliegue: Se implementa el modelo en el entorno de producción y se monitorea su desempeño. Se crea una documentación detallada y se capacita a los usuarios para utilizar el modelo. El presente documento constituye dicha documentación.

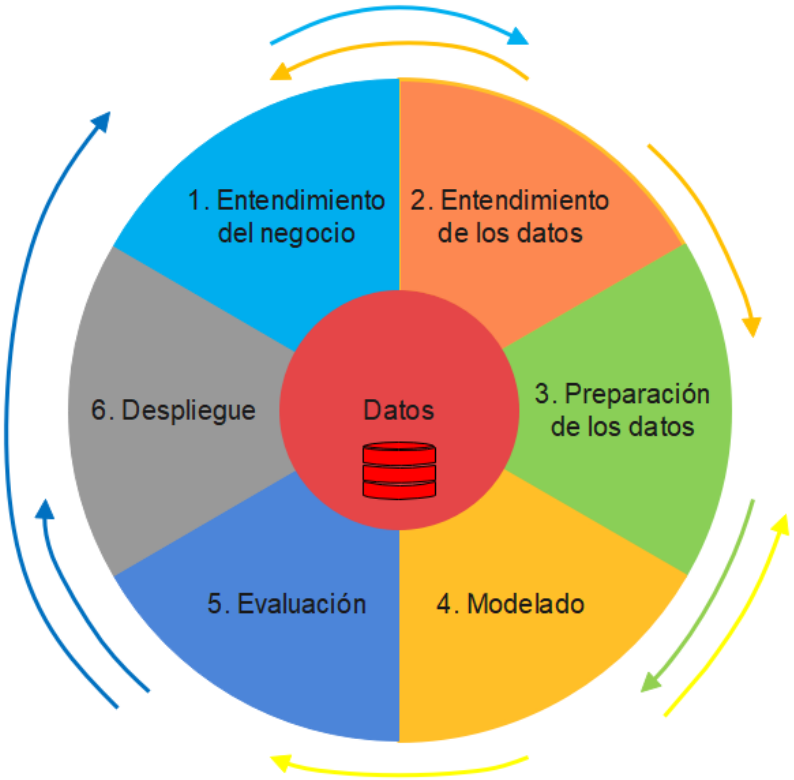

Figura 3.1: Metodología CRISP-DM [Fuente: elaboración propia].

Más allá de que se haya descrito la metodología como un proceso secuencial, es importante incidir en su carácter iterativo. Por ejemplo, la fase de modelado puede motivar un nuevo preprocesamiento de los datos que mejore los análisis realizados.

Por tanto, CRISP-DM es una metodología iterativa y flexible que se puede adaptar a las necesidades específicas de cada proyecto de minería de datos. Ayuda a los equipos de análisis de datos a seguir un proceso estructurado y eficiente para obtener los mejores resultados posibles.

3.2. Planificación

En esta sección, se lleva a cabo una previsión del proyecto, esto es, un análisis temporal y económico previo a la realización del mismo.

3.2.1. Planificación temporal

Este proyecto se realiza, en parte, como trabajo de investigación asociado a las becas de colaboración con departamentos convocadas por el Consejo Social de la Universidad de Valladolid. Es por ello que, a la hora de definir el alcance, se tiene en cuenta la carga de 210 horas exigida por el mencionado contrato. Por otra parte, este proyecto constituye la parte evaluable de dos asignaturas del Máster en Ingeniería Informática, a saber, "I+D+i en Informática" y "Trabajo de Fin de Máster". El convenio de prácticas curriculares de la primera asignatura estipula que se debe aprovechar un mínimo de 190 horas, mientras que, la segunda, tiene una dedicación mínima de 150 horas, equivalentes a 6 créditos ECTS. Como consecuencia, se deben dedicar un mínimo de 210 horas (máximo entre 210, 190 y 150) a la realización de este proyecto.

De esta forma, estableciendo el inicio del proyecto el día 09 de enero de 2023 y el cierre el 09 de julio, exceptuando la semana festiva del 03 de abril al 09 de abril, la carga horaria que debe realizarse de forma semanal (25 semanas) para ajustarse a los requerimientos temporales es de, aproximadamente, 9 horas. Se parte de este hecho para realizar la planificación temporal del proyecto, que se puede observar en el diagrama de Gantt de la Figura 3.2.

- Comprensión del negocio (09 ene - 29 ene).
- Comprensión de los datos (30 ene - 26 feb).
- Preparación de los datos (27 feb - 26 mar).
- Modelado (27 mar - 28 may).
- Evaluación (29 may - 02 jul).
- Despliegue (09 ene - 09 jul).

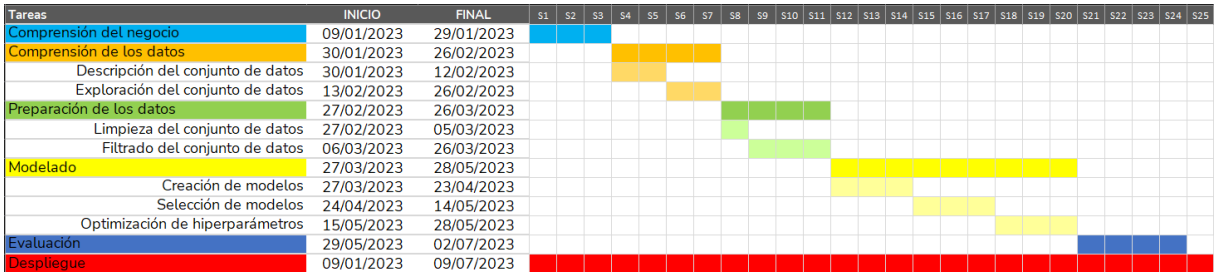

Figura 3.2: Diagrama de Gantt de la planificación temporal.

Como se observa, la planificación temporal de este trabajo se realiza semana a semana; los objetivos del proyecto se abordan por semanas en forma de subobjetivos y se va construyendo el objetivo global de forma incremental. En el diagrama de Gantt, cada fase se divide en tareas y se utiliza un color distinto para cada fase.

Cabe mencionar que en la fase de despliegue sólo se lleva a cabo la creación de una documentación detallada, que se corresponde con la memoria del TFM. Se planifica que los distintos apartados de la memoria se redacten parcialmente en la semana correspondiente al desarrollo de su contenido; ésta es la razón por la que se planifica la fase de despliegue desde el primer día hasta el último.

3.2.2. Presupuesto

Tras realizar la planificación temporal, es necesario estimar los costes del proyecto. Realizar un presupuesto inicial del proyecto que se corresponda realmente con los costes reales al final de éste resulta una tarea especialmente complicada al tratarse de un proyecto de investigación. Esto es debido, principalmente, a que no se saben exactamente las herramientas que se van a necesitar, o los contratiempos que pueden afectar directamente a las estimaciones iniciales (nuevas ideas surgidas tras el descubrimiento de algún dato, bloqueos por la inexistencia de documentación o por la ausencia de experiencia previa con la tarea, etc.). Gran parte del contenido de esta subsección se basa en el trabajo [5].

A continuación, se presenta el presupuesto desglosado en tres categorías (hardware, software y Recursos Humanos (RRHH)); se corresponden con la estimación inicial realizada al comienzo del proyecto. Cabe destacar que, para calcular el coste real asociado al uso del hardware y del software, se deben usar unas fórmulas que tienen en cuenta la frecuencia de uso, el tiempo utilizado, y el coste mensual de estas herramientas. Éstas se formulan a continuación junto con las unidades de cada magnitud entre paréntesis:

$$\text{Coste por mes}\left(\frac{€}{\text{mes}}\right) = \frac{\text{Coste total }(€)}{\text{Vida útil (meses)}} \tag{3.1}$$

$$\text{Coste real por mes}\left(\frac{€}{\text{mes}}\right) = \text{Coste por mes}\left(\frac{€}{\text{mes}}\right) \times \text{Porcentaje de uso} \tag{3.2}$$

$$\text{Coste real } (€) = \text{Coste real por mes} \left(\frac{€}{\text{mes}} \right) \times \text{Tiempo de uso (meses)} \tag{3.3}$$

Hardware

El desarrollo del proyecto se lleva a cabo con tres ordenadores distintos:

- Un ordenador personal de gama media-alta cuyas especificaciones son las siguientes: procesador Intel(*Registered Trademark* (R)) Core(*Trademark* (TM)) i7-1065G7 Unidad de Procesamiento Central o *Central Processing Unit* (CPU) @ 1.30GHz (1498 Mhz), 4 CPUs (*cores* o procesadores principales), sistema operativo Windows 11 Home de 64 bits, 16 GB de RAM, 256 GB (237 GB) de memoria Unidad de Estado Sólido o *Solid State Drive* (SSD), y 1 (*Terabyte* (TB)) (931 GB) de memoria Unidad de Disco Duro o *Hard Disk Drive* (HDD).
- Un ordenador del GSI cuyas especificaciones son las siguientes: procesador Intel(R) Core(TM) i7-2700K CPU @ 3.50GHz (3501 Mhz), 4 CPUs (*cores* o procesadores principales), sistema operativo Windows 10 Pro de 64 bits, 16 GB de RAM, 465 GB (512 GB) de memoria SSD, y 1 (TB) (931 GB) de memoria HDD.
- Una máquina virtual del GSI cuyas especificaciones son las siguientes: procesador Common KVM processor (2900 Mhz), 2 CPUs (*cores* o procesadores principales), sistema operativo Windows 10 Pro de 64 bits, 8 GB de RAM, 40 GB de memoria HDD.

El coste del ordenador personal es de 899 €, con una vida útil de unos 5 años. Dado que las características del segundo ordenador son similares, su precio se estima también en 899 €, con la misma vida útil que el anterior. Por último, por desconocimiento del coste de una máquina virtual, se estima igual que los anteriores.

Además, para conseguir una correcta comunicación con el resto del equipo, poder consultar documentación, emplear herramientas *online*, etc., es necesario disponer de una buena conexión a Internet. Por ello, este gasto también se incluye en el presupuesto. Cabe mencionar que el porcentaje de uso tan bajo de este recurso se debe a que se comparte wifi con otros dos estudiantes, por lo que se parte de un 33,3 % de uso total al mes por parte del trabajador.

El resto de los gastos corrientes, como la electricidad para alimentar los ordenadores, no se consideran para el cálculo debido al desconocimiento de su porcentaje de uso, ya que no se puede discernir del consumo no atribuible al proyecto y, por tanto, no es medible en el ámbito de éste.

Por todo ello, el coste total asociado al hardware es de 262,507 € y se desglosa según las especificaciones de la Tabla 3.1.

Herramienta	Coste total (€)	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
Ordenador personal	899	$5 \cdot 12 = 60$	80	7	83,907
Ordenador GSI	899	$5 \cdot 12 = 60$	100	2	29,967
Máquina virtual	899	$5 \cdot 12 = 60$	100	7	104,883
Conexión a Internet	25 (al mes)	-	25	7	43,75
				Total:	262,507

Tabla 3.1: Presupuesto hardware.

Software

El coste total asociado al software es de 0€ ya que todas las herramientas utilizadas tienen licencia gratuita para estudiantes como se puede observar en la Tabla 3.2.

Herramienta	Coste por mes (€)	Uso (%)	Uso (meses)	Coste real (€)
Microsoft Teams	0	80	7	0
Trello	0	100	7	0
Overleaf	0	40	7	0
Mendeley	0	60	7	0
VSCode	0	90	7	0
			Total:	0

Tabla 3.2: Presupuesto software.

Recursos humanos

A pesar de que el trabajo es realizado por una única persona, ésta toma diferentes roles a lo largo del proyecto (analista de sistema, *data scientist* y desarrollador Python).

Por esta razón, su sueldo bruto es la suma del sueldo bruto obtenido por cada rol adoptado, es decir, 5684,1€, como indica el total de la Tabla 3.3. Todos los sueldos de la Tabla 3.3 se han hallado teniendo en cuenta los datos proporcionados por el buscador de salario bruto Glassdoor [20], todos excepto el de analista de sistema, para el cual se ha utilizado el buscador de LinkedIn [32]. En todos los casos se ha considerado un trabajador en Madrid, ya que es un lugar muy genérico y que recoge unos salarios estándar a nivel europeo. También se ha tenido en cuenta que un trabajador medio realiza un total de aproximadamente 1800 horas al año.

Rol	Sueldo (€/hora)	Tiempo (horas)	Total (€)
Analista de sistema (20 %)	18,3	42	768,6
Data scientist (50 %)	20,44	105	2146,2
Desarrollador Python (30 %)	16,64	63	1048,32
		Total:	3963,12

Tabla 3.3: Sueldos (planificación).

Además, dado que es necesario dar de alta en la Seguridad Social a esta persona, se debe tener en cuenta este coste adicional, que es de $0{,}309 \cdot 3963{,}12\,€ = 1224{,}6\,€$, ya que se corresponde con el 30,9 % del sueldo bruto de la persona, al tratarse de un contrato indefinido [53]. Por todo ello, el coste total asociado al personal es de 5187,72 € y se desglosa según lo explicado en la Ecuación 3.4.

$$\text{Presupuesto RRHH} = 3963{,}12\,€\ \text{(Sueldo)} + 1224{,}6\,€\ \text{(Cotización)} = 5187{,}72\,€ \quad (3.4)$$

A partir de este estudio se prevé que el coste total del proyecto es de 5450,227 €, de los cuales 262,507 € derivan del uso de recursos hardware, y 5187,72 € de costes de personal, como se indica en la Tabla 3.4.

Hardware (€)	Software (€)	RRHH (€)	Total (€)
262,507	0	5187,72	5450,227

Tabla 3.4: Presupuesto.

3.3. Balance

En esta sección, se lleva a cabo un balance del proyecto, esto es, un análisis de la diferencia temporal y económica entre la realidad y la planificación. Puesto que se trata de una comparación con lo planificado, el balance temporal se estudia tras el final de cada semana mientras que el económico se hace de manera global al final del proyecto.

3.3.1. Balance temporal

El balance temporal final se observa en la Figura 3.3. Se añaden 6 semanas a las 25 inicialmente consideradas. El día inicial de la semana 26 es el 10 de julio y el día final de la semana 31 es el 10 de septiembre, exceptuando las semanas del 31 de julio al 13 de agosto.

- Comprensión del negocio (09 ene - 05 mar).

- Comprensión de los datos (30 ene - 05 mar).
- Preparación de los datos (06 mar - 14 may).
- Modelado (15 may - 27 ago).
- Evaluación (28 ago - 03 sep).
- Despliegue (09 ene - 10 sep).

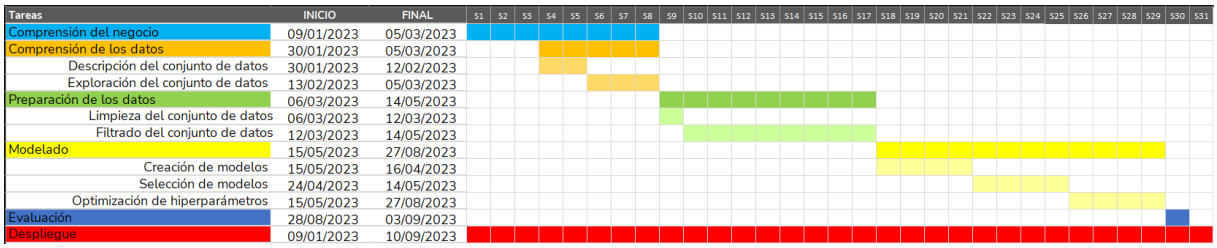

Figura 3.3: Diagrama de Gantt del balance temporal.

3.3.2. Balance económico

Los costes iniciales de software no sufren modificaciones debido a que se utilizan herramientas de software libre. Sin embargo, el número de horas añadidas debido a los retrasos en la entrega sí modifica el presupuesto hardware y los costes de personal, ya que supone otros dos mes de trabajo.

La tabla de presupuesto hardware (Tabla 3.1) se ve modificada como se observa en la Tabla 3.5. Por tanto, el coste total real asociado al hardware es de 358,913 € y se desglosa según las especificaciones de la Tabla 3.5. Así que la diferencia entre la planificación y la realidad es de 358,913 € − 262,507 € = 96,406 €.

Herramienta	Coste total (€)	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
Ordenador personal	899	$5 \cdot 12 = 60$	80	9	107,88
Ordenador GSI	899	$5 \cdot 12 = 60$	100	4	59,933
Máquina virtual	899	$5 \cdot 12 = 60$	100	9	134,85

continúa en la página siguiente

continúa desde la página anterior

Herramienta	Coste total (€)	Vida útil (meses)	Uso (%)	Uso (meses)	Coste (€)
Conexión a Internet	25 (al mes)	-	25	9	56,25
				Total:	358,913

Tabla 3.5: Balance económico hardware.

Para cuantificar de una forma más estricta la diferencia de coste que se produce en cuanto al presupuesto de RRHH, se tiene en cuenta el número exacto de horas que trabaja el estudiante, contabilizado mediante el artefacto "Cuaderno de trabajo". En total, el estudiante trabaja 397 horas. La tabla de presupuesto RRHH (Tabla 3.3) se modifica como se observa en la Tabla 3.6.

Por esta razón, su sueldo bruto es la suma del sueldo bruto obtenido por cada rol adoptado, es decir, 7492,184 €, como indica el total de la Tabla 3.6.

Rol	Sueldo (€/hora)	Tiempo (horas)	Total (€)
Analista de sistema (20 %)	18,3	79,4	1453,02
Data scientist (50 %)	20,44	198,5	4057,34
Desarrollador Python (30 %)	16,64	119,1	1981,824
		Total:	7492,184

Tabla 3.6: Sueldos (balance).

Además, dado que es necesario dar de alta en la Seguridad Social a esta persona, se debe tener en cuenta este coste adicional, que será de $0{,}309 \cdot 7492{,}184\,€ = 2315{,}085\,€$. Por todo ello, el coste total real asociado al personal es de 9807,269 € y se desglosa según lo explicado en la Ecuación 3.5.

$$\text{Coste real (RRHH)} = 7492{,}184\,€\ \text{(Sueldo)} + 2315{,}085\,€\ \text{(Cotización)} = 9807{,}269\,€ \quad (3.5)$$

Así que la diferencia entre la planificación y la realidad es de $9807{,}269\,€ - 5187{,}72\,€ = 4619{,}549\,€$.

A partir de este estudio se prevé que el coste total real del proyecto es de 10166,182 €, de los cuales 358,913 € derivan del uso de recursos hardware, y 9807,269 € de costes de personal, como se indica en la Tabla 3.7.

Hardware (€)	Software (€)	RRHH (€)	Total (€)
358,913	0	9807,269	10166,182

Tabla 3.7: Balance económico.

Así que, globalmente, la diferencia entre la planificación y la realidad es de 10166,182 € − 5450,227 € = 96,406 € + 4619,549 € = 4715,955 €.

4: Técnicas y herramientas

En este capítulo se listan las técnicas (Sección 4.1), las tecnologías (Sección 4.2) y las herramientas (Sección 4.3) utilizadas para el desarrollo del proyecto. Este desarrollo engloba la implementación de los modelos, la redacción del informe y la coordinación de la alumna con los tutores.

4.1. Técnicas

En esta sección se explica cada una de las técnicas, estadísticas y de la Inteligencia Artificial o *Artificial Intelligence* (IA), respectivamente, utilizadas en el desarrollo de los nuevos candidatos a sensores software del Brix a la salida del proceso de evaporación. También se repasan las funciones de pérdida usadas para actualizar los pesos de la red y las métricas utilizadas para calcular el error.

4.1.1. Modelo ARIMA

El modelo Media Móvil Integrada Autorregresiva o *AutoRegressive Integrated Moving Average* (ARIMA) [6] es un conocido modelo estadístico para series temporales que utiliza datos anteriores más un error para pronosticar valores futuros. Más específicamente, combina un modelo autorregresivo general, AutoRregresiva o *AutoRegressive* (AR)(p), usando valores previos de la variable dependiente para hacer predicciones y un modelo de promedio móvil general, Media Móvil o *Moving Average* (MA)(q), utilizando la media de la serie y los errores anteriores para hacer predicciones. Por tanto, un modelo ARIMA se clasifica según tres parámetros, a saber: p o número de términos autorregresivos, d o diferencias no estacionales que se necesitan para lograr la estacionariedad y q o número de errores de pronóstico retrasados en la ecuación de predicción.

En general, al utilizar un modelo ARIMA, no es necesario escalar los datos antes de hacer predicciones, ya que se basa en la estructura temporal de los datos y no en sus magnitudes absolutas. Al construir un modelo ARIMA, hay que asegurarse de que los datos cumplen con los supuestos de estacionariedad, es decir, que la media y la varianza

de los datos son constantes a lo largo del tiempo. Si los datos no son estacionarios, es necesario aplicar transformaciones como diferenciación o logaritmo.

Para comprobar si una serie de tiempo es estacionaria se realiza la prueba Dickey-Fuller Aumentada o *Augmented Dickey-Fuller* (ADF). Este *test* tiene una hipótesis nula de no estacionariedad, por lo que si el p-valor es menor que un umbral, se puede rechazar la hipótesis nula y concluir que la serie es estacionaria [61].

4.1.2. *Machine Learning*

El ML se centra en construir sistemas computacionales que a partir de datos y percepciones mejoran su rendimiento en la realización de una determinada tarea (sin haber sido explícitamente programados para dicha tarea) [39].

Dado que la variable objetivo, el Brix, toma valores númericos reales (continuos), se trata de un problema de regresión. La regresión se puede abordar creando un único modelo (simple o avanzado) o un *ensemble* de modelos. Los primeros, se consideran base porque el modelo de regresión está compuesto por un sólo método. Los segundos, están compuestos por conjuntos (*ensembles*) de regresores básicos; cuando se crea un *ensemble*, cada regresor es un método base. Su funcionamiento consiste en combinar los resultados de los regresores base (mediante *bagging* o *boosting*) para, mediante un mecanismo de agregación/selección, predecir el valor de la variable objetivo. Se basan en la idea intuitiva de que la toma de decisiones mejora cuando se contrastan opiniones diversas. Generalmente, aumentan de manera importante la precisión; sin embargo, son modelos difíciles de comprender y analizar. En esta subsección se presentan, en primer lugar, los principales métodos de regresión, seguidos por los *ensembles*.

Existen distintas estrategias para decidir los tipos de regresores base y las posibles formas de combinarlos. Los *ensembles* pueden ser homogéneos o heterogéneos según utilicen un único regresor base o varios. Existen dos estrategias de construcción: dependientes de la salida de otros regresores o independientes. Además, las instancias se pueden estimar en paralelo (todos los regresores estiman a la vez; por ejemplo, *bagging*) o secuencial (se aplican uno tras otro; por ejemplo, *boosting*) [45].

Support Vector Regression

SVR es la combinación de Máquina de Vectores de Soporte o *Support Vector Machine* (SVM) y regresión lineal. Por un lado, una SVM no funciona con los datos del Brix porque no se trata de un problema de clasificación en dos clases; no se pretende predecir una clase, sino el siguiente valor en una serie temporal. Por otro lado, una regresión lineal tampoco funciona con los datos del Brix porque presentan muchas fluctuaciones y una línea de mejor ajuste daría malas predicciones sobre los datos.

Con SVM se procura dibujar un hiperplano entre dos clases diferentes. Usando regresión lineal se intenta minimizar la función de coste usando el descenso por el gradiente (*gradient descent*). Entonces SVR, al ser la combinación de los dos, trata de minimizar el error

dentro de un cierto umbral. Una de las ventajas de SVR frente a los anteriores es que se puede aplicar para predecir valores dentro de un umbral no lineal.

SVR consiste entonces en predecir valores con precisión dentro de un cierto umbral. Por ello, se definen líneas de límite para formar un margen como $-\epsilon$ y $+\epsilon$, donde ϵ es la distancia desde el hiperplano hasta cada línea límite. El objetivo es minimizar el error y maximizar la distancia al margen [14].

Extreme Gradient Boosted Decision Tree

Para comprender el Árbol de Decisión Potenciado por Gradiente Extremo o *eXtreme Gradient Boosted Decision Tree* (XGBDT), es vital entender primero los conceptos y algoritmos de ML en los que se basa: Árbol de Decision o *Decision Tree* (DT) y *gradient boosting*.

En el aprendizaje supervisado se dispone de datos de ejemplos y sus resultados (datos etiquetados). Su objetivo es predecir la respuesta correcta para los nuevos ejemplos desconocidos [9]. Un DT es un algoritmo de ML supervisado no paramétrico, que se utiliza tanto para tareas de clasificación como de regresión. Tiene una estructura jerárquica de árbol que consiste en un nodo raíz, ramas, nodos internos (también llamados nodos de decisión) y nodos hoja (también llamados nodos terminales). Los DTs crean un modelo que predice mediante la evaluación de un árbol de preguntas, y la estimación de la cantidad mínima de preguntas necesarias para evaluar la probabilidad de tomar una decisión correcta [26]. En la Figura 4.4, se usa un DT para estimar si jugar al tenis en una pista descubierta (la etiqueta) en función del tiempo atmosférico (las características).

Tanto el RF como el Árbol de Decisión Potenciado por Gradiente o *Gradient Boosted Decision Tree* (GBDT) son *ensembles*, es decir, constituyen un modelo que consta de múltiples árboles de decisión. La diferencia está en cómo se construyen y combinan los árboles. El RF construye árboles de decisión completos en paralelo (*bagging*) a partir de muestras aleatorias de arranque del conjunto de datos. La predicción final es un promedio de todas las predicciones. Los GBDT entrenan iterativamente un conjunto de árboles de decisión poco profundos, y cada iteración usa los residuos de error del modelo anterior para ajustar el siguiente modelo (*boosting*). La predicción final es una suma ponderada de todas las predicciones. El *bagging* del RF minimiza la varianza y el sobreajuste, mientras que el *boosting* de GBDT minimiza el sesgo y el subajuste [21].

El *gradient boosting* es una extensión del *boosting* donde el proceso de generación aditiva de modelos débiles se formaliza como un algoritmo de descenso por el gradiente sobre una función objetivo. El *gradient boosting* establece resultados específicos para el próximo modelo en un esfuerzo por minimizar los errores. Los resultados previstos para cada caso se basan en el gradiente del error con respecto a la predicción.

El XGBDT es una implementación escalable y muy precisa de *gradient boosting* que se crea en gran medida para potenciar la velocidad computacional. Con este modelo, los árboles se construyen en paralelo, en lugar de secuencialmente como con GBDT. Sigue

una estrategia por niveles, escaneando valores de gradiente y usando estas sumas parciales para evaluar la calidad de las divisiones en el conjunto de entrenamiento [29].

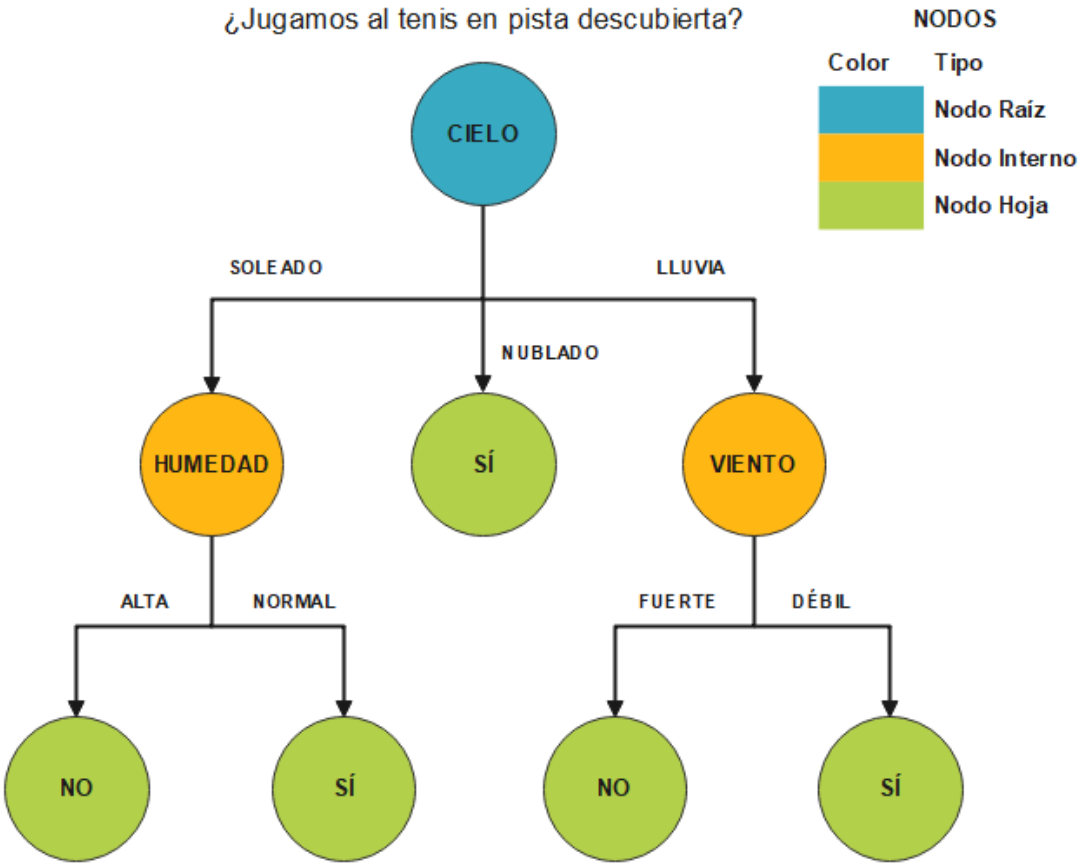

Figura 4.4: Árbol de decisión [Fuente: elaboración propia].

4.1.3. *Neural Networks*

Las NNs, también conocidas como Red Neuronal Artificial o *Artificial Neural Network* (ANN)s, son un subconjunto de ML y están en el núcleo de los algoritmos de DL. Su nombre y estructura se inspiran en el cerebro humano, e imitan la forma en la que las neuronas biológicas se señalan entre sí. Las redes neuronales son un modelo matemático en el que se basan potentes algoritmos de ML.

Desde un punto de vista más arquitectónico, una ANN es un conjunto de neuronas estructuradas en capas interconectadas [1]. Cada unidad o nodo (neurona artificial), se conecta a otras unidades a través de arcos dirigidos. Cada arco (i, j) sirve para propagar la salida de la unidad i (notada a_i), que sirve como una de las entradas para la unidad j. Cabe mencionar que las entradas y salidas son números. Cada arco (i, j) tiene asociado un peso numérico $w_{i,j}$ que determina la fuerza y el signo de la conexión. Cada unidad calcula

[1]Sólo están totalmente interconectadas en el percertrón multicapa y otras arquitecturas.

su salida en función de las entradas que recibe y la salida de cada unidad sirve, a su vez, como una de las entradas de otras neuronas. La red recibe una serie de entradas externas, llamadas unidades de entrada, y devuelve al exterior la salida de algunas de sus neuronas, llamadas unidades de salida; en este sentido se comporta como una función matemática.

La salida de cada unidad se calcula como indica la Ecuación 4.6.

$$a_j = g(\sum_{i=0}^{n} w_{i,j} \cdot a_i) \tag{4.6}$$

En la Ecuación 4.6, g es una función de activación. La función de activación g introduce cierta componente no lineal y aumenta la expresividad del modelo. Cabe mencionar que el sumatorio se hace sobre todas las unidades i que envían su salida a la unidad j, excepto para $i = 0$, que se considera una entrada ficticia, $a_0 = 1$, y un peso $w_{0,j}$ denominado *umbral* o *bias*. Intuitivamente, el umbral $w_{0,j}$ de cada unidad se interpreta como el opuesto de una cantidad cuya entrada debe superar.

En una red neuronal, se distinguen 3 tipos de capas: capa de entrada, capas ocultas (intermedias) y capa de salida. La capa de entrada recoge simplemente la entrada. Para unidades de la misma capa, la función de activación usada es la misma; entre capas distintas puede variar. La función de activación de la capa de salida varía en función del uso que se le quiera dar a la red [51]. Toda esta arquitectura puede observarse en el ejemplo de la Figura 4.5, donde las circunferencias representan cada unidad o nodo (neurona) y se colorean del mismo color las neuronas que pertenecen a la misma capa [27].

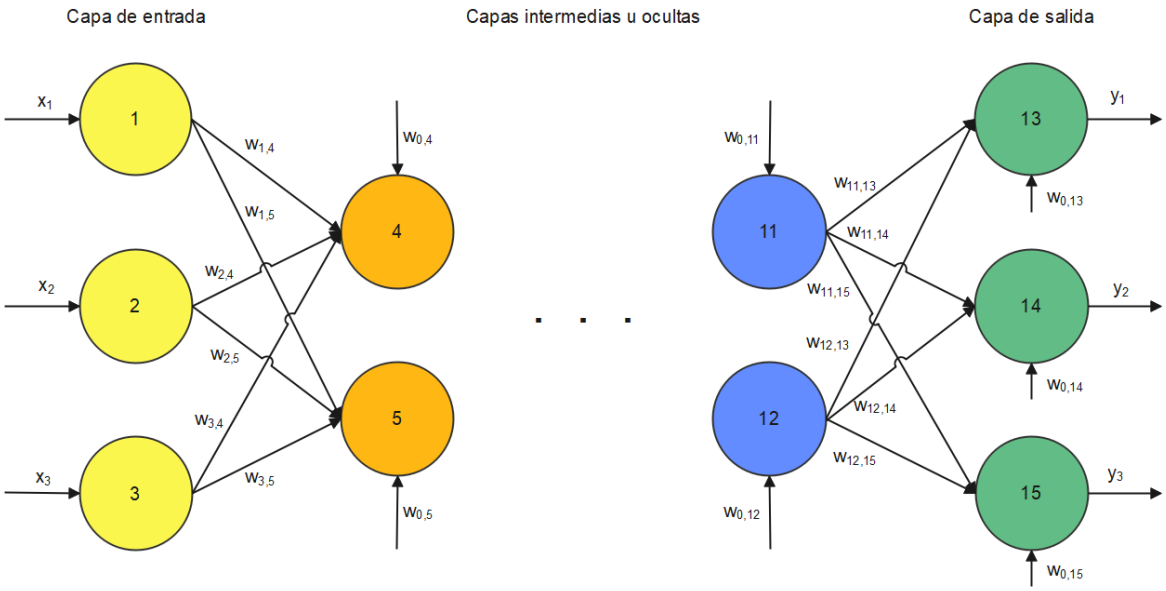

Figura 4.5: Red neuronal [Fuente: elaboración propia].

4.1.4. *Deep Learning*

En la actualidad, dentro de la IA destaca la disciplina del DL, una rama del ML que es extremadamente efectiva en el aprendizaje de patrones. Trabaja con algoritmos que extraen el conocimiento significativo de los datos mediante una jerarquía de múltiples capas que tratan de imitar las redes neuronales del cerebro. Cada capa transforma los datos de entrada en representaciones más abstractas que se van combinando según se profundiza en la red [8].

Multilayer Perceptron

El perceptrón es la red neuronal más simple, compuesta por tan sólo una capa que aúna la de entrada y la de salida. El perceptrón con una sola neurona no se puede aplicar a datos no lineales. El Perceptrón Multicapa o *Multilayer Perceptron* (MLP) se desarrolló para hacer frente a esta limitación. Es una red neuronal donde el mapeo entre entradas y salidas no es lineal. Un MLP tiene capas de entrada y salida, y una o más capas ocultas con uno o más neuronas. Y, mientras que en el perceptrón la neurona debe tener una función de activación que imponga un umbral, como Unidad Lineal Rectificada o *Rectified Linear Unit* (ReLU) o *sigmoide*, las neuronas en un MLP pueden usar cualquier función de activación arbitraria [7].

Recurrent Neural Networks

A diferencia de las redes neuronales normales, las recurrentes poseen bucles o realimentación entre entradas y salidas, lo que permite que la información persista en el tiempo. Uno de los atractivos de las RNNs es la idea de conectar la información previa con la tarea actual. Desafortunadamente, es posible que la brecha entre la información relevante y el punto donde se necesita se vuelva muy grande. En teoría, son absolutamente capaces de manejar tales "dependencias a largo plazo"; lamentablemente, en la práctica, no parecen ser capaces de aprenderlas. Ésto es debido al problema del gradiente: el valor del gradiente influye en la actualización de los pesos, y ese gradiente depende también de la activación de los elementos de proceso que, a su vez, depende de los pesos. Si el peso de la conexión de realimentación es menor que 1, los valores del estado se olvidan; se conoce como problema del gradiente que desaparece (*vanishing gradient*). Si el peso de la conexión de realimentación es mayor que 1, los valores del estado "explotan"; se conoce como problema de degeneración o explosión del gradiente (*exploding gradient*). La solución es usar valores próximos a 1 [41]. Afortunadamente, las redes LSTM no tienen este problema [43].

Una celda LSTM es un conjunto de neuronas. La clave de las LSTM es el estado de la celda, la línea horizontal que atraviesa la parte superior del diagrama de la Figura 4.6 y que se asemeja a una cinta transportadora. La LSTM tiene la capacidad de eliminar o agregar información al estado de la celda con sólo algunas interacciones lineales menores, cuidadosamente regulado por estructuras llamadas puertas. Las puertas son una forma de dejar pasar información opcionalmente. Están compuestas por una capa de red neuronal

sigmoide y una operación de multiplicación puntual. La capa *sigmoide* genera números entre 0 y 1, que describen la cantidad de cada componente que se debe dejar pasar. Un valor de 0 significa "no dejar pasar nada", mientras que un valor de 1 significa "dejar pasar todo".

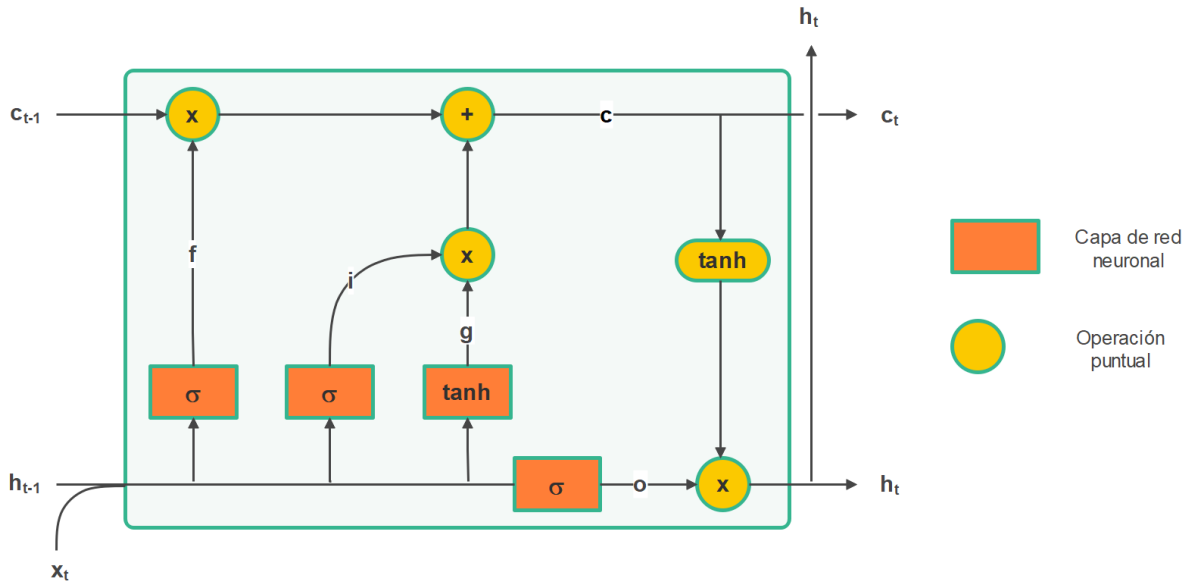

Figura 4.6: Estructura interna de una celda LSTM [Fuente: elaboración propia].

En la Figura 4.6, un fragmento de celda LSTM analiza la entrada en el instante t, x_t, y genera un valor en el instante t, h_t, siendo el estado de la celda en el instante t, c_t. En las siguientes fórmulas $\mathbf{W}$ denota la matriz de pesos y b el umbral o bias.

Una LSTM tiene tres de estas puertas para proteger y controlar el estado de la celda.

- *Forget gate* o puerta de olvido. Se trata de una capa *sigmoide* que decide qué información se va a desechar del estado de la celda. Para ello, toma como entradas h_{t-1} y x_t, y genera un número entre 0 y 1 para cada número en el estado de la celda c_{t-1}.

$$f_t = \sigma(\mathbf{W}_f \cdot [h_{t-1}, x_t] + b_f)$$

- *Input gate* o puerta de entrada. Se trata de una capa *sigmoide* que decide qué valores se actualizan.

$$i_t = \sigma(\mathbf{W}_i \cdot [h_{t-1}, x_t] + b_i)$$

  A continuación, una capa *tanh* crea un vector de nuevos valores candidatos, c'_t, que podría agregarse al estado.

$$c'_t = tanh(\mathbf{W}_c \cdot [h_{t-1}, x_t] + b_c)$$

  La combinación de estas dos partes crea una actualización del estado de la celda anterior, c_{t-1}, en el nuevo estado celular c_t y sirve para decidir qué nueva información se va a almacenar en el estado de la celda.

Se multiplica el estado anterior por f_t, olvidando lo que la puerta de olvido decidió. Luego, se agrega $i_t \cdot c'_t$, que son los nuevos valores candidatos, escalados por cuánto se actualiza cada valor de estado.

$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$$

- *Output gate* o puerta de salida. Se trata de una capa *sigmoide* que decide qué partes del estado de la celda se van a generar.

  $$o_t = \sigma(\mathbf{W}_o \cdot [h_{t-1}, x_t] + b_o)$$

  A continuación, el estado de la celda se pasa a través de una *tanh* para que tome valores entre -1 y 1. Estos valores se multiplican por la salida de la puerta *sigmoide*, de modo que sólo se emiten las partes que se decide.

  $$h_t = o_t \cdot tanh(c_t)$$

  Esta salida se basa en el estado de la celda, pero es una versión filtrada.

4.1.5. Modelo *encoder-decoder*

El llamado *encoder-decoder* es la arquitectura más comúnmente utilizada para problemas *Seq2Seq*, es decir, problemas que reciben una secuencia de datos pasado y pretenden predecir una secuencia de datos a futuro. La arquitectura *encoder-decoder* fue específicamente diseñada para predecir salidas de longitud arbitraria a partir de entradas de longitud también indefinida en el contexto de la traducción automática, pero se adaptó rápidamente a la predicción de series temporales de longitud flexible.

Este modelo consiste en dos redes: el *encoder* o codificador y el *decoder* o decodificador. El codificador toma la secuencia de entrada y la transforma en una codificación comprimida. El decodificador se inicializa con ese estado interno final comprimido del *encoder* y una serie de variables exógenas y obtiene la secuencia de salida. Esta estructura se observa en la Figura 4.7.

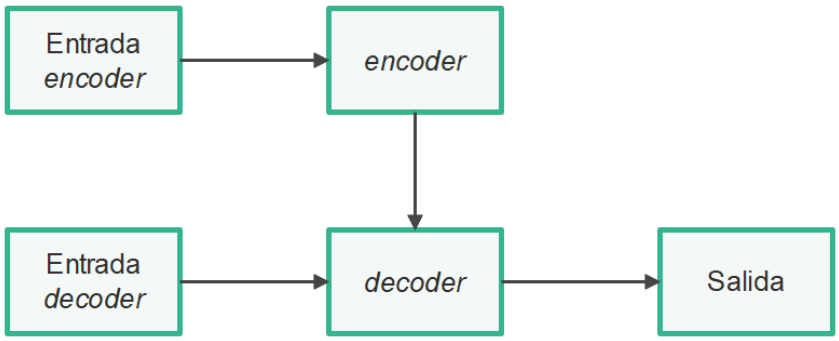

Figura 4.7: Modelo *encoder-decoder* [Fuente: elaboración propia].

4.1.6. Técnica de la ventana deslizante o *sliding window*

En la técnica de la ventana deslizante (*sliding window*), una ventana de tamaño fijo se mueve a lo largo del conjunto de datos para predecir los siguientes (flecha verde en la Figura 4.8 y en la Figura 4.9). Por cada paso de entrenamiento, la red recibe una entrada de tamaño fijo de datos históricos (pasados) y genera una salida de tamaño fijo de datos estimados. En el siguiente paso de entrenamiento, la ventana se mueve una unidad de tiempo hacia adelante (flecha negra en la Figura 4.8 y en la Figura 4.9) tanto en la entrada como en la salida y así sucesivamente. Los modelos predictivos usados en este proyecto se entrenan usando esta técnica.

En la Figura 4.8 se representan tres pasos del proceso de entrenamiento de un modelo *Many2One* con ventana deslizante de 20.

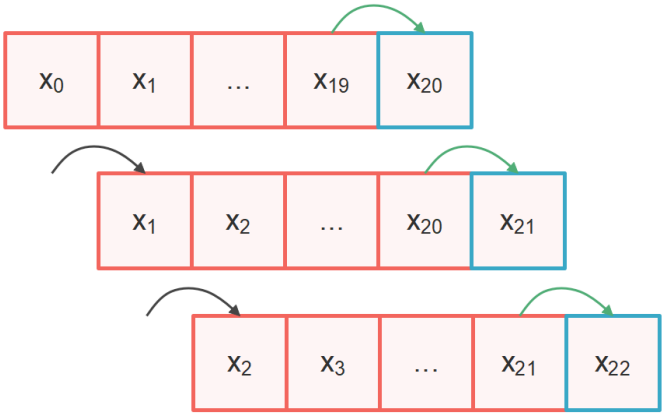

Figura 4.8: *Many2One* [Fuente: elaboración propia].

En la Figura 4.9 se representan tres pasos del proceso de entrenamiento de un modelo *Many2Many* con ventana deslizante de 20.

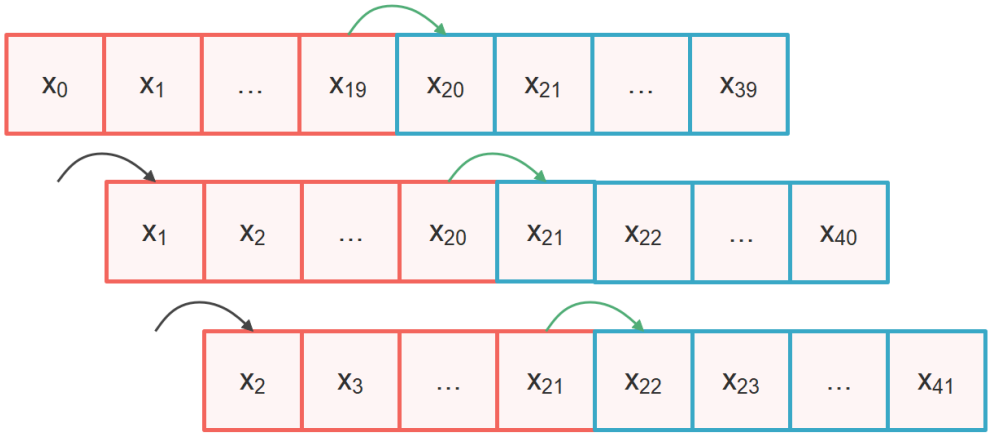

Figura 4.9: *Many2Many* [Fuente: elaboración propia].

4.1.7. Funciones de pérdida

Los algoritmos de ML se basan en minimizar una función, denominada función objetiva u objetivo. Una función de pérdida es una medida de la capacidad de predicción de un modelo de ML. Uno de los métodos más utilizados para encontrar el punto mínimo de la función es el descenso por el gradiente.

No hay una sola función de pérdida que funcione para todo tipo de datos. Depende de una serie de factores que incluyen la presencia de valores atípicos (*outliers*), la elección del algoritmo de ML, la eficiencia temporal del descenso por el gradiente, la facilidad para encontrar las derivadas y la confianza de las predicciones [24].

Las funciones de pérdida pueden ser de clasificación (predicen una etiqueta) y de regresión (predicen una cantidad). Dado que el problema objeto de estudio es de regresión, se utilizan 5 de estas funciones de pérdida, a saber: Error Absoluto Medio o *Mean Absolute Error* (MAE) (Subsección 4.1.7), MSE (Subsección 4.1.7), Raíz Cuadrada del Error Cuadrático Medio o *Root Mean Squared Error* (RMSE) (Subsección 4.1.7), Huber (Subsección 4.1.7), log-cosh (Subsección 4.1.7) y cuantil (Subsección 4.1.7), cada una de las cuales se explica en un epígrafe diferente. Existen funciones integradas en la librería de Python `sklearn` para MAE y MSE; para el resto, es necesario escribir funciones propias.

MAE

El MAE, también conocido como pérdida L1, es la suma de las diferencias absolutas entre el valor real u observado, y_i, y el estimado o predicho, $\hat{y}_i$. Para que un error con valor positivo no cancele a un error con valor negativo se usa el valor absoluto de la diferencia. Como lo que interesa es conocer el comportamiento del error de todas las observaciones, en este caso n, se hace el promedio de los valores absolutos de la diferencia. Por lo tanto, la métrica MAE mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar sus direcciones. Toma valores entre 0 e ∞, es decir, su rango es $[0, \infty)$. Su fórmula se observa en la Ecuación 4.7.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{4.7}$$

MSE

El MSE, también conocido como pérdida L2, es la suma de las distancias al cuadrado entre el valor real u observado, y_i, y el estimado o predicho, $\hat{y}_i$. Por lo tanto, mide la magnitud promedio de los errores en un conjunto de n predicciones. Toma valores entre 0 e ∞, es decir, su rango es $[0, \infty)$. Su fórmula se observa en la Ecuación 4.8.

$$\text{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{4.8}$$

Usar el error al cuadrado es más fácil de resolver, pero usar el error absoluto es más robusto frente a los valores atípicos. Dado que MSE eleva al cuadrado el error, el valor

del error aumenta mucho si es mayor que 1. Si los datos presentan un *outlier*, el error es alto y su cuadrado es mucho mayor que su valor absoluto. Esto hace que el modelo con pérdida MSE dé más peso a los valores atípicos que un modelo con pérdida MAE. Por tanto, MAE se considera más robusta para los valores atípicos que MSE.

Sin embargo, MAE presenta un gran problema, especialmente para las redes neuronales, y es que su gradiente es el mismo en todas partes [24]. El gradiente es grande incluso para valores de pérdida pequeños, lo que entorpece el aprendizaje. MSE se comporta muy bien en este caso y converge incluso con una tasa de aprendizaje fija.

RMSE

La raíz cuadrada del error cuadrático medio (RMSE) es la raíz cuadrada de MSE y está en la misma escala que MAE, es decir, está en las unidades originales de los datos. Su fórmula se observa en la Ecuación 4.9.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}} \tag{4.9}$$

Al igual que MSE, RMSE penaliza los errores grandes en la predicción. MSE/RMSE es una función diferenciable, lo cual facilita ciertas operaciones matemáticas en comparación a MAE, que no es diferenciable.

Huber

El error absoluto medio suave, también conocido como pérdida de Huber, es menos sensible a los valores atípicos en los datos que las dos anteriores. También es diferenciable en 0. Es básicamente un error absoluto (MAE) que se vuelve cuadrático (MSE) cuando el error es pequeño, concretamente menor o igual a un hiperparámetro, δ. Esto es, la pérdida de Huber se aproxima a MSE cuando $\delta \sim 0$ y MAE cuando $\delta \sim \infty$ (números grandes). Su fórmula se observa en la Ecuación 4.10.

$$\text{Huber} = L_\delta(y, x) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{si } |y - \hat{y}| \leq \delta \\ \delta \cdot |y - \hat{y}| - \frac{1}{2} \cdot \delta^2 & \text{si } |y - \hat{y}| > \delta \end{cases} \tag{4.10}$$

La pérdida de Huber soluciona el problema del gradiente constantemente grande estudiado en MAE, ya que se curva alrededor de los mínimos, lo que disminuye el gradiente. Además, es más robusto para los *outliers* que el MSE. Por lo tanto, combina buenas propiedades tanto de MSE como de MAE. Sin embargo, el problema con la pérdida de Huber es que precisa de un proceso iterativo para entrenar el hiperparámetro δ. La elección de δ es crítica porque determina lo que se va a considerar como un *outlier*. Los residuos más grandes que δ se minimizan con MAE (que es menos sensible a los *outliers* grandes), mientras que los residuos más pequeños que δ se minimizan "apropiadamente" con MSE.

Log-cosh

La pérdida de log-cosh es el logaritmo del coseno hiperbólico del error de predicción. Su fórmula se observa en la Ecuación 4.11.

$$\text{log-cosh} = L(y, \hat{y}) = \sum_{i=1}^{n} \log\left(\cosh\left(\hat{y}_i - y_i\right)\right) \tag{4.11}$$

La pérdida de log-cosh es aproximadamente igual para valores pequeños y para grandes, es decir, funciona principalmente como MSE, pero no se ve tan fuertemente afectado por una predicción ocasionalmente incorrecta. Tiene todas las ventajas de la pérdida de Huber y, además, es dos veces diferenciable en todas partes, a diferencia de la pérdida de Huber. Muchas implementaciones de modelos de ML, como XGBoost, usan el método de Newton para encontrar el óptimo, por lo que se necesita la segunda derivada (también llamada *hessiana*). Por tanto, para marcos de ML como XGBoost, las funciones doblemente diferenciables son más favorables.

Cuantil

La pérdida cuantil es, en realidad, una extensión de MAE (cuando el cuantil es percentil 50, es MAE). Su fórmula se observa en la Ecuación 4.12.

La idea es elegir el valor del cuantil θ en función de si se quiere dar más valor a los errores positivos o a los negativos. La función de pérdida intenta dar diferentes penalizaciones a la sobreestimación y subestimación en función del valor del cuantil elegido. Por ejemplo, una función de pérdida cuantil con $\theta = 0{,}25$ da más penalización a la sobreestimación y trata de mantener los valores de predicción un poco por debajo de la mediana. θ es el cuantil requerido y tiene un valor entre 0 y 1.

$$\text{cuantil} = L_\theta(y, \hat{y}) = \sum_{i=y_i<\hat{y}_i} (\theta - 1) \cdot |y_i - \hat{y}_i| + \sum_{i=y_i \geq \hat{y}_i} (\theta) \cdot |y_i - \hat{y}_i| \tag{4.12}$$

Las funciones de pérdida cuantil resultan útiles cuando se prefiere predecir un intervalo en lugar de sólo predicciones puntuales.

4.1.8. Métricas

En el contexto de los problemas de regresión se llama error a la diferencia que existe entre el valor que el modelo predice y el valor real de la observación con la que se hace el *test*. Gran parte del contenido de esta subsección se basa en el trabajo [2].

Existen diferentes métricas para conocer el error, a saber: MAE (Subsección 4.1.8), MSE (Subsección 4.1.8), RMSE (Subsección 4.1.8), R^2 (Subsección 4.1.8), R^2 ajustado (Subsección 4.1.8), Raíz Cuadrada del Error Logarítmico Cuadrático Medio o *Root Mean Squared Logarithmic Error* (RMSLE) (Subsección 4.1.8) y Error Porcentual Absoluto

Medio o *Mean Absolute Percentage Error* (MAPE) (Subsección 4.1.8), cada una de las cuales se explica en un epígrafe diferente. Existen funciones integradas de `sklearn` para MAE, MSE y R^2; para el resto, es necesario escribir funciones propias.

MAE

El error absoluto medio (MAE) se describe en la Subsección 4.1.7. Su fórmula se observa en la Ecuación 4.7.

MSE

El error cuadrático medio (MSE) se describe en la Subsección 4.1.7. Su fórmula se observa en la Ecuación 4.8.

RMSE

La raíz cuadrada del error cuadrático medio (RMSE), se describe en la Subsección 4.1.7. Su fórmula se observa en la Ecuación 4.9.

R^2

R^2 es el coeficiente de determinación e indica cuánta variación tiene la variable dependiente que se puede predecir de la variable independiente, es decir, indica la bondad del ajuste del modelo a las observaciones reales. Todas las variables independientes del modelo contribuyen al valor de R^2. Suele tomar valores en $[0, 1]$, aunque no siempre es así. El mejor valor posible de esta métrica es 1. Su fórmula se observa en la Ecuación 4.13, donde μ_y es la media de los n valores reales y.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \mu_y)} \tag{4.13}$$

Una desventaja de esta métrica es que asume que cada variable ayuda a explicar la variación en la predicción, y esto no siempre es cierto. Cada variable añadida incrementa o deja invariante el valor de R^2, pero nunca lo disminuye. Esto puede causar confusión en la interpretación, ya que parece que el modelo está mejorando cuando no necesariamente es así.

R^2 ajustado

R^2 ajustado penaliza la adición de variables independientes al modelo, compensando así la desventaja de R^2. Su fórmula se observa en la Ecuación 4.14. En ella, N es el número de filas o muestras y M es el número de columnas o variables.

$$R^2\text{ajustado} = 1 - \frac{(1 - R^2) \cdot (N - 1)}{N - M - 1} \tag{4.14}$$

Sucede siempre que si R^2 se incrementa debido a un valor significativo, R^2 ajustado también incrementa. Por otro lado, si no hay un cambio significativo en R^2, entonces R^2 ajustado disminuye.

RMSLE

El RMSLE es el RMSE de las predicciones, $\hat{y}_i$, y los valores reales, y_i, transformados por el logaritmo; mide la proporción entre ambos. Su fórmula se observa en la Ecuación 4.15.

$$\text{RMSLE} = \sqrt{\frac{\sum_{i=1}^{n} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}{n}} \tag{4.15}$$

La necesidad de usar logaritmos se remonta al problema de la sensibilidad del RMSE a los *outliers*, que pueden provocar que el valor del error se incremente mucho. Los logaritmos escalan los *outliers*, evitando tal efecto.

MAPE

El MAPE es una métrica de precisión que se presenta en forma de porcentaje y consiste en obtener el porcentaje de error para cada observación y hallar la media de todos esos valores. Su fórmula se observa en la Ecuación 4.16.

$$\text{MAPE} = \frac{100}{n} \cdot \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{4.16}$$

MAPE es independiente de la escala y fácil de interpretar. Sin embargo, puede producir valores no definidos, ∞ o muy cercanos a 0.

4.2. Tecnologías

Durante el desarrollo de la investigación han sido necesarias una serie de tecnologías, que se diferencian según su propósito. Esto es, se explican por separado las tecnologías dedicadas al desarrollo técnico del proyecto de las destinadas a la redacción del informe asociado al proyecto.

Desarrollo técnico. Por un lado, se utiliza **Python**, un lenguaje de programación interpretado, dinámico y de alto nivel utilizado en múltiples ámbitos, desde la programación de sistemas hasta el desarrollo web y la IA. Es un lenguaje de programación versátil y accesible, con una gran popularidad debido a su simplicidad y su amplia gama de aplicaciones [46].

Pandas es una librería de código abierto muy popular dentro de los desarrolladores de Python y, sobre todo, dentro del ámbito de ciencia de datos (*data science*) y ML,

ya que ofrece unas estructuras muy poderosas y flexibles que facilitan la manipulación y el tratamiento de los conjuntos de datos. Surgió como necesidad de aunar en una única librería todas las funcionalidades que un analista de datos necesitaba en su día a día, como cargar, modelar, analizar, manipular y preparar datos. Las dos estructuras de datos principales dentro del paquete Pandas son: Series, un array unidimensional etiquetado capaz de almacenar cualquier tipo de dato; y DataFrame, una estructura bidimensional con columnas que pueden ser también de cualquier tipo. Estas columnas son, a su vez, Series. Además, con Pandas es muy simple cargar datos desde diferentes tipos de archivos (Valores Separados por Comas o *Comma Separated Values* (CSV), *JavaScript Object Notation* (JSON), Lenguaje de Marcas de Hipertexto o *HyperText Markup Language* (HTML), etc.), así como guardarlos [31].

Para la implementación de las técnicas de ML se utiliza **Scikit-learn** (o Sklearn), una de las bibliotecas de Python más utilizadas para tareas de ML y la más utilizada para el ML clásico. Se podría incluir también en la discusión de la modelización estadística debido a que muchos algoritmos clásicos de ML se pueden clasificar como técnicas de aprendizaje estadístico. Scikit-learn presenta varios algoritmos de clasificación, regresión y agrupación en clústeres, incluidos SVM, RF, k-medias y Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido o *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN). Está diseñado para interoperar sin problemas con las bibliotecas numéricas y científicas NumPy y SciPy, proporcionando una gama de algoritmos de aprendizaje supervisados y no supervisados a través de una interfaz consistente. La biblioteca Scikit-learn también es lo suficientemente robusta para su uso en sistemas de producción debido a su comunidad de soporte. Con Scikit-learn se pueden realizar tareas avanzadas de aprendizaje estadístico como analizar modelos estadísticos en una cadena, generar datos aleatorios de regresión y clasificación para probar algoritmos, realizar varios tipos de codificación y transformación en los datos de entrada, buscar hiperparámetros para algoritmos complejos como SVM, etc. [54]

Para la implementación de las técnicas de DL se utiliza **PyTorch**, una biblioteca de ML de código abierto para desarrollar aplicaciones de ML. Es uno de los *frameworks* más populares para la investigación y la producción de modelos DL debido a su facilidad de uso y su flexibilidad. Permite crear modelos utilizando una sintaxis intuitiva y similar a Python, facilitando la experimentación y la creación de prototipos [47]. Cabe mencionar que se comenzó utilizando Keras, otra biblioteca de ML de código abierto de desarrollo de ML a más alto nivel que Pytorch. Sin embargo, se optó finalmente por PyTorch debido a que es notablemente más rápida. Otro motivo para su elección fue que brinda la flexibilidad necesaria para definir o modificar modelos de DL, es decir, se utiliza para crear soluciones escalables. Además, los conjuntos de datos de nivel industrial, como el *dataset* asociado a este proyecto, no son un problema para PyTorch, y puede compilar y entrenar modelos con gran facilidad [59].

NumPy [54] es el estándar de facto para la computación numérica en Python, utilizado como base para construir bibliotecas más avanzadas para aplicaciones de ciencia de datos y ML como TensorFlow o Scikit-learn. NumPy ofrece un conjunto de funciones de modelado

estadístico para realizar estadísticas descriptivas básicas y generar variables aleatorias basadas en varias distribuciones discretas y continuas.

Existen dos potentes bibliotecas de Python para tareas de visualización. Por un lado, **Matplotlib** [54] es la biblioteca base más utilizada para la visualización general. Por otro lado, **Seaborn** [54] se construye sobre Matplotlib, proporcionando *Application Programming Interface* (API)s directas para visualizaciones estadísticas dedicadas; por ello, es una de las favoritas entre los científicos de datos. Algunas de las gráficas de modelado estadístico avanzado que Seaborn proporciona son mapas de calor, violines, diagramas de dispersión con regresión lineal, ajuste e intervalos de confianza, gráficos de pares, gráficos de correlación y gráficos con facetas (es decir, visualizar una relación entre dos variables que dependen de más de una variable).

SciPy [54] es un ecosistema de software de código abierto para matemáticas, ciencias e ingeniería. De hecho, NumPy y Matplotlib son componentes de este ecosistema. Específicamente en el modelado estadístico, SciPy posee una gran colección de métodos y clases rápidos, potentes y flexibles. Destaca su uso para estadísticas inferenciales. Con SciPy se puede generar variables aleatorias a partir de una amplia selección de distribuciones estadísticas discretas y continuas, calcular la frecuencia y resumir las estadísticas de los conjuntos de datos multidimensionales, ejecutar pruebas estadísticas populares, realizar cálculos de correlación y calcular medidas estadísticas de distancia.

Más allá de la computación de estadísticas descriptivas e inferenciales básicas, se encuentra el ámbito del modelado avanzado, por ejemplo, regresión multivariante, modelos aditivos generalizados, pruebas no paramétricas, análisis de supervivencia y durabilidad, modelado de series temporales, imputación de datos con ecuaciones encadenadas, etc. El paquete **Statsmodels** [54] permite realizar todos estos análisis. Los modelos de estadística también producen tablas detalladas con valores importantes para el modelado estadístico, como p-valores, R^2 ajustado, etc.

A continuación, se listan las librerías utilizadas junto a sus versiones específicas:

- pandas, 1.5.2
- scikit-learn, 1.2.2
- torch, 2.0.1
- numpy, 1.24.1
- matplotlib, 3.6.3
- scipy, 1.10.0

Redacción de la memoria. Por otro lado, se utiliza LaTeX para la redacción de la memoria, un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contienen fórmulas matemáticas. LaTeX fue

escrito con la intención de facilitar el uso del lenguaje de composición tipográfica TEX. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos dado que la calidad tipográfica de los documentos es comparable a la de una editorial científica de primera línea. Además, se trata de software libre bajo licencia LATEX *Project Public License* (LPPL) [40].

4.3. Herramientas

Durante el desarrollo de la investigación han sido necesarias una serie de herramientas software con las que compilar el código. Como en el caso de las tecnologías, también se pueden diferenciar según su propósito. Esto es, se explican por separado las tecnologías dedicadas al desarrollo técnico del proyecto de las destinadas a la redacción del informe asociado al proyecto. Por último, se comentan las herramientas de comunicación entre los miembros del equipo de trabajo.

Desarrollo técnico. Visual Studio Code ("VSCode" para abreviar) es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web [38]. Cabe mencionar que se comenzó utilizando Google Colaboratory ("Colab" para abreviar), un producto de Google Research que permite escribir y ejecutar código arbitrario de Python [22]. Es especialmente adecuado para tareas de ML, análisis de datos y educación. Sin embargo, se optó finalmente por VSCode, debido a que incluye soporte para la depuración, algo imprescindible a la hora de crear modelos de DL desde cero (*from scratch*) y trabajar con tensores 3 Dimensiones o 3 *Dimensions* (3D).

Redacción de la memoria. Overleaf es una herramienta de publicación y redacción colaborativa en línea que hace que todo el proceso de redacción, edición y publicación de documentos científicos sea mucho más rápido y sencillo. Ofrece un editor LATEX fácil de usar con colaboración en tiempo real. La salida, completamente compilada, se produce automáticamente en segundo plano mientras se escribe [1].

Comunicación del equipo. Microsoft Teams ("Teams" para abreviar) es una aplicación de colaboración creada para el trabajo híbrido para que un equipo esté informado, organizado y conectado, todo en un mismo lugar. Puede ayudar a un equipo a comunicarse, independientemente de dónde se encuentre, a través de *chats*, canales de equipo y reuniones. Otra característica importante es que permite la compartición de archivos entre usuarios [37].

Trello es una herramienta visual que permite a los equipos gestionar cualquier tipo de proyecto y flujo de trabajo, así como supervisar tareas. Solo hay que registrarse, crear un tablero y personalizarlo añadiendo archivos, *checklists* o, incluso, automatizaciones, según las necesidades del equipo [3].

4.4. Resumen

La Tabla 4.8 resume este capítulo, asociando a cada técnica o parte del proyecto (columnas) las tecnologías y herramientas utilizadas para su desarrollo (filas).

	SVR	XGBDT	MLP	LSTM	ED	Memoria	Comunicación
VSCode	X	X	X	X	X		
Python	X	X	X	X	X		
Pandas	X	X	X	X	X		
Scikit-learn	X	X					
PyTorch			X	X	X		
NumPy	X	X	X	X	X		
Matplotlib	X	X	X	X	X		
SciPy	X	X	X	X	X		
Overleaf						X	
LaTeX						X	
Teams							X
Trello							X

Tabla 4.8: Tecnologías y herramientas utilizadas en cada parte del proyecto.

5: Conceptos teóricos

En este capítulo se sintetizan los conceptos teóricos del dominio de conocimiento del problema, necesarios para la comprensión y el desarrollo del proyecto. Para ello, se ha consultado la tesis de la Referencia [36].

5.1. Industria azucarera

El objetivo de una planta azucarera es la producción de cristales de azúcar partiendo de remolacha azucarera. Para ello, es necesaria la extracción del azúcar contenido en la fibra vegetal o difusión del azúcar desde la remolacha cortada hacia el jugo de difusión, la depuración del jugo obtenido para eliminar las sustancias que no son sacarosa (impurezas), y la supresión del exceso de agua existente en el jugo para finalmente cristalizar y secar los cristales de azúcar. En el proceso de producción de azúcar intervienen diversos tipos de sustancias y en los 3 estados físicos, así como multitud de operaciones básicas.

El proceso de producción de azúcar puede dividirse en dos partes fundamentalmente; el cuarto de remolacha y el cuarto de azúcar. El primero abarca desde la recepción de la remolacha hasta el final de la sección de evaporación, mientras que el segundo incluye la cristalización, centrifugación y secado del azúcar. Esta división, que se observa en la Figura 5.10, procede de las diferencias en la naturaleza de estas partes; el cuarto de remolacha es un proceso fundamentalmente continuo, mientras que el cuarto de azúcar es un proceso esencialmente por lotes. Esto permite su independización de forma relativamente sencilla.

En las fábricas pueden existir dos salas de control independientes para cada una de estas partes. En ocasiones, entre estas dos secciones existe una zona de almacenamiento de jarabe, que absorbe las posibles variaciones que puedan producirse en cada una de ellas.

Además de estas dos agrupaciones existen otras zonas de la planta que no pertenecen a la línea principal del proceso de producción de azúcar pero que son fundamentales para el proceso de producción, como son las calderas, la depuradora, el taller de lechada, torres de enfriamiento, alcoholera, etc. Algunas de estas secciones, como es el caso de las calderas, se consideran de vital importancia a la hora de operar el proceso de forma adecuada, utilizando el vapor de forma óptima.

La remolacha se analiza al llegar a fábrica y se lava para eliminar el material silíceo que la acompaña. A continuación, mediante unos molinos de platos horizontales, se corta en tiras delgadas llamadas cosetas. Estas cosetas se llevan a la etapa de difusión, en la que se utiliza agua caliente para extraer la sacarosa de las mismas.

En la etapa de difusión se obtiene un jugo verde y un residuo agotado, conocido como pulpa. La pulpa contiene entre un 92 % y un 98 % de agua y debe secarse bien y pronto, para evitar su destrucción por los microorganismos. La pulpa se prensa (método mecánico) antes del secado con lo que se consigue un considerable ahorro de energía y la recuperación de parte el azúcar que aún contiene. La pulpa prensada se lleva a los secaderos para eliminar el agua restante. Para realizar el secado se utilizan los gases de combustión producidos en un hogar. La pulpa seca obtenida, que contiene un 90 % de humedad, se utiliza para la alimentación del ganado. Por otro lado, el jugo verde obtenido debe purificarse al máximo posible para eliminar las sustancias que no son sacarosa. Para ello, se introduce en la sección de depuración. En esta etapa se consiguen precipitar los no azúcares mediante un tratamiento químico, que son separados del jugo por filtración.

El jugo depurado obtenido se concentra en la sección de evaporación mediante el aporte de calor, por medio de vapor, evaporando parte del agua que contiene. El jugo concentrado que se obtiene se lleva finalmente al cuarto de azúcar.

En el cuarto de azúcar tiene lugar la cristalización del azúcar, que se produce mediante la evaporación de parte del agua que aún contiene el jarabe. La cristalización de la sacarosa se produce llevando el jarabe a condiciones de sobresaturación. Este proceso se repite en dos o tres ocasiones hasta que no puede extraerse más azúcar de la melaza final. A continuación, se separan los cristales de las aguas madres mediante centrifugación. Finalmente, el azúcar es secado y almacenado, para su empaquetado, distribución y venta.

En la Figura 5.10 se muestra un esquema general del proceso azucarero descrito.

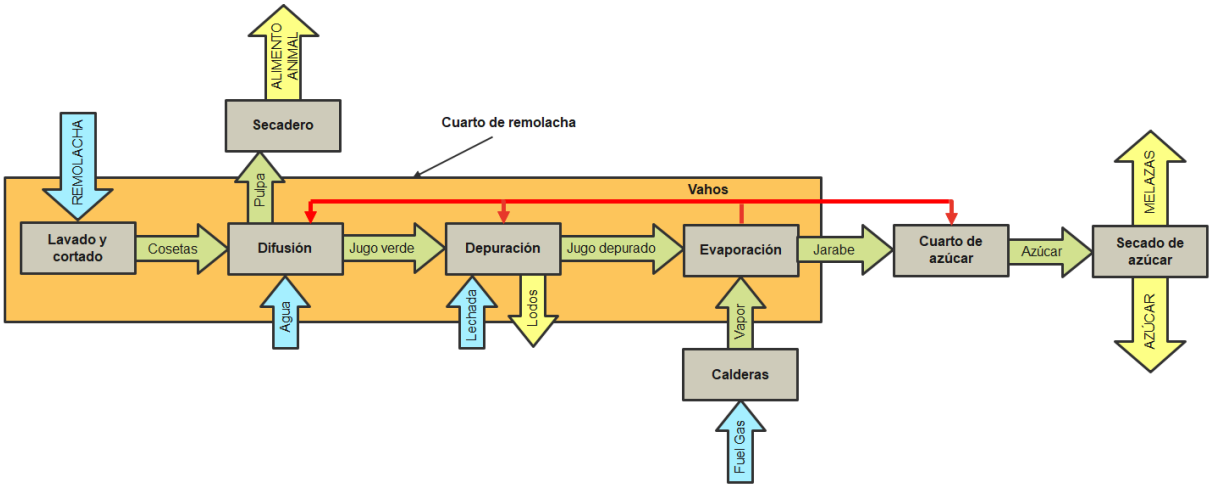

Figura 5.10: Esquema del proceso azucarero [Fuente: elaboración propia].

5.2. Evaporación

Una vez se ha producido la depuración del jugo azucarado, es necesario disminuir la cantidad de agua presente en el mismo, de forma que se facilite la posterior cristalización de los granos de azúcar. Esta concentración se realiza en la sección de evaporación. Este flujo se representa en la Figura 5.11.

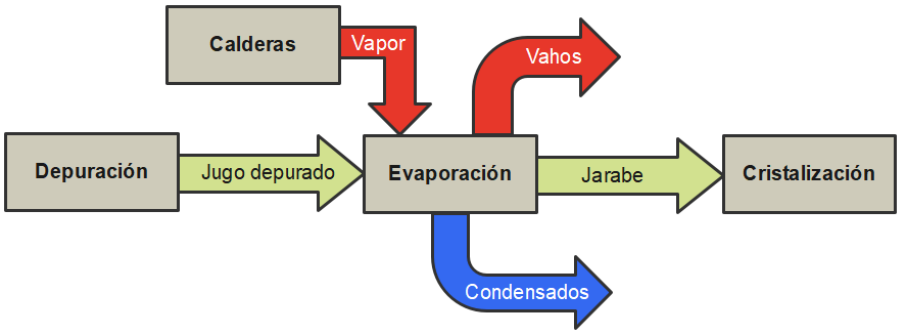

Figura 5.11: Esquema externo de la evaporación [Fuente: elaboración propia].

La sección de evaporación de una industria azucarera es una de las etapas clave en el proceso de producción del azúcar. Su objetivo es la concentración del jugo depurado, de forma que se obtenga un jarabe de características óptimas, tanto en lo que se refiere al Brix como al color. Los procesos que tienen lugar durante la evaporación del jugo son relativamente sencillos pero llevan asociados un alto consumo de vapor, por lo que un buen funcionamiento y operación de esta sección resulta fundamental de cara a la economía del proceso. El vapor que se produce en esta sección sirve además como medio de calefacción en otras zonas del proceso, por lo que la inestabilidad en la evaporación se verá reflejada en otras secciones.

El Brix es una medida de la concentración de sólidos disueltos en una disolución azucarada, que indica cuán concentrada está dicha solución. Resulta fundamental para controlar los procesos de evaporación y cristalización en la producción de azúcar, zumos, concentrados de tomate y bebidas azucaradas [35].

En la industria azucarera, el Brix se calcula utilizando la concentración de sacarosa y no azúcares, esto es, el cociente total de materia seca disuelta en un líquido. Por ejemplo, una disolución de 25 °Bx tiene 25 g de azúcar por cada 100 g de líquido o, dicho de otro modo, hay 25 g de azúcar y 75 g de agua en los 100 g de disolución [15].

El calor necesario para la evaporación del agua contenida en el jugo es aportado por la condensación de vapor de agua en unos equipos denominados evaporadores. Este vapor procede de calderas industriales en las que se produce vapor de alta presión que se aprovecha para la generación de energía eléctrica en turbinas. El vapor de salida de las turbinas, a una presión menor, es utilizado como fluido calefactor en la sección de evaporación.

Para mejorar la economía del vapor se utiliza la denominada "evaporación de efecto múltiple". Ésta consiste en la utilización de varios evaporadores, a distintas presiones, en los que el vapor producido en unos evaporadores se utiliza en otros, a menor presión, de forma que, con la misma cantidad de vapor, se consigue evaporar una mayor cantidad de agua del jugo. Para poder utilizar el vapor que sale de un efecto para evaporar agua en el efecto siguiente, es necesario que la presión disminuya a medida que se avanza en los efectos, de tal forma que la temperatura de ebullición del jugo sea menor. El evaporador o conjunto de evaporadores a una misma presión se denominan efectos. El jugo fluye desde un efecto hacia el siguiente debido a la diferencia de presión entre ellos. Cabe mencionar que existen otras alternativas para los sistemas de evaporación.

Los vahos no utilizados llegan hasta un condensador barométrico. Los condensados que salen de las cámaras de calefacción de los evaporadores se envían a unos balones de expansión. Como los balones están a una presión inferior a la de saturación de los condensados, una parte de los mismos se evapora al entrar en los balones. Este vapor se reintroduce en efectos posteriores de la evaporación, aumentando aún más la economía del proceso. Luego una parte de los vahos producidos en la evaporación se utilizan también en otras partes de la planta como elemento calefactor.

Cada evaporador está formado por una cámara y una serie de tubos. Los evaporadores que trabajan a la misma presión forman parte del mismo efecto y su fuente de vapor es la misma. Los evaporadores de un mismo efecto se identifican con un número que indica su posición dentro del circuito de efectos. Cada evaporador se distingue de su compañero mediante una letra mayúscula que indica el orden en el que actúa dentro de dicho efecto, usando un orden alfabético. Por tanto, la notación de cada evaporador es `{número}{letra}`.

Aunque los principios de funcionamiento son los mismos, la disposición de los efectos, la utilización del vapor, las presiones que se utilizan, etc. pueden variar de unas fábricas a otras. Tanto el efecto múltiple como el número de evaporadores por cada efecto se pueden implementar en diferentes niveles, dependiendo del diseño de la planta y de los requerimientos del proceso. En algunas plantas, se utilizan dos o tres efectos para la evaporación del jugo, mientras que en otras se pueden utilizar hasta seis efectos. En general, un sistema de 2 efectos puede tener entre 4 y 10 evaporadores en cada efecto, mientras que un sistema de 6 efectos puede tener hasta 40 evaporadores en total. Es importante tener en cuenta que, a medida que aumenta el número de evaporadores en cada efecto, aumenta la eficiencia energética del sistema y se reduce el consumo de vapor y energía. Sin embargo, también se requiere una mayor inversión en equipos y una mayor complejidad en el diseño y operación del sistema.

El jugo procedente de la depuración se introduce en el depósito de anteevaporación y, desde ahí, se bombea hacia el primer efecto. Este depósito tiene un sistema de seguridad que introduce un caudal de agua si el nivel baja del 10 %, de tal forma que no exista peligro de que los evaporadores se queden sin líquido. El caudal de salida del depósito de anteevaporación está fijado mediante un controlador de caudal, con lo que se garantiza el caudal que entra en la evaporación, pero no el Brix ni la temperatura.

Antes de entrar en los evaporadores, el jugo se precalienta en una serie de recalentadores que utilizan como medio de calefacción una parte de los vahos que se producen en los distintos efectos. Como resulta lógico, se utiliza el vapor de menor calidad para el jugo con temperaturas más bajas y el de mayor calidad para el jugo más caliente. Con esto, se consigue que el jugo entre en los evaporadores aproximadamente a su temperatura de ebullición. De esta forma, el calor dentro de los evaporadores se emplea en evaporar y no en calentar el jugo, con lo que se aumenta el rendimiento de los mismos.

Respecto a la disposición de los efectos, el jugo circula en serie por todos los evaporadores. En el ejemplo de la Figura 5.12, el recorrido sería Ia, Ib, IIa, IIb, IIc, IIIa, IIIb, IVa, IVb, IVc, Va, Vb y VI; por lo que, a la salida de la evaporación, el jugo ha atravesado 13 evaporadores. El vapor circula en paralelo entre los evaporadores de un mismo efecto y en serie a través de los distintos efectos, de tal forma que el vapor que se produce en el efecto I se envía al efecto II, el vapor que se produce en el efecto II se envía al efecto III, y así sucesivamente. Las flechas de color azul en la Figura 5.12 representan el flujo del jugo, mientras que las de color rojo hacen referencia a la circulación del vapor de agua.

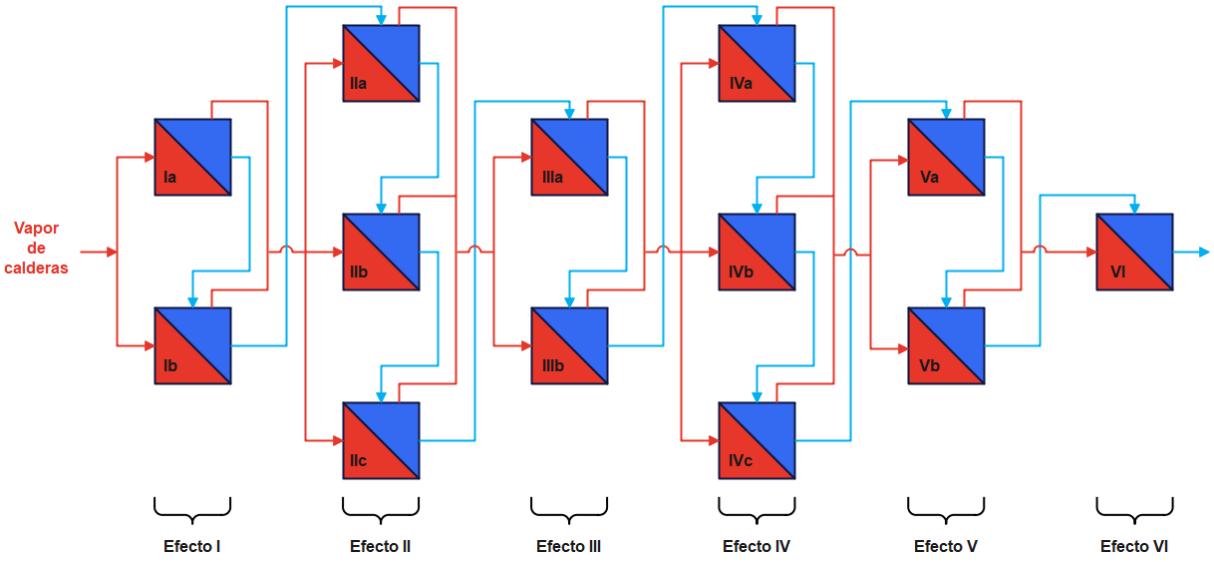

Figura 5.12: Esquema interno de la evaporación [Fuente: elaboración propia].

Durante el proceso de evaporación se mide la concentración de azúcar en el jugo mediante el uso del refractómetro, que indica el porcentaje de sólidos solubles en el jugo. El Brix del jugo aumenta a medida que se reduce la cantidad de agua en el jugo. Una vez que se alcanza la concentración de jugo deseada, el jugo concentrado se transfiere a la siguiente etapa del proceso, que puede ser la cristalización o la producción de melaza, dependiendo de los requerimientos del proceso de obtención del azúcar en la azucarera.

El objeto principal de la sección de evaporación es la obtención de un jarabe con un Brix elevado y lo más estable en el tiempo posible. Por ello, se trata de un sistema controlado, pues el Brix se controla de forma indirecta a través del agua que se está evaporando. Para

el correcto funcionamiento de este sistema de control, así como el cumplimiento de sus objetivos, es necesario medir el Brix a la salida del último evaporador. Esto motiva el uso de sensores, en especial, sensores software como los que se desarrollan en este proyecto.

La cantidad de agua que se evapora depende, esencialmente, de dos factores: el caudal y la presión del vapor que se introduce en el primer efecto y el vacío existente en el último efecto.

Por un lado, se mide el valor de la presión del vapor de agua a la entrada del primer evaporador, pues interesa que sea fijo. A la evaporación, concretamente al primer efecto, llegan dos corrientes de vapor procedentes de la zona de calderas; una de las corrientes es de alta presión y procede directamente de calderas; la otra, ha pasado previamente por las turbinas y, por tanto, tiene una presión menor. Para controlar la presión a la entrada de la evaporación, se actúa sobre el caudal de vapor directo de calderas, de tal forma que, si la evaporación necesita más presión, se abre esta válvula y entra más vapor directo de calderas. La utilización de vapor directo de calderas resulta poco económica, ya que se trata de un vapor de mucha calidad. Por tanto, los objetivos de este control son llevar a cabo una respuesta rápida ante perturbaciones en la presión y minimizar el vapor directo de calderas que se utiliza.

Por otro lado, se controla el caudal de líquido que entra en cada evaporador. En el último evaporador, se regula la cantidad de vapor de agua que pasa por el recalentador dependiendo de la presión que se quiera conseguir. El objetivo de este control, también llamado control de vacío, es asegurar que el salto de presiones entre cada efecto y, por tanto, la circulación del jugo, son correctos.

En resumen, se controla la presión en los extremos del circuito. Para ello, como se observa en la Figura 5.13, se utilizan válvulas Controlador Indicador de Presión o *Pressure Indicating Controller* (PIC) y válvulas Controlador Indicador de Flujo o *Flow Indicating Controller* (FIC).

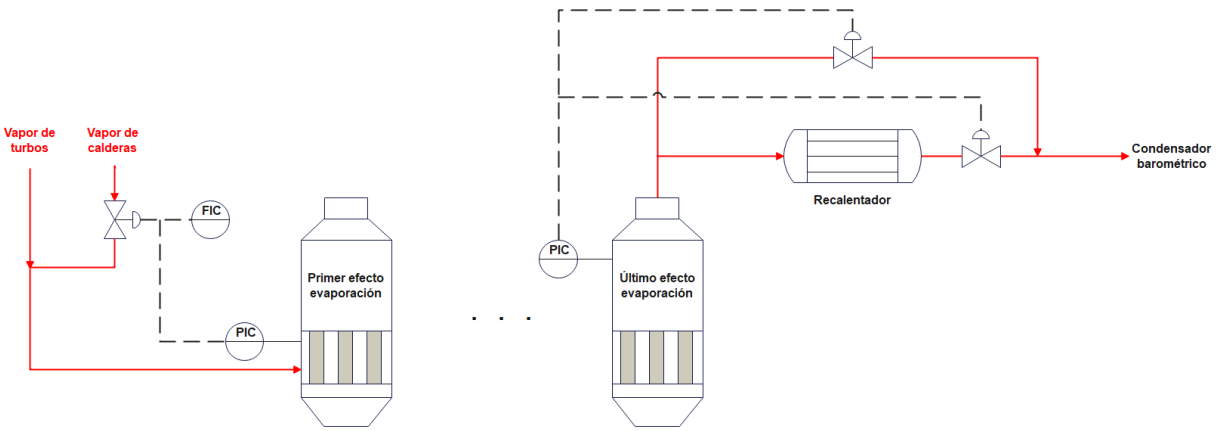

Figura 5.13: Control de vapor en el primer efecto y de vacío en el último efecto [Fuente: elaboración propia].

6: Trabajos relacionados

En este capítulo se estudian los trabajos realizados hasta el momento por otros investigadores para solucionar el problema objetivo del proyecto, con el fin de disponer de unos resultados que sirvan de punto de partida y que se puedan mejorar. Concretamente, se explican los antecedentes de medida (sensores físicos y sensores químicos) del Brix en la Sección 6.1 y los de estimación (sensores software) del Brix en la Sección 6.2.

6.1. Medición: sensores físicos y sensores químicos

Existen dos clases de sensores clasificados según el tipo de información que sean capaces de transformar. Los sensores físicos son dispositivos que detectan cambios en parámetros físicos como la temperatura, la presión, el flujo de masa, etc. Por su parte, los sensores químicos detectan cambios de pH, concentración, composición, etc. Actualmente, la mayor parte de sensores utilizados para el control de procesos industriales son físicos, pero la necesidad de obtener información química ha favorecido la investigación y el desarrollo del segundo tipo de sensores. Sin embargo, la aplicación de los sensores químicos en el campo industrial es muy limitada, ya que su fabricación se suele realizar manualmente y, por tanto, lleva asociado un elevado coste [50]. Los únicos sensores químicos que se tratan en esta sección son los biosensores.

Existen numerosos métodos capaces de medir el contenido de materia seca: la mayoría de ellos consisten en estudios realizados en laboratorios (Subsección 6.1.1), pero también se han desarrollado tecnologías que permiten una medición en tiempo real (Subsección 6.1.2).

6.1.1. Estudios en laboratorio

Los estudios realizados en laboratorios son, en su mayoría, simples y poco costosos. Estos estudios necesitan extraer una muestra, transportarla hasta el laboratorio, realizar el análisis y esperar hasta la obtención de los resultados. Este proceso provoca una demora en el tiempo entre la extracción y la medición que ralentiza la toma de decisiones [13]. Por

tanto, la principal desventaja de estos métodos es el retraso en las mediciones. Además, son inadecuados para monitorizar un proceso, debido a que el tamaño de la muestra que se analiza es muy pequeño.

En la actualidad destacan seis estudios en laboratorio, cada uno de los cuales se explica a continuación.

Polarimetría

Es un método de análisis químico que permite medir el cambio que sufre el plano de luz polarizada cuando atraviesa un medio transparente formado por sustancias ópticamente activas [2]. Si el plano de polarización se desvía respecto de la dirección inicial en el sentido de las agujas de un reloj, la sustancia se denomina dextrógira, y si es en sentido contrario, levógira [40].

Los azúcares son compuestos ópticamente activos y cada uno tiene una rotación específica característica: algunos, como la fructosa, son levorrotatorios; otros, como la glucosa, son dextrorrotatorios. El comportamiento del azúcar de cocina (sacarosa) respecto a la desviación de la luz en el polarímetro a diferentes concentraciones experimenta distintos ángulos de rotación, todos ellos hacia la derecha [10].

Cromatografía

La cromatografía es un método de separación y análisis cuantitativo de mezclas de sustancias. Consiste en la distribución diferencial de los componentes de una mezcla entre dos fases, una fase móvil y una fase estacionaria. Durante el proceso, los componentes de la muestra se separan en función de su interacción con estas fases, lo que resulta en la formación de bandas o picos separados. Según este enfoque, la base de la técnica cromatográfica está formada por tres componentes: una fase estacionaria, que siempre está compuesta por una capa sólida o una capa de líquido adsorbida en la superficie de un soporte sólido; una fase móvil, que siempre está compuesta por un componente líquido o gaseoso; y las moléculas separadas. El propósito de aplicar la cromatografía es lograr una separación satisfactoria en un intervalo de tiempo adecuado [12].

Biosensores

Un biosensor es un dispositivo que utiliza componentes biológicos para detectar y medir la presencia de sustancias químicas o biomoléculas específicas en una muestra. Estos componentes biológicos pueden ser enzimas, anticuerpos, tejidos vivos u otros materiales biológicos que interactúan con la sustancia objetivo y generan una señal detectable [30].

[2] Las sustancias ópticamente activas son aquellas que producen un giro del plano de polarización de la luz.

Hidrómetro

Un hidrómetro es un dispositivo utilizado para medir la densidad o la gravedad específica de un líquido. También es conocido como gravímetro, densímetro o aerómetro. Generalmente, consiste en un tubo de vidrio con un bulbo en un extremo y una escala graduada en el otro extremo. El bulbo se sumerge en el líquido y la densidad del líquido hace que el hidrómetro flote a diferentes niveles. La escala graduada permite leer la densidad o gravedad específica del líquido [34].

NIRS

Una Espectroscopia del Infrarrojo Cercano o *Near-Infrared Spectroscopy* (NIRS), es una técnica no invasiva utilizada para medir la absorción de luz en la región del infrarrojo cercano por tejidos biológicos [48].

Refractómetro

Un refractómetro es un dispositivo utilizado para medir el índice de refracción de un material, que es una medida de la velocidad de la luz al pasar a través de ese material en comparación con su velocidad en el vacío. El índice de refracción proporciona información sobre la forma en que la luz se curva al pasar por el material, lo cual es útil para determinar la concentración, la pureza o la calidad de ciertas sustancias [49].

6.1.2. Medición en tiempo real

A diferencia de las anteriores, estas técnicas son capaces de proporcionar resultados en tiempo real. Sin embargo, los sensores que incorporan suponen un elevado coste, lo que impide su propagación y obliga a situarlos únicamente en puntos críticos. Por tanto, la principal desventaja de estos métodos es su elevado coste y mantenimiento.

En la actualidad destacan tres mediciones del Brix en tiempo real, cada una de los cuales se explica a continuación.

Refractómetro digital

A diferencia del utilizado en los laboratorios, el refractómetro digital proporciona el valor de la medida automáticamente de manera digital, sin necesidad de hacer la lectura de la medición. Este tipo de refractómetros también suelen ser portátiles [28].

Tecnología por microondas

El resonador de microondas funciona en base al principio de que la permitividad dieléctrica compleja de un líquido afecta la respuesta electromagnética del sensor. Al analizar los cambios en la frecuencia de resonancia y el ancho de banda, se puede determinar la composición de la solución líquida. Los métodos de perturbación resonante ofrecen una mayor precisión en comparación con las técnicas de banda ancha.

En general, el sensor resonante de microondas ofrece una solución rentable y no invasiva para mediciones de concentración en diversas industrias, como alimentos, petróleo, medicina y productos farmacéuticos. Proporciona monitoreo en tiempo real de la calidad del proceso y flexibilidad en comparación con procedimientos químicos más complejos [19].

Principio de Coriolis

El funcionamiento básico de los caudalímetros Coriolis se basa en los principios de la mecánica del movimiento. A medida que el líquido se mueve a través de un tubo que vibra es obligado a acelerar mientras se mueve hacia el punto de vibración de amplitud pico. En cambio, el líquido que desacelera se mueve lejos del punto de la amplitud pico a medida que sale del tubo. El resultado es una reacción de giro del tubo de caudal durante las condiciones de flujo mientras atraviesa cada ciclo de vibración [16].

La tecnología Coriolis *Micro Motion* y los equipos de medición de densidad de Emerson ofrecen medición y supervisión en tiempo real de la densidad de la materia prima, lo que permite efectuar ajustes instantáneos, de acuerdo a la compensación real del producto [17].

6.2. Estimación: sensores software

Existen diversas técnicas de desarrollo de sensores software o sensores *soft* para estimar la medida de Brix en la industria azucarera. Los sensores pueden basarse en distintos métodos como, por ejemplo, la Medida Indirecta o *Indirect Measurement* (IM) a partir de la presión de vapor y la temperatura del jugo, el uso de redes neuronales, la aplicación de PCA para reducir el coste de entrenamiento de la red, el método PLS [18] y la utilización de algoritmos de ML [44] con PCA previo. Cabe mencionar que estos estudios utilizan un conjunto de entrenamiento distinto, pero la fábrica y el Brix estimado son iguales.

6.2.1. Medida indirecta [18]

La variación en la presión de vapor se determina experimentalmente y se puede calcular utilizando las ecuaciones 6.17 a 6.21. En ellas, P_V es la presión de vapor (en bares), T es la temperatura de la solución (en grados Kelvin), w_{DS} es el contenido de sustancia seca (en g/100 g), T_S es la temperatura de ebullición en presencia de azúcar en la solución (en grados Kelvin), y k_1, k_2 y k_3 son términos de corrección que dependen del contenido de sustancia seca.

$$k_1 = -0{,}2 + e^{-1{,}5254+0{,}022962\cdot w_{DS}+0{,}0002163\cdot w_{DS}^2} \tag{6.17}$$

$$k_2 = 0{,}9985 + 0{,}01 \cdot e^{-3{,}2021+0{,}066743\cdot w_{DS}-0{,}0001161\cdot w_{DS}^2} \tag{6.18}$$

$$k_3 = 0{,}0001 \cdot e^{-1{,}4276-0{,}024362\cdot w_{DS}+0{,}0006047\cdot w_{DS}^2} \tag{6.19}$$

$$T_S = \frac{-k_2 + \sqrt{k_2^2 - 4 \cdot k_3 \cdot (k_1 - T)}}{2 \cdot k_3} \tag{6.20}$$

$$P_V = 10^{-\left(\frac{2147}{T_S + 273{,}2}\right) + 5{,}7545} \tag{6.21}$$

Estas ecuaciones establecen una relación entre la temperatura del jugo, T, y la presión de vapor saturado, P_V, con la concentración de azúcar, w_{DS}. Dado que todas estas variables suelen medirse en la planta, basta con resolver el sistema de ecuaciones de manera implícita para calcular el Brix como función de las otras dos variables.

Hay que tener en cuenta que se hacen tres suposiciones al utilizar este método para determinar el Brix, por lo que habrá un sesgo entre los valores predichos por las fórmulas y los datos del proceso. En primer lugar, estas ecuaciones se basan en mediciones de soluciones puras de sacarosa, mientras que el jugo en la planta es una solución técnica de azúcar. En segundo lugar, se supone que el sistema está en equilibrio termodinámico, pero en los procesos industriales no se dan las condiciones para alcanzarlo. Por último, se supone que la presión de vapor medida es la presión del vapor saturado en equilibrio con el jugo; sin embargo, en los procesos industriales, es probable que el vapor no esté saturado. De hecho, dependiendo de la posición en la que se encuentre el sensor, es posible que el vapor esté ligeramente sobrecalentado.

Otra de las desventajas de este método es su sensibilidad a los cambios en los parámetros, lo que implica que pequeñas variaciones en la presión de vapor o en la temperatura del jugo generan cambios abruptos en el resultado.

Existe un sesgo entre la medición de la presión proporcionada por el sensor y la presión real. Esta desviación (*offset*) se puede ajustar planteando un problema de optimización como el de la Ecuación 6.22. En ella, n es el número de muestras recolectadas y $\tilde{w}_{\mathrm{DS}_i}(\mathit{offset})$ es el Brix estimado de forma implícita y sumando el sesgo.

$$\min_{\text{offset}} \left(\sum_{i=1}^{n} \left(w_{\mathrm{DS}_i} - \tilde{w}_{\mathrm{DS}_i}(\mathit{offset}) \right)^2 \right) \tag{6.22}$$

La principal desventaja de este método, derivada de la sensibilidad descrita anteriormente, es la falta de robustez de los resultados con respecto a las desviaciones en las mediciones. Con el fin de comparar la robustez de los métodos implementados, se introduce manualmente una desviación en la presión en el sistema, concretamente, de $-0{,}0117$, aproximadamente. Este *offset* se obtiene de calcular el mínimo presente en la Ecuación 6.22, probando durante 90 minutos mediante la ejecución de un bloque de código de Python.

La Figura 6.14 muestra los resultados de aplicar el sensor basado en la resolución de las ecuaciones 6.17 a 6.21. Las variables utilizadas en este caso son las variables número 1, 7 y 50 de la Tabla 7.10, que se corresponden con el Brix, la presión del vapor y la temperatura

del jugo, pero con los datos ya filtrados (filtro de Butterworth y de media móvil con ventana de 100). En el gráfico, la línea naranja representa los datos reales recolectados de la planta y la línea azul es el resultado del cálculo del Brix resolviendo las ecuaciones 6.17 a 6.21 de manera implícita. Se puede observar que, en un primer enfoque, los resultados obtenidos no coinciden con las mediciones reales del proceso. Se observa, por tanto, ese sesgo.

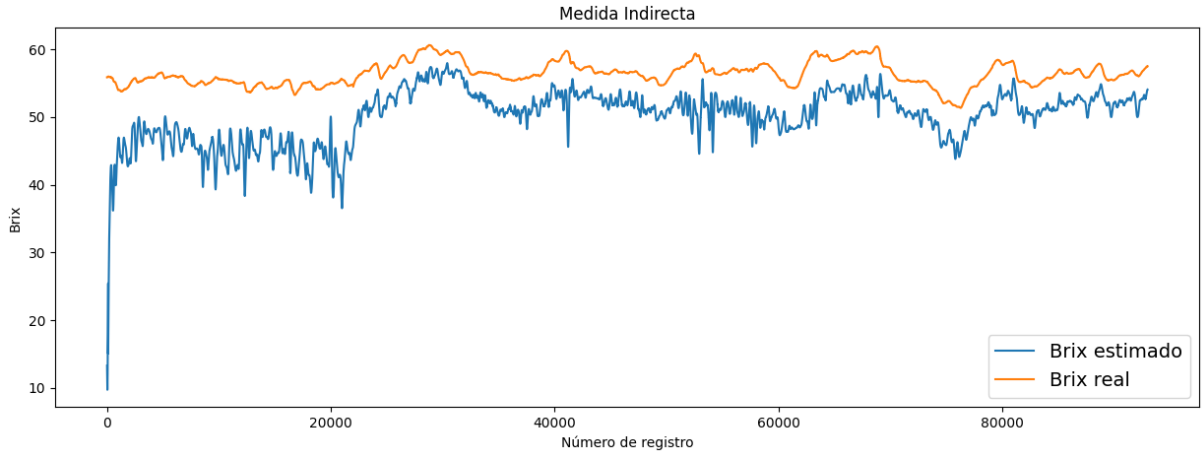

Figura 6.14: Medida indirecta.

La Figura 6.15 muestra los resultados obtenidos con el método de corrección basado en la Ecuación 6.22. El número de muestras consideradas para calcular el desplazamiento "óptimo" fue de 80000 y se ejecutó la función de minimización durante 90 minutos. Este *offset* se suma a la presión y se utilizan las ecuaciones para estimar el Brix.

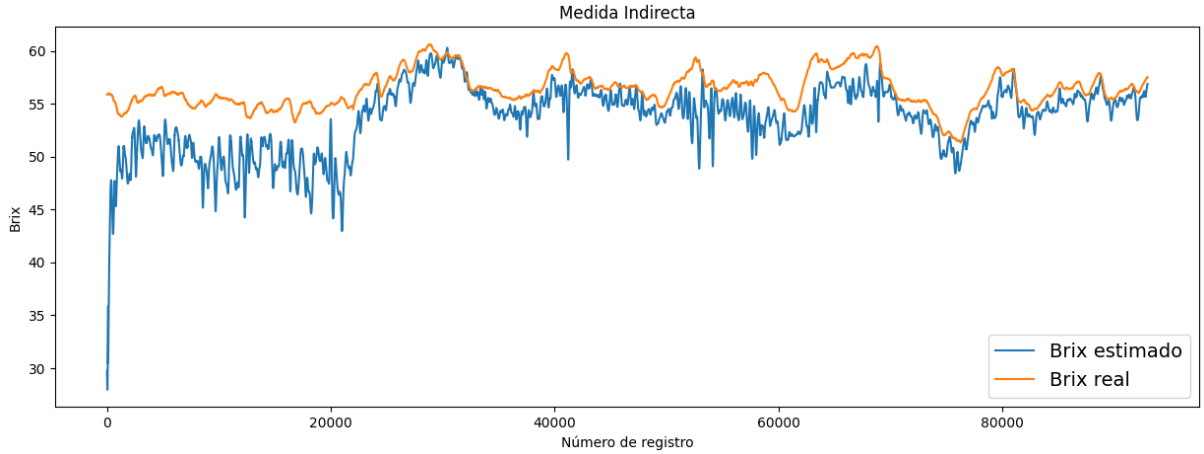

Figura 6.15: Medida indirecta con *offset*.

Los resultados obtenidos son considerablemente mejores en este caso, por lo que este método podría ser adecuado para la estimación del Brix, pero es necesario corregir

periódicamente cualquier error sistemático en las mediciones de la planta utilizando mediciones de laboratorio. Si se introduce manualmente una desviación artificial en la presión del vapor en alguna de las mediciones, la estimación del Brix se calcula con un gran error, es decir, muestra una alta sensibilidad a las perturbaciones.

6.2.2. Red neuronal [18]

La técnica aplicada al diseño de este sensor se basa en el uso de una ANN. Las redes neuronales permiten modelar no linealidades entre las variables medidas y el Brix y, como se muestra en las ecuaciones 6.17 a 6.21, el comportamiento de los parámetros es altamente no lineal.

La gran cantidad de datos del sistema dificulta la decisión sobre la mejor estructura (el número de entradas, el número de capas y el número de neuronas en cada capa) y la mejor forma de entrenar la red neuronal. Una solución es realizar un preprocesamiento de los datos mediante un análisis de correlación, para omitir aquellos datos que proporcionan poca información sobre el sistema. Si dos variables están correlacionadas, una de ellas aporta poca información para el entrenamiento de la red. Para cada par de datos del sistema (x_i, x_j), se calcula la correlación según muestra la Ecuación 6.23. En ella, μ_{x_i}, μ_{x_j} y σ_{x_i}, σ_{x_j} son el valor medio y la desviación estándar de las variables medidas x_i, x_j, respectivamente.

$$\varphi_{x_i,x_j} = \frac{E(x_i - \mu_{x_i})(x_j - \mu_{x_j})}{\sigma_{x_i} \cdot \sigma_{x_j}} \tag{6.23}$$

Cuando la correlación entre dos señales es igual o cercana a $|1|$, esto implica que ambas señales están correlacionadas. Aquellas variables que están fuertemente correlacionadas se eliminan del conjunto de datos. En el caso específico del sensor virtual, se suelen escoger valores mayores que 0,99.

Esta solución para eliminar las variables que están fuertemente correlacionadas tiene la desventaja de que puede ser una tarea costosa. Además, es posible eliminar una variable del modelo neuronal que, a pesar de tener una alta correlación con otra variable, es esencial para la estimación del Brix.

La motivación de la construcción de este sensor era diseñar una nueva configuración que tuviera en cuenta un mayor número de variables del proceso que en el caso del sensor anterior. Se lograron respuestas más robustas a las perturbaciones utilizando este método.

6.2.3. PCA y red neuronal [18]

La técnica aplicada al diseño de este sensor se basa también en el uso de una ANN. Utilizando la red neuronal, se pueden extraer las no linealidades entre las variables latentes del sistema y la medición del Brix.

Una forma de estimar las variables latentes del sistema es el uso del PCA. El PCA consiste en una transformación lineal del espacio inicial de A variables en un nuevo espacio con J ($A < J$) variables no correlacionadas e independientes para ayudar a una mejor comprensión de los datos.

Típicamente, los datos de los sistemas industriales con un alto nivel de automatización están altamente correlacionados y tienen un factor de ruido significativo. El PCA divide el conjunto de datos, recopilados en una matriz $\mathbf{X}$ de tamaño $\mathbb{R}^{KxJ}$ (K muestras de J variables), en dos partes según la Ecuación 6.24. En ella, $\mathbf{T} \cdot \mathbf{P}_{1:A}^T$ representa la variabilidad del sistema explicada por las variables latentes y se puede denotar "modelo PCA". La matriz $\mathbf{P}_{1:A}$ es la matriz de pesos o *loadings* que permite calcular las variables latentes en función de las variables medidas. La matriz $\mathbf{T}$ contiene el valor de las variables latentes o *scores* en sus columnas. $\mathbf{E}$ es una matriz que representa el ruido residual del sistema.

$$\mathbf{X} = \sum_{a=1}^{A} t_a \cdot p_a^T + \mathbf{E} = \mathbf{T} \cdot \mathbf{P}_{1:A}^T + \mathbf{E} \tag{6.24}$$

Para diseñar un modelo de PCA que represente con precisión el sistema, se debe seleccionar el número de variables latentes A que mejor represente los datos del proceso. La mejor manera de elegir este parámetro es a través de un procedimiento de validación cruzada.

Una vez que se eliminan los *outliers* del sistema, se puede establecer el modelo PCA y calcular los *scores* del conjunto de entrenamiento con la Ecuación 6.24. Se obtiene, así, $\mathbf{T} = \mathbf{X} \cdot \mathbf{P}_{1:A}$. Una vez estandarizados estos *scores*, se pueden utilizar como entradas para una red neuronal que modele la relación entre ellos y el contenido de sustancia seca, En este caso, el número de entradas en la red neuronal será tan alto como el número de variables latentes. Esto significa que la red neuronal puede ser más simple que la red utilizada en la Sección 6.2.2.

La motivación de la construcción de este sensor era utilizar las variables latentes que mejor explicasen el comportamiento de todas las variables medidas como entradas a la red neuronal. Se lograron estimaciones más robustas sin presentar un esfuerzo adicional en el entrenamiento de la red neuronal debido al gran número de entradas.

6.2.4. PLS [18]

La técnica aplicada al diseño de este sensor se basa en el uso de una de las técnicas estadísticas multivariantes más ampliamente utilizadas en la industria, conocida como regresión PLS o mínimos cuadrados parciales. Entre sus múltiples ventajas, destaca que es capaz de establecer modelos de regresión a partir de datos con alta colinealidad, permite analizar datos con más variables que individuos, proporciona modelos con una alta tasa de estabilidad en la predicción, minimizando la tasa de ajuste, puede tratar conjuntos de información con datos faltantes y es capaz de detectar *outliers* en la fuente de datos.

Como en el caso de PCA, permite limitar el espacio de búsqueda y reducir el tiempo de ejecución.

PLS establece una estructura de proyección que modela la relación entre una matriz de respuesta $\mathbf{Y}$ de tamaño $\mathbb{R}^{KxJ}$, que es la variable que se desea estimar mediante este sensor, y la matriz de predicción $\mathbf{X}$ de tamaño $\mathbb{R}^{KxJ}$. En el ejemplo de este proyecto, la matriz $\mathbf{X}$ está compuesta por las variables medidas y la matriz $\mathbf{Y}$ por las mediciones del contenido de sustancia seca (Brix). Ambas matrices pueden descomponerse en matrices más simples utilizando las ecuaciones 6.25 y 6.26. En ellas, $\mathbf{T}$ y $\mathbf{U}$ son las matrices de *scores*, $\mathbf{P}$ y $\mathbf{C}$ las de *loadings*, y $\mathbf{E}$ y $\mathbf{F}$ las de residuos de $\mathbf{X}$ e $\mathbf{Y}$, respectivamente, para un modelo con A variables latentes determinadas mediante validación cruzada.

$$\mathbf{X} = \sum_{a=1}^{A} t_a \cdot p_a^T + \mathbf{E} = \mathbf{T} \cdot \mathbf{P}^T + \mathbf{E} \tag{6.25}$$

$$\mathbf{Y} = \sum_{a=1}^{A} u_a \cdot c_a^T + \mathbf{F} = \mathbf{U} \cdot \mathbf{C}^T + \mathbf{F} \tag{6.26}$$

Como se interpreta en la Ecuación 6.27, los *scores* de X son combinaciones lineales de la matriz $\mathbf{X}$, para la primera variable latente, o la matriz de residuos $\mathbf{X}_a$ para la a-ésima variable latente. En ella, w_a es el vector de pesos de la a-ésima variable latente.

$$t_a = \mathbf{X}_{a-1} \cdot w_a, \mathbf{X}_a = \mathbf{X}_{a-1} - t_a \cdot p_a^T \tag{6.27}$$

La relación que maximiza la covarianza entre las matrices $\mathbf{T}$ y $\mathbf{U}$ se establece mediante la Ecuación 6.28. En ella, $\mathbf{B}$ es una matriz diagonal y $\mathbf{H}$ es una matriz de residuos.

$$\mathbf{U} = \mathbf{T} \cdot \mathbf{B} + \mathbf{H} \tag{6.28}$$

Una vez que se establece el modelo de PLS a partir de los datos históricos del proceso, es posible realizar la predicción de nuevas respuestas o estimaciones con la Ecuación 6.29. En ella se utiliza la matriz de cargas $\mathbf{W}^* = \mathbf{W} \cdot \mathbf{P}^T \cdot \mathbf{W}^{-1}$ para realizar el gráfico de cargas $w \cdot c[1]/w \cdot c[2]$. Este gráfico es muy útil para mostrar qué variables en X están más relacionadas con la respuesta del modelo Y, ya que proporciona información gráfica sobre la relación entre las variables medidas y las variables estimadas. Utilizando la Ecuación 6.29, es posible estimar el Brix como función de las variables de proceso medidas.

$$\hat{\mathbf{Y}} = \mathbf{T} \cdot \mathbf{C}^T = \mathbf{X} \cdot \mathbf{W} \cdot (\mathbf{P}^T \cdot \mathbf{W})^{-1} \cdot \mathbf{C}^T = \mathbf{X} \cdot \mathbf{W}^* \cdot \mathbf{C}^T \tag{6.29}$$

Al igual que en el caso de PCA, las variables medidas pueden tener diferentes rangos numéricos. Como el cálculo del modelo de PLS depende de la varianza de las variables y la varianza depende del rango numérico, resulta necesario normalizar las variables para que tengan una media de 0 y una varianza unitaria.

La motivación de la construcción de este sensor era diseñar una nueva configuración que tuviera en cuenta un mayor número de variables del proceso que en el caso del sensor de la Subsección 6.2.1. Utilizando el modelo PLS para obtener información sobre el comportamiento del proceso, se pudo observar que el Brix está altamente asociado con otras variables que no son las dos utilizadas en el sensor de la Subsección 6.2.1 (presión de vapor y temperatura del jugo). Teniendo en cuenta todas las variables obtenidas mediante este método estadístico, se lograron aproximaciones más estables, ya que la medición dependía de más variables. La predicción sigue la tendencia de los datos reales con un error aceptable para este tipo de mediciones. En definitiva, este modelo demuestra una buena capacidad para generalizar, siendo capaz de predecir de manera precisa nuevos datos. Sin embargo, se obtuvieron valores mejorables de las métricas.

6.2.5. PCA y SVR [44]

La técnica aplicada al diseño de este sensor se basa en el uso de PCA y SVR, un regresor ya explicado en la Subsección 4.1.2.

La motivación de la construcción de este sensor era diseñar una nueva configuración que utilizara un modelo de ML más simple que en el caso del sensor de la Subsección 6.2.2. Se lograron buenos resultados utilizando este método.

6.2.6. PCA y *random forest* [44]

La técnica aplicada al diseño de este sensor se basa en el uso de PCA y RF, un regresor ya explicado en la Subsección 4.1.2.

La motivación de la construcción de este sensor era diseñar una nueva configuración que utilizara un modelo de ML más simple que en el caso del sensor de la Subsección 6.2.2 y más complejo que en el caso del sensor de la Subsección 6.2.5. Se lograron unos resultados menos precisos y afectados por cierto grado de sobreajuste utilizando este método.

6.2.7. Resultados de partida

En resumen, se parte de los resultados de la Tabla 6.9 en términos de la métrica MSE.

Método	MSE
PLS	0,45
PCA + SVR	0,78

continúa en la página siguiente

continúa desde la página anterior

Método	MSE
PCA + NN	0,93
IM	1,36
PCA + RF	1,56

Tabla 6.9: Resultados de los trabajos relacionados.

Estos resultados aparecen ordenados de mejor a peor. Éstos son comparables a pesar de realizarse sobre conjuntos de *test* distintos.

7: Aspectos relevantes del desarrollo del proyecto

En este capítulo se recogen los aspectos más interesantes del desarrollo del proyecto.

7.1. Comprensión de los datos

En esta sección se describe el proceso de recopilación y análisis de la calidad de los datos disponibles mediante la Subsección 7.1.1 y la Subsección 7.1.2, respectivamente.

7.1.1. Descripción del conjunto de datos

En esta subsección se encuentra el resumen ejecutivo de los datos.

El conjunto consta de datos del proceso de evaporación de una fábrica azucarera real con 6 efectos, recogidos cada 10 segundos mediante sensores físicos. Los datos reales utilizados en este trabajo se obtuvieron mediante una herramienta Control, supervisión y adquisición de datos o *Supervisor Control And Data Adquisition* (SCADA) descrita en [52]. Una de las características de esta herramienta es que permite registrar datos históricos. En este caso, los datos se registraron con un tiempo de muestreo de 10 segundos. El sensor utilizado en la planta estudiada es un *Micro Motion* de Emerson que se basa en el principio de Coriolis. El sensor de contenido de sustancia seca en el proceso estudiado se encontraba en la salida del jugo del último efecto evaporador.

Contiene un total de 92998 registros, lo que equivale a un estudio de casi 11 días de producción de azúcar. Todas las columnas tienen valores no nulos. Los registros duplicados, 7, son válidos, ya que se pueden dar las mismas condiciones de presión, temperatura, flujo másico y Brix en distintos instantes de tiempo. Se trata de un problema de regresión. Hay un total de 51 atributos numéricos decimales de tipo *float*, que se pueden observar con detalle en la Tabla 7.10.

Nombre			Descripción	Unidad
1	CS1	Output	Contenido de sustancia seca	g/100 g
2	P1	P_vah_e1	Presión del vapor (evaporador 1)	bar
3	P2	P_vah_e2	Presión del vapor (evaporador 2)	bar
4	P3	P_vah_e3	Presión del vapor (evaporador 3)	bar
5	P4	P_vah_e4	Presión del vapor (evaporador 4)	bar abs
6	P5	P_vah_e5	Presión del vapor (evaporador 5)	bar abs
7	P6	P_vah_e6	Presión del vapor (evaporador 6)	bar abs
8	T1	T_ent_e1	Temperatura del jugo (entrada al evaporador 1)	°C
9	T2	T_ent_R10	Temperatura del jugo (entrada al intercambiador de calor R10)	°C
10	T3	T_ent_R13	Temperatura del jugo (entrada al intercambiador de calor R13)	°C
11	T4	T_ent_R14	Temperatura del jugo (entrada al intercambiador de calor R14)	°C
12	T5	T_ent_R3	Temperatura del jugo (entrada al intercambiador de calor R3)	°C
13	T6	T_ent_R3B	Temperatura del jugo (entrada al intercambiador de calor R3B)	°C
14	T7	T_ent_R8	Temperatura del jugo (entrada al intercambiador de calor R8)	°C
15	T8	T_ent_R9	Temperatura del jugo (entrada al intercambiador de calor R9)	°C
16	T9	T_jugo_anteevap	Temperatura del jugo (salida del tanque que acumula el jugo)	°C
17	T10	T_salida_R10	Temperatura del jugo (salida del intercambiador de calor R10)	°C
18	T11	T_salida_R11	Temperatura del jugo (salida del intercambiador de calor R11)	°C
19	T12	T_salida_R12	Temperatura del jugo (salida del	°C

continúa en la página siguiente

continúa desde la página anterior

	Nombre		Descripción	Unidad
			intercambiador de calor R12)	
20	T13	T_salida_R3	Temperatura del jugo (salida del intercambiador de calor R3)	°C
21	T14	T_salida_R3A	Temperatura del jugo (salida del intercambiador de calor R3A)	°C
22	T15	T_salida_R3B	Temperatura del jugo (salida del intercambiador de calor R3B)	°C
23	T16	T_salida_R4	Temperatura del jugo (salida del intercambiador de calor R4)	°C
24	T17	T_salida_R5	Temperatura del jugo (salida del intercambiador de calor R5)	°C
25	T18	T_salida_R6	Temperatura del jugo (salida del intercambiador de calor R6)	°C
26	T19	T_salida_R7	Temperatura del jugo (salida del intercambiador de calor R7)	°C
27	T20	T_salida_R8	Temperatura del jugo (salida del intercambiador de calor R8)	°C
28	T21	T_salida_R9	Temperatura del jugo (salida del intercambiador de calor R9)	°C
29	T22	T_vah_e1	Temperatura del vapor (evaporador 1)	°C
30	T23	T_vah_e2	Temperatura del vapor (evaporador 2)	°C
31	T24	T_vah_e3	Temperatura del vapor (evaporador 3)	°C
32	T25	T_vah_e4	Temperatura del vapor (evaporador 4)	°C
33	T26	T_vah_e5	Temperatura del vapor (evaporador 5)	°C
34	T27	T_vah_e6	Temperatura del vapor (evaporador 6)	°C
35	T28	T_vap_cald	Temperatura del vapor (vapor de calderas a evaporación)	°C
36	W1	W_jugo_R10	Flujo másico del jugo (hacia el intercambiador de calor R10)	t/h
37	W2	W_jugo_R3	Flujo másico del jugo (hacia el intercambiador de calor R3)	t/h
38	W3	W_jugo_R3B	Flujo másico del jugo (hacia el	t/h

continúa en la página siguiente

continúa desde la página anterior

Nombre			Descripción	Unidad
			intercambiador de calor R3B)	
39	W4	W_jugo_R4	Flujo másico del jugo (hacia el intercambiador de calor R4)	t/h
40	W5	W_jugo_R8	Flujo másico del jugo (hacia el intercambiador de calor R8)	t/h
41	W6	W_jugo_R9	Flujo másico del jugo (hacia el intercambiador de calor R9)	t/h
42	W7	W_jugo_anteevap	Flujo másico del jugo (desde el tanque que acumula el jugo)	t/h
43	W8	W_jugo_salida_e6	Flujo másico del jugo (salida del evaporador 6)	t/h
44	W9	W_vap_cald	Flujo másico del vapor (vapor de calderas a evaporación)	t/h
45	T29	e1_T_jugo_salida	Temperatura del jugo (evaporador 1)	°C
46	T30	e2_T_jugo_salida	Temperatura del jugo (salida del evaporador 2)	°C
47	T31	e3_T_jugo_salida	Temperatura del jugo (salida del evaporador 3)	°C
48	T32	e4_T_jugo_salida	Temperatura del jugo (salida del evaporador 4)	°C
49	T33	e5_T_jugo_salida	Temperatura del jugo (salida del evaporador 5)	°C
50	T34	e6_T_jugo_salida	Temperatura del jugo (salida del evaporador 6)	°C
51	P7	P_vap_cald	Presión del vapor (vapor de calderas a evaporación)	bar

Tabla 7.10: Explicación de las variables o atributos.

A continuación, se muestra una forma de agrupar estos atributos, que hace las veces de leyenda de la Tabla 7.10.

Amarillo Brix.

Verde Presión.

Azul Temperatura.

Rosa Flujo másico.

7.1.2. Exploración del conjunto de datos

En esta subsección se describe el estudio de los datos, llevado a cabo en Python.

Descripción estadística

En primer lugar, se hace una descripción estadística de cada columna, que incluye el recuento, la media, la mediana, la desviación estándar, los cuartiles y los valores mínimo y máximo. Dichos estadísticos se muestran en la Tabla 7.11. Cabe mencionar que se calculan para todas las columnas, pero se seleccionan sólamente las más relevantes según lo explicado en la Subsección 7.1.2 con el objetivo de facilitar su visualización.

	Output	e6_T_jugo_salida	P_vah_e6	T_vah_e6	W_jugo_salida_e6
count	92998	92998	92998	92998	92998
mean	56,39	96,02	0,81	92,28	126,09
std	1,73	0,34	0,01	0,49	5,71
min	51,34	94,57	0,77	91,14	112,09
25 %	55,26	95,85	0,8	91,9	121,56
50 %	56,26	96,03	0,81	92,22	126,21
75 %	57,4	96,2	0,81	92,5	130,11
max	60,65	96,83	0,88	93,94	143,52

Tabla 7.11: Descripción estadística de las columnas más relevantes del *dataset*.

Correlación

Se estudia la correlación entre las variables 2 a 2 mediante un mapa de calor. Tras observar la alta correlación entre las mismas variables físicas en distintos efectos, se decide tomar solamente los atributos de los efectos 5 y 6 (los efectos finales son los más importantes para conocer el Brix a la salida por sentido físico ligado a la cercanía), descartando el resto de variables. Dichas variables son nombradas en el código como: `P_vah_e5`, `T_vah_e5`, `e5_T_jugo_salida`, `P_vah_e6`, `T_vah_e6`, `e6_T_jugo_salida`, `W_jugo_salida_e6`. Dicho mapa de calor, reducido a las columnas de la temperatura del vapor en los distintos efectos para facilitar su visualización, se observa en la Figura 7.16. En ella se observa que el patrón de las correlaciones cambia drásticamente a partir de las variables del cuarto efecto, otro motivo para la selección de las variables de los últimos dos efectos. Se intuye entonces que los 4 primeros efectos trabajan con más independencia del Brix final y por eso están fuertemente correlacionados, mientras que los efectos 5 y 6 hacen el ajuste final para conseguir el Brix.

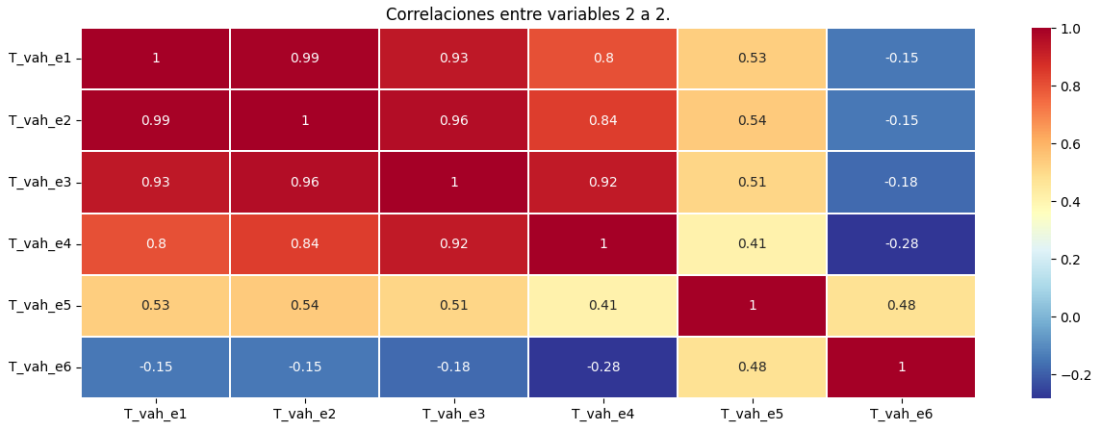

Figura 7.16: Mapa de calor de la correlación 2 a 2 de las columnas relativas a la temperatura del vapor del *dataset*.

Análisis de *outliers*

Después, se dibuja el diagrama de caja y bigotes [4] de cada una de estas columnas, donde se pueden observar los *outliers*. En este caso, se consideran valores extremos aquellos que están fuera del intervalo 7.30, siendo Q_i el i-ésimo cuartil.

$$[Q_2 - 1{,}5 \cdot (Q_3 - Q_1), Q_2 + 1{,}5 \cdot (Q_3 - Q_1)] \tag{7.30}$$

El *boxplot* de la columna `P_vah_e6` se observa en la Figura 7.17. En él, los límites de la caja indican el primer y el tercer cuartil y la línea vertical dentro de la caja es la mediana (equivalente al segundo cuartil). Los límites de los bigotes (o brazos) son los extremos del intervalo de la Ecuación 7.30 y los puntos fuera de los bigotes son los *outliers*.

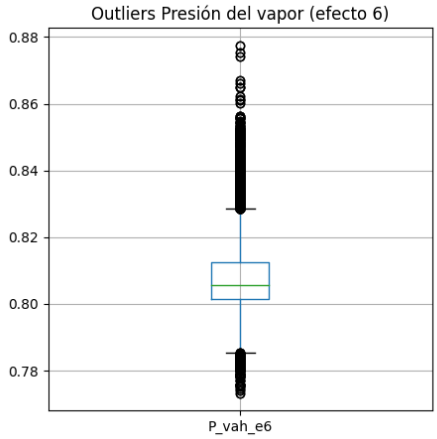

Figura 7.17: Diagrama de caja y bigotes de la columna `P_vah_e6` del *dataset*.

Estudio de la variable objetivo

Por último, se estudia la distribución de la variable objetivo mediante un diagrama de frecuencias. Dicha distribución se observa en la Figura 7.18.

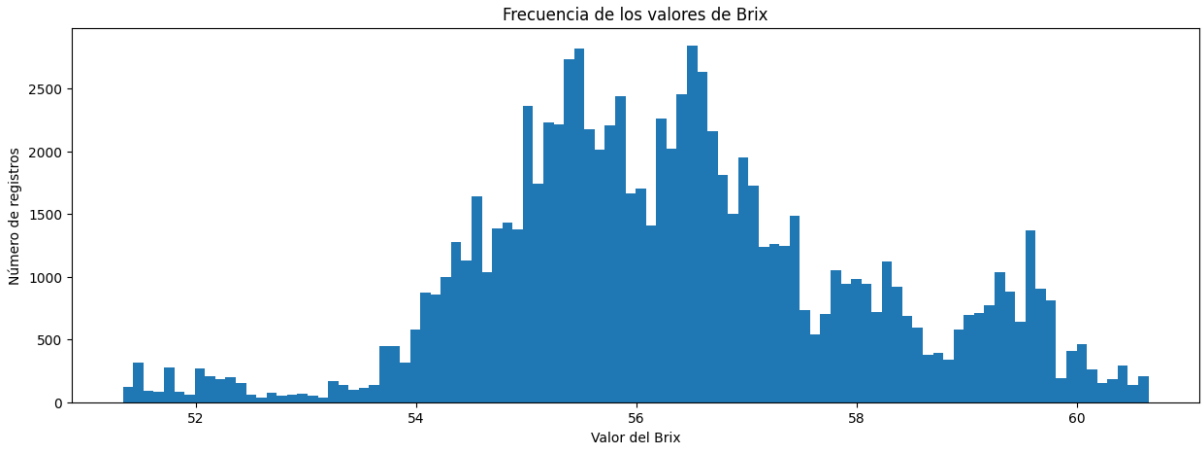

Figura 7.18: Diagrama de frecuencias de la columna `Output` del *dataset*.

También se representa temporalmente, lo que permite decidir que se tomen los primeros 80000 registros como conjunto de entrenamiento, ya que en ellos están representados todos los eventos posibles. Dicha distribución se observa en la Figura 7.19.

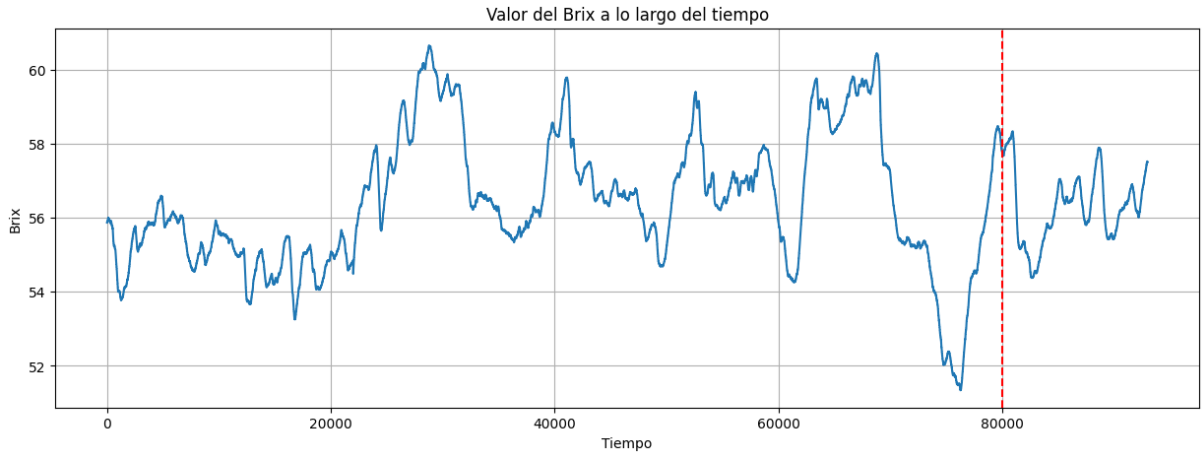

Figura 7.19: Representación temporal de la columna `Output` del *dataset*.

7.2. Preparación de los datos

En esta sección se explican los procesos de limpieza y transformación de los datos a través de las subsecciones 7.2.1 y 7.2.2, respectivamente. Se seleccionan los atributos relevantes y se eliminan los datos duplicados o innecesarios.

7.2.1. Limpieza

A pesar de que la limpieza del *dataset* es una de las tareas programadas en el diagrama de Gantt de la planificación temporal (Figura 3.2), no se lleva a cabo debido a que los datos de proceso proporcionados están en condiciones normales de operación. Se llega a esta decisión tras mostrar la representación temporal de cada columna, con los límites del intervalo de la Ecuación 7.30 representados como líneas horizontales, a un experto de la fábrica real. La representación temporal (con los límites del intervalo de confianza marcados) de la columna `P_vah_e6` se observa en la Figura 7.20.

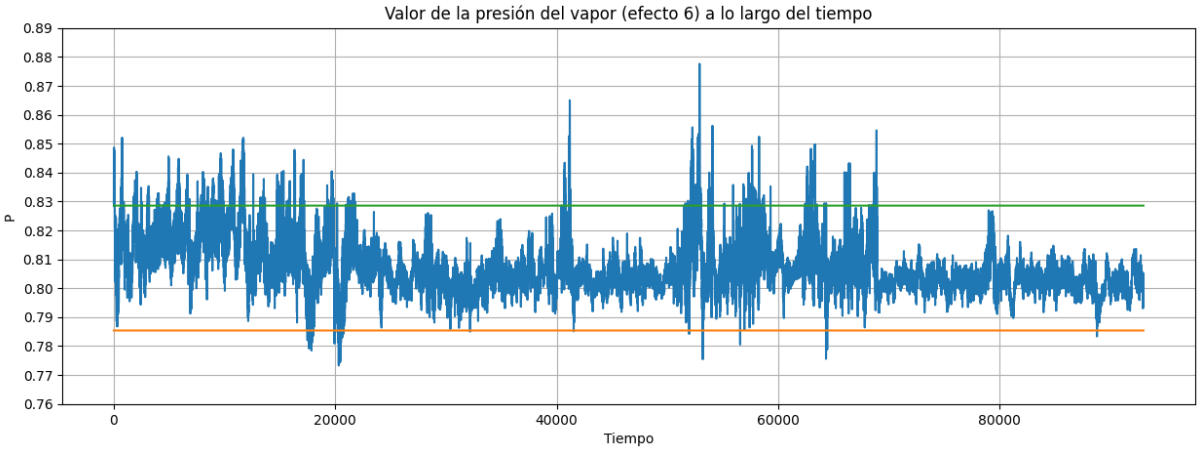

Figura 7.20: Representación temporal (con los límites del intervalo de confianza marcados) de la columna `P_vah_e6` del *dataset*.

7.2.2. Transformación

La transformación del conjunto de datos descrito en la Subsección 7.1.1 se realiza en varios pasos. En primer lugar, se toman las columnas indicadas en la Subsección 7.1.2. Luego, se realiza un filtrado de Butterworth sobre las columnas `P_vah_e6` y `e6_T_jugo_salida`. A continuación, se añade el Brix estimado mediante IM (`Output_IM`) como nueva columna del *dataset*. Después, se realiza un segundo filtrado, esta vez de media móvil, sobre cada una de las columnas del conjunto de datos a excepción del Brix de salida (`Output`), por ser el objetivo de la estimación. Una vez hecho ésto, se añade el Brix estimado mediante el modelo ARIMA (`Output_MA`) como nueva columna del *dataset*. Por último, los datos se someten a un escalado en el intervalo $(0, 1)$ o en el intervalo $(-1, 1)$, según la función de activación que se use.

Cabe mencionar que la razón por la que se aplica un filtro de media móvil a la columna `Output_IM` pero no a la columna `Output_MA` es que los datos que se toman para calcular `Output_IM` no están filtrados con media móvil y los que se utilizan para calcular `Output_MA` sí lo están. Esto es, en el primer caso se aplica el filtro después del cálculo, al contrario que para el segundo caso. Se aplican dos filtrados con el objetivo de disminuir el ruido y

conseguir estimaciones más estables con los modelos neuronales. Cabe comentar también que los datos se proporcionan con el primer filtro; el segundo es añadido en este proyecto. Esto es, el filtro de Butterworth se aplica en la fábrica, mientras que el de media móvil se usa en este proyecto por recomendación de los tutores, ya que las redes neuronales pueden funcionar mejor con variables más suavizadas.

En resumen, se utilizan las 9 variables de la Tabla 7.12 como entrada a los modelos de regresión.

Nombre			Descripción	Unidad
6	P5	P_vah_e5	Presión del vapor (evaporador 5)	bar abs
7	P6	P_vah_e6	Presión del vapor (evaporador 6)	bar abs
33	T26	T_vah_e5	Temperatura del vapor (evaporador 5)	°C
34	T27	T_vah_e6	Temperatura del vapor (evaporador 6)	°C
43	W8	W_jugo_salida_e6	Flujo másico del jugo (salida del evaporador 6)	t/h
49	T33	e5_T_jugo_salida	Temperatura del jugo (salida del evaporador 5)	°C
50	T34	e6_T_jugo_salida	Temperatura del jugo (salida del evaporador 6)	°C
52	CS2	Output_IM	Brix calculado mediante IM	g/100 g
53	CS3	Output_MA	Brix calculado mediante un modelo ARIMA	g/100 g

Tabla 7.12: Explicación de las variables o atributos de entrada a los modelos.

Filtro de Butterworth

El filtro de Butterworth [23] elimina o atenúa las frecuencias indeseadas de la señal original, dejando pasar las frecuencias deseadas dentro de la banda de paso determinada por la frecuencia de corte. Por ello, tras varias pruebas con valores diferentes de los parámetros, se crea un filtro de Butterworth de segundo orden con una frecuencia de corte de 0,003 para la columna `P_vah_e6` y uno de segundo orden con una frecuencia de corte de 0,005 para la columna `e6_T_jugo_salida` . Para ello, se usa la función `scipy.signal.butter()` de Python, la cual devuelve los coeficientes del filtro. A partir de éstos, se aplica el filtro de Butterworth a lo largo del eje 0 de la matriz de datos. Cabe mencionar que se agregan 100 muestras al principio y al final de la señal para evitar los efectos de borde en el filtrado.

Filtro de media móvil

El filtro de media móvil (también conocido como *moving average*, media rodante o *rolling mean*) [14] suaviza la señal, reduciendo el ruido y las variaciones rápidas. Por ello, tras varias pruebas con valores diferentes de los parámetros, se calcula el promedio móvil de las últimas 100 muestras de la señal (ventana deslizante de tamaño 100) con

las funciones `pandas.dataframe.rolling()` y `pandas.dataframe.mean()` de Python. Se actualiza cada columna con los valores filtrados a partir de la muestra número 99, para evitar que las primeras 99 muestras tengan valores "NaN" debido a la ventana de tamaño 100.

Modelo ARIMA

En el caso de este proyecto, para comprobar si la serie de tiempo formada por el Brix observado (columna `Output` del *dataset*) es estacionaria se realiza la prueba de ADF. La función `statsmodels.tsa.stattools.adfuller()` [57] de Python implementa ADF y devuelve estadísticas de prueba junto con valores críticos para comparar los resultados. El umbral utilizado para rechazar la hipótesis nula de no estacionariedad en caso de que el p-valor no lo supere es 0,05.

Una vez comprobada la estacionariedad de la serie, se divide el dataset en conjuntos de entrenamiento, validación y prueba y se realiza una búsqueda exhaustiva de los 3 parámetros de orden utilizando el rango $[0, 3]$ para p y q y el rango $[0, 2]$ para d. Para decidir los mejores valores se usa el conjunto de validación y la métrica MSE. Se utiliza la función s`statsmodels.api.tsa.ARIMA()` [58] de Python para calcular la salida del modelo ARIMA.

7.3. Modelado

En esta sección se explica la construcción de los modelos predictivos, la selección de los algoritmos de ML adecuados y el ajuste de los parámetros para obtener los mejores resultados mediante la Subsección 7.3.1, la Subsección 7.3.2 y la Subsección 7.3.3, respectivamente.

7.3.1. Creación de los modelos

En esta subsección se explica el desarrollo en Python de la arquitectura de cada uno de los modelos probados, sin obviar algunos detalles de la implementación de la estructura de la red.

Cabe mencionar que, aunque el objetivo inicial del trabajo era usar únicamente DL, se pensó en usar técnicas más clásicas al ver que, en otros trabajos de estimación del Brix [44], las técnicas clásicas daban también buenos resultados.

A lo largo de la memoria se van a utilizar los términos *epochs*, *batch*, *window*, *layers*, *neurons* y *features* para referirse al número de épocas, el tamaño del lote, la longitud de la secuencia, el número de capas, de neuronas y de características o variables, respectivamente. Además, se utilizan N, L, H_{in} y H_{out} para denotar el tamaño del lote, la longitud de la secuencia, el tamaño de la entrada o número de variables y el tamaño de la salida, respectivamente, por ser la notación que se utiliza en el manual de Pytorch.

SVR

Como ya se introduce en la Subsección 4.1.2 de esta memoria, SVR es una técnica de regresión basada en SVM. Para su implementación, se utiliza la clase `sklearn.svm.SVR` [55]. Entre los parámetros de entrada de esta función se hace uso de C, ϵ, *kernel*, *degree* y γ.

El parámetro C de regularización evita clasificar erróneamente cada ejemplo de entrenamiento. Para valores grandes de C como, por ejemplo, 1000, el algoritmo elige un hiperplano de margen más pequeño. Para valores pequeños de C como, por ejemplo, 0,1, el algoritmo busca un gran margen que separe el hiperplano, incluso si eso significa clasificar erróneamente algunos puntos, buscando una mayor capacidad de generalización y un menor sobreajuste.

El parámetro ϵ, ya mencionado en la Subsección 4.1.2, es la distancia desde el hiperplano hasta cada línea límite. Especifica el intervalo dentro del cual no se asocia ninguna penalización en la función de pérdida de entrenamiento.

Un núcleo o *kernel* es una función que transforma el espacio de entrada en otro en el que el problema de aprendizaje es conceptualmente más simple, linealmente separable en el mejor caso. Dicha transformación se realiza de forma implícita, sin modificar realmente los datos de entrada. De hecho, los *kernels* polinómicos aumentan la dimensionalidad del espacio, pero como la transformación es implícita no supone un problema computacional.

Existen *kernels* lineales y no lineales [56]. Dada la no linealidad del problema tratado en este proyecto, el estudio de los *kernels* se restringe a los no lineales, a saber: polinomial, en la Ecuación 7.31; Función de Base Radial o *Radial Basis Function* (RBF) en la Ecuación 7.32; y *sigmoide* en la Ecuación 7.33. En ellas, r, d y γ son parámetros específicos de cada *kernel*.

- *poly*:
$$K(X, Y) = (\gamma \cdot X^T \cdot Y + r)^d, \gamma > 0 \tag{7.31}$$
- *rbf*:
$$K(X, Y) = e^{-\frac{\|X-Y\|^2}{2\sigma^2}} = e^{-\gamma \cdot \|X-Y\|^2}, \gamma > 0 \tag{7.32}$$
- *sigmoid*:
$$K(X, Y) = tanh(\gamma \cdot X^T \cdot Y + r) \tag{7.33}$$

El parámetro *degree* es intuitivo y hace referencia al grado del *kernel* polinomial o *poly*. Se observa como d en la Ecuación 7.31.

El parámetro γ define hasta dónde llega la influencia de un sólo ejemplo de entrenamiento: lejos si el valor es bajo y cerca si el valor es alto. Esto es, con γ alto, sólo los puntos cercanos a las líneas límite se tienen en cuenta al decidir el lugar del hiperplano. Con γ bajo, los puntos que están cerca y lejos de las líneas límite se tienen en cuenta al decidir el lugar del hiperplano. En el *kernel Gaussiano* es el "radio" de las funciones *gaussianas* [14].

La motivación de este modelo era superar los resultados del otro sensor basado en SVR (Subsección 6.2.5) mediante una optimización más exhaustiva de los hiperparámetros. Cabe mencionar que el sensor de este proyecto usa un simple estudio de correlación, como el que se explica en la Subsección 7.1.2, en lugar de PCA.

XGBDT

Como ya se introduce en la Subsección 4.1.2 de esta memoria, XGBDT es una técnica de regresión avanzada, más concretamente, un *ensemble* de árboles de decisión. Para su implementación, se utiliza la clase `xgboost.XGBRegressor` [60]. XGBoost, que significa *eXtreme Gradient Boosting*, es una biblioteca de ML de GBDT distribuida y escalable. Proporciona refuerzo de árboles paralelos y es la biblioteca de ML líder para problemas de regresión y clasificación [29]. Entre los parámetros de entrada de esta función se hace uso de η, *max_depth* y *n_estimators* [55].

El parámetro η no es más que el conocido factor o tasa de aprendizaje, que permite controlar el sobreajuste mediante la contracción del tamaño del paso utilizado en la actualización de los pesos.

El parámetro *max_depth* es la profundidad máxima de los estimadores de regresión individual, es decir, de los árboles de decisión. La profundidad limita el número de nodos en el árbol. Ajustar este parámetro permite obtener un mejor rendimiento y su valor depende de la interacción de las variables de entrada.

El parámetro *n_estimators* es el número de estimadores seleccionados por detención temprana.

La motivación de este modelo era superar los resultados del otro sensor basado en un *ensemble* de DTs, el RF de la Referencia [44].

MLP

Como ya se introduce en la Subsección 4.1.4 de esta memoria, un MLP es una red en la que los elementos de proceso o neuronas artificiales se organizan por capas, no hay interconexiones entre las neuronas de una capa y cada capa tiene interconexión total únicamente con la siguiente capa. El MLP tiene, al menos, una capa oculta. La arquitectura escogida está formada por 3 capas lineales con función de activación *sigmoide*. Para su implementación, se utiliza la clase `torch.nn.Linear`, que aplica una transformación lineal de matriz $\mathbf{A}$ a los datos de entrada x, $y = x \cdot \mathbf{A}^T + b$, siendo b el vector formado por los *bias* de cada neurona e y la salida. A la salida de cada capa se utiliza la clase `torch.nn.Sigmoid`.

En este caso, el tensor 3D de entrada a la red, cuya forma es [*batch*, *window*, *features*] = [N, L, H_{in}], se aplana a lo largo de la segunda dimensión (*window*). Por tanto, la forma de la entrada es realmente [*batch*, *window* · *features*] = [N, $L \cdot H_{in}$]. Este aplanamiento de las dos últimas dimensiones en una sóla dimensión se observa en la Figura 7.21. Esto implica que el MLP construido tiene una configuración semidinámica, es decir, se entrena y se usa con el valor actual y pasado de las medidas.

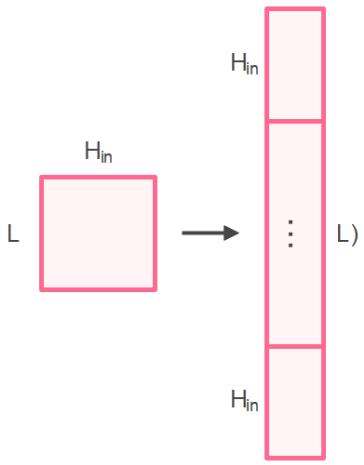

Figura 7.21: Aplanamiento de la segunda y tercera componentes del tensor de entrada al modelo MLP [Fuente: elaboración propia].

El tamaño de la salida de la capa de entrada, que coincide con el de la entrada en la segunda capa, es $[window \cdot features + features]$. El tamaño de la salida de la segunda capa, que coincide con el de la entrada en la tercera capa, es $[(window \cdot features)//2]$, siendo // el operador de la división entera. Por último, la tercera capa tiene una salida de tamaño 1, que coincide con la predicción del Brix actual. La arquitectura descrita se observa en la Figura 7.22.

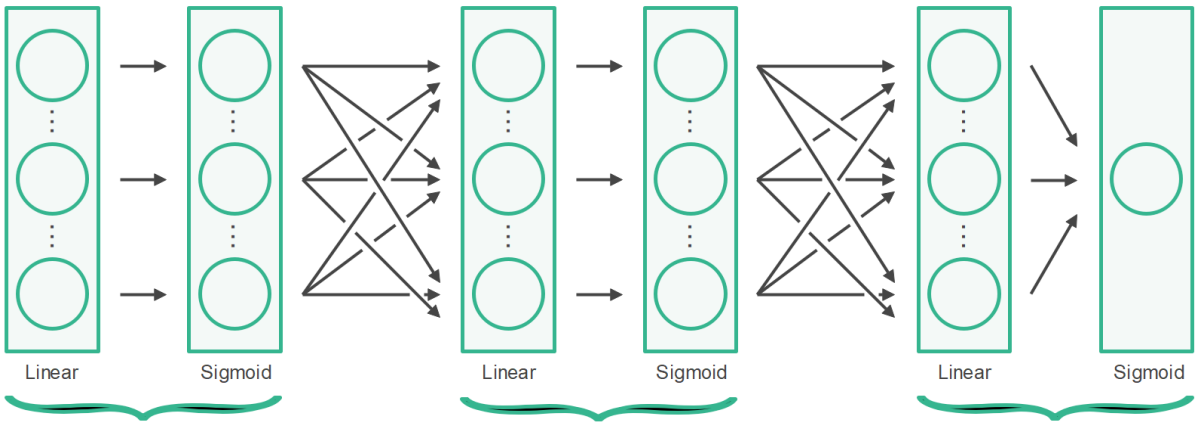

Figura 7.22: Arquitectura del modelo MLP [Fuente: elaboración propia].

LSTM

Como ya se introduce en la Subsección 4.1.4 de esta memoria, se llama LSTM a las redes neuronales recurrentes con memoria a largo plazo. Para su implementación, se utiliza la clase `torch.nn.LSTM`, que aplica una LSTM multicapa (*stacked* LSTM) a los datos

de entrada, es decir, un tensor 3D de Pytorch. Si el parámetro *batch_first* es verdadero, la forma del tensor de entrada ha de ser [*batch*, *window*, *features*] = [N, L, H_{in}]; de lo contrario, debe introducirse un tensor de la forma [*window*, *batch*, *features*] = [L, N, H_{in}]. También se pasa como parámetro el estado oculto, h, y el estado de la celda, c, que se inicializan a 0 y son tensores 3D de la forma [*layers*, *batch*, *neurons*]. El funcionamiento de la LSTM multicapa se observa en la Figura 7.23.

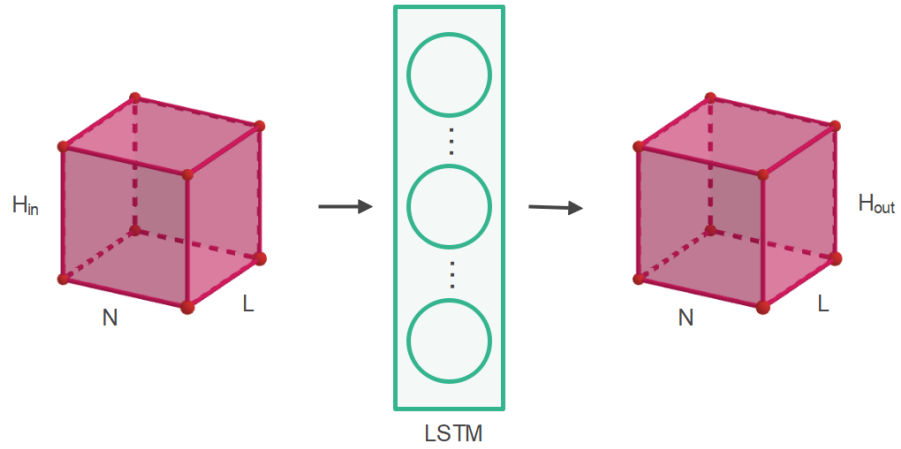

Figura 7.23: Tensores de entrada y salida al modelo LSTM [Fuente: elaboración propia].

La salida de la LSTM apilada, un tensor 3D de la forma [*batch*, *window*, *neurons*], pasa a una capa lineal con función de activación *sigmoide* (clase `torch.nn.Sigmoid`), *relu* (`torch.nn.ReLU`), tangente hiperbólica (`torch.nn.Tanh`) o *softmax* (`torch.nn.Softmax`), según corresponda. Por último, se toma el último registro (coloquialmente denominado "rodaja") de la segunda dimensión (*window*) del tensor final, que coincide con la predicción del Brix actual. La arquitectura descrita se observa en la Figura 7.24.

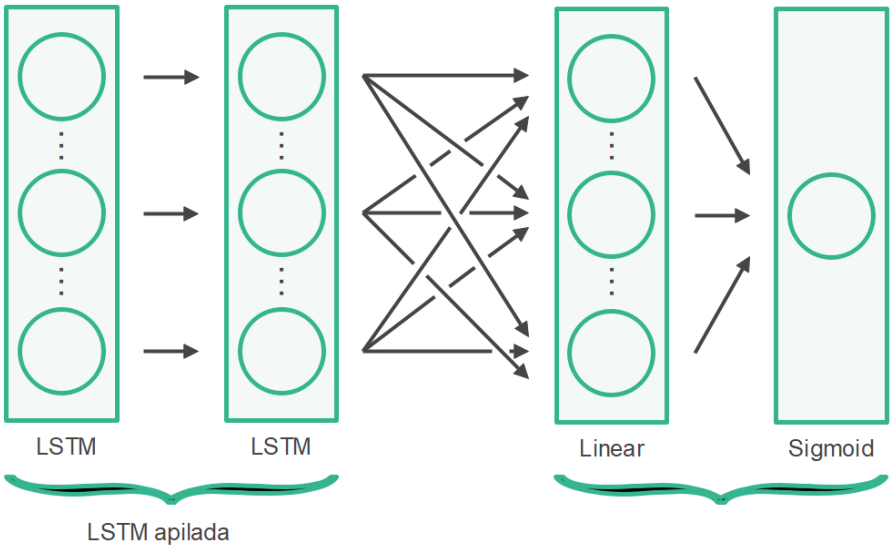

Figura 7.24: Arquitectura del modelo LSTM [Fuente: elaboración propia].

La clase `torch.nn.LSTM` tiene otros dos parámetros importantes, como son *proj_size* y *dropout*. Cuando el primero es mayor que 0, la forma del tensor de salida de la LSTM cambia a [*batch*, *window*, *proj_size*]. Las dimensiones del resto de variables, como son h y c, cambian en consecuencia. El segundo permite introducir una capa de *dropout* en la salida de cada capa de la LSTM apilada, excepto en la última capa, con una probabilidad igual a su valor.

LSTM *encoder-decoder*

Como ya se introduce en la Subsección 4.1.5 de esta memoria, el modelo ED consta de dos redes neuronales; en este caso, dos redes LSTM. La primera, o codificador (Figura 7.25), procesa una secuencia de entrada y genera un estado codificado (h y c) que resume la información de la secuencia de entrada. La segunda LSTM apilada, o decodificador (Figura 7.26), usa el estado codificado y el último registro de la primera dimensión (*window*) del tensor de entrada para producir una secuencia de salida. Cabe incidir en que la secuencia de entrada al ED tiene una longitud mayor que 1, debido a que se utiliza información del pasado, mientras que la longitud de la de salida es igual a 1, ya que se predice el Brix actual (y no a futuro).

En este caso, el codificador es una LSTM apilada con parámetro *batch_first* falso, por lo que se transponen las dos primeras dimensiones del tensor de entrada, es decir, se pasa de [*batch*, *window*, *features*] = [N, L, H_{in}] a [*window*, *batch*, *features*] = [L, N, H_{in}]. Esta transposición se observa en la Figura 7.25. El decodificador tiene la misma arquitectura que la que se explica en la Subsección 7.3.1, salvo que ya no se toma el último registro temporal del tensor final porque ya se ha reducido el tamaño a la entrada del *decoder*. Esto se observa en la Figura 7.26.

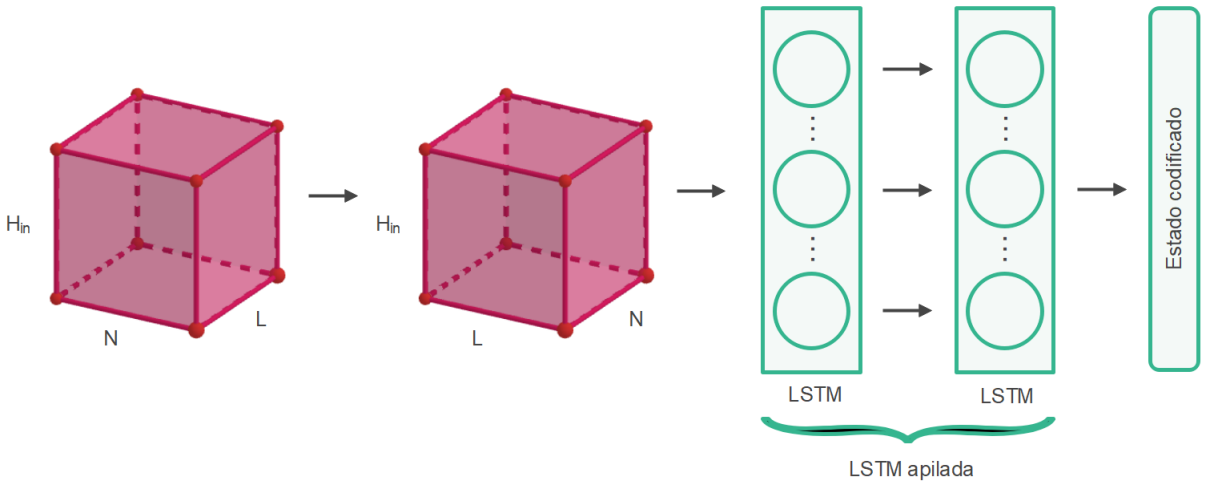

Figura 7.25: LSTM *encoder* [Fuente: elaboración propia].

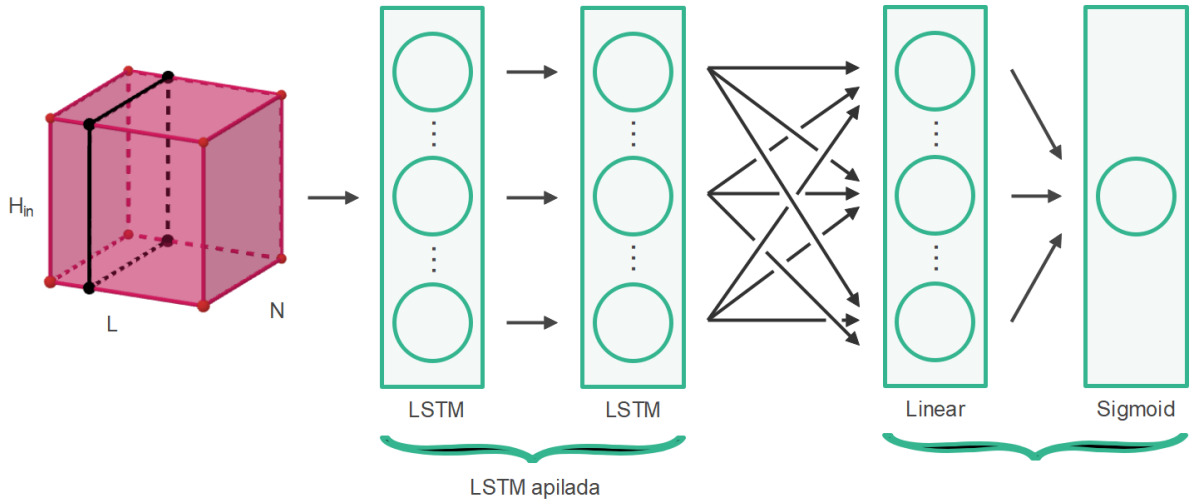

Figura 7.26: LSTM *decoder* [Fuente: elaboración propia].

LSTM *encoder-decoder* a futuro con predicción recursiva

La predicción a futuro consiste en pronosticar los valores de varias unidades de tiempo futuras dada una serie de tiempo o una secuencia de entrada.

Por su parte, la predicción recursiva durante el entrenamiento consiste en alimentar recurrentemente las entradas del decodificador un número de veces igual al número de predicciones futuras, es decir, hasta que se consigue una salida de la longitud deseada. La primera predicción del *decoder* se usa como entrada la segunda vez y así recursivamente [33], como se observa en la Figura 7.27.

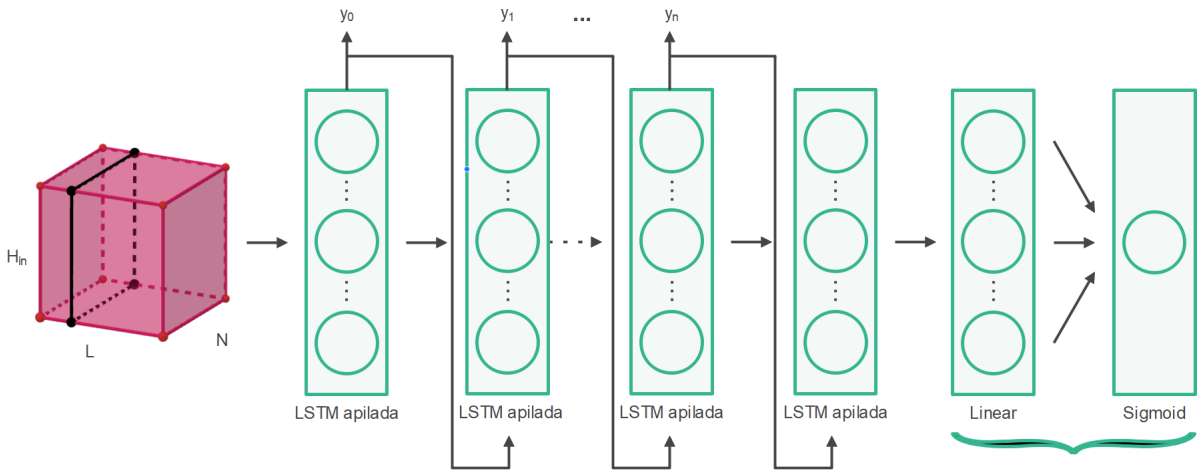

Figura 7.27: LSTM *decoder* con predicción recursiva [Fuente: elaboración propia].

Por tanto, el modelo tiene la misma arquitectura que la que se explica en la Subsección 7.3.1 a excepción de estas dos incorporaciones. Además, para proporcionar las predicciones futuras, es necesario añadir una dimensión más a la salida.

LSTM *encoder-decoder* a futuro con predicción de secuencia

A diferencia del modelo de la Subsección 7.3.1, en este caso no hay recursividad. En su lugar, la predicción a futuro se consigue pasando una estructura auxiliar de la forma $[window, batch, 1] = [L, N, 1]$ como entrada al *decoder*. Esta estructura se rellena con números naturales de forma incremental, de 0 a $L-1$, N veces, como se observa en la Figura 7.28.

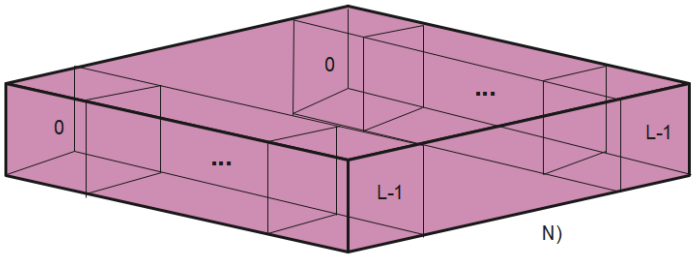

Figura 7.28: Tensor auxiliar como entrada al *decoder* [Fuente: elaboración propia].

Por tanto, Por tanto, el modelo tiene la misma arquitectura que la que se explica en la Subsección 7.3.1 a excepción de esta incorporación. El *decoder* de este modelo es el de la Figura 7.26, sustituyendo el tensor 3D de ésta por el de la Figura 7.28. Además, para proporcionar las predicciones futuras, es necesario añadir una dimensión más a la salida.

7.3.2. Selección de los modelos

En esta subsección se explica el proceso seguido hasta llegar a los modelos finalmente escogidos. Este itinerario basado en pequeñas pruebas se hace para fijar algunas arquitecturas, puesto que las posibilidades son casi infinitas.

En primer lugar, se fijan los siguientes valores para los hiperparámetros:

- 20 épocas.
- lotes de 50 registros.
- ventana deslizante de longitud 20.
- 2 capas.
- 3 neuronas por cada capa.
- MSE función de pérdida.
- optimizador Movimiento Adaptativo o *Adaptive Moment* (Adam).
- tasa de aprendizaje de 0,001.

Cabe mencionar que los registros se introducen "mezclados" (*shuffle* = *True*), pero respetando la secuencia temporal. Además, todas estas pruebas se realizan con 4 grupos distintos de columnas:

- las del efecto 6 (4 variables).
- las del efecto 6 junto a `Output_IM` (5 variables).
- las de los efectos 5 y 6 (7 variables).
- las de los efectos 5 y 6 junto a `Output_IM` (8 variables).

En primer lugar, se hacen pruebas con distintos datos, a saber:

- con filtro de Butterworth.
- con filtro de media móvil de ventana 5000 hacia atrás.
- con filtro de Butterworth y filtro de media móvil de ventana 5000 hacia atrás.
- con filtro de media móvil de ventana 100 hacia atrás.
- con filtro de Butterworth y filtro de media móvil de ventana 100 hacia atrás.

Cabe mencionar que se descartan los datos filtrados con una media móvil de 50 registros hacia atrás y 50 registros hacia adelante ya que, a la hora de aplicar el modelo en la realidad, sólo se podrían usar en el entrenamiento, pues no se dispone de datos futuros. Basta la prueba con un único modelo para descartar la media móvil de 5000. La prueba con un segundo modelo hace visibles las mejoras obtenidas al usar ambos filtros, por lo que el resto de la sección se refiere a los datos con filtro de Butterworth y filtro de media móvil de ventana 100 hacia atrás como conjunto de datos.

De esta forma, se prueban 10 modelos, a saber:

- **Modelo 1.** Una red LSTM apilada, una *sigmoide*, una lineal y otra *sigmoide*.
- **Modelo 2.** Una red LSTM apilada, una lineal y una *sigmoide*.
- **Modelo 3.** Una red LSTM apilada, una lineal y una *relu*.
- **Modelo 4.** Una red LSTM apilada, una lineal y una *tanh*.
- **Modelo 5.** Una red LSTM apilada, una lineal y una *softmax*.
- **Modelo 6.** Una red LSTM apilada y una lineal.
- **Modelo 7.** Una red LSTM apilada con parámetro $proj\_size = 1$, una lineal y una *sigmoide*.
- **Modelo 8.** Una red LSTM apilada con parámetro $proj\_size = 1$ y una *sigmoide*.
- **Modelo 9.** Una red LSTM apilada con parámetro $proj\_size = 1$ y dos *relus*.
- **Modelo 10.** Una red LSTM apilada, una lineal y dos *relus*.

Haciendo validación cruzada con 5 *folds* a partir del conjunto de datos de entrenamiento, el mejor resultado se obtiene con el Modelo 2, con un MSE de validación de 0,001 (sin desescalar). Este valor se obtiene con 8 variables. En general, en todas las pruebas se observa una mejora de los resultados a medida que se incorporan variables a las del efecto 6.

Tras esto, se modifican los valores de los hiperparámetros del Modelo 2. Estos cambios se prueban de uno en uno para poder llevar un registro más acertado, y son los siguientes:

- 50 épocas.
- lotes de 20 registros.
- ventana deslizante de longitud 100.
- 4 capas.

- 20 neuronas por cada capa.
- optimizador Movimiento Adaptativo con Decaimiento de Peso o *Adaptive Moment with Weight Decay* (AdamW).
- tasa de aprendizaje de 0,1.

También se prueba la introducción de los registros sin mezclar.

Tras observar los valores del MSE de validación, se opta por realizar los mejores cambios a la vez, esto es, se hizo una prueba con 50 épocas, lotes de 20, ventana de 20, 2 capas, 20 neuronas por capa, optimizador AdamW, tasa de aprendizaje de 0,001 y los registros "mezclados". Estos cambios producen una mejora significativa (del orden de 10^{-2}), con un MSE de validación de $2{,}443 \cdot 10^{-5}$ para la prueba con 8 variables.

Las pruebas mencionadas se realizan también sobre 2 modelos distintos de *encoder-decoder*, siendo el último modelo el elegido por sus mejores resultados:

- **Modelo 11.** Un decodificador formado por una LSTM apilada y una *sigmoide*.
- **Modelo 12.** Un decodificador formado por una LSTM apilada, una lineal y una *sigmoide*.

Con los modelos escogidos, MLP, Modelo 2 y Modelo 12, se realizan las dos últimas pruebas. La primera, para elegir la mejor función de pérdida de entre MAE, MSE, RMSE, Huber, log-cosh y cuantil [24], siendo MAE la vencedora. La segunda, para comprobar si añadir *dropout* con probabilidad 0,2 de anular neuronas mejora los resultados, obteniendo una negativa. Para el modelo MLP, se añade, con la clase `torch.nn.dropout`, una capa tras cada una de las dos primeras capas. Para el resto de modelos tan sólo se añade el parámetro *dropout* en cada LSTM existente. Para estas pruebas se calculan 4 métricas: MAE, MSE, RMSE y R^2 [2]. Además, para mayor rapidez, se sustituye la validación cruzada por una partición del conjunto de entrenamiento en 80 % *train* / 20 % *validation*.

En este punto, se decide hacer pequeñas pruebas, como las descritas anteriormente en esta subsección, con dos modelos de ML más sencillos, SVR y XGBDT, y dos modelos ED más complejos, LSTM *encoder-decoder* a futuro con predicción recursiva y LSTM *encoder-decoder* a futuro con predicción de secuencia. Tras observar que los resultados son interesantes y, en algunos casos, mejores que los obtenidos hasta el momento, se añade estos modelos a la lista.

En resumen, las arquitecturas seleccionadas para realizar sobre ellas la optimización de hiperparámetros son SVR, XGBDT, MLP, Modelo 2 o LSTM, Modelo 12 o LSTM *encoder-decoder*, LSTM *encoder-decoder* a futuro con predicción recursiva y LSTM *encoder-decoder* a futuro con predicción de secuencia.

7.3.3. Optimización de los hiperparámetros

En esta subsección se explica el proceso seguido hasta llegar a los valores "óptimos" de los hiperparámetros escogidos.

En primer lugar, conviene recordar que un hiperparámetro es un parámetro que se puede ajustar en la creación de un modelo y que debe estar bien configurado para obtener un rendimiento óptimo. Entonces, optimizar los hiperparámetros de un modelo es una tarea crucial para aumentar el rendimiento del mismo [42].

Los hiperparámetros probados para el modelo SVR se observan, junto a sus posibles valores, en la Tabla 7.13. Cabe mencionar que el parámetro *degree* depende del valor del *kernel*, afectando sólo al polinomial.

Hiperparámetro	Valores
window	[10, 20, 30]
C	[0,1; 1; 5]
epsilon	[0,1; 1; 5]
gamma	[0,1; 1; 5]
kernel	[*poly*, *rbf*, *sigmoid*]
degree	[1, 2, 3]

Tabla 7.13: Hiperparámetros de SVR.

Los hiperparámetros probados para el modelo XGBDT se observan, junto a sus posibles valores, en la Tabla 7.14.

Hiperparámetro	Valores
window	[20, 25, 30]
eta	[0,01; 0,05; 0,1]
max_depth	[10, 20, 30]
n_estimators	[200, 250, 300]

Tabla 7.14: Hiperparámetros de XGBDT.

Los hiperparámetros probados para el resto de modelos se observan, junto a sus posibles valores, en la Tabla 7.15. Cabe mencionar que los parámetros δ y θ dependen del valor de *loss_function*, afectando sólo a la función de pérdida Huber y cuantil, respectivamente.

Hiperparámetro	Valores
window	[20, 30]
batch	[25, 50]
learning_rate	[0,001; 0,01]
epochs	[20, 30]
layers	[1, 2]
neurons	[*features*, *window*]
loss_function	[MAE, MSE, RMSE, Huber, log-cosh, cuantil]
delta	[0,05; 0,2]
theta	[0,25; 0,75]

Tabla 7.15: Hiperparámetros del resto de modelos.

Por un lado, el MLP se prueba con todos los hiperparámetros excepto *layers* y *neurons*, pues el número de capas y de neuronas está fijado en este caso. Por otro lado, los modelos que predicen a futuro predicen siempre un número de valores hacia adelante igual a la longitud de la secuencia, es decir, *window*.

Se escogen valores menores y mayores a los ya probados para saber por dónde continuar la búsqueda en un futuro. El criterio de selección es el RMSE de validación por ser la métrica que mejor se adapta a este problema de regresión y a sus datos [2].

Cabe mencionar que se programa que todos los resultados se guarden en un archivo CSV. También se guardan representaciones gráficas de la función de pérdida, de las predicciones en *train* y *test* (escalados y desescalados) y el propio modelo con mejor RMSE de validación.

7.4. Resultados obtenidos y discusión

En esta sección, se presentan, explican y comparan los resultados de aplicar los diferentes enfoques.

A continuación, se observan los valores "óptimos" procedentes de la búsqueda de hiperparámetros y los tiempos de ejecución del entrenamiento con dichos valores obtenidos por cada uno de los modelos.

SVR :

window = 20; *C* = 1; *epsilon* = 0,1; *gamma* = 1; *kernel* = *poly*; *degree* = 2.

tiempo de ejecución = 1,9 segundos.

XGBDT :

window = 30; *eta* = 0,05; *max_depth* = 30; *n_estimators* = 300.

tiempo de ejecución = 8 minutos 13,2 segundos.

MLP :

window = 30; *batch* = 25; *learning_rate* = 0,001; *epochs* = 30; *loss_function* = Huber; *delta* = 0,2.

tiempo de ejecución = 11 minutos 5,5 segundos.

LSTM :

window = 30; *batch* = 25; *learning_rate* = 0,001; *epochs* = 30; *layers* = 1; *neurons* = *window*; *loss_function* = MSE.

tiempo de ejecución = 11 minutos 56,1 segundos.

LSTM *encoder-decoder* :

window = 20; *batch* = 25; *learning_rate* = 0,001; *epochs* = 30; *layers* = 2; *neurons* = *window*; *loss_function* = RMSE.

tiempo de ejecución = 15 minutos 7,7 segundos.

LSTM *encoder-decoder* a futuro (predicción recursiva) :

window = 20; *batch* = 50; *learning_rate* = 0,001; *epochs* = 30; *layers* = 2; *neurons* = *window*; *loss_function* = MSE.

tiempo de ejecución = 25 minutos 17,6 segundos.

LSTM *encoder-decoder* a futuro (predicción de secuencia) :

window = 20; *batch* = 25; *learning_rate* = 0,001; *epochs* = 30; *layers* = 1; *neurons* = *window*; *loss_function* = RMSE.

tiempo de ejecución = 6 minutos 51,7 segundos.

La Tabla 7.16 recoge el valor de las 4 métricas principales tras entrenar cada uno de los modelos seleccionados con el mejor resultado de los hiperparámetros y sobre el conjunto de validación (escalado).

Modelo	MAE	MSE	RMSE	R^2
SVR	$3{,}293 \cdot 10^{-2}$	$1{,}697 \cdot 10^{-3}$	$4{,}12 \cdot 10^{-2}$	$9{,}558 \cdot 10^{-1}$
XGBDT	$3{,}68 \cdot 10^{-4}$	$2{,}72 \cdot 10^{-7}$	$5{,}208 \cdot 10^{-4}$	1
MLP	$2{,}422 \cdot 10^{-3}$	$9{,}72 \cdot 10^{-6}$	$3{,}11 \cdot 10^{-3}$	$9{,}997 \cdot 10^{-1}$
LSTM	$1{,}089 \cdot 10^{-3}$	$3{,}19 \cdot 10^{-6}$	$1{,}777 \cdot 10^{-3}$	$9{,}999 \cdot 10^{-1}$

continúa en la página siguiente

continúa desde la página anterior

Modelo	MAE	MSE	RMSE	R^2
LSTM ED	$1{,}077 \cdot 10^{-3}$	$3{,}12 \cdot 10^{-10}$	$1{,}754 \cdot 10^{-3}$	$9{,}999 \cdot 10^{-1}$
LSTM ED a futuro (predicción recursiva)	$2{,}059 \cdot 10^{-3}$	$8{,}65 \cdot 10^{-9}$	$2{,}934 \cdot 10^{-3}$	$9{,}998 \cdot 10^{-1}$
LSTM ED a futuro (predicción de secuencia)	$2{,}142 \cdot 10^{-3}$	$8{,}9 \cdot 10^{-9}$	$2{,}978 \cdot 10^{-3}$	$9{,}998 \cdot 10^{-1}$

Tabla 7.16: Resultados sobre los datos de validación.

La Tabla 7.17 recoge el valor de las 4 métricas principales tras entrenar cada uno de los modelos seleccionados con el mejor resultado de los hiperparámetros y sobre el conjunto de *test* desescalado. Las métricas de los dos últimos modelos son especiales debido a las predicciones a futuro. Concretamente, se calculan procesando el conjunto de datos de prueba en fragmentos de tamaño *window*.

Modelo	MAE	MSE	RMSE	R^2
SVR	0,338	0,198	0,445	0,776
XGBDT	0,524	0,454	0,673	0,486
MLP	0,504	0,390	0,625	0,558
LSTM	0,461	0,370	0,608	0,581
LSTM ED	0,766	0,976	0,988	−0,104
LSTM ED a futuro (predicción recursiva)	0,492	0,390	0,625	0,558
LSTM ED a futuro (predicción de secuencia)	0,532	0,460	0,678	0,479

Tabla 7.17: Resultados sobre los datos de *test*.

Teniendo en cuenta que las 3 primeras métricas son mejores cuanto más bajo es su valor y que la última es mejor cuanto más se aproxime a 1, el mejor resultado en validación es el obtenido por el Modelo XGBDT. Sin embargo, en *test*, es mejor el SVR. Es destacable que todos los modelos mejoran todas las métricas del sensor de software basado en IM sobre el mismo conjunto de *test*:

MAE = 2,755; MSE = 13,359; RMSE = 3,655; $R^2 = -3{,}464$.

Los modelos seleccionados se pueden ordenar de mejor a peor como sigue:

1º SVR.

2º LSTM.

3º LSTM ED a futuro (predicción recursiva).

4º MLP.

5º XGBDT.

6º LSTM ED a futuro (predicción de secuencia).

7º LSTM ED.

Se puede observar un claro sobreajuste del modelo XGBDT, cuya posición en *test* dista mucho de su primer puesto en validación.

Después de analizar los resultados de todos los modelos, se puede afirmar que SVR obtiene mejores resultados que los modelos de DL. En la Figura 7.29 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo SVR con el conjunto de entrenamiento. Los datos de la imagen están escalados.

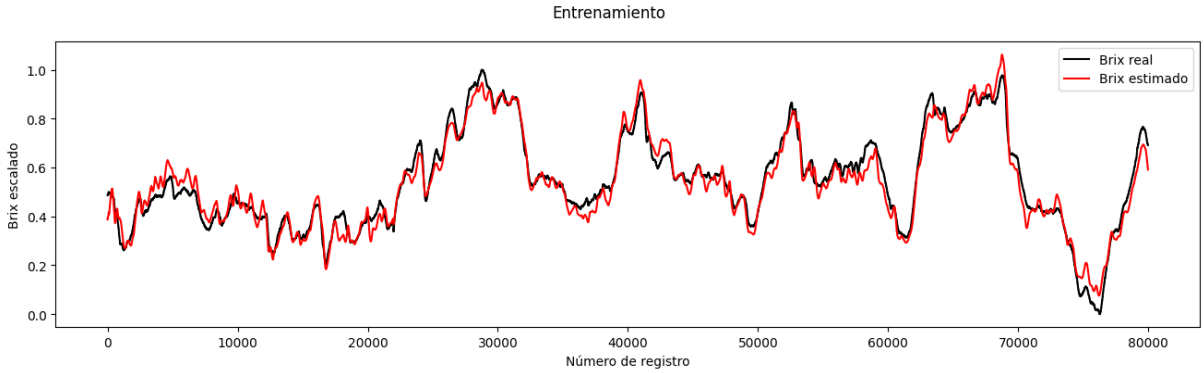

Figura 7.29: Brix real vs. Brix estimado por el modelo SVR con el conjunto de entrenamiento (datos escalados).

En la Figura 7.30 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo SVR con el conjunto de prueba. Los datos de la imagen están escalados.

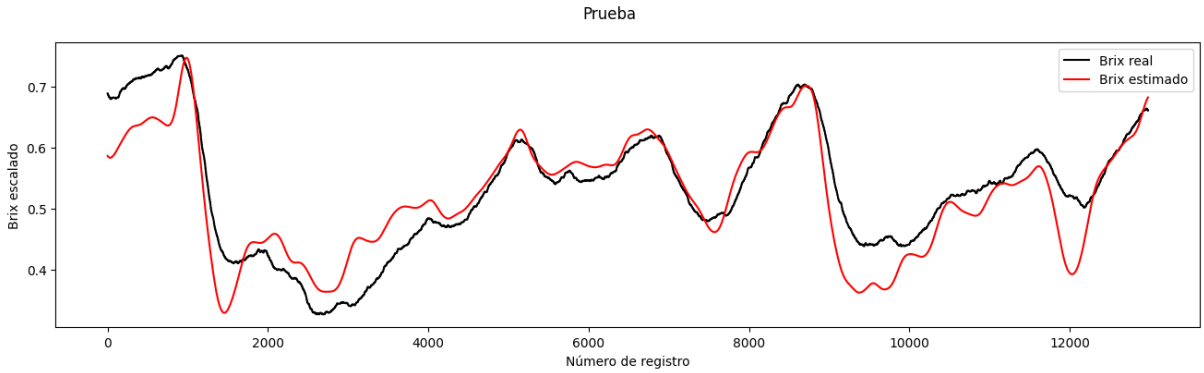

Figura 7.30: Brix real vs. Brix estimado por el modelo SVR con el conjunto de prueba (datos escalados).

En la Figura 7.31 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo SVR con el conjunto de prueba. Los datos de la imagen están en las unidades originales. En definitiva, este modelo demuestra una buena capacidad para generalizar, siendo capaz de predecir de manera precisa nuevos datos, como se puede observar en la Figura 7.31.

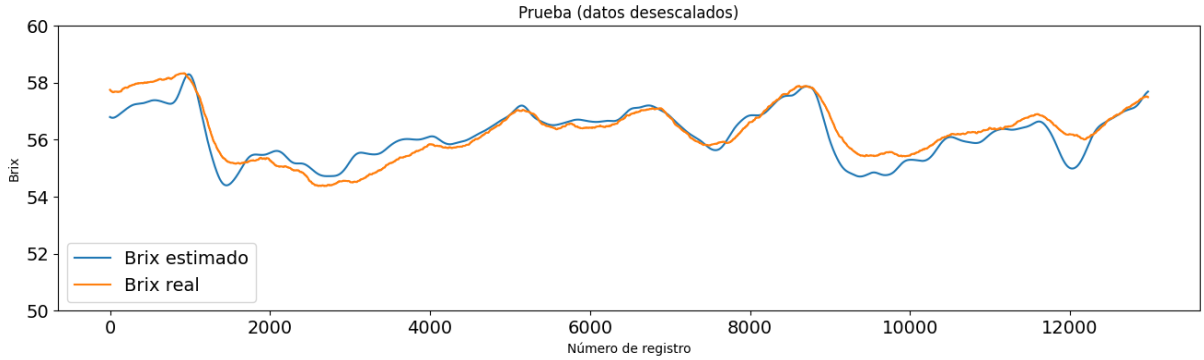

Figura 7.31: Brix real vs. Brix estimado por el modelo SVR con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.32 se puede observar una representación temporal del Brix real frente al Brix estimado por el mejor modelo del que se parte, el PLS de [44], con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

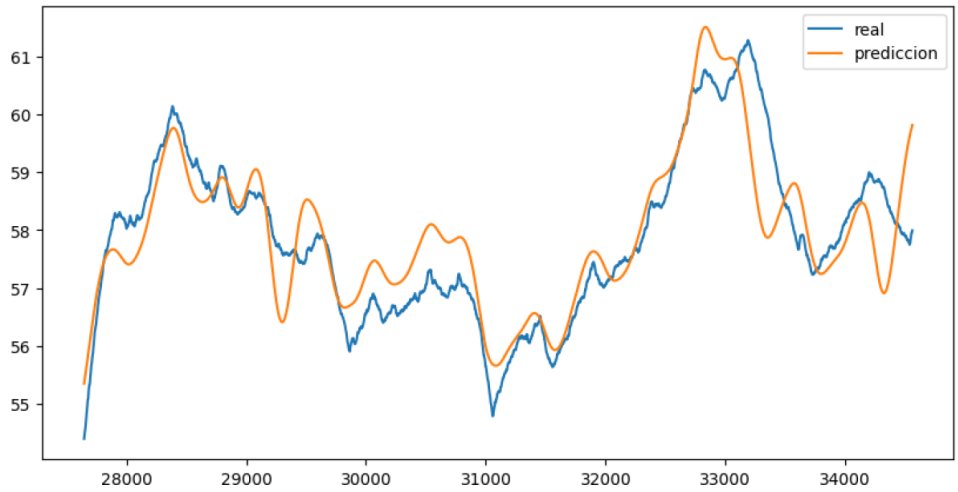

Figura 7.32: Brix real vs. Brix estimado por el modelo PLS con el conjunto de prueba (datos en las unidades originales) [44].

En la Figura 7.33 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo LSTM con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

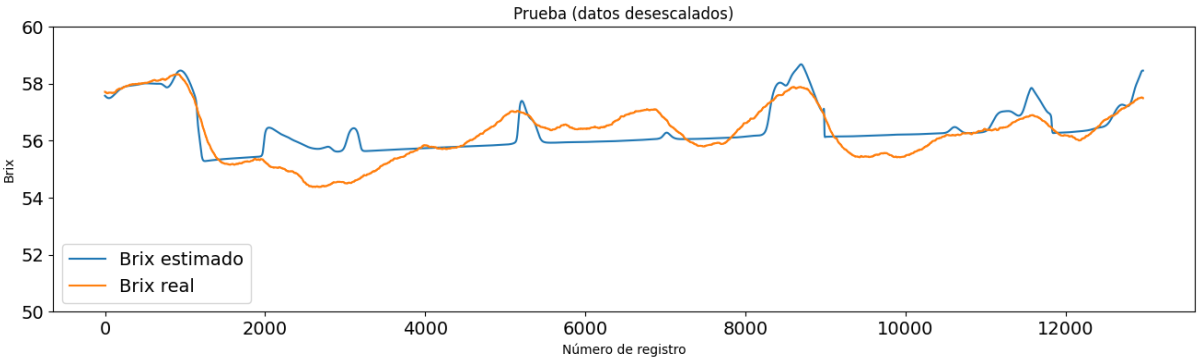

Figura 7.33: Brix real vs. Brix estimado por el modelo LSTM con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.34 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo LSTM *encoder-decoder* a futuro con predicción recursiva con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

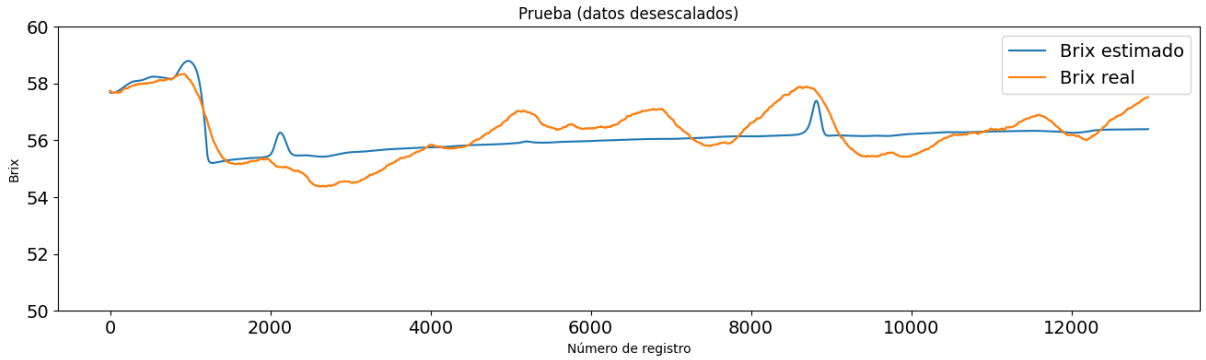

Figura 7.34: Brix real vs. Brix estimado por el modelo LSTM ED a futuro (predicción recursiva) con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.35 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo MLP con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

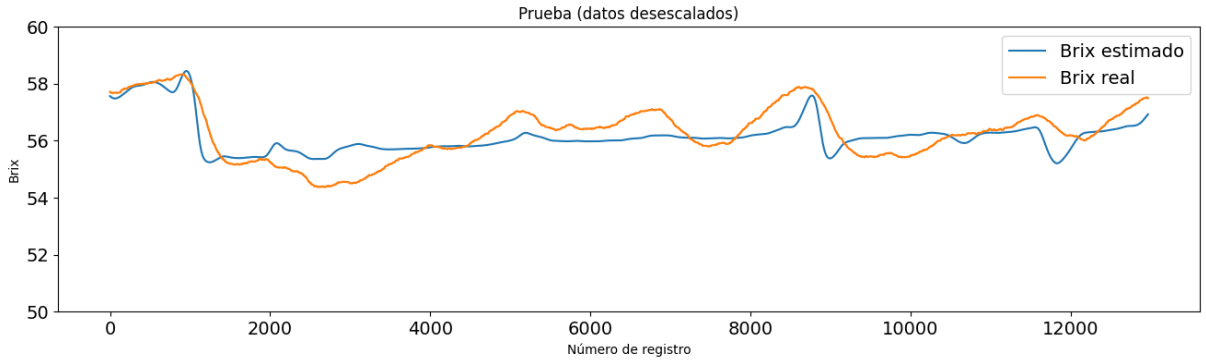

Figura 7.35: Brix real vs. Brix estimado por el modelo MLP con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.36 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo XGBDT con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

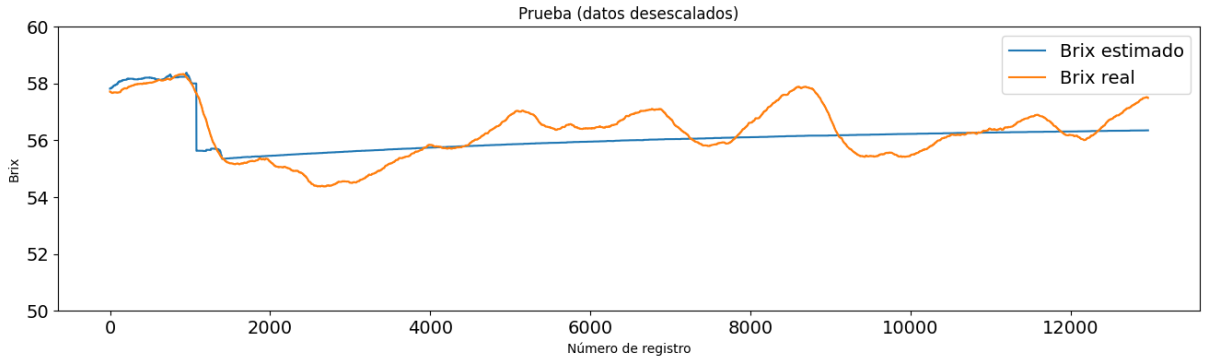

Figura 7.36: Brix real vs. Brix estimado por el modelo XGBDT con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.37 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo LSTM *encoder-decoder* a futuro con predicción de secuencia con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

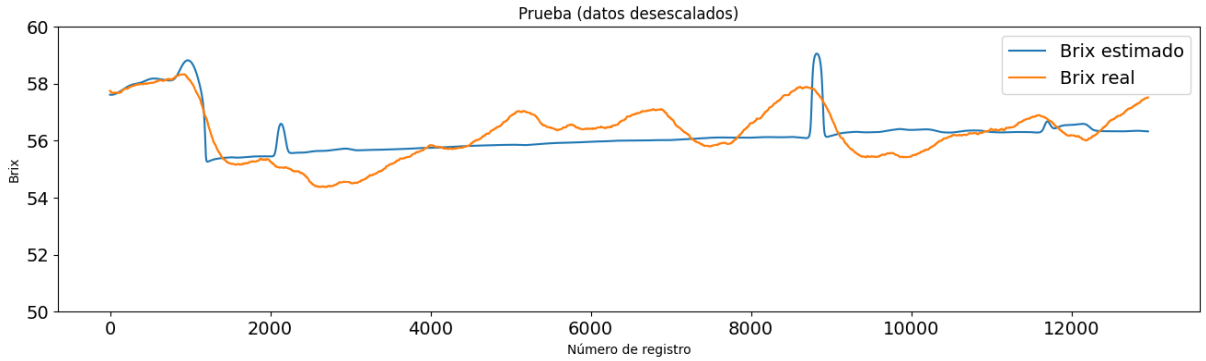

Figura 7.37: Brix real vs. Brix estimado por el modelo LSTM ED a futuro (predicción de secuencia) con el conjunto de prueba (datos en las unidades originales).

En la Figura 7.38 se puede observar una representación temporal del Brix real frente al Brix estimado por el modelo LSTM *encoder-decoder* con el conjunto de prueba. Los datos de la imagen están en las unidades originales.

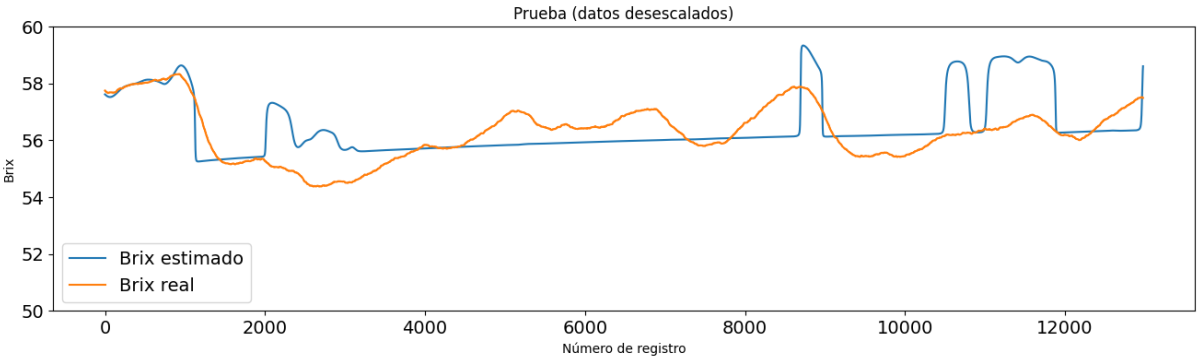

Figura 7.38: Brix real vs. Brix estimado por el modelo LSTM ED con el conjunto de prueba (datos en las unidades originales).

En definitiva, todos los modelos tienen capacidad para predecir el Brix, con diferencias que son consistentes con las métricas obtenidas, como muestran las últimas gráficas. Aunque el modelo SVR destaca sobre el resto, los demás también demuestran una buena capacidad para generalizar, siendo capaces de predecir los patrones de los datos.

Para finalizar este capítulo se resumen en la Tabla 7.18 los resultados de partida y los alcanzados. En gris están resaltados los modelos creados en este proyecto. Este repaso facilita la extracción de conclusiones.

Método	MSE
SVR	0,20
LSTM	0,37
LSTM ED a futuro (predicción recursiva)	0,39
MLP	0,39
PLS	0,45
XGBDT	0,45
LSTM ED a futuro (predicción de secuencia)	0,46
PCA + SVR	0,78
PCA + NN	0,93
LSTM ED	0,98
IM	1,36
PCA + RF	1,56

Tabla 7.18: Resultados de los trabajos relacionados y del propio proyecto.

8: Conclusiones y líneas de trabajo futuras

De este capítulo se obtienen conclusiones a partir de los resultados obtenidos (Sección 8.1), y se depositan diversas líneas de trabajo que pueden retomarse en un futuro como continuación de este proyecto (Sección 8.2).

8.1. Conclusiones

Esta sección tiene como finalidad extraer las principales conclusiones una vez analizados los resultados. Para ello, consta de dos subsecciones bien diferenciadas entre sí; una se centra en los aspectos técnicos y metodológicos del proyecto (Subsección 8.1.1) mientras que la otra se centra más en el estudiante (Subsección 8.1.2).

8.1.1. Perspectiva del proyecto

Desde un punto de vista centrado en el proyecto, se tratan tres temas por separado.

- Objetivos específicos. El objetivo marcado al principio del proyecto (Objetivo OBJ-01) se cumple satisfactoriamente. De hecho, se consigue crear hasta 4 modelos que superan el mejor de los resultados de los sensores software construidos hasta el momento por el grupo de investigación (Subobjetivo OBJ-01.01). Se han desarrollado hasta 7 modelos con técnicas de ML (Subobjetivo OBJ-01.02) utilizando la metodología CRISP-DM. Se han comparado hasta 7 nuevos sensores software (en color gris en la Tabla 7.18) con el cálculo estándar de la IM y otros sensores software desarrollados por el grupo en el pasado (en color blanco en la Tabla 7.18), con resultados satisfactorios (Subobjetivo OBJ-01.03). Cabe recordar que todos los modelos que aparecen en la Tabla 7.18 en color blanco se obtienen de [44] excepto IM, que se toma de [18]. Estos trabajos se basan en la misma azucarera, pero el *dataset* corresponde a otro período de fabricación, en condiciones de operación normal.

En cuanto a los resultados obtenidos, se extraen dos conclusiones. Por una parte, se mejoran todos los resultados de los que se parte, lo que se considera un gran avance. Por otra parte, el método más sencillo, SVR, se comporta mejor que el resto de modelos creados sobre el conjunto de datos del que se dispone en este trabajo. Se barajan tres razones por las cuales los modelos basados en las redes LSTM no superan al más simple: los datos no tienen regularidad temporal (Figura 7.19), es decir, no se aprecian patrones de los que las LSTM puedan obtener ventaja; las redes LSTM precisan una ventana temporal más grande que 20 o 30 registros para aprovechar su desempeño con dependencias a largo plazo; no se ha encontrado aún una arquitectura que se beneficie de la potencia interna de las redes LSTM para su aplicación concreta en la estimación del Brix en el proceso de evaporación de la industria azucarera.

- Utilidad para el futuro. Los distintos bloques de los cuadernos se han diseñado de forma genérica pensando en su reutilización para la configuración de nuevos modelos en base a nuevos datos (ajenos o no a la industria azucarera), tratando de facilitar y agilizar el desarrollo de nuevos sensores software basados en ML por parte de los miembros del grupo. Sin embargo, esto no significa que puedan proporcionar buenos resultados; sin un reajuste, los modelos pueden, incluso, no converger. Se trata de un tema delicado: puede que el procesamiento de los datos y los modelos creados funcionen bien con el Brix pero no con otros procesos, con dinámicas y fenómenos subyacentes que requieran técnicas totalmente distintas.

- Desarrollo del proyecto/metodología de trabajo. El uso de la metodología CRISP-DM para el desarrollo del proyecto ha permitido seguir un modelo de proceso con una serie de etapas bien definidas. Sin su flexibilidad y su carácter iterativo no hubiera sido posible identificar los defectos del proyecto y revertirlos, en consecuencia, para mejorar su calidad.

8.1.2. Perspectiva personal

Esta estancia ha sido muy gratificante y enriquecedora. He podido aprender cómo se realiza un proyecto de investigación desde cero y de qué manera se trabaja en un Grupo de Investigación Reconocido (GIR) para conseguirlo. Más aún, he podido comprobar la incertidumbre y la desesperación provocadas por los largos tiempos de cómputo, y las correspondientes esperas, de un proyecto de ML. Como anécdota relativa al esfuerzo de cómputo, se calculan aproximadamente 100000 minutos de ejecución de cuadernos de Python durante todo el proyecto, lo que equivale a unos dos meses de funcionamiento. Esto es debido a que se hicieron muchas pruebas previas a la optimización de los hiperparámetros y a que fue necesaria la repetición de varios experimentos debido a errores en el código.

Con respecto a la técnica, fue costosa la adaptación a Pytorch, ya que es una librería que trata las NNs a más bajo nivel que Keras, con la que sí que estaba familiarizada. Mientras aprendía a manejar Pytorch, pude comprobar la dificultad de los tensores 3D. Sin embargo, la tarea que me resultó más compleja fue diseñar los modelos y conseguir

que convergieran. En relación con los tensores, fue arduo encajar las dimensiones de estos con las entradas y salidas de los modelos. En especial, tuve complicaciones a la hora de construir el modelo *encoder-decoder*, lo que supuso retrasos en la planificación.

8.2. Líneas de trabajo futuras

De cara a la continuación del proyecto en un futuro y con el objetivo de mejorar los resultados obtenidos y publicarlos, se plantean las siguientes líneas de trabajo.

En primer lugar, se puede completar el estudio con otras opciones de decodificador observadas durante la revisión del estado del arte [33]. Se ha estudiado que, durante el entrenamiento, el decodificador LSTM puede hacer predicciones de, al menos, tres maneras diferentes: recursivamente, utilizando *teacher forcing* o una mezcla de las dos anteriores. Como se explica en la Subsección 7.3.1, en este proyecto se implementa la opción recursiva; la intención es probar las restantes.

En segundo lugar, se pueden estudiar otras arquitecturas y modelos, como las *augmented* RNNs [11] o lo más vanguardista en el mercado actual, los *Gated Recurrent Unit* (GRU) *Transformers*.

Por último, se pueden aplicar técnicas diferentes de optimización de hiperparámetros como el *grid*, el *random*, y la optimización bayesiana [42], que permitan una búsqueda más eficiente.

Bibliografía

[1] Albarracín, A. Gestionar bibliografía - LaTeX: redacción de documentos científicos - Guías BibUpo at Universidad Pablo de Olavide. https://guiasbib.upo.es/latex/Trabajando-con-Overleaf, 2023. Último acceso: 28/07/2023.

[2] Arrioja Landa Cosio, N. Métricas en regresión. https://medium.com/@nicolasarrioja/metricas-en-regresion-5e5d4259430b, 2021. Último acceso: 24/08/2023.

[3] Atlassian. Qué es Trello: descubre sus funciones, usos y todo lo que ofrece. https://trello.com/es/tour, 2023. Último acceso: 28/07/2023.

[4] Balderix. Diagrama de caja y bigotes. excel. https://www.probabilidadyestadistica.net/diagrama-de-caja-y-bigotes-boxplot/?utm_content=cmp-true, 2008. Último acceso: 16/08/2023.

[5] Baz Domínguez, P. Desarrollo de un sistema de generación de series temporales para propósitos de aprendizaje automático, 2022. Trabajo Fin de Grado. Grado en Ingeniería Informática de Servicios y Aplicaciones.

[6] Benites, L. ARIMA (Modelos Box-Jenkins): Media Móvil Integrada Autoregresiva en 2023 → STATOLOGOS®. https://statologos.com/arima/?utm_content=cmp-true, 2021. Último acceso: 18/06/2023.

[7] Bento, C. Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis. https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141, 2021. Último acceso: 17/08/2023.

[8] Bregón, A. Apuntes de la asignatura "sistemas inteligentes" del grado en ingeniería informática de servicios y aplicaciones de la universidad de valladolid. curso académico 21/22. tema 11: Aprendizaje - dl.

[9] Bregón, A. Apuntes de la asignatura "sistemas inteligentes" del grado en ingeniería informática de servicios y aplicaciones de la universidad de valladolid. curso académico 21/22. tema 7: Aprendizaje - introducción.

[10] Caleño Ortiz, G. P., Zapata Bermeo, J. D., and Jordan. Estudio de la polarimetría desde una perspectiva fenomenológica para la enseñanza de las ciencias. *Revista Investigación Química Vicente Garrido Capa 4*, 9 (2017), 1689–1699.

[11] Chris, O., and Shan, C. Attention and Augmented Recurrent Neural Networks. https://distill.pub/2016/augmented-rnns/, 2016. Último acceso: 11/06/2023.

[12] Coskun, O. Separation Tecniques: CHROMATOGRAPHY. *Northern Clinics of Istanbul 3*, 2 (2016), 156–160.

[13] da Costa, M. V., Fontes, C. H., Carvalho, G., and Júnior, E. C. d. M. UltraBrix: A device for measuring the soluble solids content in sugarcane. *Sustainability (Switzerland) 13*, 3 (jan 2021), 1–19.

[14] Drew Scatterday. Walking through Support Vector Regression and LSTMs with stock price prediction | by Drew Scatterday | Towards Data Science. 1–26. Último acceso: 28/08/2023.

[15] EcuRed. Brix - EcuRed. https://www.ecured.cu/Brix, 2017. Último acceso: 28/05/2023.

[16] Emerson. Caudalímetros Coriolis Micro Motion | Emerson ES. https://www.emerson.com/es-es/automation/measurement-instrumentation/flow-measurement/coriolis-flow-meters, 2022. Último acceso: 16/08/2023.

[17] Emerson. Soluciones para alimentos y bebidas La mejor medición para obtener resultados sorprendentes. Último acceso: 16/08/2023.

[18] Garcia-Alvarez, D., Merino, A., Martí, R., and Fuente, M. J. Soft sensor design for dry substance content estimation in the sugar industry. *Zuckerindustrie 137*, 10 (2012), 645–653.

[19] Gennarelli, G., Romeo, S., Scarfi, M. R., and Soldovieri, F. A microwave resonant sensor for concentration measurements of liquid solutions. *IEEE Sensors Journal 13*, 5 (2013), 1857–1864.

[20] Glassdoor. Sueldos de la empresa | Glassdoor.es. https://www.glassdoor.es/Sueldos/index.htm, 2021. Último acceso: 16/08/2023.

[21] González, L. Sobreajuste y Subajuste en Machine Learning. https://aprendeia.com/sobreajuste-y-subajuste-en-machine-learning/, 2023. Último acceso: 16/08/2023.

[22] Google. Google colab. https://research.google.com/colaboratory/intl/es/faq.html, 2021. Último acceso: 28/07/2023.

[23] Greyrat, R. Filtro de paso de banda digital Butterworth en Python – Barcelona Geeks. https://barcelonageeks.com/filtro-butterworth-de-paso-de-banda-digital-en-python/, 2022. Último acceso: 16/08/2023.

[24] Grover, P. 5 Regression Loss Functions All Machine Learners Should Know. *Medium* (2020), 1–17.

[25] Haya, P. La metodología CRISP-DM en ciencia de datos - IIC, 2021.

[26] IBM. ¿Qué es un árbol de decisión? | IBM. https://www.ibm.com/es-es/topics/decision-trees, 2022. Último acceso: 17/08/2023.

[27] IBM Cloud Education. ¿Qué son las redes neuronales? - España | IBM. https://www.ibm.com/es-es/cloud/learn/neural-networks, 2020. Último acceso: 17/08/2023.

[28] Ingenierizando. Refractómetro: qué es, partes, funcionamiento, tipos,... https://www.ingenierizando.com/laboratorio/refractometro/#Tipos-de-refractometros?utm_content=cmp-truehttps://www.ingenierizando.com/laboratorio/refractometro/, 2023. Último acceso: 16/08/2023.

[29] J. Thornton, H. What Is It and Why Does It Matter? In *The It Factor: What Makes a Teacher Great?* 2019, pp. 1–2.

[30] Jiménez C., C., and León P., D. E. Biosensores: aplicaciones y perspectivas en el control y calidad de procesos y productos alimenticios biosensors: implementation and outlook in the control and pro-cess quality and foodstuffs. *REVISTA DE LA FACULTAD DE QUÍMICA FARMACÉUTICA 16*, 1 (2009), 144–154.

[31] José Luis Chacón. Introducción a Pandas, la librería de Python para trabajar con datos. https://profile.es/blog/pandas-python/, 2021. Último acceso: 17/08/2023.

[32] LinkedIn. LinkedIn Salary: Descubre sueldos reales. Conoce hasta dónde puedes llegar | LinkedIn. https://www.linkedin.com/salary/. Último acceso: 16/08/2023.

[33] Lkulowski. GitHub - lkulowski/LSTM_encoder_decoder: Build a LSTM encoder-decoder using PyTorch to make sequence-to-sequence prediction for time series data. https://github.com/lkulowski/LSTM_encoder_decoder, 2020. Último acceso: 02/06/2023.

[34] Lorefice, S., and Malengo, A. Calibration of hydrometers. *Measurement Science and Technology 17*, 10 (2006), 2560–2566.

[35] Marti, R., Garcia-Alvarez, D., Merino, A., and Fuente, M. Diseño de un sensor soft para la estimación del Brix en la industria azucarera. In *Jornadas de Automática* (2011).

[36] Merino Gómez, A. *Libería de modelos del cuarto de remolacha de una industria azucarera para un simulador de entrenamiento de operarios*. PhD thesis, 2008.

[37] Microsoft. Introducción a Microsoft Teams - Soporte técnico de Microsoft. https://support.microsoft.com/es-es/office/introducci{ó}n-a-microsoft-teams-b98d533f-118e-4bae-bf44-3df2470c2b12, 2021. Último acceso: 28/07/2023.

[38] Microsoft. Visual studio code - code editing. redefined. https://code.visualstudio.com/, 2023. Último acceso: 15/08/2023.

[39] Mitchell, T. *Machine Learning*. McGraw-Hill, 1997.

[40] Mora, M. Análisis de Productos Agrícolas I. Tema 7 Refractometría y Polarimetría. *Análisis Químico Cualitativa, Madrid* (2015), 1–14.

[41] Moro, Q. I. Apuntes de la asignatura "deep learning y sus aplicaciones" del máster en ingeniería informática de la universidad de valladolid. curso académico 22/23. redes neuronales recurrentes (rnn).

[42] Márquez Vergara, L. A., and Naya, G. ¿Cómo hacer optimización de parámetros en Python? https://blog.escueladedatosvivos.ai/como-hacer-optimizacion-parametros-python/, 2021. Último acceso: 11/06/2023.

[43] Olah, C. Understanding lstm networks – colah's blog. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2023. Último acceso: 05/06/2023.

[44] Perrote Gómez, J. Desarrollo de un sensor software para la estimación de materia seca en un proceso industrial, 2023. Trabajo Fin de Grado. Grado en Estadística.

[45] Pulido Junquera, J. B., and Alonso González, C. J. Apuntes de la asignatura "técnicas escalables de análisis de datos en entornos big data: Clasificadores" del máster en ingeniería informática de la universidad de valladolid. curso académico 22/23. tema 5: Clasificadores avanzados en spark ml.

[46] Python Software Foundation. Welcome to Python.org. https://www.python.org/, 2016. Último acceso: 11/06/2023.

[47] Pytorch. Web de documentación de la librería Python Pytorch. https://pytorch.org/, 2016. Último acceso: 11/06/2023.

[48] Ramírez-Morales, I., Rivero, D., Fernández-Blanco, E., and Pazos, A. Optimization of NIR calibration models for multiple processes in the sugar industry. *Chemometrics and Intelligent Laboratory Systems 159*, October (2016), 45–57.

[49] Rheims, J., Köser, J., and Wriedt, T. Refractive-index measurements in the near-IR using an Abbe refractometer. *Measurement Science and Technology 8*, 6 (1997), 601–605.

[50] Ruiz, J. Desarrollo de biosensores enzimáticos miniaturizados para su aplicación en la industria alimentaria. *Barcelona España Tesis doctoral por la Universidad de Barcelona* (2006), 244.

[51] Ruiz Reina, J. L., and Martín Mateos, F. J. Apuntes de la asignatura "inteligencia artificial" del grado en ingeniería informática de la universidad de sevilla. curso académico 21/22. tema 7: Introducción a las redes neuronales.

[52] Santos, R. A., Normey-Rico, J. E., Gómez, A. M., and De Prada Moraga, C. EDUSCA (Educational SCAda): Features and applications. *IFAC Proceedings Volumes (IFAC-PapersOnline) 7*, PART 1 (2006), 614–619.

[53] Santos Pascualena, J. ¿Cuánto cuesta contratar un trabajador? - Infoautonomos. https://www.infoautonomos.com/blog/cuanto-cuesta-contratar-un-trabajador/, 2021. Último acceso: 16/08/2023.

[54] Sarkar, T. Statistical Modeling with Python: How-to & Top Libraries - Kite Blog. https://www.kite.com/blog/python/statistical-modeling-python-libraries/?msclkid=07c113aaa88a11ec8e4fd897bd84f05b, 2019. Último acceso: 17/08/2023.

[55] Scikit-learn. Web de documentación de la librería Python Scikit-learn, sklearn.ensemble.GradientBoostingRegressor — scikit-learn 1.2.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html, 2016. Último acceso: 28/08/2023.

[56] Scikit-learn. Web de documentación de la librería Python Scikit-learn, Support Vector Regression (SVR) using linear and non-linear kernels. https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html, 2016. Último acceso: 28/08/2023.

[57] Statsmodels. Web de documentación de la librería python statsmodels, statsmodels.tsa.stattools.adfuller - statsmodels 0.15.0 (+49). https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html. Último acceso: 16/08/2023.

[58] Statsmodels. Web de documentación de la librería python statsmodels, statsmodels.tsa.arima.model.arima - statsmodels 0.14.0. https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima.model.ARIMA.html, 2019. Último acceso: 16/08/2023.

[59] Vadapalli, P. Keras vs. PyTorch: Difference Between Keras PyTorch | upGrad blog. https://www.upgrad.com/blog/keras-vs-pytorch/, 2020. Último acceso: 25/07/2023.

[60] XGBoost. Web de documentación de la librería python xgboost, xgboost parameters — xgboost 1.7.3 documentation. https://xgboost.readthedocs.io/en/stable/parameter.html, 2022. Último acceso: 17/08/2023.

[61] Zach. Augmented Dickey-Fuller Test in Python (With Example). https://www.statology.org/dickey-fuller-test-python/#:$\sim$:text=ToperformanaugmentedDickey-Fullertest%2Cwecan,theadfuller%28%29functionfromthestatsmodelslibrary.https://www.statology.org/dickey-fuller-test-python/, 2021. Último acceso: 16/08/2023.

Apéndices

Apéndice A

Acrónimos

Adam	Movimiento Adaptativo o *Adaptive Moment*
AdamW	Movimiento Adaptativo con Decaimiento de Peso o *Adaptive Moment with Weight Decay*
ADF	Dickey-Fuller Aumentada o *Augmented Dickey-Fuller*
ANN	Red Neuronal Artificial o *Artificial Neural Network*
API	*Application Programming Interface*
AR	AutoRregresiva o *AutoRegressive*
ARIMA	Media Móvil Integrada Autorregresiva o *AutoRegressive Integrated Moving Average*
BD	Base de Datos o *Database*
3D	3 Dimensiones o 3 *Dimensions*
CPU	Unidad de Procesamiento Central o *Central Processing Unit*
CRISP-DM	*Cross-Industry Standard Process for Data Mining*
CSV	Valores Separados por Comas o *Comma Separated Values*
DBSCAN	Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido o *Density-Based Spatial Clustering of Applications with Noise*
DL	Aprendizaje Profundo o *Deep Learning*
DT	Árbol de Decision o *Decision Tree*
ECTS	Sistema Europeo de Transferencia y Acumulación de Créditos o *European Credit Transfer System*

ED	Codificador-Decodificador o *Encoder-Decoder*
FIC	Controlador Indicador de Flujo o *Flow Indicating Controller*
GB	*Gigabyte*
GBDT	Árbol de Decisión Potenciado por Gradiente o *Gradient Boosted Decision Tree*
GIR	Grupo de Investigación Reconocido
GRU	*Gated Recurrent Unit*
GSI	Grupo de Sistemas Inteligentes
HDD	Unidad de Disco Duro o *Hard Disk Drive*
HTML	Lenguaje de Marcas de Hipertexto o *HyperText Markup Language*
IA	Inteligencia Artificial o *Artificial Intelligence*
IDE	Entorno de Desarrollo Integrado o *Integrated Development Environment*
I+D+i	Investigación, Desarrollo e Innovación o *Research and Development*
IM	Medida Indirecta o *Indirect Measurement*
IU	Interfaz de Usuario o *User Interface*
JSON	*JavaScript Object Notation*
LPPL	LaTeX *Project Public License*
LSTM	*Long Short-Term Memory*
MA	Media Móvil o *Moving Average*
MAE	Error Absoluto Medio o *Mean Absolute Error*
MAPE	Error Porcentual Absoluto Medio o *Mean Absolute Percentage Error*
ML	Aprendizaje Automático o *Machine Learning*
MLP	Perceptrón Multicapa o *Multilayer Perceptron*
MSE	Error Cuadrático Medio o *Mean Absolute Error*
NIRS	Espectroscopia del Infrarrojo Cercano o *Near-Infrared Spectroscopy*
NN	Red Neuronal o *Neural Network*
PCA	Análisis de Componentes Principales o *Principal Component Analysis*

PLS	Mínimos Cuadrados Parciales o *Partial Least Squares*
PIC	Controlador Indicador de Presión o *Pressure Indicating Controller*
R	*Registered Trademark*
RAM	Memoria de Acceso Aleatorio o *Random Access Memory*
RBF	Función de Base Radial o *Radial Basis Function*
ReLU	Unidad Lineal Rectificada o *Rectified Linear Unit*
RF	Bosque Aleatorio o *Random Forest*
RFR	Regresión de Bosque Aleatorio o *Random Forest Regression*
RMSE	Raíz Cuadrada del Error Cuadrático Medio o *Root Mean Squared Error*
RMSLE	Raíz Cuadrada del Error Logarítmico Cuadrático Medio o *Root Mean Squared Logarithmic Error*
RRHH	Recursos Humanos
RNN	Red Neuronal Recurrente o *Recurrent Neural Network*
SCADA	Control, supervisión y adquisición de datos o *Supervisor Control And Data Adquisition*
SICUE	Sistema de Intercambio entre Centros Universitarios de España
SSD	Unidad de Estado Sólido o *Solid State Drive*
SVM	Máquina de Vectores de Soporte o *Support Vector Machine*
SVR	Regresión de Vectores de Soporte o *Support Vector Regression*
TB	*Terabyte*
TFG	Trabajo Fin de Grado
TFM	Trabajo Fin de Máster
TM	*Trademark*
TSER	Regresión Extrínseca de Series Temporales o *Time Series Extrinsic Regression*
XGBDT	Árbol de Decisión Potenciado por Gradiente Extremo o *eXtreme Gradient Boosted Decision Tree*

Apéndice B

Contenido adjunto

En este apéndice se enumera el contenido adjunto a la memoria, disponible en el siguiente enlace de GitLab:

https://gitlab.inf.uva.es/patbazd/tfm_modelosmlestimarbrix

A continuación se explica la jerarquía de directorios y ficheros entregada.

- `Datos` Carpeta que almacena los archivos relacionados con la preparación del *dataset*.
  - `NewData_without_outliers.csv` Archivo con 92998 filas y 51 columnas con datos de medidas del proceso azucarero, es decir, se trata del conjunto de datos inicial proporcionado para el proyecto. Como su nombre indica, está exento de *outliers*. Es el archivo usado para la exploración.
  - `NewData_without_outliers_filtered_areas56_IM.csv` Archivo con 92998 filas y 9 columnas con datos de medidas del proceso azucarero. Como su nombre indica, está exento de *outliers*, filtrado (con Butterworth) y consta de las variables relacionadas con los efectos 5 y 6 y la predicción del Brix mediante IM. Es el archivo usado para el filtrado de media móvil.
  - `NewData_without_outliers_filtered_butterworth_w100.csv` Archivo con 92998 filas y 9 columnas con datos de medidas del proceso azucarero. Como su nombre indica, está exento de *outliers* y filtrado con Butterworth y media móvil de ventana 100. Consta de las variables relacionadas con los efectos 5 y 6 y la predicción del Brix mediante IM. Es el archivo usado para el modelo ARIMA.
  - `exploracion.ipynb` Cuaderno de Python para el filtrado de los datos según la Subsección 7.1.2.
  - `IM.ipynb` Cuaderno de Python con el método IM proporcionado por el GIR y retocado por la alumna.
  - `filtrado.ipynb` Cuaderno de Python para el filtrado de los datos según la Subsección 7.2.2. Contiene gráficas muy interesantes de comparación entre filtrados.

  - `MA.ipynb` Cuaderno de Python con el modelo ARIMA.
  - `dataset.csv` Archivo con 92998 filas y 9 columnas que se usa como entrada a los modelos, es decir, se trata del conjunto de datos preparado (limpiado y transformado, aunque no escalado).

- `SVR` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo SVR.
  - `SVR.ipynb` Cuaderno de Python que genera el resto de archivos de la carpeta.
  - `SVR.pkl` Mejor modelo obtenido mediante validación cruzada.
  - `SVR.csv` Resultados de la validación cruzada.
  - `train.png` Representación realidad vs. predicción del Brix con los datos de *train*. Se trata de datos escalados.
  - `test.png` Representación realidad vs. predicción del Brix con los datos de *test*. Se trata de datos escalados.
  - `test_denorm.png` Representación realidad vs. predicción del Brix con los datos de *test*. Se trata de unidades originales.

- `XGBDT` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo XGBDT.
  - `XGBDT.ipynb`
  - `XGBDT.pkl`
  - `XGBDT.csv`
  - `train.png`
  - `test.png`
  - `test_denorm.png`

- `MLP` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo MLP.
  - `MLP.ipynb`
  - `MLP.pkl`
  - `MLP.csv`
  - `loss.png` Representación de la función de pérdida a lo largo de las épocas durante el entrenamiento.
  - `train.png`
  - `test.png`
  - `test_denorm.png`

- `LSTM` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo LSTM.
  - `LSTM.ipynb`
  - `LSTM.pkl`
  - `LSTM.csv`
  - `loss.png`
  - `train.png`
  - `test.png`
  - `test_denorm.png`
- `ED` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo LSTM *encoder-decoder*.
  - `ED.ipynb`
  - `ED.pkl`
  - `ED.csv`
  - `loss.png`
  - `train.png`
  - `test.png`
  - `test_denorm.png`
- `ED_futuro_recursiva` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo LSTM *encoder-decoder* con predicción recursiva.
  - `ED_futuro_recursiva.ipynb`
  - `ED_futuro_recursiva.pkl`
  - `ED_futuro_recursiva.csv`
  - `loss.png`
  - `train.png`
  - `test.png`
  - `test_denorm.png`
  - `test_denorm_futuroi.png` con i en $[0, 19]$ Representación realidad vs. i-ésima predicción (i registros a futuro) del Brix con los datos de *test*. Se trata de datos escalados.
- `ED_futuro_auxiliar` Carpeta que almacena los archivos relacionados con el entrenamiento del modelo LSTM *encoder-decoder* con predicción de secuencia.
  - `ED_futuro_auxiliar.ipynb`

- `ED_futuro_auxiliar.pkl`
- `ED_futuro_auxiliar.csv`
- `loss.png`
- `train.png`
- `test.png`
- `test_denorm.png`
- `test_denorm_futuroi.png` con i en $[0, 19]$