COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

INTERPRETABLE UNIT COMMITMENT

Autor: Daniel Elechiguerra Batlle

Director: Sara Lumbreras Sancho

Co-Director: Andrés Ramos Galán

Madrid

Agosto de 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Interpretable Unit Commitment

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020-2021 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada

de otros documentos está debidamente referenciada.

Fdo.: Daniel Elechiguerra Batlle Fecha: 27/ 08/ 2021

Elechiguerra

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Sara Lumbreras Sancho Fecha: 29/ 08/ 2021

[illegible]

EL CO-DIRECTOR DEL PROYECTO

Fdo.: Andrés Ramos Galán Fecha: 31/ 08/ 2021

[illegible]

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

INTERPRETABLE UNIT COMMITMENT

Autor: Daniel Elechiguerra Batlle

Director: Sara Lumbreras Sancho

Co-Director: Andrés Ramos Galán

Madrid

Agosto de 2021

UNIT COMMITMENT INTERPRETABLE

Autor: Elechiguerra Batlle, Daniel.
Director: Lumbreras Sancho, Sara.
Co-Director: Ramos Galán, Andrés.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto propone la aplicación de las técnicas de machine learning interpretable al problema del unit commitment con el fin no solo de predecir las soluciones óptimas sino también entender cómo se han obtenido dichos resultados. De este modo, el modelo podrá ser empleado para explicar el funcionamiento del problema.

Palabras clave: Unit commitment, machine learning interpretable, árbol de decisión, model tree, regresión lineal, clustering, operación del sistema eléctrico

1. Introducción

La planificación de la generación es una de las tareas más importantes dentro de la operación del sistema eléctrico, pues permite satisfacer la demanda al mínimo coste, en base a las previsiones de demanda y generación renovable [1].

La herramienta más extendida para abordar esta tarea es el unit commitment (UC). Se trata de un problema de optimización cuyo objetivo es planificar la conexión y la producción de las unidades del sistema en un horizonte determinado, minimizando los costes totales de operación, y respetando ciertas restricciones físicas y temporales de los generadores y las líneas de transporte, a la vez que se garantizan los requisitos de seguridad del sistema [2].

Dada su importancia, el unit commitment se ha convertido en uno de los problemas más estudiados en el sector eléctrico. En los últimos años, numerosos estudios han intentado reducir el tiempo computacional necesario para resolver este problema y mejorar su capacidad para modelar los detalles cada vez más complejos de los sistemas eléctricos, especialmente la creciente incertidumbre asociada a la aceleración de la penetración de las fuentes de energía renovables, con el fin de obtener resultados progresivamente más eficientes [3], [4].

A pesar de la gran cantidad de investigaciones relacionadas con el problema del unit commitment y, en particular, con la aplicación del machine learning (ML) a su resolución, no existen trabajos dedicados al desarrollo de modelos que permitan comprender las soluciones generadas por los modelos de UC. La interpretación de dichos resultados sigue requiriendo un profundo conocimiento del tema, dado que el algoritmo de optimización no explica de forma transparente cómo ha obtenido la solución encontrada.

2. Definición del proyecto

En consecuencia, este proyecto pretende desarrollar un conjunto de modelos basados en las técnicas de machine learning interpretable que puedan estimar los valores de las variables y variables duales de las soluciones óptimas del problema del unit commitment de una manera comprensible por el ser humano. Estos modelos serán empleados para explicar el funcionamiento de dicho problema.

3. Metodología

En primer lugar, se deben procesar los tanto los datos de entrada del UC como los resultados obtenidos para adaptarlos a las necesidades de los modelos de ML. Por un lado, las salidas del UC simplemente serán escaladas para poder trabajar con ellas de manera conjunta.

Por otro lado, debemos construir la matriz de entrada del modelo, a partir de la información proporcionada al UC. De toda la información disponible, tomaremos la demanda y la generación eólica, y con ella obtendremos la demanda neta en cada nodo con generación eólica, y la demanda neta total del sistema. De nuevo, escalaremos estas variables para homogeneizar los rangos de las mismas. Por último calcularemos los incrementos de demanda neta en los tres periodos anteriores y posteriores respecto del periodo en cuestión, para que el modelo pueda capturar posibles relaciones inter-temporales.

Con este proceso, dispondremos de una matriz de entrada de hasta 77 variables. Como estas dimensiones resultarían poco manejables o interpretables, entrenaremos un random forest para cada conjunto de variables de salida del mismo tipo, y seleccionaremos las 15 variables que este algoritmo haya considerado más importantes. Por lo tanto, cada conjunto de variables de salida empleará una matriz de entrada diferente. La Ilustración 1 muestra un ejemplo de esta selección de variables.

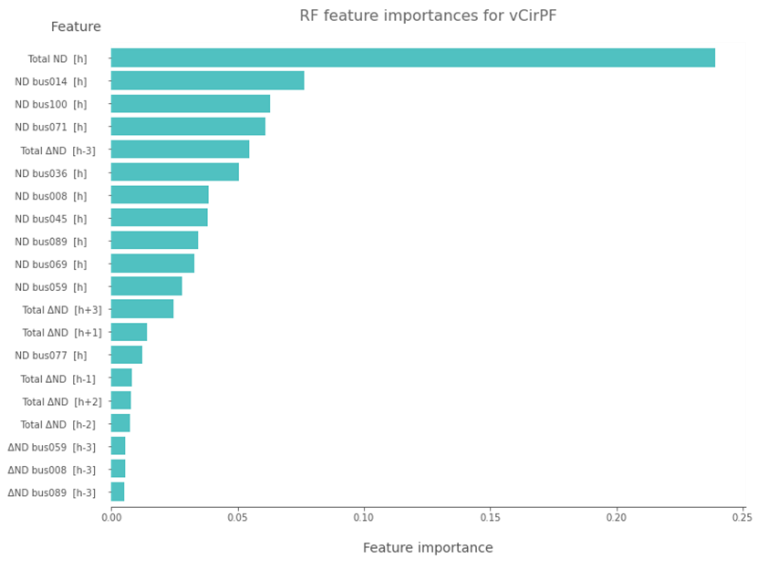

Ilustración 1 – Importancia de las variables de entrada para el conjunto de variables de salida vCirPF.

Como el problema del UC está constituido tanto por variables continuas como binarias, será necesario desarrollar dos clases de modelos diferentes, uno de regresión y uno de clasificación. Ambas clases de modelos estarán basados en árboles de decisión con salidas múltiples, y los emplearemos para predecir conjuntamente cada grupo de variables del mismo tipo. Esta predicción conjunta tiene una gran ventaja desde el punto de vista de la interpretabilidad, puesto que facilita la representación y comprensión de los resultados y permite respetar, en mayor medida, las relaciones entre las variables de salida.

El siguiente paso será la selección de los hiperparámetros óptimos para cada modelo. Como queremos garantizar la interpretabilidad de los modelos no llevaremos a cabo la optimización de los hiperparámetros de manera global, sino que encontraremos la combinación óptima de hiperparámetros para distintas profundidades del árbol de decisión aplicando una optimización bayesiana. En función de los resultados, se seleccionará la profundidad que permita obtener el balance más adecuado entre interpretabilidad y precisión. Para los modelos presentados en este proyecto, la profundidad escogida ha sido de 5 en el caso de los de regresión, y 6 para los de clasificación.

Sobre los modelos preliminares obtenidos, trataremos de ganar precisión o interpretabilidad aplicando técnicas de machine learning complementarias en sus nodos terminales.

En cuanto a modelos de regresión, entrenaremos regresiones lineales de una única variable en cada uno de los nodos terminales. Esto permitirá al árbol capturar las relaciones lineales que puedan existir en las particiones resultantes del espacio de variables, cosa que no puede lograr un simple árbol de decisión. Este es un método similar al conocido como *model tree* (Ilustración 2), pero se ha simplificado su implementación debido a la elevada carga computacional que supondría su implementación completa. Adicionalmente, la variable escogida para estas regresiones no será una común, sino que cada nodo empleará la que arroje los mejores resultados.

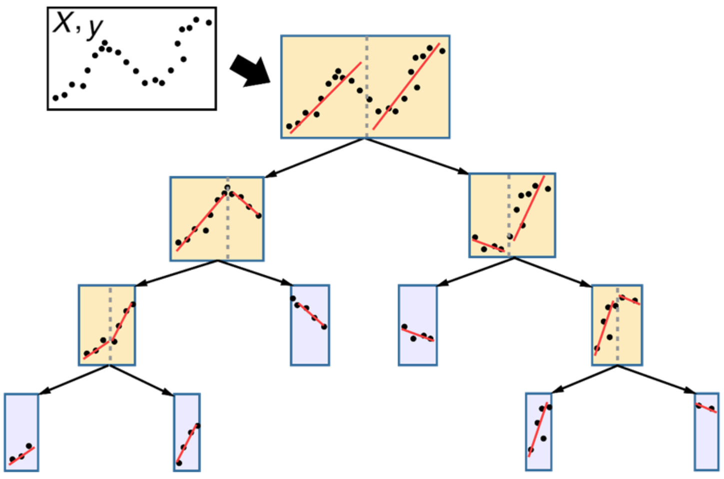

Ilustración 2 – Model tree basado en regresiones lineales [5].

Por el contrario, para los modelos de clasificación trataremos de mejorar su interpretabilidad, lo que nos permitirá partir de árboles más profundos que para los modelos de regresión. Un problema habitual de los árboles de decisión aplicados a tareas de clasificación es la duplicidad de nodos y ramas. Por ello, llevaremos a cabo un clustering (usando el algoritmo de K-modes) de los nodos terminales para reducir el número de parámetros únicos del modelo, lo cual facilitará sustancialmente su interpretación. Los nodos se identificarán con etiquetas correspondientes a cada cluster, y las salidas se representarán en una tabla adjunta. El mismo proceso se aplicará a las variables de salida dado que, en muchas ocasiones, existe una gran correlación entre estas.

4. Resultados

Una vez implementados los modelos descritos, se pueden entrenar empleando el conjunto de datos de entrenamiento y predecir los resultados del conjunto de evaluación, con los cuales se analizará su desempeño. Estos resultados se compararán, además, con los obtenidos mediante otros métodos, para aportar una mejor perspectiva de cómo de buenos son los resultados obtenidos.

En la Tabla 1 se muestran los resultados obtenidos por el model tree para tres variables del problema. En general, el modelo implementado no solo es el más interpretable de todos, sino que también logra los menores errores después del gradient boosting decision tree (GBDT), el cual se ha empleado como referencia del mínimo error posible. Por lo tanto, los resultados obtenidos son muy positivos.

La única única excepción la encontramos en el caso de los flujos de potencia por las líneas (vCirPF), debido a la complejidad y dimensión de estas variables. En este caso, el modelo no logra mejores resultados que una regresión lineal, ni se acerca al desempeño del GBDT.

Tabla 1: Desempeño del model tree comparado con otros métodos.

	vProduct1		**vCirPF**		**eMinOutput**	
Variable	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**
Worst case	0.1353	0.1588	0.1269	0.0887	0.0910	0.0091
Linear regression	0.0720	0.0798	0.0645	0.0518	0.0628	0.0033
Decision tree (5)	0.0541	0.0659	0.0928	0.0686	0.0364	0.0021
Model tree (5)	0.0496	0.0641	0.0732	0.0587	0.0327	0.0017
GBDT	0.0401	0.0508	0.0510	0.0385	0.0295	0.0022

Adicionalmente, la Figura 3 representa la relación en uno de los nodos del modelo entre la variable escogida para la regresión lineal y una de las variables de salida. En ella, se puede apreciar la ventaja del método empleado frente a un simple árbol de decisión, que únicamente puede predecir un valor constante.

- 5 -

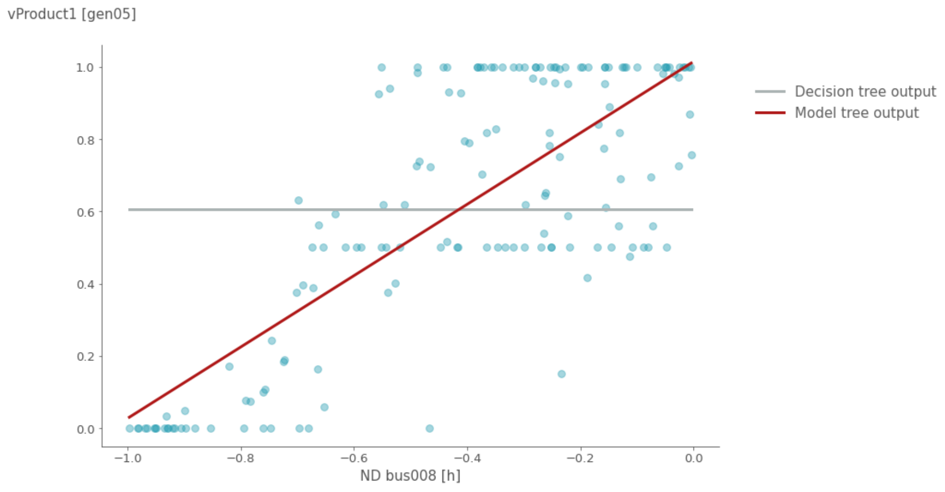

Figura 3: Representación de la relación entre vProduct1 [gen05] y ND bus008 [h] en el nodo 5 del model tree.

Por último, los resultados obtenidos por el modelo de clasificación se muestran en la Tabla 2. En este caso, la precisión del modelo es, evidentemente, inferior al del árbol de clasificación original. No obstante, la pérdida de precisión es completamente despreciable, especialmente en comparación con la interpretabilidad lograda. Además, los resultados vuelven a ser comparables con los obtenidos por el GBDT, y mejores que los del resto de métodos. En consecuencia, el modelo cumple con los objetivos previstos.

Tabla 2: Desempeño del modelo de clasificación comparado con otros métodos.

	vCommit		**eMinOutput**	
Variable	**mean accuracy**	**std accuracy**	**mean accuracy**	**std accuracy**
Worst case	0.9102	0.1249	0.6376	0.0963
Logistic regression	0.9662	0.0549	0.8927	0.0434
Decision tree (6)	0.9677	0.0470	0.9219	0.0227
Clustered DT (6)	0.9673	0.0481	0.9207	0.0242
GBDT	0.9768	0.0334	0.9466	0.0176

5. Conclusiones

Los resultados obtenidos al aplicar ambos modelos al problema del UC definido en este proyecto muestran que, en general, logran un equilibrio óptimo entre desempeño e interpretabilidad, superando habitualmente en ambos aspectos al resto de algoritmos intrínsecamente interpretables. De hecho, la precisión de estos modelos no se aleja excesivamente de la obtenida utilizando algoritmos más complejos y no interpretables. En consecuencia, los modelos implementados cumplen satisfactoriamente los objetivos que se han definido al inicio del proyecto.

Estas conclusiones se pueden aplicar a la mayoría de las variables del problema del UC, especialmente las relacionadas con los generadores térmicos. Sin embargo, los resultados obtenidos para los flujos de potencia a través de las líneas eléctricas sugieren que estos modelos no son lo suficientemente flexibles para modelar un conjunto de variables tan amplio y complicado.

No obstante, esta limitación no resta relevancia a los resultados obtenidos y a la idoneidad de los modelos desarrollados para estimar el resto de las variables de salida, las cuales son, además, las más importantes del problema.

Finalmente, aunque los modelos presentados permiten comprender mejor el funcionamiento del UC y explicar cómo se obtienen las soluciones óptimas, su desempeño no permite considerarlos como modelos completamente equivalentes al propio problema de optimización en sí, lo cual sucede incluso con los modelos más complejos y precisos desarrollados hasta el momento. Por lo tanto, los modelos interpretables desarrollados deben entenderse como herramientas complementarias al problema del unit commitment.

6. Referencias

[1] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem," *IEEE Transactions on Power Systems,* vol. 28, no. 1, p. 52–63, 2013.

[2] D. A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos, "Which Unit-Commitment Formulation is Best? A Comparison Framework," *IEEE Transactions on Power Systems,* vol. 35, no. 4, pp. 2926-2936, 2020.

[3] M. Silbernagl, M. Huber, and R. Brandenberg, "Improving Accuracy and Efficiency of Start-Up Cost Formulations in MIP Unit Commitment by Modeling Power Plant Temperatures," *IEEE Transactions on Power Systems,* vol. 31, no. 4, pp. 2578-2586, 2016.

[4] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra, "Large-scale Unit Commitment under uncertainty," *4OR,* vol. 13, no. 2, p. 115–171, 2015.

[5] A. Wong, "Building Model Trees - GitHub," 2020. [Online]. Available: https://github.com/ankonzoid/LearningX/tree/master/advanced_ML/model_tree.

INTERPRETABLE UNIT COMMITMENT

Author: Elechiguerra Batlle, Daniel.
Supervisor: Lumbreras Sancho, Sara.
Co-Supervisor: Ramos Galán, Andrés.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project proposes the application of interpretable machine learning techniques to the unit commitment problem in order not only to predict the optimal solutions but also to understand how these results have been obtained. In this way, the model can be used to explain how the unit commitment works.

Keywords: Unit commitment, interpretable machine learning, decision tree, model tree, linear regression, clustering, power system operation

1. Introduction

Generation planning is one of the most important tasks within power system operation, which allows meeting the demand at minimum cost, based on demand and renewable generation forecasts [1].

The most widespread tool to address this task is the unit commitment (UC), which is an optimization problem whose objective is to plan the commitment and production of the system units over a given horizon, minimizing the total operating costs, and respecting certain physical and temporal constraints from generators and transmission lines while guaranteeing system security requirements [2].

Given its importance, the unit commitment has become one of the most studied problems in the electricity sector. In recent years, numerous studies have tried to reduce the computational time required to solve this problem and to improve its ability to model the increasingly complex details of power systems, especially the growing uncertainty associated with the accelerating penetration of renewable energy sources, in order to obtain progressively more efficient results [3], [4].

Despite the large amount of research related to the unit commitment problem and, particularly, the application of machine learning (ML) to its resolution, there are no works dedicated to developing models that can be used to understand the solutions generated by UC models. The interpretation of such results still requires a deep knowledge of the topic since the optimization algorithm does not transparently explain how it obtained the solution found.

2. Project definition

Consequently, this project aims to develop a set of models based on interpretable machine learning techniques that can estimate the values of the variables and dual

variables of the optimal solutions of the unit commitment problem in a human-understandable way. These models will be used to explain how the unit commitment problem works.

3. Methodology

First, both the UC input data and the generated outputs must be processed to adapt them to the needs of the ML models. On the one hand, the UC outputs will simply be scaled in order to be able to work with them jointly.

On the other hand, we must build the input matrix of the model from the information provided to the UC. From all the available information, we will take the demand and wind generation and compute the net demand at each node with intermittent generation and the total net demand of the system. Again, we will scale these variables to homogenize their ranges. Finally, we will compute the net demand increases in the three periods before and after compared to the corresponding period so that the model can capture possible inter-temporal relationships.

After this process, we will have an input matrix of up to 77 variables. As these dimensions would be unmanageable and hardly interpretable, we will train a random forest for each set of output variables of the same type and select the 15 variables that the algorithm deems more important to predict the outputs. Therefore, each set of output variables will use a different input matrix. Figure 1 shows an example of this variable selection.

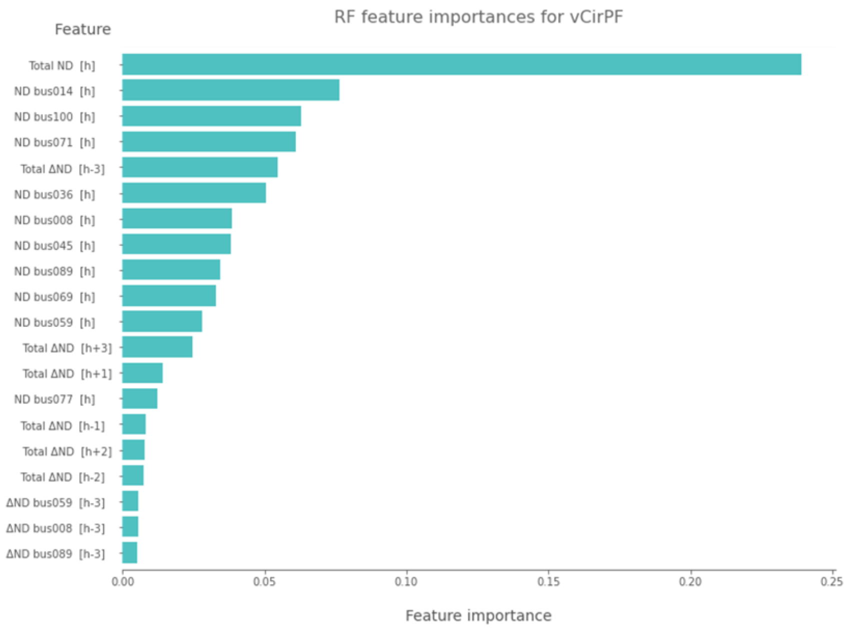

Figure 1 – Feature importances for the vCirPF output variable.

Since the UC problem consists of both continuous and binary variables, it will be necessary to develop two different classes of models, one for regression and another for classification tasks. Both classes will be based on multi-output decision trees, and we will use them to predict each group of variables of the same type together. This joint prediction has a great advantage from the interpretability point of view since it facilitates the representation and understanding of the results and allows us to respect, to a greater extent, the relationships between output variables.

The next step will be the selection of the optimal hyperparameters for each model. As we want to guarantee the interpretability of the models, we will not carry out the optimization of the hyperparameters globally. Instead, we will find the optimal combination of hyperparameters for different depths of the decision tree by applying a Bayesian optimization. Based on the results, we will select the depth that allows us to obtain the most appropriate balance between interpretability and accuracy. The depth chosen for the models presented in this project was 5 and 6 for the regression and classification models, respectively.

We will try to gain accuracy or interpretability on the preliminary models obtained by applying complementary machine learning techniques on their terminal nodes.

As for regression models, we will train single-variable linear regressions on each of the terminal nodes. This will allow the tree to capture the linear relationships in the resulting partitions of the feature space, something that a simple decision tree cannot achieve. This method is similar to the one known as a model tree (Figure 2), but its implementation has been simplified due to the high computational burden that its full implementation would entail. In addition, the variable chosen for these regressions will not be a unique one, but each node will use the one that yields the best results.

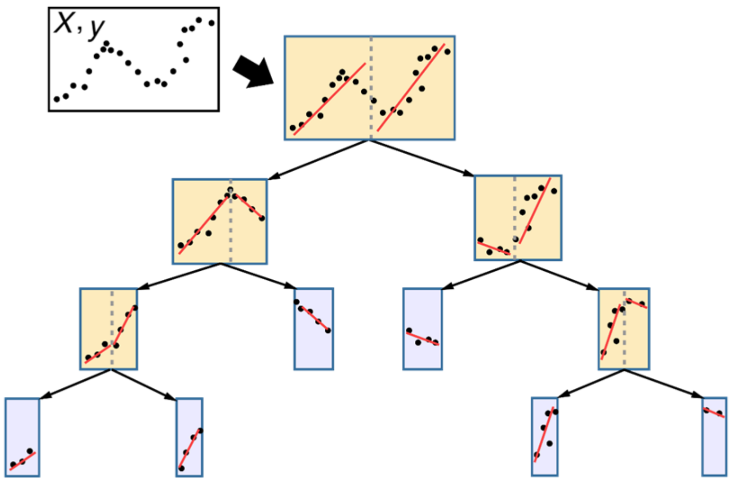

Figure 2 – Linear regression model tree [5].

Concerning classification models, we will try to improve their interpretability, allowing us to start from deeper trees than for regression. A common problem with decision tree classifiers is the duplication of nodes and branches. Therefore, we will perform a clustering (using the K-modes algorithm) of the terminal nodes to reduce the number of unique parameters of the model, which will substantially facilitate its interpretation. The nodes will be identified with labels corresponding to each cluster, and the outputs will be represented in an attached table. The same process will be applied to the output variables since, in many cases, there is a high correlation between them.

7. Results

Once the models described above have been implemented, they can be trained using the training set, and the results of the evaluation set can be predicted, with which the performance will be analyzed. These results will also be compared with those obtained by other methods to better understand how good the results obtained are.

Table 1 shows the results obtained by the model tree for three variables of the problem. In general, the implemented model is not only the most interpretable of all but also achieves the lowest errors behind the gradient boosting decision tree (GBDT), which has been used as a reference for the minimum possible error. Therefore, the results obtained are very positive.

The only exception is the group of variables corresponding to the flows through power lines (vCirPF) due to the dimension and complexity of these variables. In this case, the model does not achieve better results than linear regression, nor does it approach the performance of the GBDT.

Table 1: Performance of the model tree compared to other approaches.

	vProduct1		**vCirPF**		**eMinOutput**	
Variable	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**
Worst case	0.1353	0.1588	0.1269	0.0887	0.0910	0.0091
Linear regression	0.0720	0.0798	0.0645	0.0518	0.0628	0.0033
Decision tree (5)	0.0541	0.0659	0.0928	0.0686	0.0364	0.0021
Model tree (5)	0.0496	0.0641	0.0732	0.0587	0.0327	0.0017
GBDT	0.0401	0.0508	0.0510	0.0385	0.0295	0.0022

In addition, Figure 3 represents the relationship in one of the terminal nodes of the model between the variable chosen for the linear regression and one of the output variables. It clearly illustrates the advantage of the employed method over a simple decision tree, which can only predict a constant value.

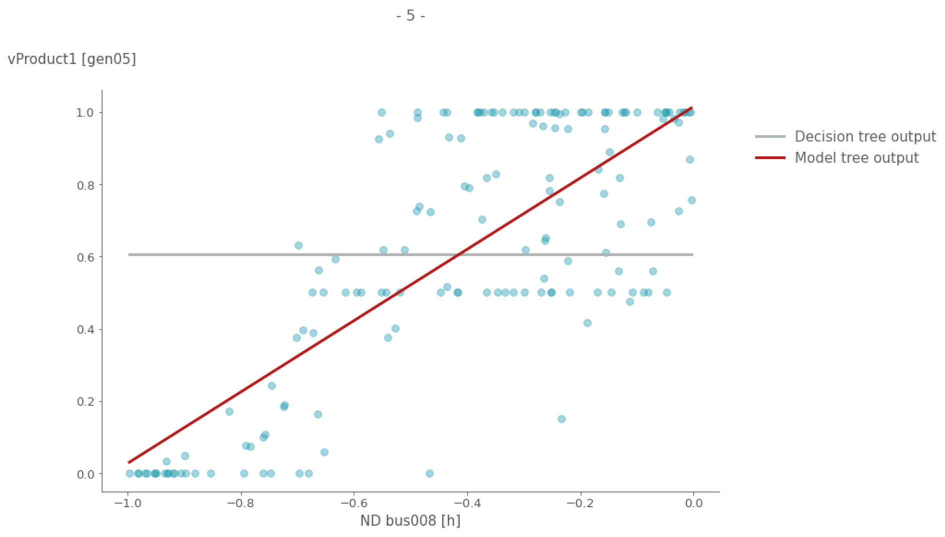

Figure 3: Representation of the relationship between vProduct1 [gen05] and ND bus008 [h] in node 5 of the model tree.

Lastly, the results obtained by the classification model are shown in Table 2. In this case, the model's accuracy is obviously lower than that of the original classification tree. However, the loss of accuracy is completely negligible, especially in comparison with the interpretability achieved. Moreover, the results are again comparable with those obtained by the GBDT and better than those of the other methods. Consequently, the model meets the expected objectives.

Table 2: Performance of the clustered decision tree classifier compared to other approaches.

	vCommit		**eMinOutput**	
Variable	**mean accuracy**	**std accuracy**	**mean accuracy**	**std accuracy**
Worst case	0.9102	0.1249	0.6376	0.0963
Logistic regression	0.9662	0.0549	0.8927	0.0434
Decision tree (6)	0.9677	0.0470	0.9219	0.0227
Clustered DT (6)	0.9673	0.0481	0.9207	0.0242
GBDT	0.9768	0.0334	0.9466	0.0176

4. Conclusions

The results obtained by applying both models to the UC problem defined in this project show that, in general, they achieve an optimal balance between performance and interpretability, usually outperforming the rest of the intrinsically interpretable algorithms in both aspects. In fact, the accuracy of these models does not depart excessively from that obtained using more complex and non-interpretable algorithms. Consequently, the implemented models satisfactorily meet the objectives defined at the beginning of the project.

These conclusions can be applied to most of the variables of the UC problem, especially those related to the thermal generators. However, the results obtained for the flow through the power lines suggest that these models are not flexible enough to model such a large and complicated set of variables.

Nevertheless, this limitation does not detract from the relevance of the results obtained and the suitability of the models developed to estimate the rest of the output variables, which are, besides, the most important variables of the problem.

Finally, even though the models presented allow a better understanding of the functioning of the UC and explain how the optimal solutions are obtained, their performance does not allow us to consider them as models completely equivalent to the optimization problem itself, which is the case even with the most complex and accurate models developed so far. Therefore, the implemented interpretable models should be understood as complementary tools to the unit commitment problem.

5. References

[1] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem," *IEEE Transactions on Power Systems*, vol. 28, no. 1, p. 52–63, 2013.

[2] D. A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos, "Which Unit-Commitment Formulation is Best? A Comparison Framework," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 2926-2936, 2020.

[3] M. Silbernagl, M. Huber, and R. Brandenberg, "Improving Accuracy and Efficiency of Start-Up Cost Formulations in MIP Unit Commitment by Modeling Power Plant Temperatures," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2578-2586, 2016.

[4] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra, "Large-scale Unit Commitment under uncertainty," *4OR*, vol. 13, no. 2, p. 115–171, 2015.

[5] A. Wong, "Building Model Trees - GitHub," 2020. [Online]. Available: https://github.com/ankonzoid/LearningX/tree/master/advanced_ML/model_tree.

Table of Contents

List of Figures

List of Tables

Nomenclature

Formula symbols and units

Symbol	**Meaning**	**Unit**
b	Pumping power consumption	MW
D	Power demand	MW
IG	Intermittent generation	MW
ND	Net demand	MW
p	Power generation	MW
pns	Power not served	MW
rd	Downward reserve	MW
ru	Upward reserve	MW
s	Power spillage	MW
sd	Shutdown decision variable	{0,1}
su	Startup decision variable	{0,1}
uc	Commitment status variable	{0,1}
w	Reservoir level	MWh

Indexes and abbreviations

Symbol	**Meaning**
AI	Artificial intelligence
ANN	Artificial neural network
DT	Decision tree
GBDT	Gradient boosting decision tree
IML	Interpretable machine learning
LogR	Logistic Regression
LR	Linear regression
MIP	Mixed-integer programming
ML	Machine learning
PCA	Principal components analysis
RF	Random forest
RL	Reinforcement learning
RMSE	Root-mean-square error
RSS	Residual sum of squares
SDG	Sustainable Development Goal
SL	Supervised learning
SO	System operator
UC	Unit commitment
UL	Unsupervised learning

1 Introduction

Generation planning is one of the most important tasks within power system operation, which allows meeting the demand at minimum cost, based on demand and renewable generation forecasts [1].

Moreover, any imbalance between total generation and demand in real-time can entail a risk to system stability. Therefore, the system operator must take corrective measures, such as adjusting the production of committed generators, to return the system to equilibrium. These measures can involve significant costs for the system, so the operator must consider the uncertainty associated with generation and demand when scheduling generation in order to deal with unforeseen events as efficiently as possible [1].

The most widespread tool to carry out this task is the unit commitment (UC). It is an optimization problem whose objective is to plan the commitment and production of the system units over a given horizon, minimizing the total operating costs, and respecting certain physical and temporal constraints from generators and transmission lines while guaranteeing system security requirements [2].

Given its importance, the unit commitment has become one of the most studied problems in the electricity sector. In recent years, numerous studies have tried to reduce the computational time required to solve this problem and to improve its ability to model the increasingly complex details of power systems, especially the growing uncertainty associated with the accelerating penetration of renewable energy sources, in order to obtain progressively more efficient results [3], [4].

One of the main lines of research in this sense has been the application of machine learning techniques to the UC problem. Such techniques have been applied to improve demand and intermittent generation predictions, reducing the impact of their associated uncertainty [5]; to predict which constraints of the problem are active in the optimal solution and, therefore, consider them when solving the optimization problem [3]; or to obtain preliminary solutions and provide the optimization problem with these close-to-optimal solutions in order to reduce the computation time [4].

Despite the large amount of research related to the unit commitment problem and, particularly, the application of machine learning to its resolution, there are no works dedicated to developing models that can be used to understand the solutions generated by UC models. The interpretation of such results still requires a deep knowledge of the topic since the optimization algorithm does not transparently explain how it obtained the solution found.

The problem of lack of transparency is not unique to optimization models. The increasing complexity of machine learning algorithms has resulted in a loss of their ability to explain how they carry out their estimations in a human-understandable way. For this reason, these complex algorithms have been considered as black boxes, a concept that can also be applied to optimization models.

This loss of accountability can be a major problem when these models are used as tools for decision-making since they can lead to undesirable results, which can be difficult to detect and solve. To overcome these limitations, interpretable machine learning (IML) has emerged in the past few years as a response to the need for developing models that can be both accurate and interpretable so as they can be used for critical decision-making processes [6].

Consequently, in this project, we propose the development of models based on interpretable machine learning techniques that can not only generate the optimal solutions to the unit commitment problem but also explain how these results have been obtained.

2 State of the art

2.1 The unit commitment problem

In power systems, the balance between generation and demand is critical for system stability due to the difficulty of storing electrical energy. Any imbalance between generation and demand in real-time deviates the system's frequency from its nominal value and must be fixed by the system operator by adequately adjusting the output of the generators at its disposal. In the most extreme imbalance cases, the system operator may be forced to apply specific corrective measures, such as connecting fast-startup generators or shedding part of the load, which may entail significant additional costs for the system [1].

Therefore, optimal generation planning is one of the essential tasks in power systems operation, and unit commitment is the most popular tool to address this issue. Figure 1 depicts the day-ahead generation schedule segmented by technologies of the Spanish system for April 5th, 2021, elaborated with data published by the Spanish system operator, Red Eléctrica de España [7].

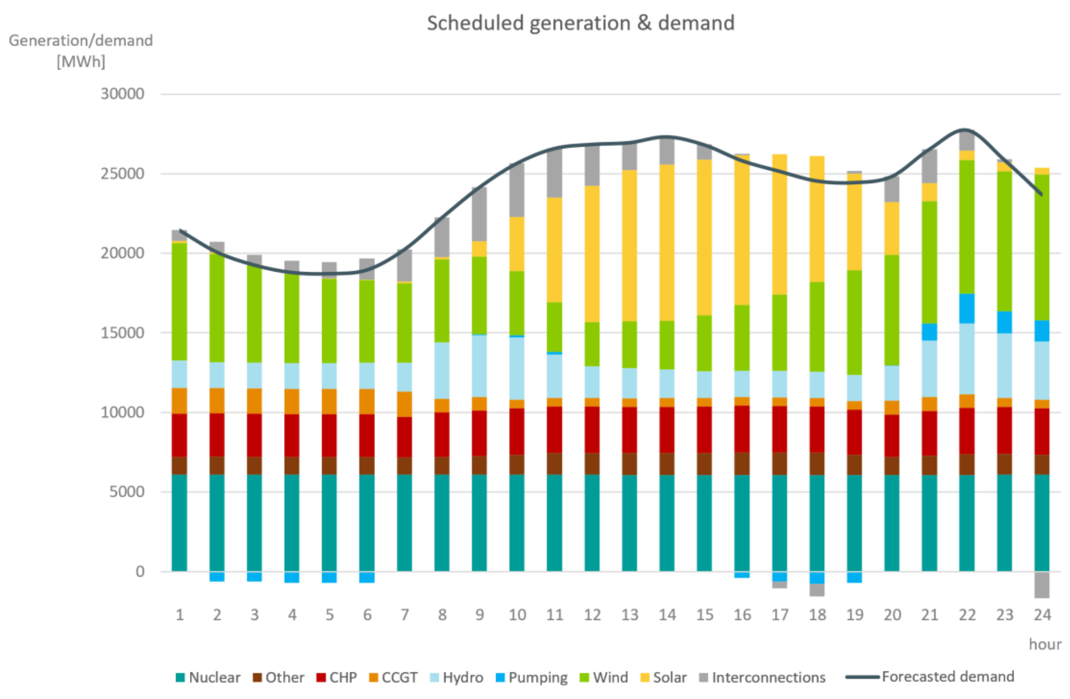

Figure 1: day-ahead scheduled generation and demand of the Spanish system for April 5th, 2021.

The unit commitment problem is an optimization problem that deals with the (generally) short-term centralized scheduling of the commitment of units, as well as their output, with the purpose of meeting the forecasted demand while minimizing the operational costs of the system over a given horizon, and complying with certain system requirements (e.g., spinning reserve) and physical, temporal constraints (e.g., technical limits of the generators) [2].

The UC problem is mathematically formulated as a non-convex mixed-integer programming (MIP) problem, belonging to the NP-hard category, which makes it difficult to efficiently obtain the optimal solution. This issue has been magnified over the past decades due to the significant increase in the size and complexity of power systems [8].

Consequently, multiple formulations of the UC problem have been proposed in recent years, seeking to more realistically model system constraints and reduce the computation time required to solve them, which have opened the door to applying these formulations to increasingly larger systems, achieving more accurate and efficient results. However, despite the former advances, the computation time required to solve large-scale UC problems is still one of the major limitations when it comes to accurately modeling it, forcing system operators to either simplify the system constraints or restrict the computation time of the problem resolution. As a result, sub-optimal solutions are obtained, which involves extra costs for the system compared to the optimal generation planning [2], [3], [9].

Additionally, uncertainty management has become a central matter concerning the scheduling of generation to meet demand at the lowest cost. Traditional UC formulations have modeled the problem as deterministic, in which the forecasted demand, renewable generation, and real-time availability of generators and other elements of the system are assumed to be known at the moment of the problem resolution. Renewable, non-dispatchable generation is modeled along with demand so that generators must meet the net demand of the system (demand minus renewable generation). In this class of formulations, the uncertainty associated with these variables is usually managed by imposing a contingency generation reserve requirement that could absorb the real-time generation-demand imbalances resulting from these sources of variability [1], [4].

These formulations have historically offered satisfactory results due to the relative predictability of demand and the low generation share of renewable resources, allowing cost-effective management of real-time imbalances. However, the increasing penetration of intermittent generation and price-responsiveness of demand in electric power systems leads to new challenges regarding the efficient contingency of their intrinsic variability, given the uncontrollable nature of these elements. Besides, by only considering a single reference scenario, which is usually the most likely one, the obtained generation planning might turn to be completely inefficient considering the conditions that finally occur in real-time [1].

Thus, alternative UC formulations have been developed in order to take into account the uncertainty of these variables when solving the optimization problem, of which three are worth mentioning. First, stochastic optimization is carried out employing multiple scenarios that reflect the probability distribution of the uncertainty and searches for the solution yielding the most efficient overall results. Second, robust optimization deals with the uncertainty by considering a specified variability in the input parameters and finds the best solution which is feasible in all the possible scenarios. Lastly, in chance-constrained optimization, problem constraints need only to be respected above a certain probability level [1], [4].

However, even though these variants could allow system operators to carry out a more efficient generation planning considering the intrinsic uncertainty of generation and demand, these variants generally require the model to consider multiple scenarios or combinations of input parameters. This significantly increases the complexity of the optimization problem in comparison to the deterministic approach, which could already suffer from computational efficiency issues. As a result, the employment of such formulations for real-world applications is not yet extended, and further improvements in this field will be necessary to make it more attractive [4].

2.1.1 Elements of the unit commitment

Despite the wide variety of existing UC formulations, most of them have some common elements presented below.

2.1.1.1 Time horizon and periods

The time horizon defines the timespan to be modeled through the UC, which will be divided into periods according to specified time steps. The time horizon usually ranges from one day (typical day-ahead scheduling) to one week, while the time step is generally equal to one hour and, occasionally, 15 or 30 minutes. As the computational burden of solving UC problems has improved over the past years, the interest in reducing the length of time periods has correspondingly grown due to the possibility of modeling system flexibility constraints with greater detail [10], [11].

2.1.1.2 Load profile

In general, the forecasted demand to be satisfied by generation must be defined either for the whole system or at each node if a network representation is included. Although system operators seek to meet the entire demand, this is not always economically efficient or technically possible. Consequently, a cost (penalty) for not serving electricity load must be generally introduced so as to account for this issue, and its value should be related to the utility of demand.

Finally, as it has been introduced, the load profile is an important source of uncertainty and even the main one in small systems with low penetration of renewable energy sources. UC formulations conceived to deal with uncertainty usually require the characterization of multiple demand scenarios, considering their probability of occurrence, for the model to yield the optimal generation planning [10].

2.1.1.3 Generating units

The definition and modeling of generation units are some of the fundamental parts of the UC formulation. Typically, three types of units are distinguished. The primary ones are thermal units, such as nuclear, coal, or gas power plants. Thermal generation commonly stands for the most remarkable portion of short-term operation costs, if not the whole, which implies that their accurate representation is vital for obtaining the most economically efficient generation schedule [10], [11].

Thus, the main distinguishing factors of generating units are their startup, shutdown, and generation costs (or cost curves). Depending on these costs, some generators will more efficiently satisfy the base load, while others will be more suited to cope with daily electricity demand peaks. In addition, thermal generators are usually subject to several technical constraints such as generation capacity or flexibility, which need to be considered since they restrict their actual operation capability [12].

In contrast, renewable generation such as wind or solar generation has an intermittent, non-dispatchable nature, meaning that system agents cannot control their output. In consequence, this kind of generation is normally considered deterministic, in the same way as demand [1], [10]. Furthermore, its operating cost is generally considered to be zero, and therefore will be the first resource used to meet demand. However, under certain circumstances, such as when there is a critical excess of generation compared to electricity load and reducing the thermal output is not cost-efficient, part of renewable generation might be curtailed to minimize system costs [13].

Finally, hydro units are also considered in a large number of UC problems. These are generally the most flexible units of the system, and their generation cost can be assumed to be zero. However, this assumption may only lead to efficient generation schedules when the hydro energy to be produced during the pertinent time horizon has already been pre-allocated employing a long-term model that can efficiently account for the *water value*, which reflects the cost of the system that can be avoided by producing the available hydro energy in the future. Alternatively, a cost for hydro generation could be introduced to consider this issue instead of fixing the hydro energy to be produced [14].

There are three main types of hydro units, depending on their characteristics. However, they can all be modeled under the same generic formulation through a combination of constraints related to their power generation capacity and the level limits of their water reservoirs. Conventional hydro plants rely on a water reservoir to store water to be used in the most cost-efficient periods, such as expensive peak-load periods. In contrast, run-of-river hydro plants have little to no water storage capacity. Their electricity production highly depends on the river's water flow in which they are located, thus behaving in a similar way to the mentioned renewable generation. Lastly, pumped-storage hydro plants have both an upper and a lower water reservoir. They can pump water from the lower to the upper one during *cheaper* hours in order to use it to produce electricity during *expensive* hours [12].

2.1.1.4 Transmission network

In certain systems, particularly weakly meshed ones, the transmission network can play a decisive role in the system's operation, forcing the startup and shutdown of some generators in order to satisfy the demand at every node without exceeding the capacity limits of power lines. In the case of strongly meshed networks, the effect of the transmission network is sometimes neglected, which is known as modeling it as a copperplate network [11], [15].

In many power systems, the market clearance is carried out financially without considering the impact of the transmission grid, which may not be technically possible. Subsequently, the system operator is in charge of determining a feasible and reliable generation schedule by adjusting the commitment of generators as well as their output. These adjustments imply an increase in generation costs relative to the optimal configuration resulting from the non-restricted optimization problem and may not represent the most efficient solution considering the network topology [11], [15].

Therefore, the decoupling between generation planning and network management may result in inefficient solutions that further increase the cost of system operation. The inclusion of network constraints in the UC model allows obtaining the most economical feasible dispatches, effectively overcoming the stated issue [10], [11].

2.1.1.5 Objective function

Generally, the UC problem's purpose is to minimize the system operation costs over the studied horizon, satisfying certain technical, inter-temporal, and reliability constraints or requirements [10]. Therefore, the objective function of the mathematical formulation of the UC problem should effectively reflect the different costs that result from the operation of the power system, such as the cost of electricity generation (CF_t and CV_t), the startup (CSU_t) and shutdown costs (CSD_t) of generating units, and the cost of not satisfying part of the demand ($CPNS_n$), as shown in (1). In this expression, renewable and hydro generation costs are neglected and, therefore, they are not reflected in the objective function. If this was not the case, their operational cost could be modeled similarly to a thermal generator. Any other cost or penalty associated with the system operation, such as the cost of secondary reserves, may also be included in the objective function [2].

$$\min \sum_{nt} CF_t uc_{nt} + \sum_{nt} CV_t p_{nt}^T + \sum_{n} CPNS pns_n + \sum_{nt} CSU_t su_{nt} + \sum_{nt} CSD_t sd_{nt} \tag{1}$$

2.1.1.6 Constraints

Constraints imposed in the optimization problem aim to model, as loyally as possible, the characteristics and constraints of the elements that make up the power system. Depending on the definition of these constraints, the resulting model may be simpler but computationally more efficient or more accurate, therefore resulting in a more economically efficient solution, at the cost of requiring more time to be solved [3].

In the following, some of the most common constraints used in UC problems will be presented. These constraints are mainly based on the Tight and Compact UC formulation described in [2], complemented with the representation of hydro units extracted from [16], and network constraints from [17].

- **Generation and demand balance requirement**

One of the critical requirements for the UC is to schedule the necessary generation to meet demand at every period. Therefore, the balance between generation and demand (2) is one of the fundamental equations of any UC formulation: The sum of thermal generation (p_{nt}^T), net hydro production ($p_{nh} - b_{nh}$), forecasted intermittent generation (IG_n), and the non-served demand (pns_n) must equal the total demand of the system (D_n) for each period [2], [16]. This same principle can be applied to network-constrained UC formulations (3).

$$\sum_{t} p_{nt}^T + \sum_{h} (p_{nh} - b_{nh}) + IG_n + pns_n = D_n \quad \forall n \tag{2}$$

$$\sum_{t} p_{nt}^T + \sum_{h} (p_{nh} - b_{nh}) + \sum_{i} IG_{ni} + \sum_{i} pns_{ni} = \sum_{i} D_{ni} \quad \forall n \tag{3}$$

- **Commitment, startup, and shutdown logic**

This equation (4) is used to link the variation of the commitment state (uc_{nt}) of a thermal unit t in period n compared to the previous period, and its startup (su_{nt}) and shutdown decisions (sd_{nt}) for the same period [2]. Concerning these variables, 1 means that unit *t* is committed, starts up or shuts down at period *n*, respectively.

$$uc_{nt} - uc_{n-1,t} = su_{nt} - sd_{nt} \quad \forall nt \tag{4}$$

- **Minimum up and down time constraints**

Some thermal units are required to remain committed or offline for a certain number of periods once they have been started up or shut down, respectively. Expressions (5) and (6) model the minimum *up* and *down* time constraints of these generators [2].

$$\sum_{n'=n+1-TU_t}^{n} su_{n't} \leq uc_{nt} \quad \forall nt \tag{5}$$

$$\sum_{n'=n+1-TD_t}^{n} sd_{n't} \leq 1 - uc_{nt} \quad \forall nt \tag{6}$$

- **Total output**

It is common to model the output of thermal units (p_{nt}^T) as the production above the technical minimum of the generator ($\underline{P}_t$), considering the commitment state of the unit, so as the output variable (p_{nt}) represents the *flexible* production of the unit (7) [2].

The previous equation assumes that thermal units can both start up and shut down in just one period, i.e., one hour. However, some generators such as coal plants may actually need more time to start up (shut down) and will increasingly (decreasingly) produce energy before being fully committed (offline). Therefore, more detailed UC models take into account these startup (PSU_{nt}) and shutdown trajectories (PSD_{nt}) to compute the total output of thermal units (8) [2].

$$p_{nt}^T = p_{nt} + \underline{P}_t uc_{nt} \quad \forall nt \tag{7}$$

$$\begin{aligned} p_{nt}^T = p_{nt} + \underline{P}_t uc_{nt} + \sum_{n'=1}^{TSD_t} PSD_{n't} sd_{n-n'+1,t} \\ + \sum_{n'=1}^{TSU_t} PSU_{n't} su_{n-n'+1+TSU_t,t} \quad \forall nt \end{aligned} \tag{8}$$

- **Minimum production constraint**

Thermal units are designed to produce above a minimum level, known as the generator's technical minimum. Expression (9) ensures that the production of any thermal unit t at any period n minus the downward generation reserve (rd_{nt}) is always above its technical minimum [2].

The same expression would be valid for hydro units, provided their output has been defined above their minimum production. However, if this is not the case, then this constraint would be expressed as in (10). Most conventional hydro plants have a virtually zero minimum production but others, namely run-of-river hydro plants, have a running water flow requirement that leads them to generate a minimum of electricity which may vary over different periods. Pumped-storage hydro plants are able to pump water from a lower to a higher reservoir by consuming electricity, and expression (11) ensures that this consumption is non-negative at any period [16].

$$p_{nt} - rd_{nt} \geq 0 \quad \forall nt \tag{9}$$

$$p_{nh} - rd_{nh} \geq \underline{P}_{nh} \quad \forall nh \tag{10}$$

$$b_{nh} \geq 0 \quad \forall nh \tag{11}$$

- **Maximum production capacity constraint**

Generating units have a maximum technical output which sets an upper bound on their generation capacity. As a consequence, the production of a unit t above its technical minimum plus its upward generation reserve (ru_{nt}) at any period n must be lower than its flexible production capacity ($\overline{P}_t - \underline{P}_t$) when the generator is committed, and below 0 when it is offline (12) [2].

If startup and shutdown trajectories are considered (13), the available production capacity the period before starting up or shutting down will be restricted by the maximum startup (SU_t) or shutdown capacities (SD_t), respectively, and not by the maximum output of the unit ($\overline{P}_t$) [2].

Analogously to (12), the production plus its upward reserve of a hydro unit h at any period n must be below the maximum production capacity of the unit ($\overline{P}_h$) (14). Expression (15) ensures that the consumption of pumped-storage hydro plants is below their maximum limit ($\overline{B}_h$) [16].

$$p_{nt} + ru_{nt} \leq uc_{nt}\left(\overline{P}_t - \underline{P}_t\right) \quad \forall nt \tag{12}$$

$$p_{nt} + ru_{nt} \leq uc_{nt}\left(\overline{P}_t - \underline{P}_t\right) - sd_{n+1,t}\left(\overline{P}_t - SD_t\right) - su_{n+1,t}\left(\overline{P}_t - SU_t\right) \quad \forall nt \tag{13}$$

$$p_{nh} + ru_{nh} \leq \overline{P}_h \quad \forall nh \tag{14}$$

$$b_{nh} \leq \overline{B}_h \quad \forall nh \tag{15}$$

- **Ramping constraints**

Additionally, thermal units can only change their production from one period to another up to a defined upward or downward limit (RU_t). Therefore, the generation increase of unit *t* at period *n* compared to the previous period plus the upward reserve for the same period (since it may be required to increase its production in real-time operation) must be lower than the maximum upward ramp (RU_t) (18). Similarly, (19) restricts the downward ramp of thermal generators to be below their maximum (RD_t).

$$p_{nt} - p_{n-1,t} + ru_{nt} \leq RU_t \quad \forall nt \tag{16}$$

$$p_{nt} - p_{n-1,t} - rd_{nt} \geq -RD_t \quad \forall nt \tag{17}$$

- **Up and down secondary reserve requirements**

Some UC models include spinning reserve requirements which are employed to adjust committed generators' output to match the actual demand during real-time system operation. For every period, the sum of all generators up (18) and down secondary reserves (19) must be above the total required up (DU_n) and down reserves (DD_n) [2]. These hourly requirements are usually set relative to specific parameters such as the greatest committed generating unit of the system, a percentage of the demand, or a percentage of wind generation.

$$\sum_t ru_{nt} + \sum_h ru_{nh} \geq DU_n \quad \forall n \tag{18}$$

$$\sum_t rd_{nt} + \sum_h rd_{nh} \geq DD_n \quad \forall n \tag{19}$$

- **Hydro reservoir level and available hydro energy**

Hydropower plants' water availability is usually modeled through their reservoir levels, which are typically expressed in terms of the energy that can be extracted from a volume of water rather than the volume itself. According to (20), the reservoir level (w_{nh}) of hydro plant h at period n is equal to the level of the previous period, minus the variation during the period due to the operation of the hydro plant. The reservoir level will decrease when water is turbined (p_{nh}) to produce electricity and, in the case of pumped-storage hydro plants, increased when water is pumped up (b_{nh}) –corrected by the efficiency of the pumping process (η_h). Hydro plants can also spill water (s_{nh}) without generating electricity in cases where they have an excess of water in their reservoirs [16].

This reservoir level is normally required to match a defined value at the last period, limiting the amount of water that can be turbined during the UC horizon (21). Alternatively, simpler formulations directly represent a fixed available hydro energy to be produced over the scheduling horizon (22). This last formulation serves as a good approximation for short-term UC problems, but may lead to unfeasible results when longer horizons are considered.

Finally, more complex (and less short-term) hydro formulations include time-space relationships of hydro basins (cascaded reservoirs). These formulations are similar to (20) but include the increase of the reservoir level due to natural water inflows as well as water that has been turbined at an upper hydro plant of the same basin, reduced by an efficiency factor, and delayed a number of periods related to the time it takes this water to flow from one plant to the other [16].

$$w_{nh} = w_{nh-1} - (p_{nh} - \eta_h b_{nh} + s_{nh}) \quad \forall nh \tag{20}$$

$$w_{nh} = w_h^* \quad n = N \tag{21}$$

$$E_h \geq \sum_n (p_{nh} - \eta_h b_{nh} + s_{nh}) \quad \forall h \tag{22}$$

- **Hydro reservoir level requirements**

Hydro reservoir levels are generally required to be kept within certain security limits, which may prevent hydro plants from producing electricity if the reservoir level reaches its minimum level or force the power plant to produce electricity and even spill water if the reservoir level rises to its upper limit. When hydro reservoirs are being modeled, (23) and (24) are used to ensure that these limits are not exceeded [16].

$$w_{nh} \geq \underline{W_h} \quad \forall nh \tag{23}$$

$$w_{nh} \leq \overline{W}_h \quad \forall nh \tag{24}$$

- **Maximum transmission capacity constraints**

Finally, network-constrained UC models require to limit the maximum power ($\underline{F_k}$ and $\overline{F}_k$) that can flow through the transmission lines of the power system. In (25) and (26), the power flow of any line k is computed as the sum of the contributions of the net generation (thermal, hydro, and intermittent generation minus pumping consumption and demand) of every node excluding the slack node, weighted by the sensitivity factor ($\tilde{Q}_{ki}$) of the line with respect to each of these nodes [17].

$$\underline{F_k} \leq \sum_{i \neq s} \tilde{Q}_{ki} \left(\sum_{t \in \Omega_i} p_{nt} + \sum_{h \in \Omega_i} (p_{nh} - b_{nh}) + IG_{ni} - D_{ni} \right) \quad \forall nk \tag{25}$$

$$\overline{F}_k \geq \sum_{i \neq s} \tilde{Q}_{ki} \left(\sum_{t \in \Omega_i} p_{nt} + \sum_{h \in \Omega_i} (p_{nh} - b_{nh}) + IG_{ni} - D_{ni} \right) \quad \forall nk \tag{26}$$

2.2 Interpretable machine learning

Before discussing in detail the concept of interpretable machine learning, it will be helpful to provide a general overview of machine learning, as well as its main classes and algorithms, since it will serve as a basis for the following sections.

2.2.1 Machine learning

Machine learning (ML) is the branch of artificial intelligence (AI) that studies the development of computer algorithms whose objective is to learn from a "training data set" in order to make predictions about new data through inference and generalization of relationships between samples, input features, and outputs. More specifically, one of the most widespread definitions of ML was proposed by T. Mitchell: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.” [18]

The origin of machine learning, as we know it today, dates back to the late 1950s with the invention of the "perceptron" by F. Rosenblatt, which later enabled the development of artificial neural networks (ANN). However, some of the techniques used in the field of machine learning were developed much earlier, such as the least-squares method or the Bayes' Theorem [19].

Since then, this field has experienced exponential growth, both thanks to the development of new algorithms and their continuous improvement, which has allowed them to significantly increase their predictive accuracy and reduce their training time. As a result, machine learning techniques have gained widespread popularity and recognition, finding endless applications in virtually all areas of knowledge, including, certainly, power systems operation [19].

Depending on the task to be performed and the nature of the data used, machine learning algorithms can be classified into three main groups:

- **Supervised learning**

In supervised learning (SL) tasks, the algorithm is trained to predict desired outputs based on their corresponding inputs by extracting general relationships between them. Consequently, the goal of supervised learning is to infer a general function that is able to map inputs to outputs as accurately as possible. SL algorithms can be further grouped into classification and regression analysis, depending on the output datatype [20].

When the output to be predicted is quantitative (numeric value), the task is defined as regression. In addition to estimating the desired outcome for a given input, the algorithm can be used to analyze the relationship between input features and the output variable by inspecting how the output varies when one or more input features are changed. Figure 2 (left) presents an example of a one-feature regression task. In contrast, when the output variable is qualitative (categorical), the task is said to be a classification problem. The purpose of classifier algorithms is to predict the class to which an instance belongs based on its attributes. Figure 2 (right) shows an example of a two-dimensional binary classification problem.

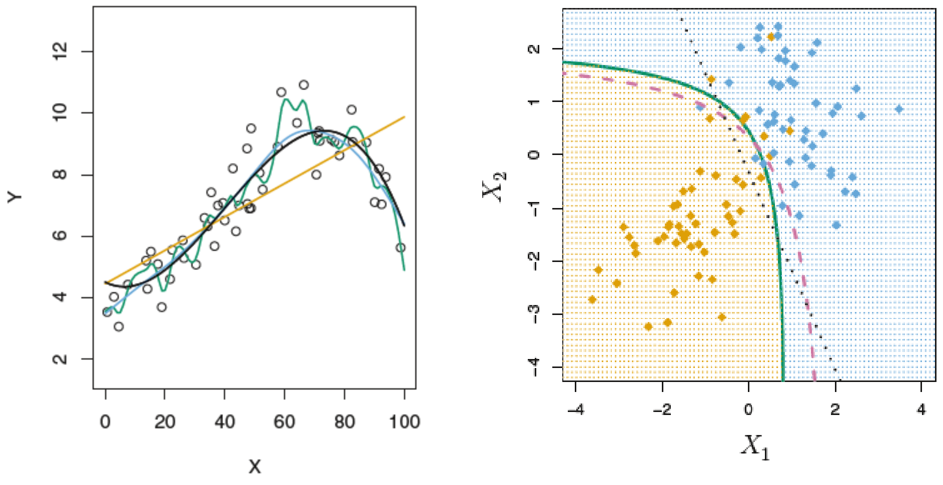

Figure 2: Examples of regression analysis (left) and classification (right) [18].

Up to now, a wide variety of supervised learning algorithms have been developed based on very diverse techniques. Some of the most popular algorithms include linear regression, logistic regression, decision trees and other tree-based methods such as random forests and gradient boosting decision trees, support-vector machines, and artificial neural networks. The kind of relationships modeled by the algorithm will highly depend on its nature and complexity. As a consequence, there are generally no better or worse algorithms for every possible task. Certain algorithms will be better suited for certain problems than others, which makes the selection of the most suitable model for a given task one of the most critical steps of developing a supervised learning model [21]. A typical example to visualize the presented issue is shown in Figure 3, where the linear model clearly outperforms the decision tree classifier in the first problem but not for the second.

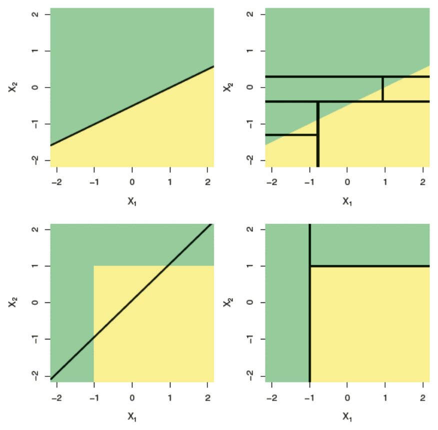

Figure 3: Comparison of linear and decision tree classifiers on different two-dimensional classification tasks [20].

- **Unsupervised learning**

As opposed to the previous case, in unsupervised learning (UL), the output label of data is not known or available. Thus, the purpose is to infer relationships either between features or samples. Within unsupervised learning, the most important types are clustering and principal components analysis (PCA) [20].

On the one hand, clustering algorithms are meant to group observations that share attributes and to identify patterns across them. These are commonly used to determine groups of *similar* samples and extract a representative sample for the whole group, which allows to significantly reduce the amount of data to be stored. Figure 4 (left) illustrates a two-dimensional clustering with 3 clusters. On the other hand, principal component analysis deals with the computation of the principal components of the data distribution, i.e., the orthonormal linear combination of features that best fits the distribution, minimizing the square distance to every axis of the new space. PCA methods, such as the one depicted in Figure 4 (right), are usually used for dimensionality reduction losing the least possible amount of information from the original features since the first few principal components tend to comprise most of the information [20].

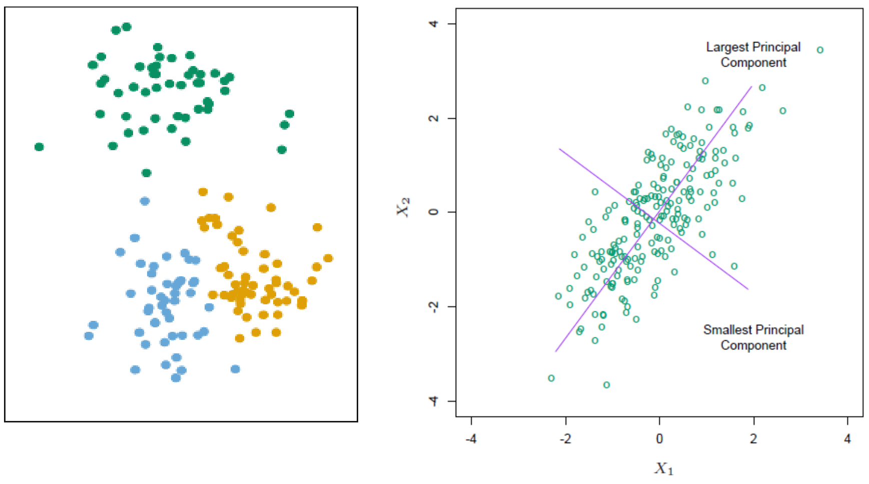

Figure 4: Examples of clustering (left) [20] and principal component analysis (right) [22].

- **Reinforcement learning**

Reinforcement learning (RL) is the last of the main classes of machine learning methods. In this case, the algorithm is provided with an environment with which it can interact. Depending on the actions it takes within the given environment, it will receive different rewards. Therefore, the purpose of the algorithm is to learn how to optimally behave in the provided environment by maximizing its reward function according to the reinforcement signals it receives. RL is widely used for control systems [23].

2.2.2 From ML to interpretable machine learning

The constant research to improve the accuracy of machine learning models has led to the development of new, better-performing algorithms, which are able to learn increasingly more sophisticated relationships. Because of this flexibility requirement, algorithms have become more and more complex, resulting in a loss of transparency regarding their underlying behavior [24].

Therefore, many of these algorithms are seen as *black-box models*, i.e., models that do not have the ability to explain how they come up with the predictions and whose parameters cannot be directly interpreted or understood by humans. Due to their inherent complexity, some of the algorithms that are commonly considered black-box models are deep neural networks, gradient boosting decision trees, random forests, and support-vector machines. In addition, proprietary models are also regarded as black boxes because only predicted outcomes are available to users and not the model itself [25].

This opacity may not imply a problem in itself, as some tasks may only require the best possible accuracy, regardless of how the algorithm comes up with the predictions. However, ML models are progressively used as relevant tools for decision making, and the lack of accountability for predictions made by black-box models has become a highly controversial issue, especially when the outcome of the model can have an impact on human lives. In recent years, multiple ML models have been reported to generate *undesired* outcomes because of systematic biases related to ethnicity, gender, or sexual identity. Some examples of these controversial issues include racially biased legal sentencing advice and face-recognition models or gender-biased salary estimating algorithms [26].

This problem has been labeled as algorithmic bias, and it is strongly related to the data used to train the model. This means that if for whatever reason, the training data present certain correlations between any of the controversial features and the desired output, the model will interpret these relationships as causal and will replicate them when generating actual predictions. The inability to account for how the algorithm carries out its predictions can make it impossible to fix or even identify issues such as the ones presented above. Thus the black-box model may not be reliable enough to be used for critical decision making [26].

In this context, interpretable machine learning (IML) has emerged in the past few years as a response to the need for developing models that can be both accurate and interpretable in order to be used as valuable and fair tools for crucial decision-making processes. These models are popularly referred to as white-box models since they are transparent in how they compute their outcomes [6].

2.2.3 Interpretability and algorithm classification

Since interpretability is the central issue concerning these models, a formal definition is needed in order to properly identify ML models as interpretable. According to T. Miller, interpretability can be defined as "the degree to which a human can understand the cause of a decision" [27]. Thus, the easier it is for someone to understand how the model computes its predictions, the more interpretable it is.

Depending on the point at which the prediction interpretability is achieved, two main categories of interpretable machine learning techniques can be identified:

- **Intrinsic interpretable models**

Model-based or intrinsic interpretability is based on the use of inherently interpretable algorithms. This kind of interpretability's primary challenge is to implement models easily understood by humans while achieving high accuracy [28]. Some of the most common intrinsic interpretable models are decision trees, rule-based models, and linear regression [29].

Nevertheless, even these *simpler* models can become less interpretable if their internal complexity is not effectively restricted by, for example, limiting the number of features they use or by constraining key hyperparameters. Examples of these interpretability constraints include forcing sparsity in the model (such as lasso regression), imposing monotonicity constraints in classifiers, and limiting the number of leaves or depth of decision trees. Including interpretability constraints in ML models often results in a trade-off between interpretability and performance since more complex models may achieve more accurate results for specific tasks than the constrained ones [29].

Intrinsic interpretable machine learning models will be the most relevant IML techniques for the purpose of this project. One of its greatest exponents in recent years has been C. Rudin, from which two main works applied to decision-making in medicine and criminal justice can be highlighted due to the implications of the obtained results.

In the first one, she developed a model based on decision lists called Bayesian Rule Lists to predict stroke risk in patients [30]. This interpretable model consists of a series of *if/else if/else... then...* statements, where the first part defines a partition of the feature space and the second one assigns a prediction for that partition. Despite being a simple model, its accuracy was even higher than the previous stroke risk prediction approach without compromising its interpretability, challenging the general belief that simpler models necessarily result in worse predictions than those obtained using complex algorithms.

In the second of these works, she proposes a transparent and sufficiently accurate classification model for predicting recidivism in the context of criminal justice, which was called Supersparse Linear Integer Model [31]. It is a scoring system formulated as an optimization problem in which the objective function, composed both of the prediction error and an interpretability penalty, is minimized using mixed-integer programming. The implemented algorithm proved intrinsic interpretable ML models to effectively overcome the issues related to algorithmic bias and lack of transparency that were experienced with previous approaches.

Further on, some of the main intrinsic interpretable models will be introduced, as they will be the ones to be studied in the context of this project.

- **Post-hoc interpretability**

In contrast, post-hoc interpretability consists of creating interpretable models or using diverse methods to explain the predictions made by a black-box model. These methods seek to shed light on how complex models work and to provide more trust towards them. In this case, the resulting predictions can be highly accurate because it does not depend on the interpretable model itself, but on the more complex black-box model [29]. Post-hoc interpretability techniques can also be used to improve the accuracy of the underlying ML models, as they could help identify relationships learned by the model that are known to be incorrect [28]. However, the reliability of the explanations obtained by this method can be limited, as it is still an approximation [29].

Post-hoc interpretability methods are usually categorized into global and local interpretability methods, sometimes referred to as dataset-level and prediction-level interpretations, respectively. Global interpretability methods provide an explanation of how the model works internally, i.e., how outcomes are generated. These methods can provide complex models with a greater level of transparency. Local interpretability methods are used to explain how the model came out with individual predictions, analyzing causal relationships between inputs and output for specific samples [29].

These techniques can be further classified into model-agnostic and model-specific explanations. On the one hand, model-agnostic explanations are widely applicable to any machine learning algorithms because they only use their input and output data. They treat the model as if it were strictly a black box, regardless of its actual implementation. On the other hand, model-specific explanations are unique to a single machine learning model, usually examining its internal parameters and structure [29].

One of the main model-agnostic interpretation methods is the surrogate model, an ML model trained to replicate the predictions of another –original– model. Surrogate models can be either global or local, and they are widely used in a number of engineering applications to replace more complex, expensive, or time-consuming models and understand how they work. In fact, machine learning algorithms used to generate unit commitment solutions, generally with the purpose of reducing the computational burden of solving the corresponding optimization problem, fall into the category of global surrogate models. In the context of model agnostic interpretation methods, the surrogate model is an intrinsic interpretable model used to explain how a black-box model works internally by approximating its behavior [6].

Another popular model-agnostic technique is known as permutation feature importance, and it is used to assess the impact of each input feature in the generation of predictions. Given an already trained model and a test set, values within each input feature are iteratively shuffled. The prediction accuracy is evaluated for each of the permuted test sets, with the advantage of not needing features to be normalized. The decrease of accuracy obtained for the permuted test sets compared to the non-permuted one reflects the importance level of these features [6], [29].

Likewise, there are multiple comparable feature importance and feature interaction methods, which yield similar insights regarding input features [6]. Some of them are model-specific, such as feature importance methods applied to tree-based ensemble models. In this case, importance values are computed by assessing how many times each feature is used to split the feature space or how much accuracy increases considering all the splits corresponding to each feature [29]. All these approaches are easily interpretable and can provide valuable insights regarding what the model is learning from the data. However, they have significant drawbacks, such as the fact that they do not explain how these features are being used by the model and *how* do predictions change with respect to variations in input features, or the decrease of importance observed when input features present any kind of correlation, obscuring their relationship with the output variable [6].

Besides, there has been a significant effort to provide artificial neural networks with interpretability in the past years. The main reason is that ANNs are able to obtain unrivaled performances in certain highly complex problems, but their internal structure is virtually impossible to be understood by humans. Thus, post-hoc model-specific methods are used to explain how they come up with predictions without harming the predictive accuracy. Most of them are conceived as visualizations of the internal parameters of different algorithm layers, which are supposed to depict their specific function. Global approaches extract the weights from these layers to explain which is the relevant information and how it is being transformed, while local approaches (probably the most popular ones, especially in image recognition models) consider the actual values of each layer resulting from the prediction of individual samples [28].

Some authors, such as C. Rudin [25] and W. J. Murdoch [28], have expressed their critical opinion on post-hoc methods, emphasizing that the interpretation they provide should be carefully analyzed before being used as high-stake decision tools.

According to [25], intrinsic interpretable models should always be employed in these situations instead of black-box ML models. In her work, Rudin presents that the interpretability of ML models does not come at the cost of lower accuracy, particularly when structured data is available, and features contain meaningful information. Moreover, she stresses the fact that post-hoc techniques provide explanations that cannot be trusted to represent how the original model computes its predictions because it is just an approximation of the original model. Finally, post-hoc explanations cannot explain how a black box model would perform when fed with outlying inputs (inputs significantly different from those used to train both models), which is a critical issue when these models are used for decision-making purposes [25].

Lastly, any of these two main interpretation techniques (model-based and post-hoc) is considered to have two fundamental requirements: accuracy and relevance. First, an interpretable model needs to be accurate since otherwise, it is impossible to claim that the model provides appropriate explanations for the underlying relations between inputs and outputs because it could not reproduce them. This accuracy requirement is identified as predictive accuracy for intrinsic interpretable models to describe the ability of the model to approximate the underlying data relationships, and descriptive accuracy for post-hoc interpretation methods (surrogate models) to define their ability to approximate the relationships that the original model has learned. Achieving the necessary accuracy level may be especially challenging when the problem has highly complex relationships, or complex black-box models have been used as predictors [28]. Figure 5 depicts the impact of both interpretability methods on predictive and descriptive accuracies.

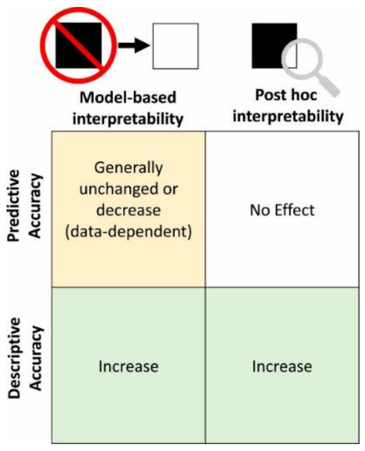
Model-based interpretability
Post hoc interpretability
Predictive Accuracy
Descriptive Accuracy
Generally unchanged or decrease (data-dependent)
No Effect
Increase
Increase

Figure 5: Intrinsic and post-hoc interpretability methods impact accuracy [28].

Second, the extracted information needs to be relevant, insightful, and consistent for the model to be trustworthy. In certain applications, e.g., medicine or policy making, the developer of an algorithm may notice that it is learning relationships that are not of interest for the purpose of the model, or they can even be known to be incorrect relationships, as it was the case for the mentioned biased algorithms. Thus, relevancy should be closely analyzed in order to assess whether or not it is fair and reliable. Focusing on relevancy can result in a loss of accuracy, as some relationships may be artificially concealed to the model but can also yield benefits such as when used to improve feature engineering [28].

2.2.4 Interpretable machine learning algorithms

This section will introduce some of the most common interpretable machine learning algorithms, i.e., linear regression, logistic regression, decision trees, and decision lists.

2.2.4.1 Linear regression

Linear regression (LR) was one of the first techniques to be used to predict outputs and extract relationships with their inputs, long before the concepts of ML and, of course, interpretable IML took hold. These models assume the output variable (y) to be a linear combination of the input features (x_j), and can therefore be formulated as

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j, \tag{27}$$

where $\beta_j, j \in (1, p)$ are the feature weights, and β_0 is a constant term known as the intercept. Input features are generally quantitative variables and transformations of these variables (square-root, logarithm, exponentiation, etc.), but can also take the form of binary variables, qualitative inputs encoded into numeric or *dummy* variables, or interactions between variables [22].

The most popular technique to estimate the LR parameters from (27) is the ordinary least squares method, in which coefficients $\boldsymbol{\beta} = \left(\beta_0, \beta_1, \dots, \beta_p\right)^T$ are computed to minimize the residual sum of squares (RSS), i.e., the sum of the squared differences between actual and estimated outputs, of the training set [22]:

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i - f(\boldsymbol{x}_i))^2 = \sum_{i=1}^{N} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 \tag{28}$$

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \right) \right)^2 \tag{29}$$

Formula (29) allows obtaining the optimal feature coefficients while imposing additional terms and conditions to problem formulation. For instance, *shrinkage methods* such as lasso and ridge regression include in (29) the terms $\lambda \sum_{j=1}^{p} \left|\beta_j\right|$ and $\lambda \sum_{j=1}^{p} {\beta_j}^2$, respectively, to penalize the value of the feature weights. λ is known as the complexity parameter. These extensions of the original LR formulation can be useful to avoid the disproportionately high feature weights that can be obtained when input features are strongly correlated, to decrease the number of features to be used by the model (favoring interpretability), and to reduce overfitting in cases where the number of variables p is greater than the number of observations N [22].

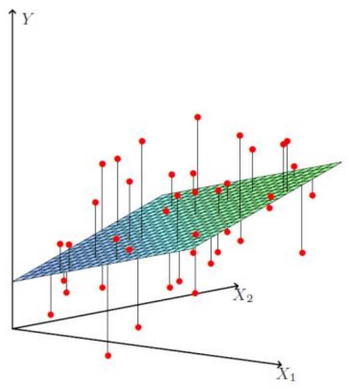

Figure 6: Least squares linear regression with $\boldsymbol{X} \in \mathbb{R}^2$ [20].

Regarding the interpretability of LR, the main advantages of the model are its simplicity as well as its transparent, linear relation between inputs and output, which is expressed through the model coefficients. Nevertheless, depending on the data type of each feature, the interpretation of the coefficients has slight differences, which must be considered [6]:

- Numerical feature: The output of the model changes by a factor equal to the feature weight when the input feature is increased in one unit. Ordinal categorical features encoded as integers can also be understood in the same way.
- Binary feature and one-hot-encoded categorical feature: The feature weight expresses the output variation when the variable takes value 1 compared to the situation where its value is 0.
- Intercept β_0: The intercept parameter is a particular case that can be understood as the weight of a *constant feature*, equal to 1 for all samples. It can be interpreted as the predicted value of an instance where all features are zero (even if this situation was not possible).

However, the validity of LR models is subject to certain assumptions regarding the data that may not hold in particular problems. These assumptions are linearity between features and the target, normality of the distribution of the target value relative to input features, homoscedasticity (homogeneous variance of the error over the feature space), independence of instances, and absence of multicollinearity between input features. Therefore, LR models are generally not suited to model complex, nonlinear relationships, and its coefficients may not result intuitive when features present high correlation levels [6].

2.2.4.2 Logistic regression

When the task to be carried out is a classification problem, logistic regression (LogR) is one of the most straightforward approaches that can be used. The logistic regression models a linear relationship between inputs and the logit transformation of the probability of belonging to each of the K classes except from one (since all the probabilities must add up to 1), as defined in (30). The left-hand side of the expression is also known as the *log odds*, which represents the logarithm of the *odds* function, i.e., the probability of the event divided by the probability of a different event) [22].

$$\log \frac{\Pr\,(Y = k|X = \boldsymbol{x})}{\Pr\,(Y = K|X = \boldsymbol{x})} = \beta_{k0} + \boldsymbol{\beta}_k^T \boldsymbol{X}, \quad k = 1, \dots, K-1 \tag{30}$$

Consequently, predicted probabilities can be expressed as follows [22]:

$$\Pr\,(Y = k|X = \boldsymbol{x}) = \frac{\exp(\beta_{k0} + \boldsymbol{\beta}_k^T \boldsymbol{x})}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \boldsymbol{\beta}_l^T \boldsymbol{x})}, \quad k = 1, \dots, K-1 \tag{31}$$

$$\Pr(Y = K|X = \boldsymbol{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \boldsymbol{\beta}_l^T \boldsymbol{x})} \tag{32}$$

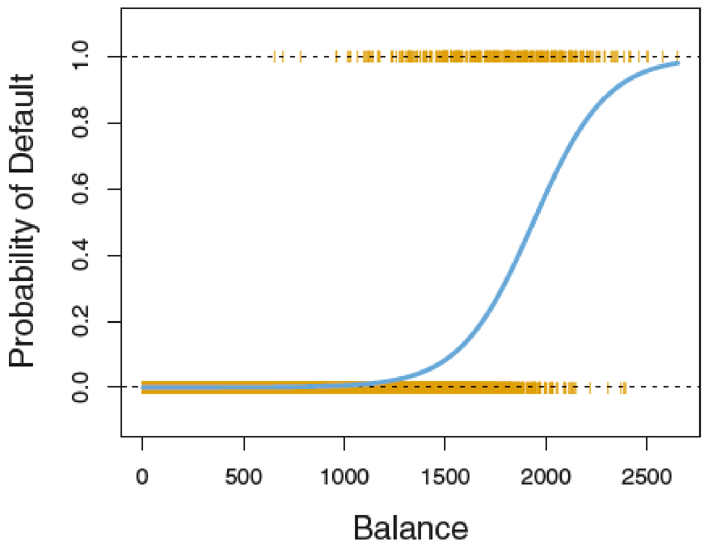

Figure 7: Logistic regression function vs. actual labels for a one-variable problem.

The estimation of the coefficients in logistic regression models is usually carried out through the maximum likelihood method. First, we define the log-likelihood for a set of N instances as

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p_{y_i}(\boldsymbol{x}_i; \boldsymbol{\theta}), \tag{33}$$

where $\boldsymbol{\theta} = \{\beta_{10}, \boldsymbol{\beta}_1^T, \ldots, \beta_{K-1,0}, \boldsymbol{\beta}_{K-1}^T\}$ comprises the entire set of parameters and $p_k(\boldsymbol{x}_i; \boldsymbol{\theta}) = \Pr(Y = k|X = \boldsymbol{x_i}; \boldsymbol{\theta})$ represents the conditional likelihood of Y given X considering the parameter set $\boldsymbol{\theta}$ [22].

Focusing on the binary classification ($y \in \{0,1\}$), which is the relevant case for the purpose of this project, (33) can be expressed as

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{i=1}^{N} \{y_\text{i} \log p(\boldsymbol{x}_i; \boldsymbol{\beta}) + (1 - y_\text{i}) \log(1 - p(\boldsymbol{x}_i; \boldsymbol{\beta}))\} \\ &= \sum_{i=1}^{N} \{y_\text{i} \boldsymbol{\beta}^T \boldsymbol{x}_i - \log(1 + e^{\boldsymbol{\beta}^T x_i})\}, \end{aligned} \tag{34}$$

assuming that the input vector $\boldsymbol{x}_i$ includes a constant term equal to 1 to integrate the intercept.

In order to maximize the log-likelihood, we must differentiate (34) and set it to zero (35) so as to compute the optimal coefficients of the model.

$$\frac{dl(\boldsymbol{\beta})}{d\boldsymbol{\beta}} = \sum_{i=1}^{N} \boldsymbol{x}_i(y_i - p(\boldsymbol{x}_i; \boldsymbol{\beta})) = 0 \tag{35}$$

The logistic regression shares most of the advantages and disadvantages with linear regression. Moreover, it is important to stress that the outcome of the logistic regression model is not a specific class but the probability of belonging to each of the possible classes. The main benefit of this kind of prediction comes from the fact that the output probability threshold to decide whether an instance belongs to each of the categories can be selected, which allows to intentionally reduce the most undesired kind of error (false positives or false negatives) [22].

Finally, the logistic regression model is less intuitive than linear regression since inputs are not linearly related to the output odds but their log-odds. Again, depending on the data type of each feature, the interpretation of the parameters is different [6]:

- Numerical feature: A change of a feature in one unit varies the log-odds by a factor of β_j, and the estimated odds by $\exp(\beta_j)$.
- Binary feature and one-hot-encoded categorical feature: Switching from the reference category to the other (or another) one varies the output odds by a factor of $\exp(\beta_j)$.

- Intercept β_0: The intercept represents the odds of an instance where all features are zero, although it is not usually relevant.

2.2.4.3 Decision trees

As opposed to the previous algorithms, decision trees (DTs) are more suited to model nonlinear relationships between inputs and outputs as well as the interaction between features and can be used both for regression and classification problems [6].

Figure 8 illustrates a decision tree trained for a regression task. Decision trees consist of a series of splitting decision rules (known as *internal nodes*) that successively divide the feature space into multiple complementary sub-regions (R_1, …, R_5 in the left figure). These partitioned regions are known as the *terminal or leaf nodes* of the decision tree, and an output value is assigned to each of them, i.e., the expected value of observations belonging to each terminal node. In regression tasks, this output is numerical, and it is usually computed as the average of the target values from the training observations that fell into that feature space partition. In contrast, classification decision trees assign to each of the leaf nodes either the mode of these observations or a vector containing the proportion of instances belonging to each of the possible classes [20].

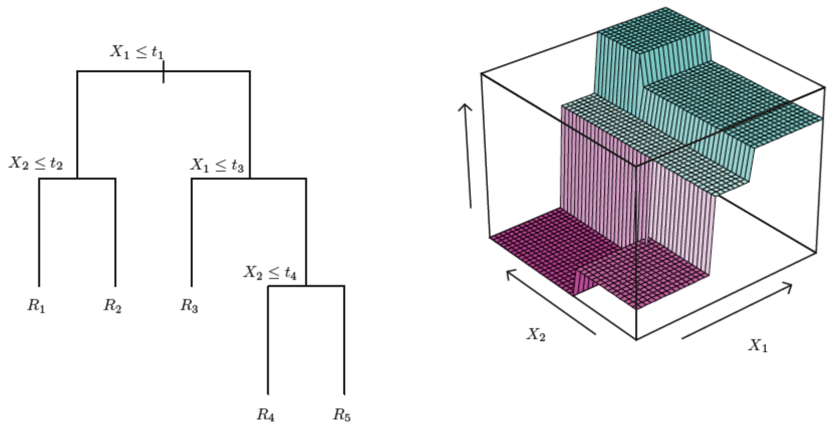

Figure 8: Representation of the decision tree rules (left) and the resulting space partition (right) for a two-dimensional regression task [20].

The building of DTs through the successive splits is carried out by selecting the splits that minimize their target loss function, i.e., the RSS (36) in the case of regression tasks, and the Gini index (37) or cross-entropy (38) functions for classification problems, until a given stopping criteria is met. The stopping criteria can be based on multiple parameters such as the maximum number of leaf nodes, maximum depth of the tree, minimum error decrease, and minimum samples per leaf. These are known as hyperparameters, and they are used to control the learning process of the algorithm [20].

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 \tag{36}$$

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{37}$$

$$\mathrm{D} = -\sum_{i=1}^{N} \hat{p}_{mk} \log \hat{p}_{mk} \tag{38}$$

The principal advantage of decision trees is their intrinsic interpretability since they can be easily represented as a graph (Figure 2 left). In fact, their sequential decision nature is closer to human decision-making than any other machine learning algorithm, which makes them valuable models to be used within the context of IML. Besides, they can handle complex relationships and interactions, any type of input data, and can be used both for regression and classification tasks [20].

Nevertheless, the intrinsic interpretability of DTs is relatively limited by their own size, i.e., trees with a very large number of input features, leaf nodes, or depth may become challenging to represent and to be understood as a whole. As a result, the selection of the abovementioned hyperparameters is not only a critical task to maximize the accuracy of the model but also to ensure its transparence and interpretability [6].

In addition to explaining how outputs are obtained, decision trees can transparently show how each of the decisions contributed to reducing the training error, and thus, how relevant were the associated features for the model. This information can be extracted in aggregate for each input feature using the feature importances tool, introduced in section 2.2.3, which provides a general insight into how much these variables contribute to explaining the target variable. Decision trees' (and their derivate algorithms) feature importances tool is so simple and effective that it is extensively used as an embedded method for feature selection purposes when building ML models with high-dimensional datasets [32].

Finally, DTs may not be the best choice for certain tasks, such as when the outcome has a linear relationship with the input (Figure 3), and can easily suffer from the overfitting of the training set, which usually harms the predictive accuracy of these algorithms. Several tree-based algorithms have been developed to overcome these limitations, e.g., random forests and boosted gradient decision trees, which rely on bagging and boosting methods, respectively. Compared to simple decision trees, these extended algorithms can achieve significant performance improvements, but they also involve a critical loss of interpretability that virtually excludes them from being used as interpretable machine learning models [20].

2.2.4.4 Decision lists

Decision lists are a series of hierarchically sorted decision rules, equivalent to conditional statements in computer programming, which can generally be applied to both regression and classification tasks. Each decision rule consists of an *if (else if)* statement with a single condition or logical conjunction of conditions, known as the antecedent, and a *then* statement, the prediction. To make a prediction, decision rules are sequentially applied until one of the *if* statements is satisfied, and the corresponding prediction value is assigned to the instance. If none of these rules apply to a specific instance, a default value is assigned. This is known as the default rule, which is equivalent to an *else* statement [6].

The usefulness of individual decision rules can be measured by two elements. On the one hand, the support or coverage of a rule measures the percentage of training instances that have been matched by it. On the other hand, the accuracy or confidence of a rule measures how accurately does the rule predict the instances to which it applies. Both measures usually face a similar accuracy-precision trade-off as the one faced by decision trees. Increasing the number of rules could improve their accuracy but would decrease their support [6].

The concept behind decision lists is very similar to decision trees in that they both apply a sequence of rules which split the feature space and assign an outcome to each partition. In fact, any decision tree can be expressed as a decision list. However, decision tree's partitions are mutually exclusive and collectively exhaustive by definition, meaning that they do not overlap and cover the entire feature space. In contrast, rules from decision lists could overlap, a conflict that is solved thanks to the hierarchy of decision rules, and the exhaustivity is generally achieved through the application of the default rule [6].

The actual implementation of decision lists can vary significantly depending on how decision rules are selected, but three of them are the most widespread. The first approach is OneR, which only uses the most explanatory feature from the dataset and creates a decision rule for each feature value. It only supports categorical inputs, and it has been designed for classification tasks but could deal with continuous features and targets if these are discretized into intervals [6].

The process followed by sequential covering, on its behalf, is closer to the one followed by decision trees, and it is perhaps the most intuitive. In this case, the algorithm finds the best rule based on the training dataset and computes the prediction for the rule. Then, the samples to which the rule applies are removed, regardless of how accurate the rule prediction was. This procedure is sequentially repeated until all the training data has been covered or a specific stop condition is met. The way in which the sequential covering selects each rule can take multiple forms depending on the algorithm employed, but, in any case, some requirements concerning the support or accuracy of decision rules are needed in order to define how to find optimal partition at every step [6].

Figure 9 illustrates the learning process of a sequential covering decision list, which provides a clear example of how these models work. Besides, the non-exclusivity and non-exhaustivity of decision rules can be observed [6].

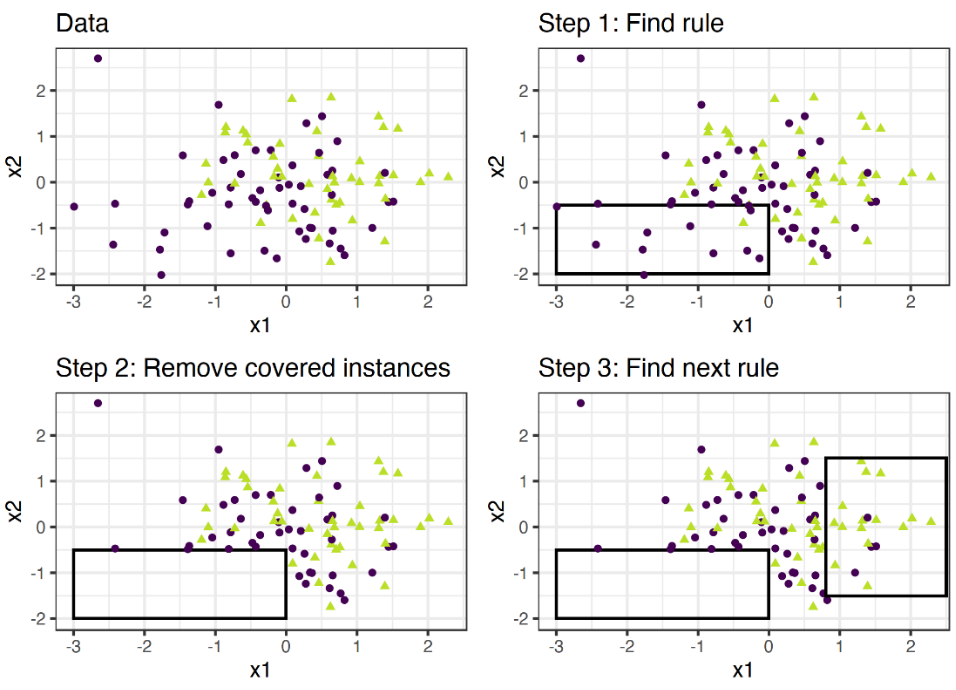

Figure 9: Decision rule learning process of a sequential covering decision list [6].

The third approach is known as Bayesian rule lists, which is the one employed by C. Rudin in [30]. This kind of decision lists first pre-mine a number of frequently occurring patterns present in the training data. These *prior* patterns can be obtained either from single feature values or a combination of them, and their frequency is measured by their support in the dataset. Then, the Bayesian rule list algorithm builds an accurate decision list by selecting the pre-mined conditions and outcomes that maximize the *posterior* probability of the resulting model through an iterative process, which prioritizes shorter lists and conditions.

Decision lists' main advantage is also their interpretability, which is comparable to that of decision trees. They can even yield more compact results than decision trees when these suffer from imbalanced and duplicated sub-trees, typically in classification tasks. However, they are only manageable if the number of rules and conditions is relatively small, which is the same complexity issue as seen for decision trees [6].

Finally, despite the fact that decision lists can be the best option for certain tasks, they present important limitations. Their main drawback is that most implementations require inputs to be categorical, which drastically restricts their range of application. Besides, even though they can be applied to regression tasks, they are usually more suited for classification problems since the number of possible outputs is directly related to the number of decision rules [6].

2.3 Unit commitment & machine learning

The challenges that have arisen in recent years, mainly due to the increase in the size and complexity of power systems, as well as the penetration of renewable energies and other sources of uncertainty, have led to an increasing difficulty in generating optimal generation schedules at a reasonable computational cost. However, the importance of resource planning in the electricity sector has opened the door to numerous innovative studies aimed at developing more computationally efficient models, without renouncing to a detailed modeling of these systems, in order to generate economically efficient schedules.

Unsurprisingly, one of these research lines has been the application of machine learning techniques to the unit commitment problem. ML has a wide range of applications closely linked to this subject, such as the prediction of electricity generation from non-dispatchable renewable sources, which allows reducing the uncertainty of the problem and, thus, obtaining more efficient results [5], [33]. Nonetheless, the application of ML to the resolution of the UC with the aim of totally or partially avoiding the computational cost of solving the optimization problem every time a new planning is required is the one of greatest interest in the context of this project.

The existing literature on the topic comprises a wide variety of approaches. The majority of these research projects address the UC problem as a supervised learning task, in which precomputed solutions of the optimization problem are used to train an ML model with the aim of replicating them. With few exceptions attempting to develop strictly surrogate models [34], the general purpose of these works is not to create ML models that can be used as a solver themselves, but rather to provide tools that can assist the optimization problem. According to [35], this approach usually yields the best results.

Most of the proposed methods rely on the development of ML models trained to generate complete or partial solutions to the UC problem, which are then used as starting points for the optimization problem. The result in all the proposed approaches is a substantial reduction of the computation time is achieved without significantly moving away from the optimal solution of the problem.

Among the existing supervised learning algorithms, artificial neural networks stand out as the most popular ones to address this task. For instance, a genetic-based ANN model is developed in [36] to compute a set of preliminary solutions that are subsequently optimized to obtain the optimal generation planning for a thermal power system. In [37], ANNs are used to predict the commitment of generating units, and dynamic programming is employed to assist the ML model for units where the predicted outcome is not certain enough. Similarly, an ANN is developed in [38] to predict the discrete variables of the problem, e.g., the commitment of the thermal generators, and an optimization approach is applied to obtain the continuous ones, e.g., the production of generating units.

Alternative proposals which are not based on neural networks can be found in [39] and [40], where a multi-target random forest and a k-nearest neighbors regression models are developed, respectively, to obtain warm-start solutions for the UC problem. This last study also proposes an alternative approach, where another k-nearest neighbor algorithm is developed to predict the transmission constraints that should be included in the optimization problem, leaving out the non-critical ones. A similar method is explored in [35], but extending it to a wider variety of constraints.

Besides the supervised learning approach, different models have been developed addressing the UC problem from other perspectives. On the one hand, unsupervised learning can be used to reduce the dimension of the problem and thus save execution time. For instance, the model proposed in [41] addresses the UC problem under uncertainty by clustering the given scenarios and applying the UC optimization model to these clusters. The result is an intermediate solution between stochastic and scenario-based unit commitment, which can significantly reduce the size of the problem. In [42], the clustering of decision variables such as the commitment of multiple units is performed without significantly increasing the total cost of the generation planning compared to the base formulation.

On the other hand, there are multiple studies, such as those developed in [43] – [46], that address the resolution of the UC using reinforcement learning algorithms, widely used in the field of optimization. Besides the reduction of execution time, the main advantage of these approaches is that they can generate feasible and efficient solutions without the need for a precise UC model definition and, in any case, without having to generate the precomputed scenarios that are necessary for other approaches [46].

Finally, this project seeks to point out that, despite the abundance of existing studies related to the unit commitment problem and, in particular, to the application of ML for its resolution, none of them has targeted the development of models that can be used to explain the solutions generated by UC models. The ML models that are usually used to replace or assist the UC model, and the optimization problem itself, are conceived as black boxes whose only purpose is to obtain the optimal generation schedules, regardless of their underlying behavior. Consequently, the application of interpretable machine learning to this subject constitutes a virtually unexplored horizon so far.

3 Case study description

The development of an interpretable machine learning model applied to the Unit Commitment problem requires the elaboration of a reasonable number of scenarios, as well as their corresponding resolution, in order to build a data set with which to train the model and evaluate its performance. Hence, the main input features that will be used to carry out the optimization will be employed to build the input set of the interpretable machine learning model. Lastly, this information will be used to predict some of the most relevant decision variables extracted from the optimal solutions of the proposed scenarios.

In the following, the main elements of the case study will be described, as well as the mathematical formulation of the Unit Commitment problem.

3.1.1 Power system and wind generation scenarios

The power system analyzed in this project consists of a modified IEEE 118-bus test system, presented in Figure 10, that allows the consideration of the electricity production of multiple wind farms throughout the system, and which has been widely used in UC studies, for example, [47], [48], and [49]. The primary source of data used to model the power system is [50]. This modified IEEE 118-bus system comprises 118 buses, 186 transmission lines, 54 thermal units, 91 loads, and three wind units (base case).

All detailed parameters, such as generator characteristics, transmission network, load distribution profile, system-wide power demand, and wind power scenarios, are available in [51].

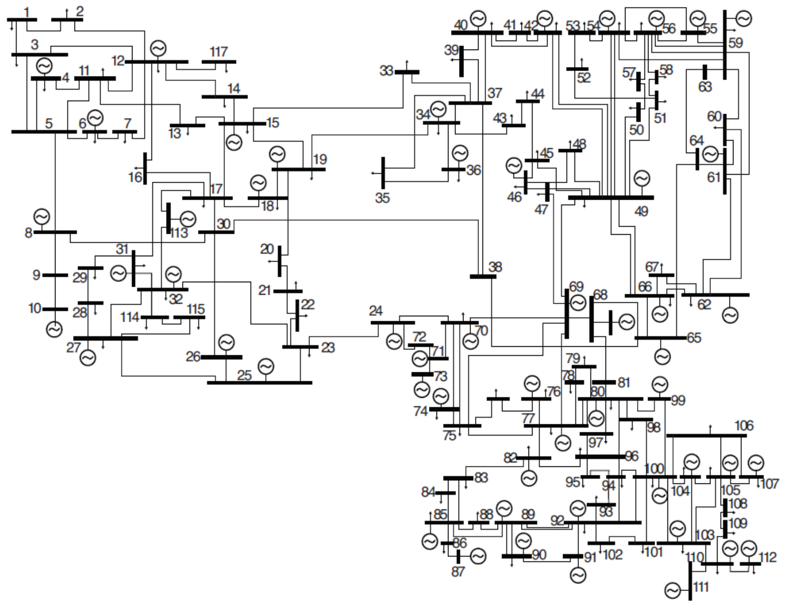

Figure 10: Unifilar diagram of the IEEE 118-bus power system.

The operation planning will be carried out for a 24-hour horizon, divided into 24 hourly periods. A common load profile has been defined for all scenarios within this time horizon, with an average and maximum level of 3991 MW and 5592 MW, respectively, as shown in Figure 11. This aggregate system demand is distributed among the 91 loads of the power system, according to the load factor assigned to each node. Furthermore, it is worth noting that the cost of not satisfying part of the demand ($CPNS$) has been set to 10000 €/MWh.

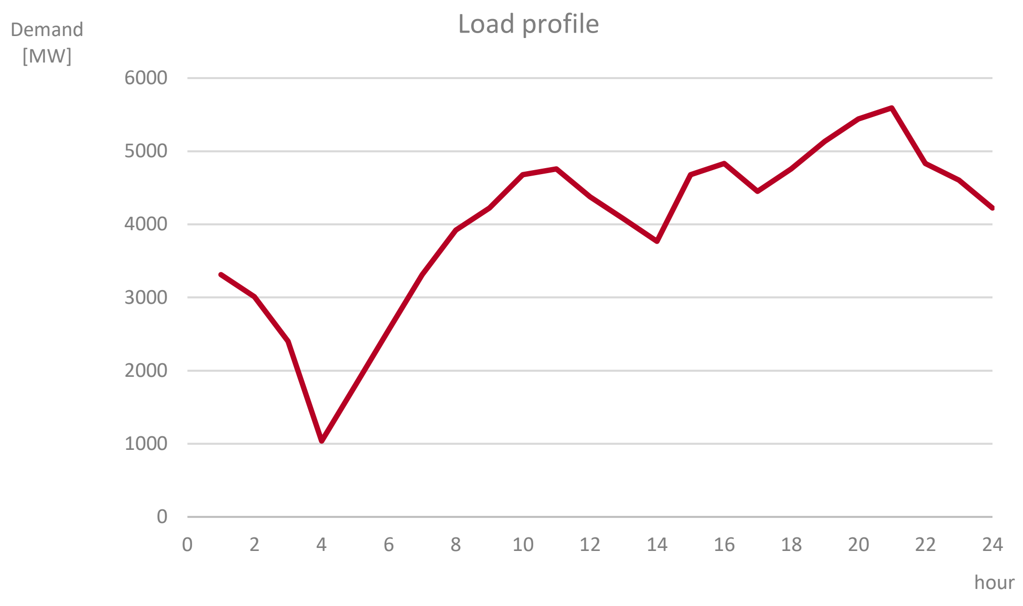

Figure 11: Case study’s system load profile.

The differentiation between the case study’s scenarios lies in the generation profile of wind units. 365 wind generation scenarios have been considered, using the hourly wind capacity factors for different Spain's regions, based on MERRA-2. These scenarios simulate the present-day fleet of wind farms, the near-term future, and long-term future fleets, as described in [52].

Consequently, the base case, in which ten wind units are spread over the system, presents an aggregated average and maximum production of 1713 MW and 4848 MW, respectively. In addition to this base case, a low wind penetration case with another 365 scenarios and three units has been elaborated in order to analyze the impact of wind penetration in the model performance. The aggregated average and maximum production of this alternative case are 584 MW and 1522 MW, respectively. Figure 12 and Figure 13 illustrate the aggregate hourly wind generation distribution and average wind generation per node, respectively, for both wind penetration cases.

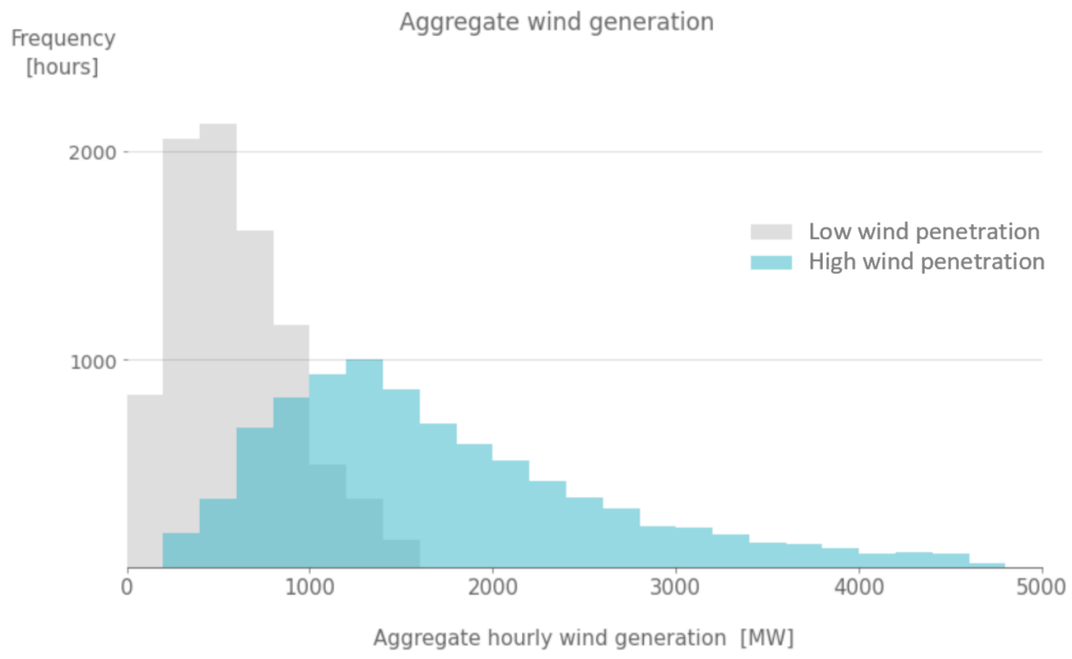

Figure 12: Aggregate wind generation comparison between the low and the high wind penetration cases.

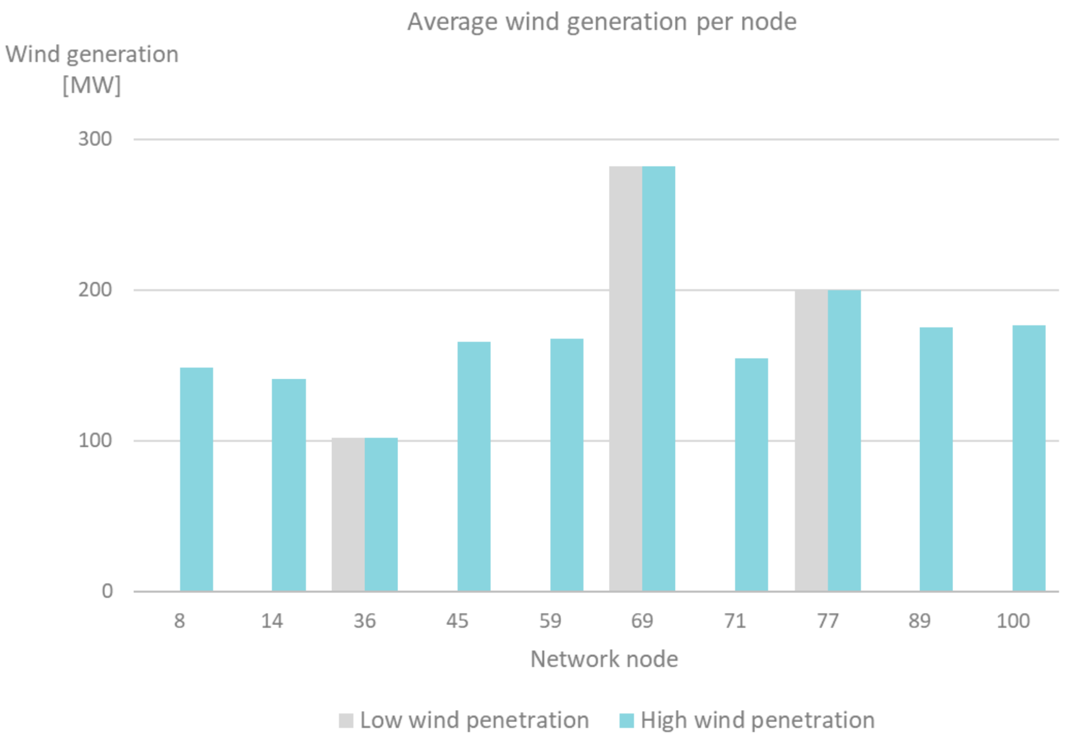

Figure 13: Average wind generation per node in the low and high wind penetration cases.

3.1.2 Case study UC formulation

The UC model used to generate the optimal solutions for this study (39) is based on the Tight and Compact formulation described in [2], and the resolution of the optimization problems has been carried out using GAMS software.

The objective function of the UC problem considers the fixed and variable costs of thermal electricity generation, startup, shutdown costs, non-served energy, and secondary reserve costs.

Moreover, the formulation of the problem includes the following constraints and equations: generation and demand balance; up and down secondary reserve requirements; maximum power flow through transmission lines; and technical constraints of generators, which encompass maximum and minimum production limit constraints, total output and commitment logic equations, up and down ramping constraints, and minimum up and down time constraints.

Finally, the impact of the transmission network on the overall performance of the model will be assessed by comparing the results obtained when network constraints (25) (26) are included in the UC formulation and when the network is modeled as a single node (copperplate network approach). This comparison will be carried out only for the base (high) wind penetration case.

$$\min \sum_{nt} CF_t uc_{nt} + \sum_{nt} CV_t p_{nt}^T + \sum_{n} CPNS_n pns_n$$
$$+ \sum_{nt} CSU_t su_{nt} + \sum_{nt} CSD_t sd_{nt} + \sum_{nt} CRU_t ru_{nt} + \sum_{nt} CRD_t rd_{nt}$$

$s.t.$

$$\sum_t p_{nt}^T + \sum_i IG_{ni} + \sum_i pns_{ni} = \sum_i D_{ni} \quad \forall n$$

$$\sum_t ru_{nt} \geq DU_n \quad \forall n$$

$$\sum_t rd_{nt} \geq DD_n \quad \forall n$$

$$uc_{nt} - uc_{n-1,t} = su_{nt} - sd_{nt} \quad \forall nt$$

$$\sum_{n'=n+1-TU_t}^{n} su_{n't} \leq uc_{nt} \quad \forall nt$$

$$\sum_{n'=n+1-TD_t}^{n} sd_{n't} \leq 1 - uc_{nt} \quad \forall nt \tag{39}$$

$$p_{nt}^T = p_{nt} + \underline{P}_t uc_{nt} + \sum_{n'=1}^{TSD_t} PSD_{n't} sd_{n-n'+1,t} + \sum_{n'=1}^{TSU_t} PSU_{n't} su_{n-n'+1+TSU_t,t} \quad \forall nt$$

$$p_{nt} + ru_{nt} \leq uc_{nt}\left(\overline{P}_t - \underline{P}_t\right) - sd_{n+1,t}\left(\overline{P}_t - SD_t\right) - su_{n+1,t}\left(\overline{P}_t - SU_t\right) \quad \forall nt$$

$$p_{nt} - rd_{nt} \geq 0 \quad \forall nt$$

$$p_{nt} - p_{n-1,t} + ru_{nt} \leq RU_t \quad \forall nt$$

$$p_{nt} - p_{n-1,t} - rd_{nt} \geq -RD_t \quad \forall nt$$

$$\underline{F}_k \leq \sum_{i \neq s} \tilde{Q}_{ki} \left(\sum_{t \in \Omega_i} p_{nt} + IG_{ni} - D_{ni} \right) \quad \forall nk$$

$$\overline{F}_k \geq \sum_{i \neq s} \tilde{Q}_{ki} \left(\sum_{t \in \Omega_i} p_{nt} + IG_{ni} - D_{ni} \right) \quad \forall nk$$

4 Alignment with Sustainable Development Goals

The Sustainable Development Goals (SGDs) are a set of 17 global goals established by the United Nations General Assembly in 2015 [53], aiming to “achieve a better and more sustainable future for all by 2030.” Two years later, specific targets and indicators were assigned to each goal in order to facilitate addressing these objectives and measuring the achieved impact [54].

This project seeks to shed some light on understanding such a studied optimization problem as the unit commitment. The possibility of developing models that can explain how optimal dispatches are obtained and which are the critical variables of the problem can become a valuable tool for decision-making in the path towards a more sustainable power sector.

Subsequently, since the power sector is a cornerstone to fight climate change and achieve global sustainable development, this project can be associated with three interrelated Sustainable Development Goals:

- **Goal 7: Affordable and clean energy**

  This goal seeks to “ensure access to affordable, reliable, sustainable, and modern energy for all.” To achieve this purpose, it will be necessary to increase the global share of renewable generation through increasing investment in clean energy infrastructure and energy efficiency. However, the high integration of renewable generation sources into power systems often requires significant network reinforcement and can entail challenging stability issues due to its intermittent nature. These kinds of issues make system operation even more necessary than before. Therefore, understanding how integrating more intermittent generation into power systems can impact the behavior and optimal operation of the system can lead to more efficient, fair, and reliable decisions.

- **Goal 11: Sustainable cities and communities**

  SDG 11 encourages countries to “make cities and human settlements inclusive, safe, resilient, and sustainable.” Many different dimensions are considered by this goal, among which achieving more sustainable cities and reducing their environmental impact are the ones that can be associated with this project. Distributed generation will play a key role in decarbonizing cities. Similar to the previous goal, integrating these resources will require significant investments and an efficient operation of power systems. Consequently, developing models to explain how UC optimal solutions are obtained will also contribute to this Sustainable Development Goal.

- **Goal 13: Climate action**

  Finally, the climate action goal urges countries to “take urgent action to combat climate change and its impacts by regulating emissions and promoting developments in renewable energy.” As the United Nations described this goal, one of the essential pillars for combating climate change is reducing greenhouse gas emissions through the transition to renewable generation sources. Therefore, this objective is directly linked to what is indicated in SGD 7, which is committed to enhancing clean energy sources, and everything that was mentioned for that goal is also applicable in this case.

7 AFFORDABLE AND CLEAN ENERGY
11 SUSTAINABLE CITIES AND COMMUNITIES
13 CLIMATE ACTION

Figure 14: Main Sustainable Development Goals related to the project (7, 11, and 13).

5 Methodology

The prediction of UC optimal solutions can be addressed from multiple perspectives, as described in section 2.2. However, since we would like our model to explain the resolution of the unit commitment problem and how it comes up with its solutions, we will approach the problem as a supervised learning task, relying on the algorithms presented in section 2.2.4. With this purpose, we will implement a set of regression and classification algorithms in order to predict the most relevant outputs of the unit commitment, such as the commitment and production of thermal generators, or which power lines are constrained.

In a broad sense, we propose developing *surrogate* models, relatively aligned with the concept defined in section 2.2.3 but whose purpose is to replace an optimization model instead of a complex ML algorithm. The models therein described aim to replicate the behavior of another ML model (a black box), which has been trained on original data. In such cases, there is the possibility to directly use an interpretable model instead, eliminating the need for an auxiliary algorithm. However, this possibility does not exist in the case of the unit commitment since its optimal solutions obtained by solving the corresponding optimization problem are necessary to train the ML model. The only way to avoid an explanatory model relying on an original UC model would require implementing a transparent optimization model that can naturally explain how it generates the optimal schedules, which not only is outside the scope of this work but also entails exploring a completely new branch of optimization.

Therefore, the outputs of the UC model can be considered the real outcomes we are seeking to predict (and not only to explain), meaning that the development of an IML model applied to the UC problem shares characteristics with both intrinsic interpretable and surrogate models.

In this chapter, the methodology followed to implement such interpretable models will be described, including the processing of the necessary dataset. The entire work has been entirely carried out in Python 3.8.5, mainly relying on *NumPy*, *pandas*, and *scikit-learn* libraries.

5.1 Data processing

5.1.1 Input variables

Original data used to train machine learning models seldom presents the necessary format to carry out the intended task and, therefore, require to be previously processed. This is a critical stage in the elaboration of ML models as it can considerably impact their capability to infer relationships from the input data and, in the end, the performance of the models.

Data processing usually includes multiple steps such as outlier detection, instance selection, handling of missing values, feature normalization and transformation, feature selection, or feature extraction [55]. In the context of this project, the input data to be used by the model will need to be built from the input information generated for the optimization problem (section 3.1.1), which is already a clean dataset. Therefore, we can skip the first three of the mentioned steps and directly dive into the remaining ones.

From all the information employed by the model, e.g., intermittent generation, demand, thermal generators' characteristics, and network parameters, only the inputs varying across the different time periods and scenarios will provide useful information for the model. Intermittent generation (IG_{sni}) and demand (D_{ni}) both satisfy this requirement and, thus, will be selected as the basis for our dataset. This means that the model will not be provided with knowledge about the parameters of the network topology and its generators, even though they have a decisive influence on the resulting solutions. Consequently, the IML model will be network-dependent, and its validity will be restricted to the original parameters employed to solve the scenarios from which the ML model will be trained.

We will then use these variables to compute the net demand at each of the nodes where there is a wind generation unit ($\boldsymbol{ND^i}$), which constitutes a more suitable set of features for the model since it represents the energy that needs to be fulfilled by thermal generators (40). The total net demand of the system ($\boldsymbol{ND^T}$) will also be included in the input features (41) since this is likely to become a significant variable of the model due to its relevance in the solution of the UC problem when no network constraints are active. Moreover, this will be the feature employed to build the input matrix for the single-node UC problem, as the nodal decomposition of the net demand is not relevant for this task.

$$ND_{sn}^{i} = D_{ni} - IG_{sni} \qquad \forall s, n, i \in \Omega_{IG} \tag{40}$$

$$ND_{sn}^{T} = \sum_{i} D_{ni} - \sum_{i} IG_{sni} \qquad \forall s, n \tag{41}$$

Each of the resulting features presents a different range of values, particularly the system net demand. This variety of feature scaling can negatively impact the performance of several machine learning algorithms and should be normalized [56]. In this case, only the scaling of the input features has been carried out by dividing by the maximum absolute value of the feature since we are interested in preserving the sign of the original variable:

$$nd_{sn}^{i} = \frac{ND_{sn}^{i}}{\max\limits_{s',n'}\left(\left|ND_{s',n'}^{i}\right|\right)} \qquad \forall s, n, i \in \{T, \Omega_{IG}\} \tag{42}$$

With these scaled features, we could already build a simple input matrix with which to train our machine learning model. However, the temporal interdependence between time periods is one of the main challenges of UC problems, especially given the ramps and trajectory constraints that are being modeled in the presented case study. This is, in fact, the major difficulty faced by surrogate models that need to predict each hour separately as if they were not related to each other.

To mitigate this problem, information from adjacent hours can be provided to the model so as it can capture intertemporal relationships. More precisely, we will include the variation of the net demand at the three previous and three following periods relative to the net demand at the period for which the output will be predicted (43). This will be carried out for both wind generation nodes ($\boldsymbol{\Delta ND^{i,j}}$) and the whole system ($\boldsymbol{\Delta ND^{T,j}}$).

The problem with shifting the net demand features is that there is not information to perform these calculations for the first and last periods of the scheduling horizon. This is not an issue for certain machine learning algorithms such as decision trees and other tree-based ensembles, but most algorithms cannot handle missing data, e.g., linear regression and logistic regression.

Consequently, depending on the model that will be used to predict the desired outcomes, we will follow different imputing strategies: if the model can deal with missing data, we will assign a NaN value, representing that the true value is missing. Otherwise, we will impute 0 –this is equivalent to assuming that the net demand is kept equal to the corresponding first or last period in the periods outside the optimization horizon, which is the best assumption we could make.

$$
\begin{aligned}
&if\ (n+j \geq 1)\ and\ (n+j \leq 24): \\
&\qquad \Delta ND_{s,n}^{i,j} = ND_{s,n+j}^{i} - ND_{s,n}^{i} \qquad \forall i \in \{T, \Omega_{IG}\} \\
&else: \\
&\qquad \Delta ND_{s,n}^{i,j} = IMP \qquad \forall i \in \{T, \Omega_{IG}\} \\
&\qquad\qquad \forall s, n, j \in \{\pm 1, \pm 2, \pm 3\}
\end{aligned}
\tag{43}
$$

After constructing these new features, we have built an input matrix (X^{NC}) where rows are identified by the corresponding scenario and time period, and columns correspond to all the net demand features that have been created. For the single-node problem, only system-wide features will be needed to build the input matrix (X^{SN}).

$$
\begin{aligned}
X^{NC} &= \underset{i \in \{T, \Omega_{IG}\}}{concat}([\boldsymbol{\Delta ND^{i,-3}}, \boldsymbol{\Delta ND^{i,-2}}, \boldsymbol{\Delta ND^{i,-1}}, \boldsymbol{ND^{i}}, \boldsymbol{\Delta ND^{i,1}}, \boldsymbol{\Delta ND^{i,2}}, \boldsymbol{\Delta ND^{i,3}}]) \\
X^{SN} &= [\boldsymbol{\Delta ND^{T,-3}}, \boldsymbol{\Delta ND^{T,-2}}, \boldsymbol{\Delta ND^{T,-1}}, \boldsymbol{ND^{T}}, \boldsymbol{\Delta ND^{T,1}}, \boldsymbol{\Delta ND^{T,2}}, \boldsymbol{\Delta ND^{T,3}}]
\end{aligned}
\tag{44}
$$

As a result, X^{SN} will be composed of 7 variables, whereas the number of input features obtained for the network-constrained UC problem (X^{NC}) will range from 28 in the low wind penetration scenarios (3 wind generation nodes) to 77 in the high wind penetration scenarios (10 wind generation nodes). To avoid the loss of interpretability of the IML model that this large number of features could imply, an initial feature selection will be included as part of the model in order to restrict its dimensionality.

5.1.2 Output variables and equations

The results obtained from the Unit Commitment optimization model comprise both the value of the decision variables of the optimal solution and the marginal value of the dual variables of the problem constraints. These results are presented in Table 1 and Table 2, which include the number of dimensions and values per scenario for each type of output, their description, and the unit in which they are defined.

Table 1: Output variables generated by the UC optimization model.

Variable	Number of dim.	Count (network)	Count (single node)	Description	Unit
v2ndResDW	2	1296	1296	2nd reserve down allocation	GW
v2ndResUP	2	1296	1296	2nd reserve up allocation	GW
vCirPF	4	4464	0	Power flow through a line	GW
vCommit	2	1296	1296	Commitment of the unit [0-1]	-
vConsump	2	1296	1296	Consumption of the unit	GW
vENS	2	2832	2832	Energy not served per node	GW
vIG	2	2832	2832	Intermittent generation per bus	GW
vProduct	2	1296	1296	Output of the unit	GW
vProduct1	2	1296	1296	Output of the unit above min. load	GW
vShutDown	2	1296	1296	Shutdown of the unit [0-1]	-
vStartUp	2	1296	1296	Startup of the unit [0-1]	-
vTotalVCost	0	1	1	Total variable cost of the system	M$

Table 2: Output equations obtained from the UC optimization model.

Equation	Number of dim.	Count (network)	Count (single node)	Description	Unit
e2ReserveDw	1	24		Downwards operating reserve req.	GW
e2ReserveUp	1	24		Upwards operating reserve req.	GW
eBalance	1	0	24	System load-generation balance	GW
eBalanceBus	2	2832	0	Load-generation balance per bus	GW
eCirPF	4	4464	0	Maximum power flow through a line	GW
eMaxOutput	2	1296		Maximum output of a committed unit	GW
eMinOutput	2	1296		Minimum output of a committed unit	GW
eMinTDown	2	1296		Minimum down time	-
eMinTUp	2	1296		Minimum up time	-
eRampDw	2	1296		Maximum ramp down	GW
eRampUp	2	1296		Maximum ramp up	GW
eTotOutput	2	1296		Total output of a committed unit	GW
eUCStrShut	2	1296		Commitment-startup-shutdown relationship	-

These outputs of the UC problem will be the targets of the IML models that will be described later. Therefore, we will need to adapt their format into one that suits the machine learning algorithm. The data extracted from the UC model presents the following generic structure, which is shared by both variables and equations:

Table 3: Output format example of the UC variables and equations.

Sce.	Result	dim1	dim2	dim3	dim4	Lower bound	Value	Upper bound	Marginal
sc001	vProduct1	h002	gen03			**0**	**0.015**	**0.025**	0
sc125	vCommit	h005	gen21			**0**	**1**	**1**	0.001
sc010	vCirPF	h017	bus108	bus109	c1	**-0.175**	**0.027**	**0.175**	0
sc200	eMinOutput	h012	gen09			0	0	inf	**0**
sc001	eRampUp	h001	gen09			-inf	0.020	0.025	**0**
sc365	eCirPF	h023	bus001	bus003	c1	2.796	2.796	2.796	**0.012**

According to the given structure, time periods will be defined by their scenario, and the first dimension, which is always the hour (except for the total cost of the system, *vTotalVCost*, not considered in this project). Then, each of the variables and equations whose type is reflected in the *Result* column are defined by the remaining dimensions, whenever they apply. It will be convenient to normalize these target variables so that we can handle together multiple variables of the same type, regardless of their original range. The normalization procedure that has been carried out varies depending on the type of target:

- **Variables**

Decision variables are expressed as values (v) that can range from their corresponding lower bound (LB_v) to their upper bound (UB_v), being both of them constant for each variable. Therefore, we will normalize these values employing their corresponding bounds:

$$v_i^{norm} = \frac{v_i - LB_v}{UB_v - LB_v} \tag{45}$$

The only exceptions will be binary variables, such as *vCommit*, since they do not need to be normalized, and *vCirPF*, as it is the only variable that can take both positive and negative values, and we would like to preserve its original sign. In this last case, the absolute value of both bounds is equal, and the normalized variable can be obtained by dividing its value by the corresponding upper bound:

$$v_i^{norm} = \frac{v_i}{UB_v} \tag{46}$$

- **Equations**

Regarding equations, we are interested in their *Marginal* value (m), not the value of the equation itself, which is the one located in the *Value* column. In this case, there are no bounds that can be used to normalize them, so we will divide the values of each equation (grouping them by their dimensions, not just the type of equation) by their maximum absolute value. This will also preserve the sparsity of the original marginal values. Additionally, if all the observed values for a given dual variable were zero, then we would just assign zero to the variable since it would mean that the constraint is never active.

$if\ \max_j(|m_j|) > 0$:

$$v_i^{norm} = \frac{m_i}{\max_j(|m_j|)} \tag{47}$$

$else$:

$$v_i^{norm} = 0$$

Furthermore, we are commonly more interested in determining whether or not constraints are active in the optimal solution of the UC problem rather than in their specific marginal value. Employing regression models to obtain these kinds of insights might not satisfactorily yield the desired results since non-active constraints would frequently be assigned close-to-zero values but not zero, and a certain threshold would be needed. Instead, it would be more effective to directly address the issue as a classification problem. For this reason, the dual variables of equations have also been converted to binary variables, creating an alternative target for the presented task.

$$
\begin{aligned}
&if\ m_i = 0: \\
&\qquad v_i^{norm} = 0 \\
&else: \\
&\qquad v_i^{norm} = 1
\end{aligned}
\tag{48}
$$

After normalizing the output data, the previous example would now present the structure displayed in Table 4. This structure will allow us to work indifferently with both variables and dual variables of equations. Thus, we will just use the term variables to refer to both of them from now on.

Table 4: Normalized output example of the UC variables and equations.

Sce.	Result	dim1	dim2	dim3	dim4	Norm. value
sc001	vProduct1	h002	gen03			**0.600**
sc125	vCommit	h005	gen21			**1**
sc010	vCirPF	h017	bus108	bus109	c1	**0.154**
sc200	eMinOutput	h012	gen09			**0**
sc001	eRampUp	h001	gen09			**0**
sc365	eCirPF	h023	bus001	bus003	c1	**0.128**

Finally, since we want the output of the ML model to be the value of a specific variable for a given time period, we would like the structure of the output data to fit the following scheme: *{Scenario, hour} x {Result, dim2, dim3, dim4}*. This can be easily achieved by unstacking the *Result, dim2, dim3,* and *dim4* columns.

5.2 Interpretable Unit Commitment model

Once we have prepared the dataset, we can focus on developing the interpretable machine learning model. As mentioned before, this project proposes the implementation of multiple models that, as a whole, can predict all the UC variables of interest in a human-understandable way. To this end, it will be necessary to develop both a regression model for continuous variables and a classification model for binary ones.

There are countless alternatives to address this problem, with significant differences depending on the class of algorithms employed. The most straightforward approach would be to predict each variable separately, training an individual model for each of them, and analyzing results independently. All the algorithms described in section 2.2.4 could be used for this approach.

However, the size of the problem would become rapidly unmanageable, a similar problem to having an excessive number of features. If we tried to separately predict the commitment and production of each of the 54 thermal generators; the power flows through the 186 power lines; and which are the active constraints in an optimal solution, the resulting complexity of the approach could hardly be described as interpretable. Such a procedure would only be reasonable if we wanted to explain particular UC variables but not the whole problem.

Therefore, the most appropriate option would be developing multi-target models, enabling the joint prediction of all variables of the same type so that only one algorithm would be necessary to model each kind of variable. Moreover, this approach would also preserve the relationships between the multiple outputs, which is a very interesting attribute for the modeling of the unit commitment, as it was proved in [39]. If independent models are trained for each target variable, correlations between these variables are difficultly preserved and can result in unfeasible solutions. Employing a single model drastically reduces these kinds of issues, though they do not necessarily avoid them completely.

Given these considerations, decision trees prove to be the most suitable of the described algorithms, allowing a common framework for both continuous and binary variables. In a broad sense, the decision tree will split the feature space and assign a particular combination of outputs to the scenarios belonging to each space partition.

In addition, this algorithm has significant advantages over others, such as logistic and linear regression. For instance, UC is a highly complex problem in which data relationships are often nonlinear, and tree-based algorithms are more suited for such kinds of tasks. Furthermore, it will also be easier to represent and interpret the model, making it an excellent approach to accomplish the purpose of this project.

Based on decision trees, we will propose some model extensions, aiming to either simplify the obtained solutions or extract additional information that may be relevant to understand UC optimal solutions.

Lastly, we have divided the dataset into two parts: 80% of the scenarios will serve for selecting hyperparameters and training the model, and the remaining 20% to evaluate its predictive performance.

It is important to stress the fact that we must randomly segment the dataset by complete scenarios and not by individual periods when carrying out the mentioned division. The reason is that the performance of a model should never be assessed using samples related to others from the training dataset (meaning that they share some kind of information) since this situation could not occur in reality and, hence, the resulting performance would not be representative. This is a common issue in machine learning tasks popularly known as *leakage* and needs to be avoided.

5.2.1 Feature selection

The first step will be the feature selection in the case of the network constrained unit commitment. As mentioned above, the number of input features will range from 28 to 77, which can substantially hamper the interpretability of the model. The goal of this step will be to reduce this number to only 15.

In order to select these variables, we will use the feature importance tool (section 2.2.3) of a random forest model, which is usually able to achieve reasonable results without the need to optimize its hyperparameters. We will train the random forest on the set of outputs we want to predict, introducing all the available input variables. The algorithm will infer relationships between these inputs and the outputs, assigning importance values to each of the features.

Using these values, we will choose the 15 most relevant features for the RF and discard the others. Since this process is carried out for each type of output to be predicted, the resulting feature importances will differ, and a different input matrix for each model will be obtained.

Finally, random forests have the advantage of being based on decision trees since their implementation is based on the combination of multiple DTs trained with the same outputs. Consequently, the extracted variables will be closely related to the DT's most explanatory variables for each task.

5.2.2 Hyperparameter selection

As introduced at the beginning of this chapter, the machine learning model for both regression and classification tasks will be a multi-target decision tree, implemented using the *scikit-learn* Python package. Even though we could directly fit the models with the training data, the resulting trees would most likely have an unmanageable dimension, and their performance could be far from optimal due to the overfitting of the training set. To avoid these undesired results, we must carefully choose the hyperparameters of the algorithm, which are the parameters that control its learning process.

This selection has two essential factors. First, we need to artificially constrain the complexity of the algorithm in order to ensure its interpretability. As introduced in section 2.2.4.3, the main parameter controlling the complexity of decision trees is the *maximum depth*. Therefore, we must define a range of values for this parameter that will result in a sufficiently interpretable model.

For this particular application, we will consider that an interpretable decision tree should have a depth lower than or equal to 6. Nonetheless, we will analyze the results with up to a maximum depth of 10 since this will allow us to quantify the accuracy that we could obtain if we did not restrict the depth of the algorithm. Additionally, the minimum value will be set to 3, as lower values will not be able to achieve satisfactory results.

Second, we want the model to generate as accurate predictions as possible, respecting the ranges of the values we want to assign to the mentioned parameters. This is achieved by tuning the model hyperparameters. Hyperparameter tuning consists of training and evaluating an ML model using different combinations of hyperparameters in order to select the one resulting in the best performance on data that has not been employed to train the model.

Besides the maximum depth, we will consider three additional hyperparameters. The first of them will be the *maximum number of leaves*, strongly related to the complexity and interpretability of the algorithm, as it restricts the number of feature space partitions. It is important to note that the number of leaf nodes in a fully grown tree is equal to $2^{max.\ depth}$, so setting a greater value would have no effect on the learning process as the maximum depth constraint would always apply before. Therefore, the allowed values for such parameter will be limited to the maximum value corresponding to each depth, and its minimum value will be set to 16.

Then, we will include the *minimum number of samples per leaf*, which restricts the minimum number of training instances that can be assigned to a terminal node. This parameter seeks to prevent the algorithm from learning non-general relationships, which could minimize the training error but yield worse results for out-of-sample data. Unlike the previous cases, the selection of this value needs to be carried out considering the size of the training set. For this application, the allowed values will range from 2 to 200.

The last hyperparameter that will be considered will be the *minimal cost-complexity* parameter. Once the decision tree has been trained, this hyperparameter is used to remove the subtrees resulting in an overall impurity decrease lower than the value defined for it. Thus, it only allows the model to learn the most explanatory relations. The minimum and maximum bounds selected for this parameter will be 10^{-8} to 10^{-4}, respectively.

These last hyperparameters are not explicitly related to the interpretability of the tree (even though they can also constrain its depth and number of nodes), but they have a crucial role in the generalization capacity of the algorithm since they prevent it from overfitting the training set, positively impacting its predictive accuracy.

The selected bounds for the considered hyperparameters are summarized in Table 5, additionally including their type of variable, i.e., integer or continuous.

Table 5: Decision tree hyperparameters and their allowed values.

Hyperparameter	Minimum	Maximum	Type
Max. depth	3	10	integer
Max. leaf nodes	16	$2^{max.\ depth}$	integer
Min. samples per leaf	2	200	integer
Cost-complexity param.	10^{-8}	10^{-4}	continuous

The most straightforward approach to perform hyperparameter tuning are grid search, which consists of the exhaustive evaluation of all the possible combinations resulting from some pre-defined values for the hyperparameters, and random search, which randomly evaluates only a specified number of these combinations. However, they are computationally expensive as the number of possible combinations exponentially grows with the number of predefined values and will not necessarily yield the optimal results due to the need for discretizing the hyperparameter space [57].

Alternatively, the Bayesian optimization approach is usually able to find the optimal hyperparameters in significantly less time than the previous approaches. This approach analyzes the performance achieved by previously evaluated combinations of hyperparameters and maps the remaining hyperspace by estimating the probability of other possible combinations to yield a better result than the best performance achieved so far, and evaluates the most promising one. As a result, the algorithm can only focus on the best-performing hyperparameter values, avoiding losing time with ineffective combinations. Besides, hyperparameters do not need to be discretized into a few possible values since the evaluation space is only defined by its bounds, as we have defined them previously [57].

However, we are not only interested in obtaining the best performance of the model, but rather we would like to analyze how does the performance vary with the complexity of the decision tree or, more precisely, with its maximum depth. Therefore, we will carry out a Bayesian optimization for each maximum depth value. This will allow us to estimate the best expectable results depending on the depth of the decision tree and select the final combination of hyperparameters for each type of output, considering both expected accuracy and interpretability.

Lastly, the analysis of these hyperparameters will be carried out using cross-validation, removing the need to use a portion of the data only for this purpose. This method splits the training set into k parts or *folds*, usually between five and ten, in the same way as we did for the training and test sets. Starting from these partitions, the algorithm is iteratively trained using all but one of the folds, and the validation prediction is made on the left-out fold. The validation error will be obtained by averaging the errors obtained in each of these iterations. This way, we can assess the model performance on out-of-sample data while still using the training set. The selection of the optimal hyperparameters for the presented task will be carried out using six folds.

After analyzing the results obtained for each combination of hyperparameters, we will select the ones yielding the best balance between accuracy and interpretability. With these optimal hyperparameters, we can train the interpretable model using the complete training set, and finally, assess its performance on the test set.

5.2.3 Linear regression model tree

Nevertheless, in this project, we propose an extension of the previous model for the regression tasks (continuous outputs), which can bring a very different perspective to the modeling of the UC problem. One of the major disadvantages of decision trees is the lack of sensitivity in their terminal nodes since they are assigned with a single value. To overcome this limitation, we propose the training of linear regression models at each terminal node, which can approximate some of the specific relationships existing in each hyperspace partition. For example, if there is a unit that is marginal for the partition corresponding to one terminal node, the output of that generator will have a direct relationship with the net demand, and a linear regression could capture this effect.

The use of additional algorithms on the leaves of a decision tree is known as a model tree [58] and is a concept close to the idea of a piecewise linear regression in a multidimensional space but without requiring output continuity between adjacent partitions. Figure 15 provides an illustrative example of a model tree based on linear regression, similar to the one we will try to implement in this project [59].

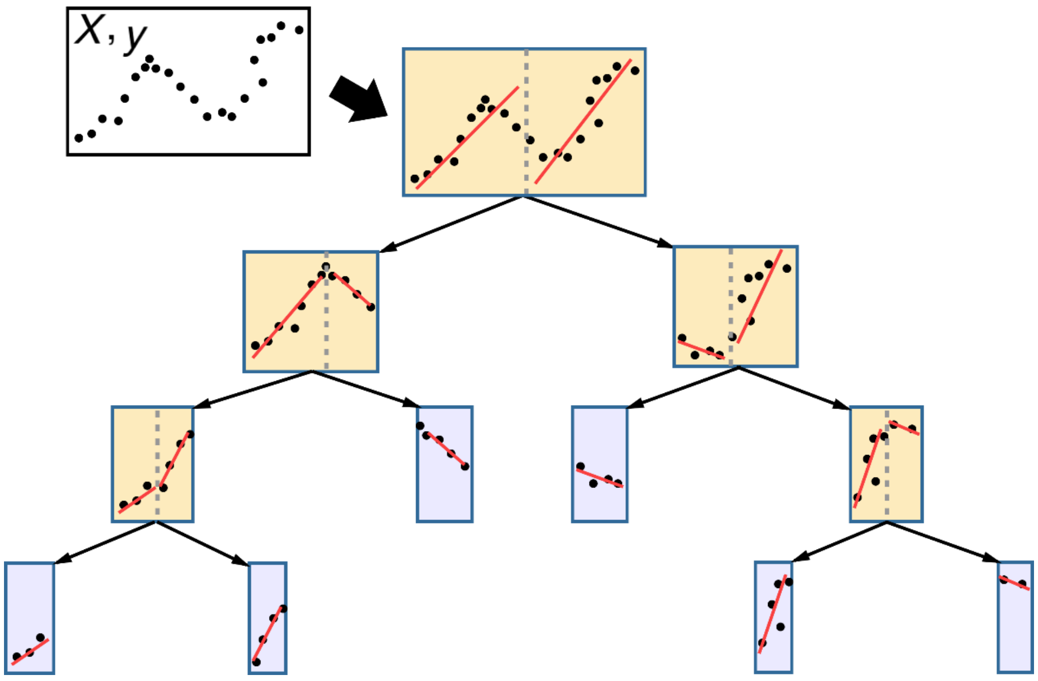

Figure 15: Graphical representation of a linear regression model tree for a one-dimensional problem [59].

Implementing a model tree in the strict sense must carry out the partitions of the feature space, taking into account the linear regressions that will result at its nodes so that it can actually perform the splits that will result in the lowest total error. This is precisely what is illustrated in the preceding figure. However, evaluating the optimal splits in a model tree has a remarkably higher computational cost than that of a simple decision tree or just training the complementary model itself. This, in practice, limits the number of options that the model is allowed to evaluate before selecting the best split [59].

Additionally, model trees are not a widespread class of algorithms, and few implementations are available. Tests carried out with the most complete implementations available, such as the one found in [59], have not yielded satisfactory results comparable to those of a simple decision tree.

In consequence, the implementation carried out in this project will consist of training linear regressions at the terminal nodes of a previously built decision tree regressor, significantly simplifying the training process. Even though the terminal nodes of the decision tree will not have been optimally selected for subsequently training linear regressions, the information extracted from these models will still be very useful to assess the relationship between inputs and outputs in the different partitions of the feature space and understand the corresponding optimal solutions of the UC problem.

However, it is important to note that this technique runs the risk of considerably difficulting the interpretability of the model, particularly when an excessive number of variables are included in the linear regressions. It should be noted that, by including m variables in the linear regression, the number of parameters located at the terminal nodes is multiplied by $m + 1$ (due to the intercept) in comparison to the original tree. This means that a decision tree with linear regressions with a single variable and the intercept at its terminal nodes would have a complexity comparable to that of a tree with one more level. Nevertheless, the information provided by the linear regression could be more valuable than that of an additional tree level.

Considering the above, we propose using a single variable at each node, even though more variables could be employed. The variable to be selected will not be common to all nodes since this would substantially limit the effectiveness of this extension. Instead, we will look for the most explanatory feature at each node, i.e., the one resulting in the smallest validation error.

5.2.4 Terminal nodes clustering

The development of the decision tree classifier, used to predict the binary variables of the unit commitment, will be essentially identical to the regression model except that, in this case, a model tree will not be built from it.

Instead of integrating an additional model at the terminal nodes to provide a different sort of interpretability, we will focus on facilitating the visualization of the output variables of each node and even reducing the number of different output combinations. As a result, we can make the decision tree much easier to understand, and thanks to this, we could assume a greater depth of the tree.

First, due to the binary nature of the output variables, multiple terminal nodes of the decision tree classifier will likely have the same combination of outputs. Consequently, it would be much simpler –and more interpretable– to assign each existing combination of output variables an identifier and represent these unique outcomes in a look-up table, thus avoiding repeatedly representing the same outputs in different nodes and simplifying the identification of these equivalent nodes.

Moreover, in other cases, there will be leaves whose output is practically identical to those of other terminal nodes. Therefore, in this project, we propose applying a clustering algorithm to the outputs of the model in order to group those that are nearly equal and thus further reduce the dimension of the model. The selected algorithm for carrying out the clustering of terminal nodes will be K-modes, which is a modified version of K-means that allows clustering data with categorical instead of numerical features. K-modes uses the Hamming distance to measure the similarity of data points which computes the number of positions where two arrays of the same length are different [60].

Finally, this idea can also be applied to the output variables themselves that are totally or highly correlated, which is the circumstance, for instance, of some generators' commitment status. This is actually a similar strategy to the one implemented in [42] (section 2.3), where multiple variables were clustered to reduce the number of decision variables of the UC optimization problem.

However, it is particularly important to note that it would not be appropriate to perform the aggregation of variables before training the model since this would reduce the weight of those variables in the multi-target objective function and could involuntarily undermine the model's accuracy as a whole.

Finally, we will gather in the mentioned look-up table all the variables that have been grouped in this step.

5.2.5 Decision tree representation

The last phase of developing intrinsic interpretable models is their representation since their actual interpretability considerably depends on it. This is a crucial aspect because such models, if they are not provided with an understandable representation, will be equivalent to a black-box model, losing all their added value.

Scikit-learn's decision trees have many useful visualization functions that allow representing the resulting models after being trained. However, they have some limitations regarding the information that can be included in each node and are not compatible with the extended regression and classification decision trees that have been presented in the previous sections.

Therefore, a tool based on these functionalities has been carefully elaborated, allowing us to flexibly adapt the representation of the developed decision and model trees, which will be used to illustrate the results in the following chapter.

Some of this tool’s features include, for instance, selecting the colors of the tree, the format of the leaves, whether to display the error or accuracy and the number of training samples at each node, the specific variables to be included, or whether to display them in a separate look-up table through the labeling of terminal nodes.

6 Analysis of results

The interpretable machine learning models presented in the previous section were trained using the training set (80% of the scenarios) and subsequently applied to predict the results corresponding to the test set (the remaining 20% of scenarios).

In this section, the resulting outcomes for some representative variables, such as vProduct1, vCommit, vCirPF, and eMinOutput, will be studied. First, we will analyze and compare the feature importances obtained with the random forests for these variables.

Second, we will evaluate the validation performance of the regression model relative to its depth in order to select an appropriate balance between interpretability and accuracy. Then, the resulting model will be compared to other approaches with different levels of interpretability, allowing us to assess how good the predicted outcomes are for each variable. Additionally, the model's performance will be compared with the one obtained for the single-node and low wind penetration scenarios to analyze the impact of these variations compared to the base case. The trained model will be represented together with specific terminal nodes in order to understand the behavior of the model tree.

Finally, a similar process will be followed for the classification model. We will select a reasonable depth for an interpretable decision tree based on the validation performance. We will then analyze the impact of clustering terminal nodes and generators and represent the resulting models.

6.1 Feature selection

As explained in section 5.2.1, the number of input features for the network-constrained cases is too large to be handled by the IML. Therefore, we have carried a selection of the 15 most relevant variables based on the feature importances extracted from random forests fitted on the training set. In the following, the results obtained for the 20 most significant features for the high wind penetration scenarios will be presented for each studied variable.

Regarding the output of generators (vProduct1), Figure 16 shows that there is essentially one primary variable in the model, which is the total net demand of the corresponding period (Total ND [h]). This is a reasonable result because when generators' output is not limited by any constraint, it will directly depend on the total net demand of the system. Following the total net demand, the main features correspond to the increase of total net demand in adjacent periods, which provide helpful information for the model to account for the load gradient constraints of generators. Lastly, we can also appreciate how the random forest algorithm considers node-related features. These will usually be related to areas of the network where power lines congestion is likely to happen, which would constrain the generation of units located in those areas. However, the low relative importance of these variables makes it difficult to draw precise conclusions about them.

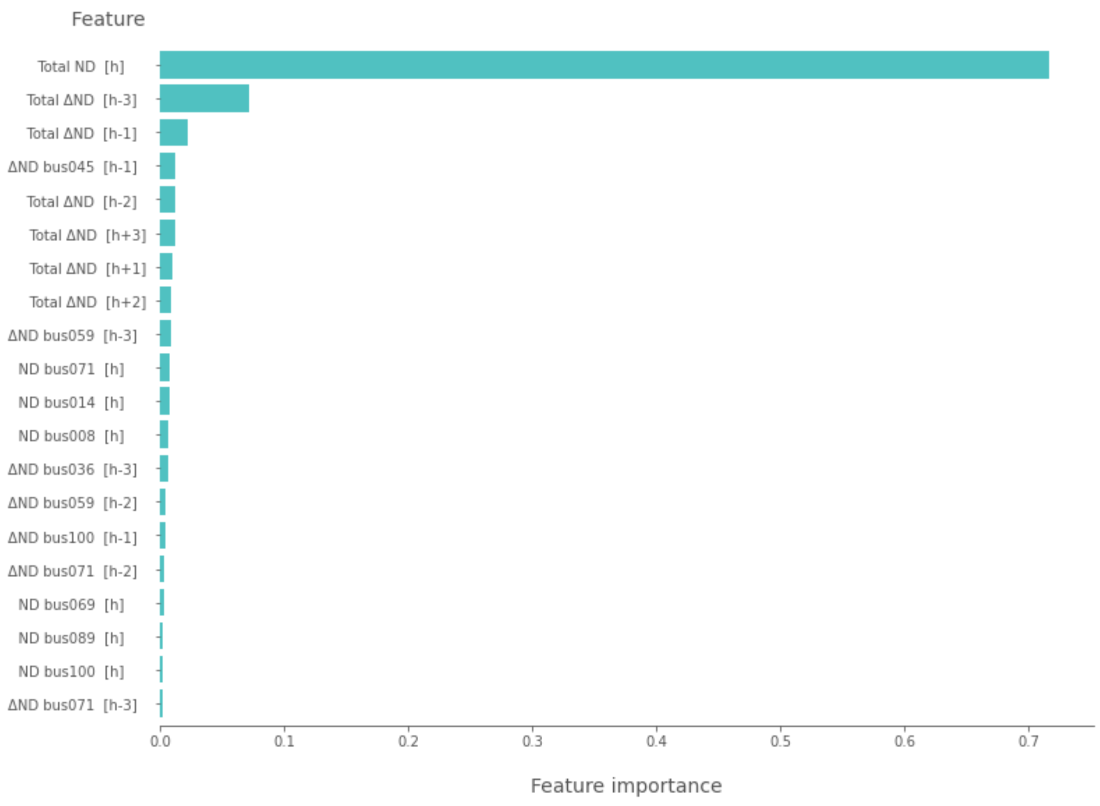

Figure 16: Random Forest feature importances for vProduct1 in the network-constrained high wind penetration UC problem.

In the case of the power flow through the lines (vCirPF), the main variable is again the total net demand in the corresponding period, as shown in Figure 17. This is due to the fact that, especially in nodes far from wind generation, the flow through power lines depends mainly on thermal generation, and this, in turn, on the total net demand, as explained above. However, after the previous one, the most relevant variables are not related to the total net demand but the net demand in the nodes with renewable generation in the corresponding period since the power flow through nearby lines will largely depend on it.

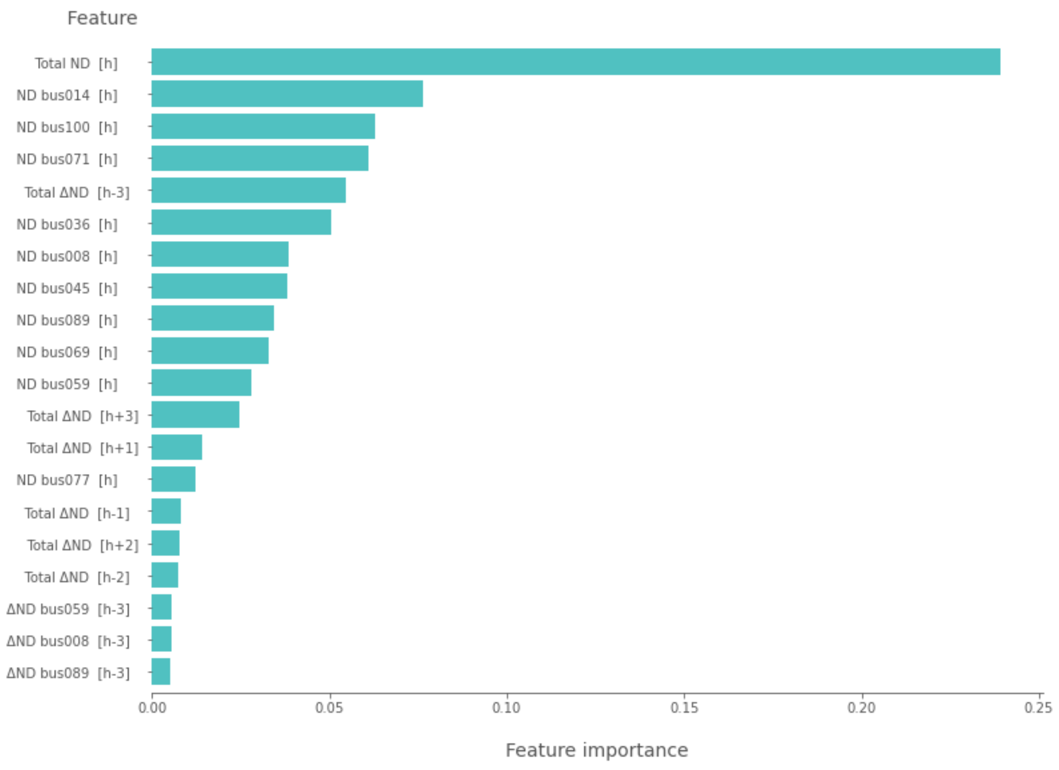

Figure 17: Random Forest feature importances for vCirPF in the network-constrained high wind penetration UC problem.

Subsequently, Figure 18 presents the results obtained for the commitment of generators (vCommit). Unlike the previous cases, it is a binary variable, not a continuous one. In general, it can be seen how the importance of the variables is much more diluted. For example, total net demand is the most relevant variable, but its relative importance is much lower than in previous cases. This may be counter-intuitive since one might expect much greater importance of the variables related to total net demand, as was the case for vProduct1.

However, as will be seen later, vCommit has relatively low variability for most generators. Analyzing the generators with the highest variability in more detail, it has been observed that these are generators close to wind generation. In these cases, the commitment of the generators is limited by the capacity of the power lines to evacuate energy when the renewable generation is very high, which explains why the variables associated with these nodes are of great importance in the model.

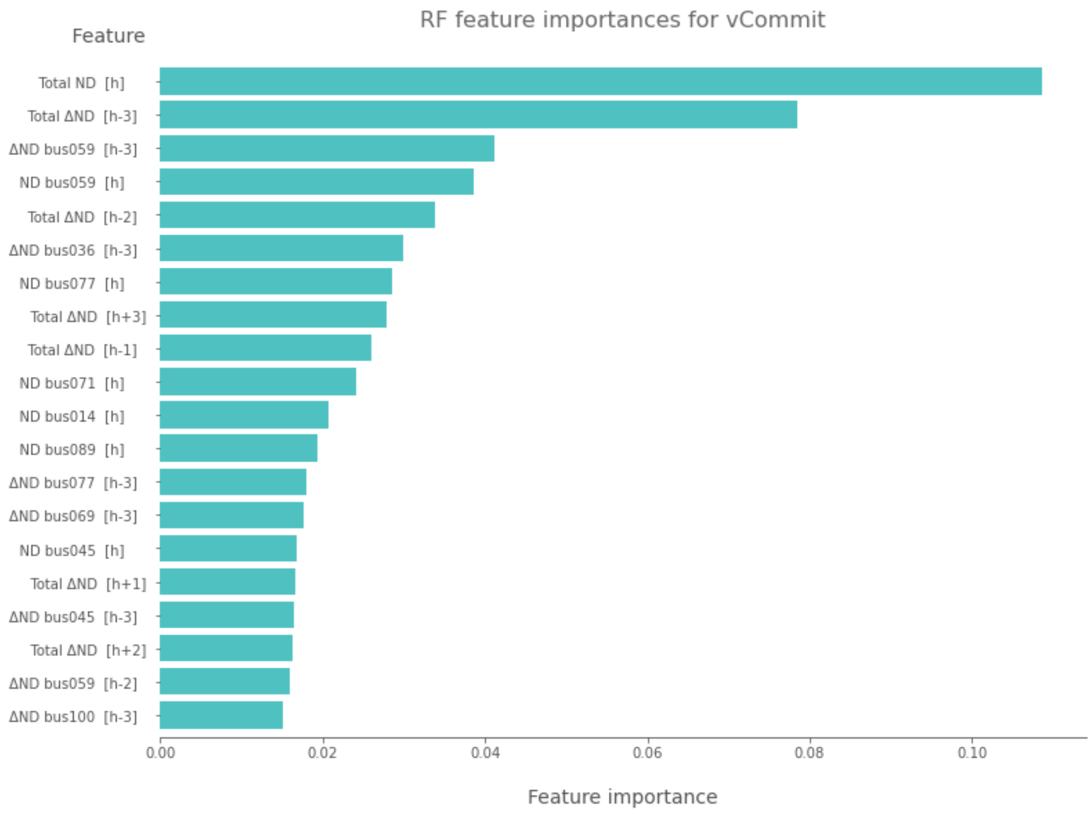

Figure 18: Random Forest feature importances for vCommit in the network-constrained high wind penetration UC problem.

In addition to the above variables, we will also study the results obtained for the dual variable of the minimum generation constraint (eMinOutput), both in its continuous and binary form. The feature importances for both cases are shown in Figure 19 and Figure 20, respectively. The feature importances are very similar to those obtained for vProduct1 and vCommit, respectively, and analogous reasoning can be applied. However, the results are more difficult to interpret globally since this constraint can be active regardless of whether the generator is connected or not.

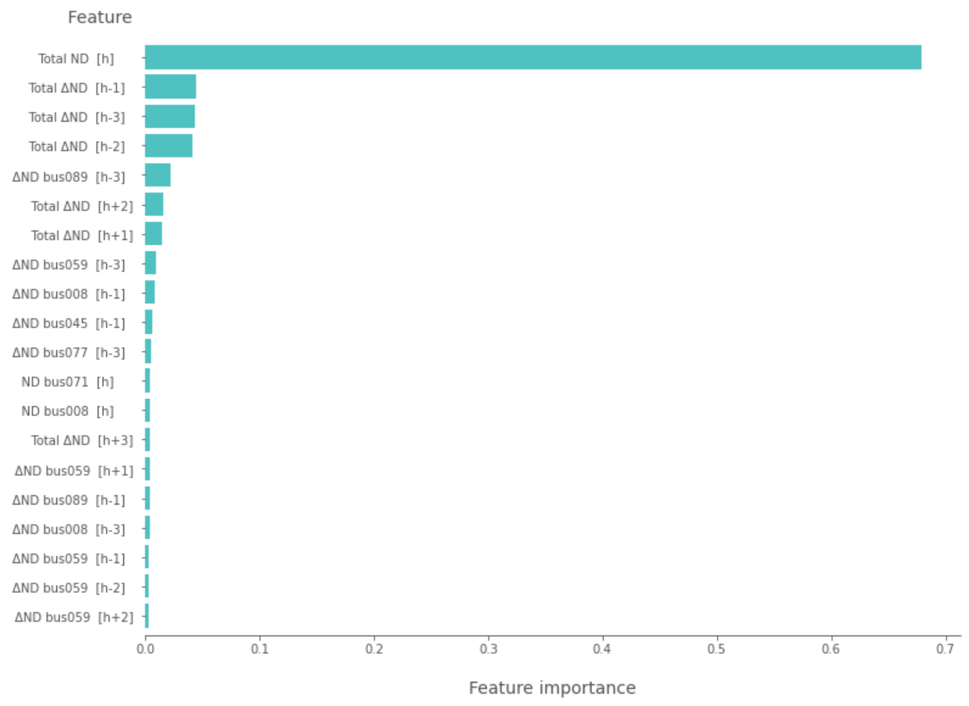

Figure 19: Random Forest feature importances for eMinOutput in the network-constrained high wind penetration UC problem.

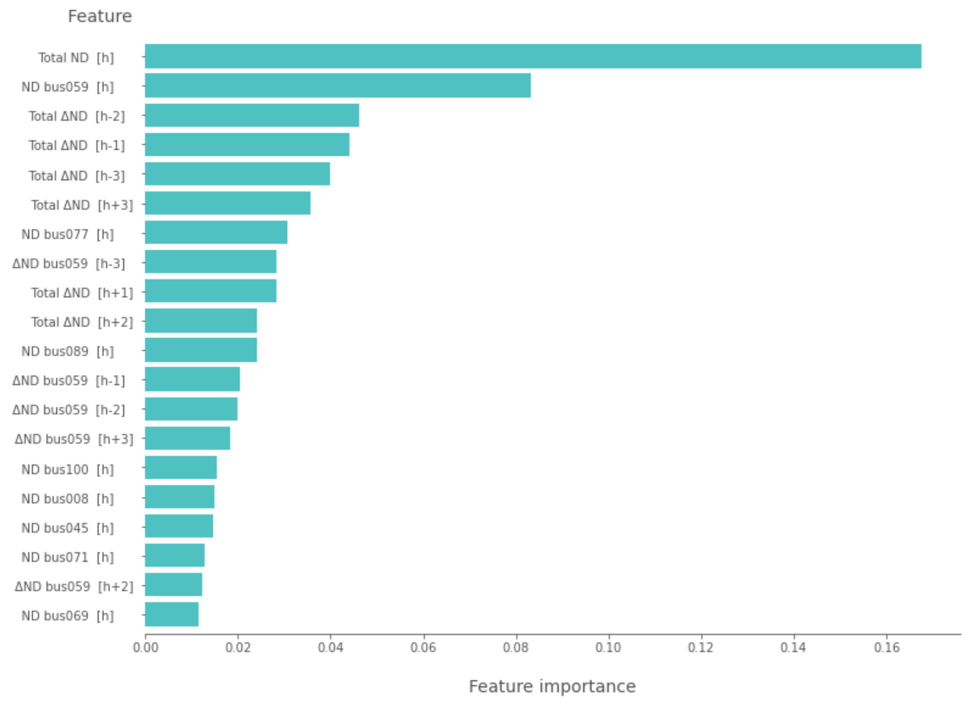

Figure 20: Random Forest feature importances for eMinOutput (binary) in the network-constrained high wind penetration UC problem.

6.2 Regression model

After selecting the main features to be used by the model tree for each of the outputs, we must select the combination of hyperparameters that will allow us to obtain an optimal balance between accuracy and interpretability. For this purpose, the validation performance of the model tree has been analyzed as a function of the depth of the tree on which the algorithm is based.

The performance on each variable in the regression models will be evaluated using the root-mean-square error (RMSE), as it is one of the most common metrics to quantify the estimation error in continuous variables (49). Consequently, since each model has multiple outputs –as many as there are variables of each type–, the global performance of each model will be analyzed by computing the mean RMSE obtained for each output.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2}{N}} \tag{49}$$

Applying this metric to the variables corresponding to vProduct1, we obtain the results presented in Figure 21. Additionally, the figure includes the mean RMSE obtained both by using a single decision tree for all outputs and one tree for each of them.

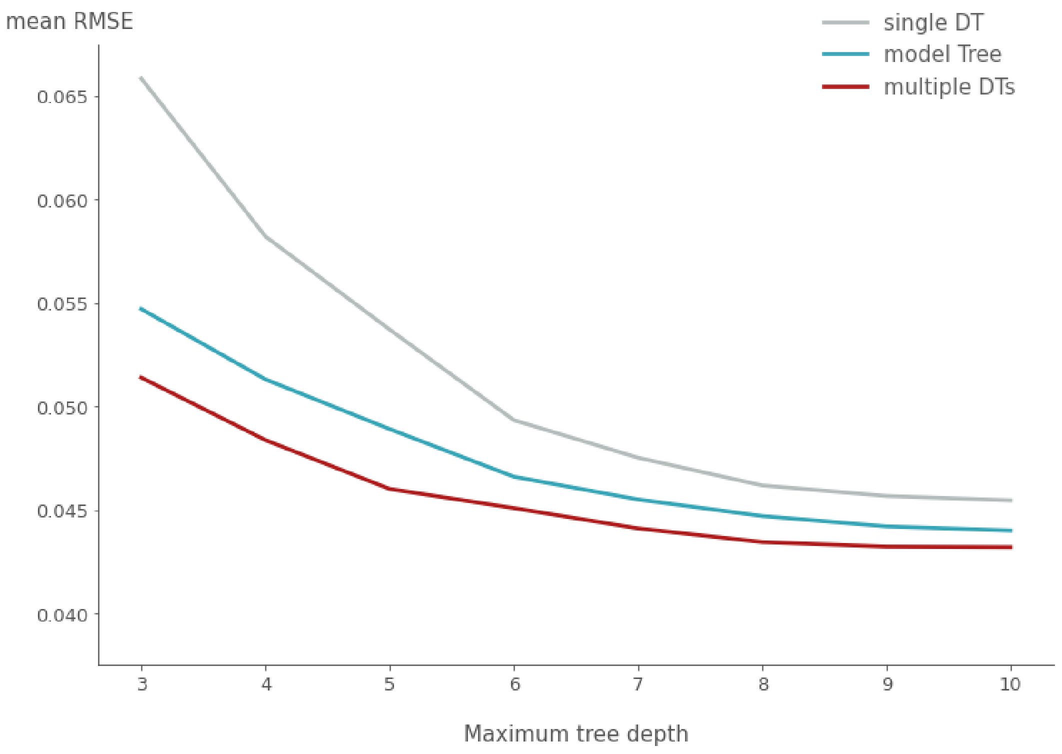

Figure 21: Model tree and decision tree RMSE relative to its maximum depth for vProduct1 in the network-constrained high wind penetration UC problem.

As can be seen in this figure, the model tree manages to significantly reduce the error compared to the case of a single decision tree, between 5 and 20%, approximately, depending on the depth. As the maximum depth of the tree increases, this difference is progressively reduced. One of the main reasons for this effect is that the tree itself begins to capture some of the linear relationships in the data. This is one of the main disadvantages of training linear regression *a posteriori*, rather than considering it during model training. If that were the case, better results could probably be achieved. Additionally, as the depth of the tree increases, the number of samples at the terminal nodes decreases, which reduces the information available for training the linear regressions, and their accuracy may be negatively affected. This effect can be solved simply by using a larger number of instances to train the model (if they are available or can be generated).

An interesting fact to note from this graph is that the model tree can obtain better results than a decision tree with a higher depth. This is important because, as indicated in section 6.2, training linear regressions with one feature and an intercept at the terminal nodes could double the number of model parameters at these nodes, which also occurs when the maximum depth of the tree is increased by one level. However, the interpretability of both models is not necessarily comparable. On the one hand, in the case of the model tree, it is not required to split the node, thus avoiding the addition of a new condition at each branch, and the linear regression at a terminal node provides more relevant information than just the mean value of the leaves into which the node would be split. On the other hand, not all variables at each terminal node will have a linear relationship with the feature selected at that node. In these cases, the model tree will maintain the mean value of the previous decision tree, thus having no negative impact on the interpretability of the model. Considering the above, it can be concluded that the model tree can achieve a better balance between interpretability and accuracy than the decision tree itself.

Additionally, the performance obtained by the model tree is comparable to that achieved by training a decision tree with the same maximum depth for each output. This difference is initially below 6%, and it is reduced as the depth increases, as the model tree has greater flexibility to capture the variability of the output. The comparison shown allows us to conclude that it would not be effective to train a decision tree for each variable (one model for each of the 54 generators) since this would imply an excessive and unmanageable complexity compared to the use of a single model tree with a greater level of depth, capable of obtaining results comparable to the previous ones, but with much lower complexity.

Finally, using the results shown in Figure 21, the maximum depths of the model tree that would achieve the best balance between accuracy and interpretability would be either 5 or 6. Of these options, the second one obviously yields better results, but its representation and interpretation would be relatively complicated, so the depth selected for the final model tree will be 5.

Table 6 shows the set of hyperparameters selected for each of the regression models studied in this section. In all cases, the best results are obtained by training a fully-grown tree. On the contrary, the minimum samples per leaf and cost-complexity hyperparameters present a higher variability, depending on the generalization capacity needed in each case.

Table 6: Selected hyperparameters for the regression models in the network-constrained high wind penetration UC problem.

Hyperparameter	vProduct1	vCirPF	eMinOutput
Max. depth	5	5	5
Max. leaf nodes	32	32	32
Min. samples per leaf	10	21	12
Cost-complexity param.	$6\cdot10^{-7}$	$2\cdot10^{-7}$	$3.5\cdot10^{-7}$

Using these hyperparameters, we can train the model with the full training set and estimate the test outcomes. To properly evaluate the suitability of the obtained results, it is not enough to simply compute the error made by the model. It should be put in context. To do so, we will compare these results with those obtained through alternative approaches.

First, we will calculate the maximum error that we should expect in each of the tasks. This error is obtained by using the mean value observed in the training set as the prediction for each variable. If a model cannot improve this value (the worst case), it is completely useless.

Secondly, we will compare the results with other interpretable models, which will allow us to address, from another perspective, the balance between interpretability and performance. In this case, the models will be a linear regression and a decision tree with the same depth as the model tree.

Finally, we will compute the performance in the test set of a gradient boosting decision tree. GBDTs are not interpretable, but they are algorithms that usually achieve good results in a wide variety of tasks. Therefore, we will use it as a reference of what could be the minimum possible error for the given task.

These results are summarized in Table 7.

Table 7: Performance of the developed model tree compared to other approaches for the network-constrained high wind penetration UC problem.

	vProduct1		vCirPF		eMinOutput	
Variable	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**
Worst case	0.1353	0.1588	0.1269	0.0887	0.0910	0.0091
Linear regression	0.0720	0.0798	0.0645	0.0518	0.0628	0.0033
Decision tree (5)	0.0541	0.0659	0.0928	0.0686	0.0364	0.0021
Model tree (5)	0.0496	0.0641	0.0732	0.0587	0.0327	0.0017
GBDT	0.0401	0.0508	0.0510	0.0385	0.0295	0.0022

As for vProduct1, the model tree yields the best result among the interpretable models, with a mean RMSE below the decision tree's and substantially lower than that of the linear regression. Analyzing the results from a general perspective, it can be observed that it achieves an error relatively close to that of the GBDT and almost three times lower than that of the worst case. Although the difference in performance relative to the GBDT is still notable, it is important to remember that this is a much more complex and non-interpretable model, so the result obtained by the model tree can be considered reasonably good. Therefore, it can be concluded that this approach achieves an optimal balance between interpretability and performance.

The case of eMinOutput is practically analogous to the previous one. The main difference is that both the decision tree and the model tree obtain results closer to the GBDT than with vProduct1, making this model even more appropriate.

However, the results obtained for vCirPF are less promising. Even though the mean RMSE of the model tree is below the worst case’s, it is far from that of the GBDT. In fact, the linear regression obtains a better result for this set of outputs. This fact may seem surprising, but it should be remembered that the linear regression model is fitted with the whole set of input features, while the model tree chooses a single variable for each terminal node. This characteristic is a major limitation for this model since the flow through the power lines is a much more complex variable than those associated with generators. The model tree does not have enough flexibility to capture all the existing relationships in the data. In any case, although linear regression achieves better performance, the size of the problem (15 features with their corresponding parameters and intercept for each of the 186 outputs) substantially complicates the interpretability of the model, so it would not necessarily be optimal.

Additionally, we can compare the above results with those obtained in the alternative UC problems defined in section 3. As shown in Table 8, the mean RMSE obtained when the impact of the network is neglected, is lower than in the base case. This is reasonable since, by eliminating the grid, the limitations that these may have on the operation of the generators are eliminated, and it will be easier to estimate these values.

Similarly, reducing the penetration of wind generation reduces one of the main sources of variability in the problem, which can also lead to limitations on the generators, both due to the load ramp and their influence on grid constraints. Therefore, it is reasonable that the results also improve in this alternative case.

Table 8: Performance of the developed model tree in the proposed UC problem alternatives.

	vProduct1		**vCirPF**		**eMinOutput**	
Variable	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**	**mean RMSE**	**std RMSE**
Single node HWP	0.0423	0.0567	–	–	0.0295	0.0013
Network-const. HWP	0.0496	0.0641	0.0732	0.0587	0.0327	0.0017
Network-const. LWP	0.0409	0.0495	0.0536	0.0469	0.0308	0.0016

Subsequently, we will represent the model tree developed for vProduct1, selecting some representative generators in order to illustrate its interpretability (Figure 22). In line with what was observed in the feature selection stage, most of the partitions of the input space are based on the total net demand features since they have the most influence on the output of the generators. The variables associated with specific nodes appear at the bottom of the tree, where the tree has been able to segment more local relationships, which only take place in specific partitions of the input space.

Noteworthy in this figure is that not all outputs have a linear dependence on the variable associated with each terminal node. Consequently, the resulting model is quite sparse, which facilitates its interpretation since the previous value of the decision tree is maintained.

Additionally, one can observe expression $ord(h) > i$, with $i \in \{1,2,3\}$, as a criterion for tree partitioning. This is not because the period number is included as an input variable but because the model identifies that information is missing for certain variables and decides to segment the training samples when this occurs. Therefore, this logic is equivalent to that shown in the figure, with the cutoff time being dependent on the variable taken by the model.

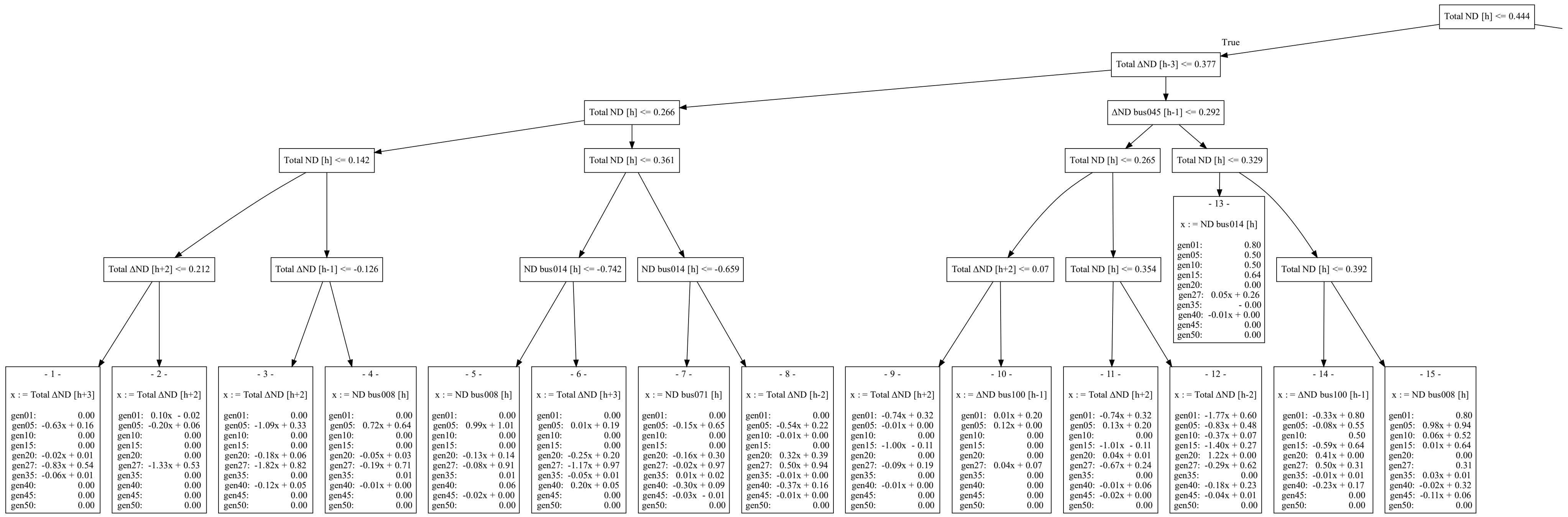

Figure 22 (1): Developed linear regression model tree for vProduct1 in the network-constrained high wind penetration UC problem.

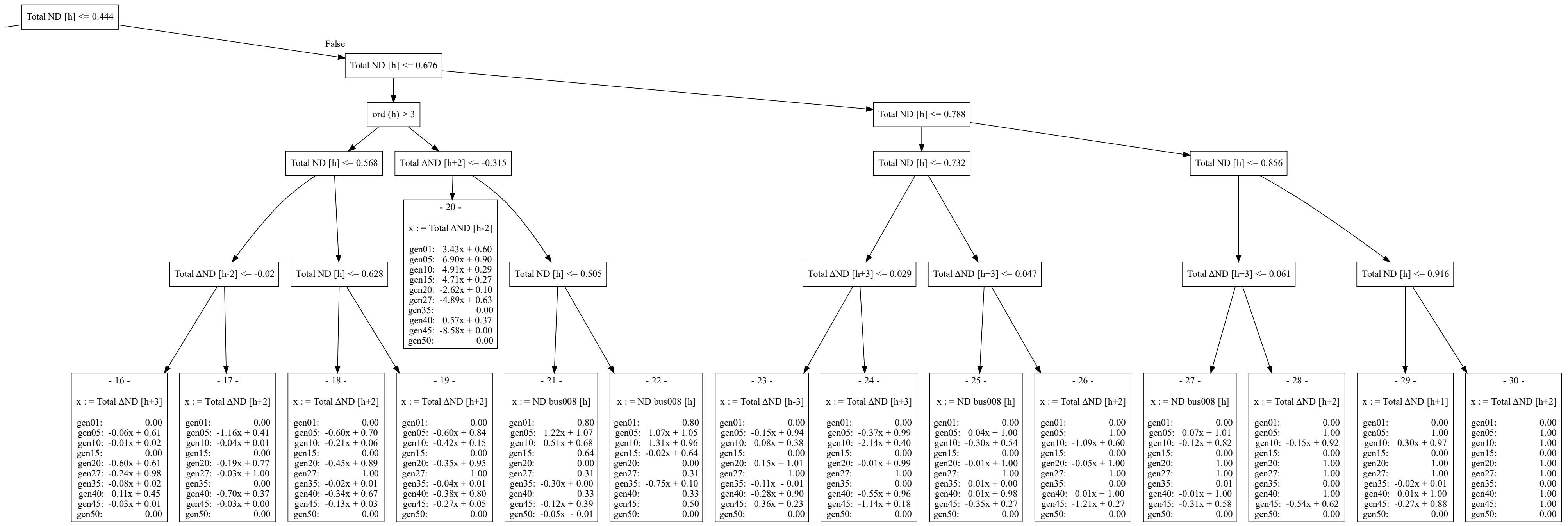

Figure 22 (2): Developed linear regression model tree for vProduct1 in the network-constrained high wind penetration UC problem.

Finally, we will briefly analyze the resulting linear regression models for some generators at certain terminal nodes in order to appreciate this model's advantages over a decision tree.

In the first of these examples, Figure 23, we can see how the model tree is able to capture the linear relationship between the increase in the total net demand in period $h - 3$ compared to period h and the production of generator 27. The decision tree, on the contrary, could not capture this relationship.

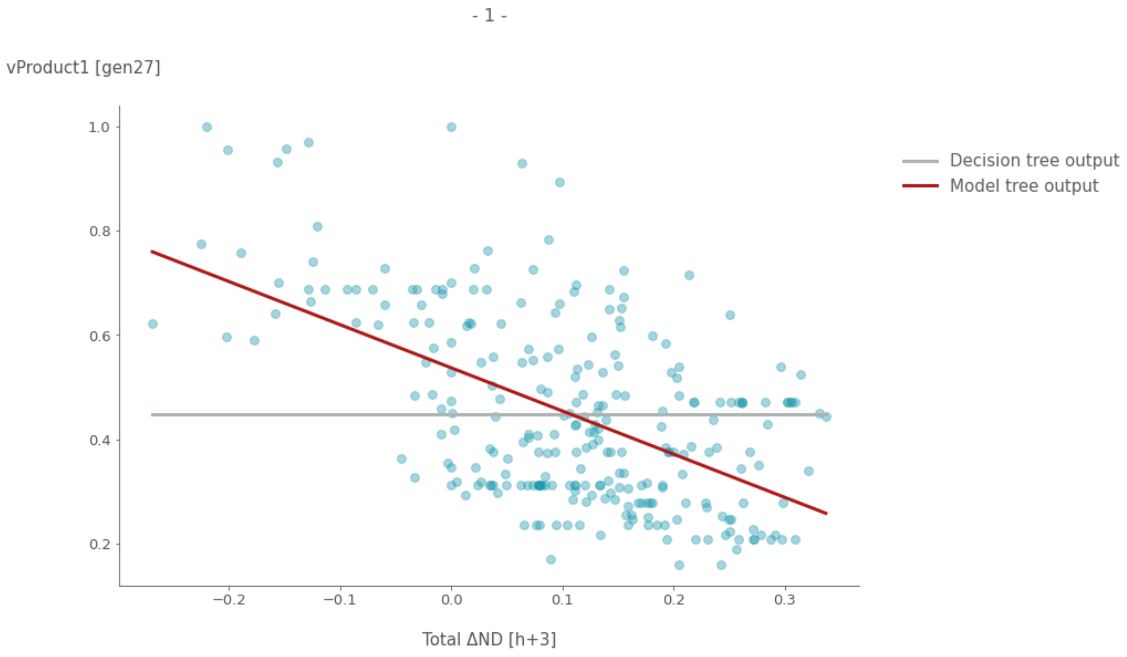

Figure 23: Representation of the relationship between vProduct1 [gen27] and Total ΔND [h+3] in node number 1 of the model tree.

Figure 24 and Figure 25 show the relationship between the output of generator 5 relative to the net power demand at node 8, located in the same area as the generator, at two different terminal nodes. As can be seen, the linear regression of the model tree due to the distribution of the training data is practically identical, with the production increasing proportionally with the net demand at that node. This shows that this is a relevant relationship, which manifests itself in multiple partitions of the space. While this does not mean that both nodes are repeated, as differences are manifested in other variables, it shows that decision tree regressors may present redundancies in some sub-trees. However, this is not usually a relevant problem in regression tasks.

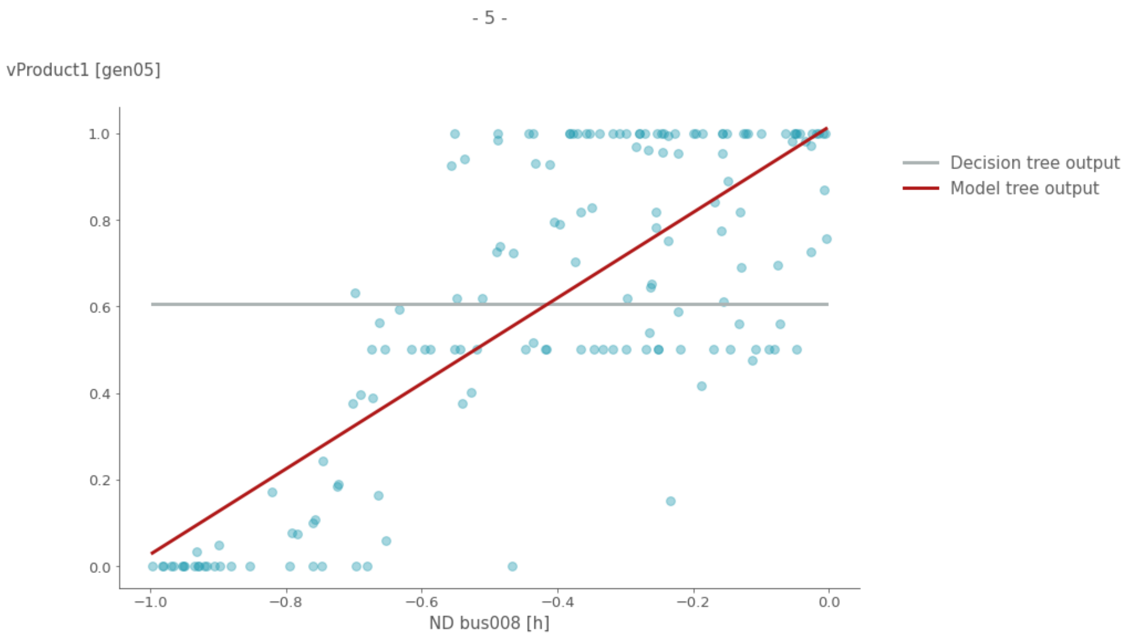

Figure 24: Representation of the relationship between vProduct1 [gen05] and ND bus008 [h] in node number 5 of the model tree.

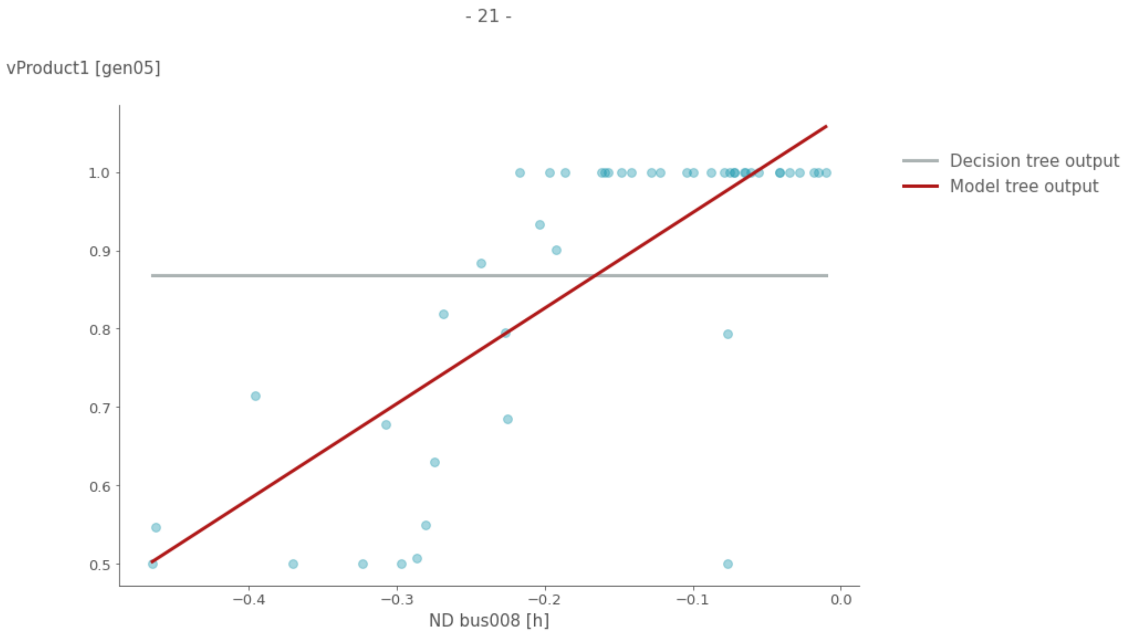

Figure 25: Representation of the relationship between vProduct1 [gen05] and ND bus008 [h] in node number 21 of the model tree.

Figure 26 illustrates another interesting case. The production of generator 15 is practically discrete and shows a clear non-linear relationship with the variation of the total net demand two periods later compared to the current period. When the total net demand decreases, the generator increases its output. Because of its discrete nature, an appropriate partition of the node followed by the assignment of the corresponding mean values, which could be achieved by increasing the depth of the tree by one level, would result in nearly perfect modeling of this relationship. However, a tree with the depth shown is completely incapable of capturing that information. Therefore, linear regression is a truly effective solution for modeling relationships such as the one shown without the need to increase the depth of the base tree, even if the relationships are not strictly linear.

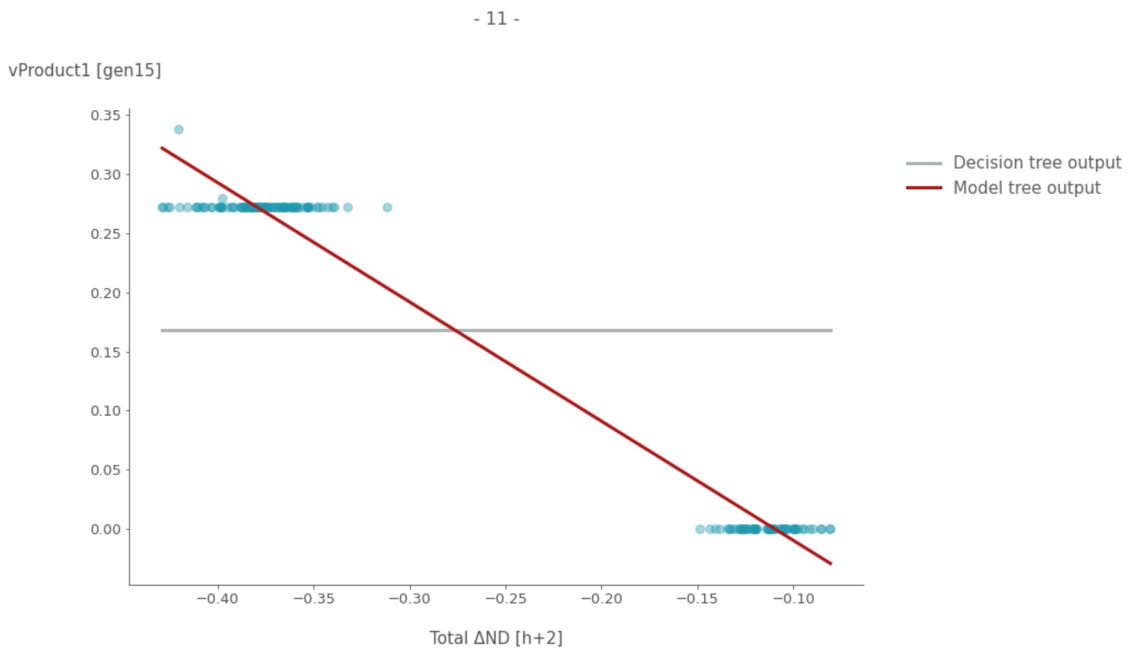

Figure 26: Representation of the relationship between vProduct1 [gen15] and Total ΔND [h+2] in node number 11 of the model tree

Finally, Figure 27 and Figure 28 show how the output of the model tree exactly matches that of the decision tree when there is no relationship between the output and the variable assigned to that terminal node. This does not mean that the variable is not useful since it will be used to model other outputs within the same space partition. It simply has no impact on the analyzed outputs.

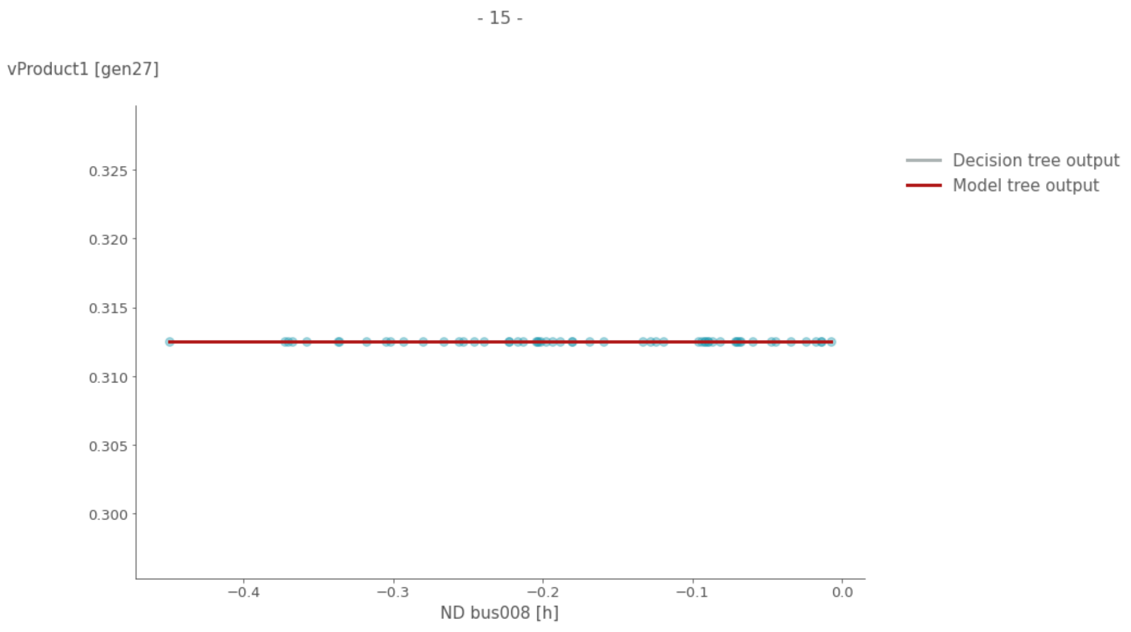

Figure 27: Representation of the relationship between vProduct1 [gen27] and ND bus008 [h] in node number 15 of the model tree.

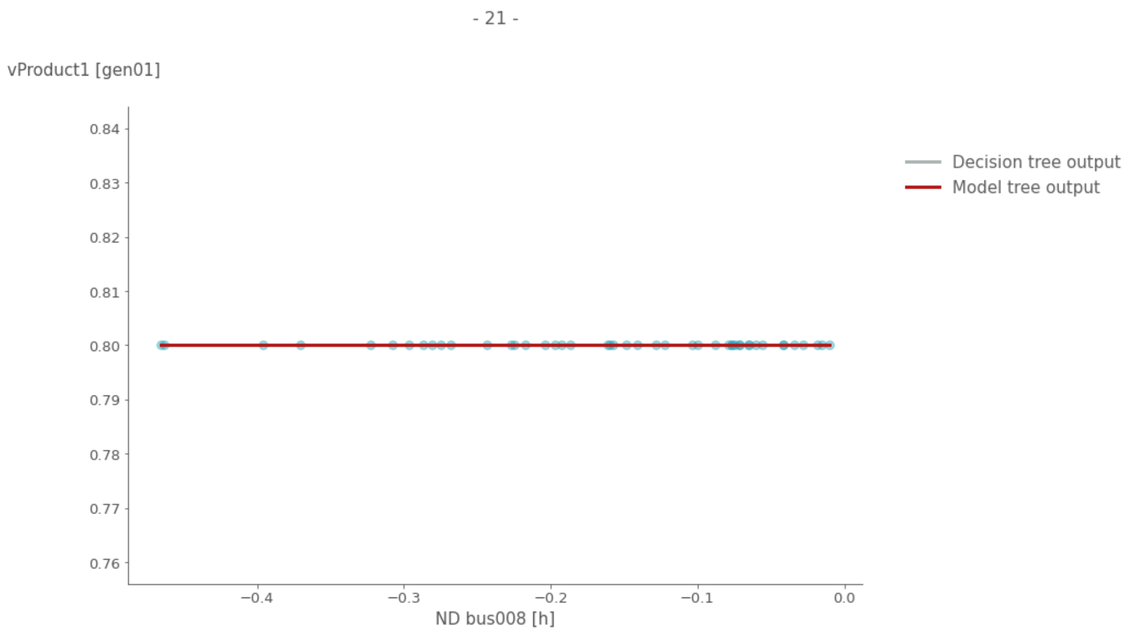

Figure 28: Representation of the relationship between vProduct1 [gen01] and ND bus008 [h] in node number 21 of the model tree.

6.3 Classification model

In the same way that we have carried out the selection of hyperparameters for the regression model, we must repeat the process for the classification model. In this case, the metric used will be the accuracy, which measures the proportion of samples that have been correctly classified (50). Therefore, we will evaluate the performance of the models by calculating the average accuracy of the set of output variables.

$$accuracy = \frac{TP + TN}{N}, \tag{50}$$

where TP stands for true positives, i.e., samples correctly labeled as 1, and TN stands for true negatives, i.e., samples correctly labeled as 0.

Figure 29 shows the average accuracy of a single decision tree for all variables versus that obtained by training one model per variable. For lower depths, the performance of the single decision tree is significantly worse than that of the multiple DTs since the latter have much greater flexibility. However, this difference is drastically reduced as the maximum depth of the trees increases.

Again, it can be concluded that a single decision tree is more effective in terms of interpretability and accuracy than the use of one tree for each variable, but in this case, the depth required to achieve comparable results is approximately two levels higher than that of the multiple trees.

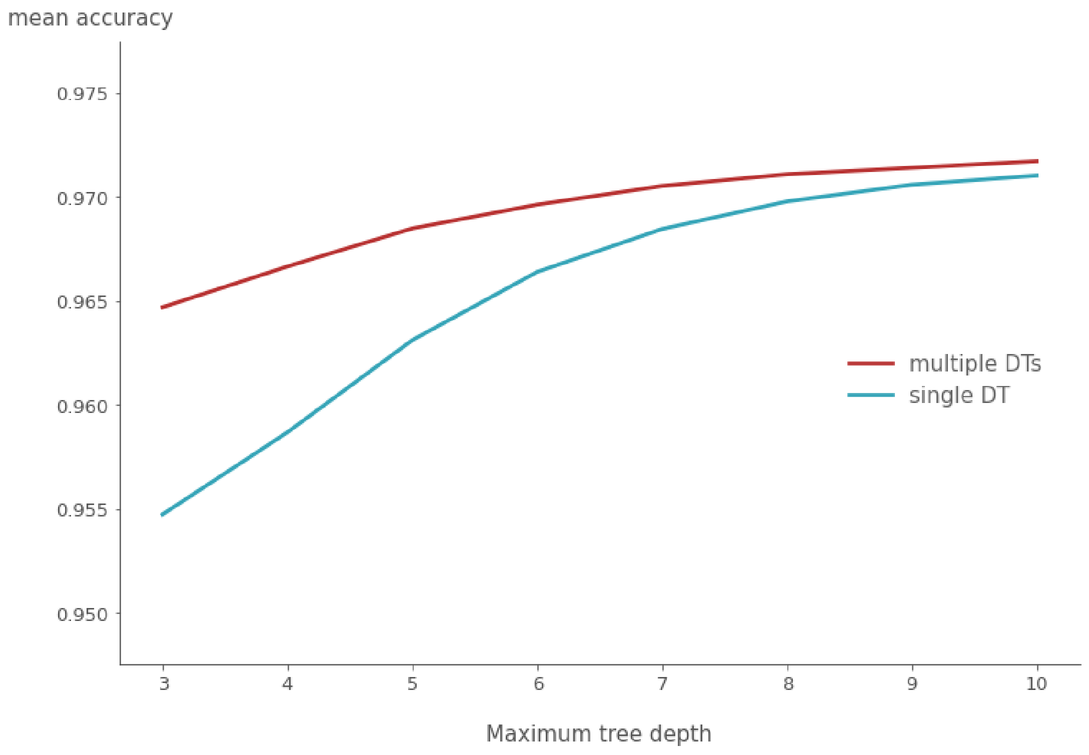

Figure 29: Decision tree accuracy relative to its maximum depth for vCommit in the network-constrained high wind penetration UC problem.

Analyzing the previous graph, a maximum depth of 6 has been chosen for the decision tree classifiers. Although the representation of a tree of these dimensions can be complicated, the clustering of terminal nodes will allow compacting the result in an easier way to analyze.

The resulting hyperparameters for the models studied are shown in Table 9. The values obtained are relatively equivalent to those corresponding to the regression model, although taking into account that, in this case, the depth has been increased by one level. One element to note is that, in this case, the maximum depths do not correspond to those of fully-grown trees. Instead, a smaller maximum number of terminal nodes has been selected.

Table 9: Selected hyperparameters for the classification models in the network-constrained high wind penetration UC problem.

Hyperparameter	vCommit	eMinOutput
Max. depth	6	6
Max. leaf nodes	56	60
Min. samples per leaf	12	6
Cost-complexity param.	5·10^-7	1.5·10^-6

Employing the hyperparameters shown in the previous table, we will train a decision tree for each output type. In this case, we do not intend to develop a model tree but to cluster both the terminal nodes and the output variables themselves.

Figure 30 illustrates the average accuracy of the model as a function of the number of unique terminal node output combinations. As can be seen, from 44 unique nodes onwards, the accuracy remains constant. This implies that there are 12 terminal nodes whose output combinations are exactly the same as those of other leaves in the tree. This kind of redundancy, more typical in classification tasks, does have an important impact on the model and makes it difficult to understand.

Additionally, the number of unique terminal nodes can be further reduced without significantly affecting the model's performance since the outputs in some of them are virtually identical. For vCommit, the number of unique terminal nodes chosen will be 39. Interestingly, even though the depth of the tree is 6, the number of different nodes will be close to the 32 that a fully-grown decision tree of depth 5 would have, which has great benefits from an interpretation point of view.

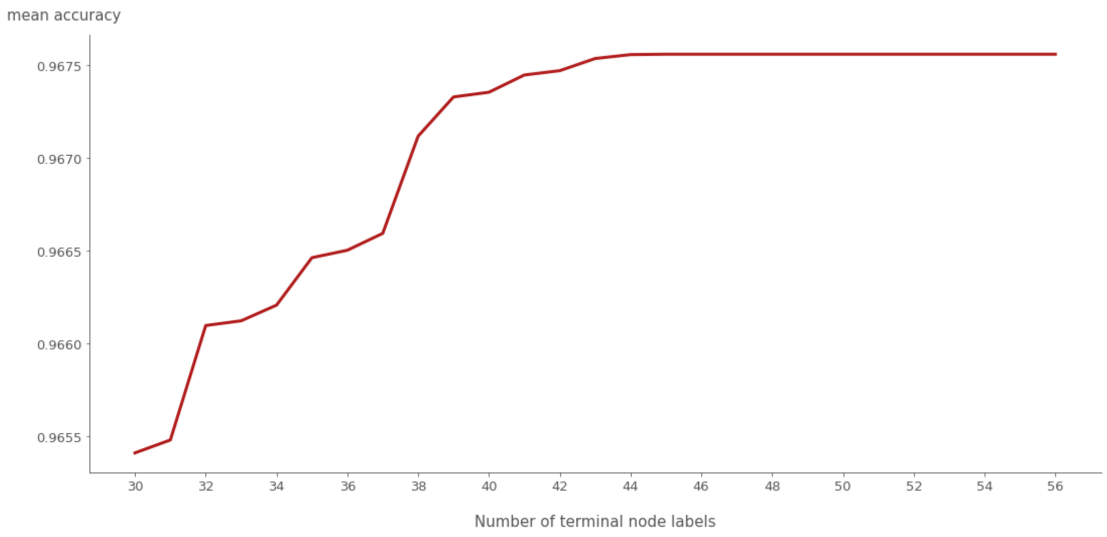

Figure 30: Mean validation accuracy of the vCommit decision tree classifier relative to the number of unique terminal node output combinations.

Similarly, Figure 31 shows the average model accuracy as a function of the number of generator clusters. From the beginning, it can be observed that approximately 30 of these generators show a commitment behavior identical to that of other generators in the system. Most of these are generators that remain disconnected in all the scenarios considered. In addition, we can find another 5 generators whose operation is very similar, so aggregating them will not imply a relevant loss of accuracy. Consequently, the number of generator clusters used for this model will be 19.

Considering both factors, we have reduced the dimension of the decision tree from 56 terminal nodes with 54 generators to only 39 unique terminal nodes with 19 generator clusters. This represents a reduction in the number of unique parameters at the terminal nodes of more than 75% compared to the original case, with virtually no change in model performance (the mean accuracy has been reduced by an amount well below 1%).

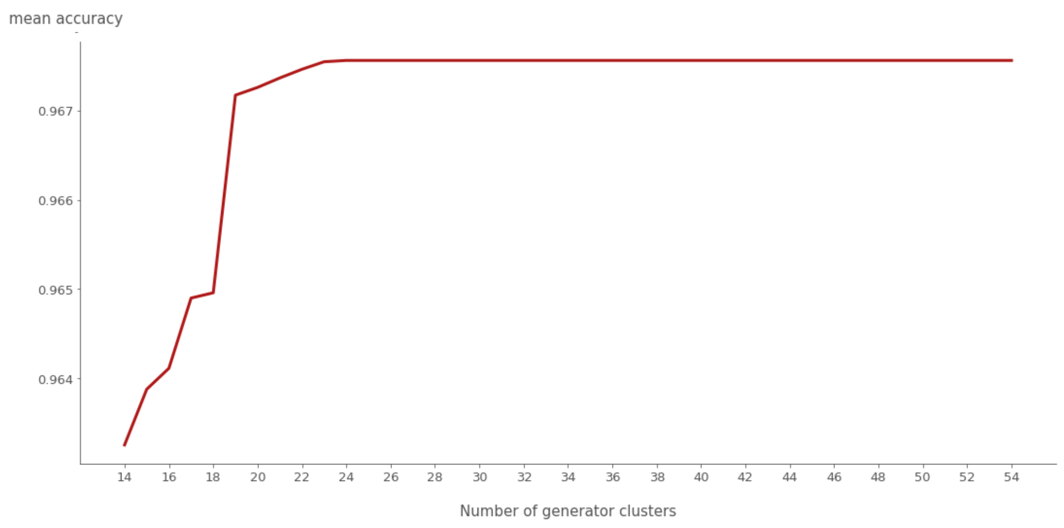

Figure 31: Mean validation accuracy of the vCommit decision tree classifier relative to the number of generator clusters.

Using this methodology, we can evaluate the accuracy of the model using the test set. Additionally, we will compare these results with those obtained using alternative approaches (Table 10), as we did with the model tree.

The minimum expected mean accuracy for each task will be obtained by calculating the accuracy we would obtain by predicting the mode of each of the output variables in the training set.

As alternative interpretable models, we will include a logistic regression and the decision tree equivalent to the one elaborated, but without performing the clustering processes. Therefore, the latter results can be expected to be just slightly above those obtained by the proposed model since the impact of clustering is very small.

Finally, we will use a GBDT as a reference of the accuracy that could be achieved using a model without any limitation in terms of complexity.

Table 10: Performance of the developed decision tree classifier compared to other approaches for the network-constrained high wind penetration UC problem.

	vCommit		eMinOutput	
Variable	**mean accuracy**	**std accuracy**	**mean accuracy**	**std accuracy**
Worst case	0.9102	0.1249	0.6376	0.0963
Logistic regression	0.9662	0.0549	0.8927	0.0434
Decision tree (6)	0.9677	0.0470	0.9219	0.0227
Clustered DT (6)	0.9673	0.0481	0.9207	0.0242
GBDT	0.9768	0.0334	0.9466	0.0176

Firstly, we can observe that the variability of vCommit is relatively low since the minimum expected accuracy is above 90%. The three interpretable models presented achieve very similar performance and are satisfactorily close to that obtained by the GBDT, which validates all the approaches. However, the interpretation of the logistic regressions is much less evident than that of the linear regressions and certainly less than that of the decision trees, so their presented performance would not justify their use. As for the two decision tree classifiers, it is clear that the former will achieve a slightly superior result. However, it can be seen that the difference between both models is practically negligible, so it can be concluded that the clustered decision tree presents by far the best balance between interpretability and accuracy.

Second, we analyze the results obtained for the estimation of eMinOutput in its binary form, which indicates whether the associated constraint is active or not. In this case, the variability is much higher since the minimum accuracy has a considerably low value. Once again, the clustered DT manages to greatly improve this result and stands out as the optimal interpretable algorithm among those presented here. However, despite the fact that the resulting accuracy is not excessively far from that obtained by the GBDT in relative terms, it is a remarkable difference in absolute terms.

Concerning the comparison of the results obtained for the different cases studied, the conclusions drawn are analogous to those of the regression model. Either by omitting the impact of the network or by reducing the penetration of wind generation, the operation of generators is less limited by external constraints, which facilitates the modeling of their production. In particular, Table 11 seems to indicate that the constraints due to renewable generation in the base case have a greater, or are more complicated to model than those of the power grid, since reducing this factor improves the resulting accuracy to a greater extent than eliminating the network.

Table 11: Performance of the developed decision tree classifier in the proposed UC problem alternatives.

	vCommit		**eMinOutput**	
Variable	**mean accuracy**	**std accuracy**	**mean accuracy**	**std accuracy**
Single node HWP	0.9704	0.0493	0.9331	0.0245
Network-const. HWP	0.9673	0.0481	0.9207	0.0242
Network-const. LWP	0.9749	0.0437	0.9615	0.0261

Finally, we will represent the clustered decision tree elaborated for the set of vCommit outputs. As can be seen in Figure 32, the clustered nodes have been identified with unique labels. The outputs associated with these terminal node clusters are shown in Table 12 as a look-up table. Finally, the mapping of the reference generators that are shown in the previous table allows identifying the rest of the generators in the same cluster is found in Table 13.

Analyzing the representation of the decision tree, it can also be seen that the system-wide input features are the ones that dominate the initial partitions of the input space, with the features related to specific nodes appearing at the bottom of the tree, where they play an important role in identifying sets of scenarios that present similar behavior.

Another relevant observation involves the duplicity of terminal nodes, which are generally (but not exclusively) found in nearby branches according to the tree representation. This is one of the usual problems of decision tree classifiers, as mentioned above. While in certain contexts, decision lists can overcome this problem by obtaining more interpretable results than a decision tree itself, this problem still has many unique nodes. For that purpose, a set of conditions would have to be defined to replicate the space's corresponding partitions. This would result in an excessive number of complex conditions that would negatively impact the interpretability of the model. In this case, the exclusivity and exhaustivity of the decision three partitions are important advantages.

Finally, Table 13 allows us to understand how the generators have been grouped in each cluster. The most remarkable group is the one identified with generator 7, which gathers the set of generators that are not connected in any or practically none of the periods with which the model has been trained. Similarly, generator 5 represents the generators that will always be connected according to the decision tree classifier. The rest of the clusters correspond to either single generators or groups of generators that, due to their characteristics, present a very high correlation in their mode of operation.

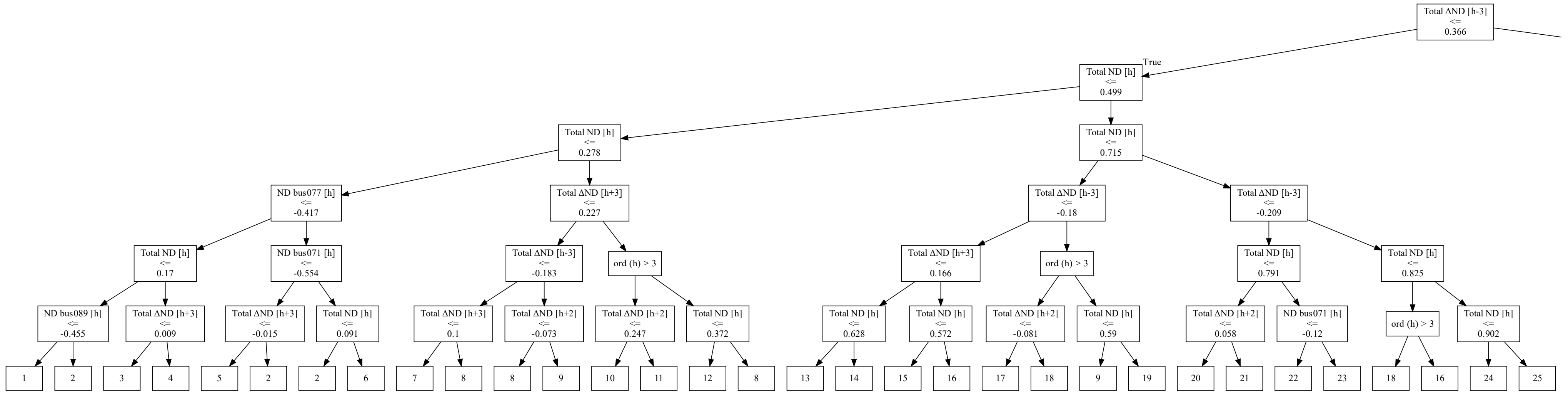

Figure 32 (1): Developed clustered decision tree for vCommit in the network-constrained high wind penetration UC problem.

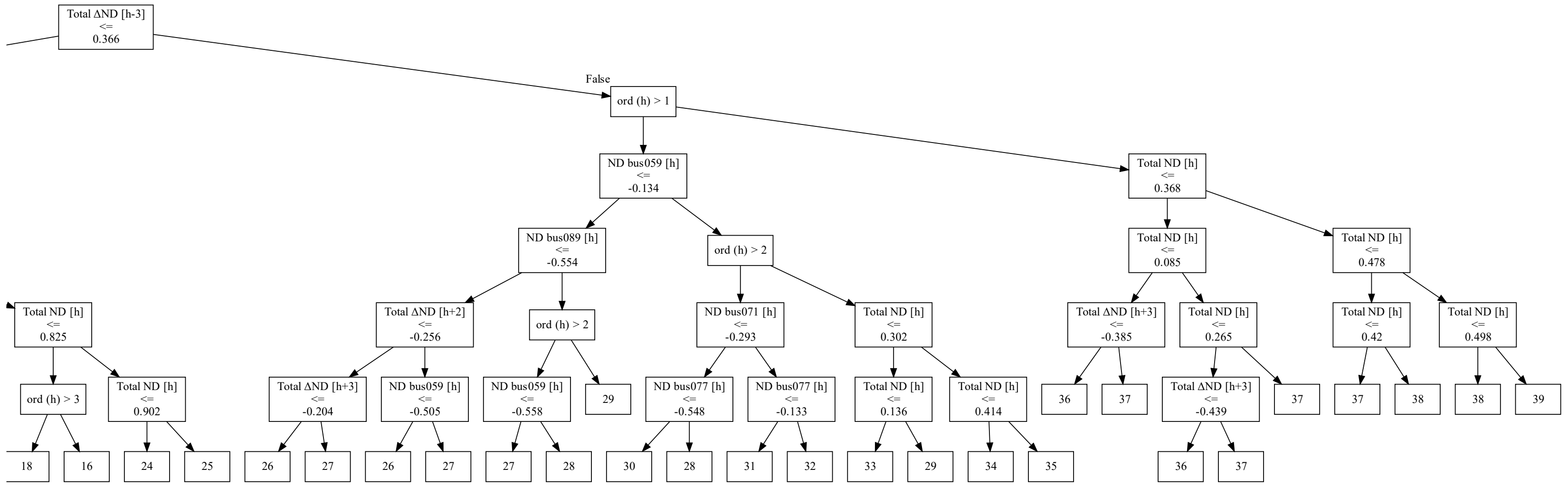

Figure 32 (2): Developed clustered decision tree for vCommit in the network-constrained high wind penetration UC problem.

Table 12: Unique terminal node vCommit outputs by cluster label.

	Clustered terminal node output																																						
Gen.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	1	1	1	1	1
11	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	1	1	1	0	1	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1
20	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
35	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
37	0	0	1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
40	0	1	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	1	1
44	0	0	0	0	0	1	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	0	1	1	0	0	1	1
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1

Table 13: Mapping of the existing generators with their corresponding cluster reference generator.

Reference generator	Clustered generators
1	1, 2, 3, 8, 9
4	4
5	5, 27, 28
6	6, 30
7	7, 12, 13, 16, 17, 18, 19, 22, 23, 25, 26, 31, 32, 33, 38, 41, 42, 46, 47, 48, 49, 50, 51, 52, 53, 54
10	10
11	11, 21, 39
14	14
15	15
20	20
24	24
29	29
34	34
35	35
36	43
37	37
40	40
44	44
45	45

7 Conclusions and Outlook

7.1 Conclusions

In this project, interpretable machine learning models have been developed to explain in a human-understandable way how the unit commitment problem applied to a specific system generates the optimal solutions, as well as to independently predict the decision variables and dual variables of the problem.

Specifically, two models have been developed, a regression and a classification model, to estimate the continuous and binary variables, respectively. Both are based on multi-output decision trees since they offer the best balance between performance and interpretability among all intrinsically interpretable algorithms. On the one hand, the UC problem is complex and highly nonlinear, so decision trees have some advantage over other linear models in addressing the task. On the other hand, the possibility of jointly predicting variables of the same type makes the interpretation of the resulting model extremely easier than with any other approach. Training a single algorithm for each output would not be manageable, and, therefore, neither would they be properly interpretable. In addition, much of the interaction between the output variables would be lost.

Even though the UC problem is characterized by important nonlinear relationships, it also presents numerous fully or partially linear relationships, which cannot be easily modeled by a decision tree, especially if its maximum depth is restricted to ensure interpretability. Therefore, the regression model proposed in this work is a linear regression model tree, which is a decision tree in whose terminal nodes linear regressions models have been trained. In this case, the linear regressions will only include one feature. This approach makes it possible to capture a large amount of information that would escape a normal decision tree without compromising its interpretability.

Conversely, the problem faced by decision tree classifiers is related to the redundancy of sub-trees and terminal nodes, which repeat information unnecessarily. This hinders the interpretability of the model since it implies repeatedly representing the same parameters multiple times. Therefore, in this project, we propose clustering these nodes, which will be compactly represented in an attached look-up table. In this way, nodes presenting an identical combination of outputs can be immediately identified. Furthermore, a significant redundancy has also been detected in the output variables themselves. Therefore, they have also been grouped by generator clusters, further reducing the number of parameters needed to describe the classification model.

The results obtained by applying both approaches to the UC problem studied in this work show that, in general, they achieve an optimal balance between performance and interpretability, usually outperforming in both aspects the rest of the intrinsically interpretable algorithms. In fact, the mean RMSE or accuracy of these models are not excessively far from those obtained using black-box machine learning models, as is the case of GBDTs. Therefore, the implemented models satisfactorily meet the objectives that had been identified at the beginning of the project.

These conclusions apply to most of the variables of the UC problem, especially those related to the thermal generators. However, the results obtained for the estimated flows through power lines suggest that these models are not flexible enough to replicate such a large and complex set of variables.

However, this limitation does not detract from the relevance of the obtained results and the suitability of the developed models to estimate the remaining output variables, which are, besides, the most important variables of the problem.

A meaningful reflection related to the models that have been developed aiming to explain the general operation of the UC problem is that these not only depend on the topology of the system with which the optimal solutions of the problem have been obtained, but they also depend closely on the scenarios that have been elaborated to obtain these solutions. These models, by default, try to extract the most general and useful relationships they find in the data.

Therefore, they will tend to learn the behavior of the UC model primarily in the most common sets of scenarios while tending to disregard extreme or rare cases, which may not be found relevant either because of their low frequency or the error incurred when they are omitted.

It could be argued that if we want to understand the performance of the UC in both common and rare situations, the solution would be to balance the training set so that the model perceives all of them as equally frequent. However, this solution would not be the most appropriate since we would be forcing the model to replicate specific relationships that, according to the dataset, are a minority without providing more helpful information for the model to carry out such a task. Therefore, the algorithm will try to learn these relationships, even if they are not actually general enough.

Analyzing the results in such a context would be meaningless since, if we also balance the test set itself, we are modifying the variability of the test set in our favor. If, on the other hand, we evaluate the model with the original distribution, results will necessarily be worse since we have focused the model on learning certain relationships, which are not the most general ones. While interpretability is paramount, even above the algorithm's own accuracy, it should not be completely ignored. If the model is not able to replicate the data it is intended to explain, it cannot be guaranteed to provide a reliable explanation of how the results were obtained [61].

Consequently, in order to carry out analyses of this type, it would be more appropriate to develop a set of scenarios that properly reflect the behaviors we seek to explain through the application of an IML model so that the model can suitably discern those cases.

Finally, it is important to mention that, even though the presented models allow shedding light on the functioning of the UC problem and explain how it obtains the optimal solutions, the obtained results do not allow us to consider them as surrogate models completely equivalent to the optimization problem itself. Not even the models based on GBDTs, which have been employed as a reference in this project, would be sufficiently accurate to consider such a possibility.

This conclusion is the same as the one reached by numerous studies [35]. Therefore, the developed interpretable models should be understood not as substitutes but as complements to the UC problem. They are truly useful tools to understand, in a general way, the operation of the optimization problem; to estimate which constraints will be active in the optimal solution of a scenario and for what reasons; or to predict approximate solutions, which can also be used as a starting point for the optimization itself.

7.2 Outlook

One of the main challenges of intrinsically interpretable models, especially when dealing with such complex tasks as unit commitment, is to achieve results that can *compete* with other highly accurate but completely opaque models.

As indicated, interpretability has value in itself, meaning that it is not necessary to aim for identical results. However, any effort to improve its accuracy will validate it as a reliable explanation of the underlying UC model.

Further efforts following this work could be focused on searching for alternative models, equally interpretable, which can yield more accurate results than those presented here. However, the simplicity of using a single decision tree for each set of output variables makes these models practically unbeatable from the interpretability point of view. Therefore, the proposed alternatives are aimed at improving the accuracy of the implemented algorithms. Alternative models would only make sense in the case of variables such as vCirPF, for which these models may lack the necessary flexibility.

As for the classification model, the results show that the methodology used can obtain considerably accurate and interpretable estimates. Nevertheless, there is still room for improvement. One of the best alternatives would be trying to increase its depth without compromising its interpretability. For this purpose, the best option would be to carry out the tree depth selection together with that of the maximum number of terminal nodes. The current implementation does not allow this simultaneity since it determines the optimal maximum number of leaves for each depth independently, and this will tend to correspond to that of a fully-grown tree. However, restricting the number of terminal nodes while giving more flexibility from the depth side may allow modeling more complex feature space partitions without exponentially increasing the number of model parameters.

Nevertheless, the main potential improvement is related to the model tree proposed in this project. Implementing a model tree involves an enormous computational burden. For each possible partition of the feature space, the model must train the corresponding complementary algorithm on each of the hypothetically resulting nodes and select the optimal partition. In this work, the complementary model is a single feature linear regression, which iteratively selects the most appropriate input variable. This further increases the time complexity of the approach.

Given these obstacles, the model has been simplified by training the linear regressions on a previously built decision tree. However, the partitions carried out by the DT are far from optimal for the linear regression models since the DT will try to split the linear relationships into pieces, while the model tree could automatically handle them in a single terminal node.

Therefore, it would be highly recommended to explore the implementation of a linear regression model tree that is more faithful to the original concept. To do so, it will be necessary to increase the efficiency of the processes involved. For instance, the dimension of the feature space could be further reduced, or the search for the optimal variable in linear regressions could be parallelized.

A more efficient implementation of the model tree would allow a hyperparameter selection more adapted to the actual performance of the algorithm and substantially improve the results obtained for continuous variables.

8 References

[1] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive Robust Optimization for the Security Constrained Unit Commitment Problem," *IEEE Transactions on Power Systems,* vol. 28, no. 1, p. 52–63, 2013.

[2] D. A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos, "Which Unit-Commitment Formulation is Best? A Comparison Framework," *IEEE Transactions on Power Systems,* vol. 35, no. 4, pp. 2926-2936, 2020.

[3] M. Silbernagl, M. Huber, and R. Brandenberg, "Improving Accuracy and Efficiency of Start-Up Cost Formulations in MIP Unit Commitment by Modeling Power Plant Temperatures," *IEEE Transactions on Power Systems,* vol. 31, no. 4, pp. 2578-2586, 2016.

[4] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra, "Large-scale Unit Commitment under uncertainty," *4OR,* vol. 13, no. 2, p. 115–171, 2015.

[5] A. M. Foley, P. G. Leahy, A. Marvuglia, and E. J. McKeogh, "Current methods and advances in forecasting of wind power generation," *Renewable Energy,* vol. 37, no. 1, pp. 1-8, 2012.

[6] C. Molnar, Interpretable machine learning. A Guide for Making Black Box Models Explainable, Leanpub, 2019.

[7] Red Eléctrica, "Balance de generación programada - PVP 05/04/2021," ESIOS, https://www.esios.ree.es/es/balance?date=05-04-2021&program=PVP&agg=hour.

[8] P. Bendotti, P. Fouilhoux, and C. Rottner, "On the complexity of the Unit Commitment Problem," *Annals of Operations Research,* vol. 274, no. 1, pp. 119-130, 2018.

[9] M. Carrion and J. M. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1371-1378, 2006.

[10] I. Abdou and M. Tkiouat, "Unit Commitment Problem in Electrical Power System: A Literature Review," *International Journal of Electrical and Computer Engineering,* vol. 8, no. 3, pp. 1357-1372, 2018.

[11] A. J. Conejo and L. Baringo, Power System Operations, Springer, 2018.

[12] Power Optimisation, "Unit Commitment and Economic Dispatch Software to Optimise the Short-Term Scheduling of Electrical Power Generation," Beaconsfield.

[13] E. Du et al., "Operation of a High Renewable Penetrated Power System With CSP Plants: A Look-Ahead Stochastic Unit Commitment Model," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 140-151, 2019.

[14] L. S. M. Guedes, P. de Mendonça Maia, A. C. Lisboa, D. A. G. Vieira, and R. R. Saldanha, "A Unit Commitment Algorithm and a Compact MILP Model for Short-Term Hydro-Power Generation Scheduling," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3381-3390, 2017.

[15] Y. Fu, M. Shahidehpour, and Z. Li, "Security-Constrained Unit Commitment With AC Constraints," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1001-1013, 2005.

[16] T. Li and M. Shahidehpour, "Price-based unit commitment: a case of Lagrangian relaxation versus mixed integer programming," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 2015-2025, 2005.

[17] C. C. Marín-Cano, J. E. Sierra-Aguilar, J. M. López-Lezama, Á. Jaramillo-Duque, and W. M. Villa-Acevedo, "Implementation of User Cuts and Linear Sensitivity Factors to Improve the Computational Performance of the Security-Constrained Unit Commitment Problem," *Energies*, vol. 12, no. 7, p. 1399, 2019.

[18] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[19] A. L. Fradkov, "Early History of Machine Learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385-1390, 2020.

[20] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning, Springer, 2013.

[21] J. R. Rice, The Algorithm Selection Problem, Advances in Computers, 1976.

[22] T. Hastie, R. Tibshirani, and J. H. Friedman, The Elements of Statistical Learning, Springer, 2001.

[23] T. O. Ayodele, New Advances in Machine Learning, InTech, 2010.

[24] C. Molnar, G. Casalicchio, and B. Bischl, Interpretable Machine Learning - A Brief History, State of the Art and Challenges, ECML PKDD 2020 Workshops, 2020.

[25] C. Rudin, Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead, Nature Machine Intelligence, 2019.

[26] A. Koene, "Algorithmic Bias: Addressing Growing Concerns," *IEEE Technology and Society Magazine*, vol. 36, no. 2, pp. 31-32, 2017.

[27] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1-38, 2019.

[28] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 44, pp. 22071-22080, 2019.

[29] M. Du, N. Liu, and X. Hu, "Techniques for Interpretable Machine Learning," *Communications of the ACM*, vol. 63, no. 1, pp. 68-77, 2019.

[30] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model," *Applied Statistics*, vol. 9, no. 3, pp. 1350-1371, 2015.

[31] J. Zeng, B. Ustun, and C. Rudin, "Interpretable classification models for recidivism prediction," *Royal Statistical Society*, vol. 180, no. 3, pp. 689-722, 2017.

[32] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1532-4435, 2003.

[33] M. Hossain, S. Mekhilef, M. Danesh, L. Olatomiwa, and S. Shamshirband, "Application of extreme learning machine for short term output power forecasting of three grid-connected PV systems," *Journal of Cleaner Production*, vol. 167, pp. 395-405, 2017.

[34] M. H. Sendaula, S. K. Biswas, A. Eltom, C. Parten, and W. Kazibwe, "Application of artificial neural networks to unit commitment," *Proceedings of the First International Forum on Applications of Neural Networks to Power Systems*, pp. 256-260, 1991.

[35] F. Mohammadi, M. Sahraei-Ardakani, D. Trakas, and N. D. Hatziargyriou, "Machine Learning Assisted Stochastic Unit Commitment during Hurricanes with Predictable Line Outages," *IEEE Transactions on Power Systems*, 2021.

[36] S. J. Huang and C. L. Huang, "Application of genetic-based neural networks to thermal unit commitment," *IEEE Transactions on Power Systems*, vol. 12, no. 2, pp. 654-660, 1997.

[37] Z. Ouyang and S. M. Shahidehpour, "A hybrid artificial neural network-dynamic programming approach to unit commitment," *IEEE Transactions on Power Systems,* vol. 7, no. 1, pp. 236-242, 1992.

[38] R. Nayak and J. D. Sharma, "A hybrid neural network and simulated annealing approach to the unit commitment problem," *Computers & Electrical Engineering,* vol. 26, no. 6, pp. 461-477, 2000.

[39] K. Baker, "Learning Warm-Start Points For Ac Optimal Power Flow," *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing,* pp. 1-6, 2019.

[40] A. S. Xavier, F. Qiu, and S. Ahmed, "Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems," *Informs Journal on Computing,* vol. 33, no. 2, pp. 419-835, 2020.

[41] I. Blanco and J. M. Morales, "An Efficient Robust Solution to the Two-Stage Stochastic Unit Commitment Problem," *IEEE Transactions on Power Systems,* vol. 32, no. 6, pp. 4477-4488, 2017.

[42] J. Meus, K. Poncelet, and E. Delarue, "Applicability of a Clustered Unit Commitment Model in Power System Modeling," *IEEE Transactions on Power Systems,* vol. 33, no. 2, pp. 2195-2204, 2018.

[43] P. G. Latha, "An Efficient Machine Learning Algorithm for Unit Commitment Problem," *International Journal of Applied Engineering Research,* vol. 13, no. 3, pp. 135-141, 2018.

[44] J. E.A., I. A. T.P., and J. R. V.P., "Reinforcement Learning Solution for Unit Commitment Problem through Pursuit Method," *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies,* pp. 324-327, 2009.

[45] C. A. Coronado, M. R. Figueroa, and C. A. Roa-Sepulveda, "A reinforcement learning solution for the unit commitment problem," *2012 47th International Universities Power Engineering Conference,* pp. 1-6, 2012.

[46] G. Dalal and S. Mannor, "Reinforcement learning for the unit commitment problem," *2015 IEEE Eindhoven PowerTech,* pp. 1-6, 2015.

[47] F. Aminifar, M. Fotuhi-Firuzabad, and M. Shahidehpour, "Unit commitment with probabilistic spinning reserve and interruptible load considerations," *IEEE Transactions on Power Systems,* vol. 24, no. 1, p. 388–397, 2009.

[48] B. Hu, L. Wu, and M. Marwali, "On the Robust Solution to SCUC With Load and Wind Uncertainty Correlations," *IEEE Transactions on Power Systems,* vol. 29, no. 6, pp. 2952-2964, 2014.

[49] R. Jiang, J. Wang, M. Zhang, and Y. Guan, "Two-Stage Minimax Regret Robust Unit Commitment," *IEEE Transactions on Power Systems,* vol. 28, no. 3, pp. 2271-2282, 2013.

[50] C. Sahin, M. Shahidehpour, and I. Erkmen, "Allocation of hourly reserve versus demand response for security-constrained scheduling of stochastic wind energy," *IEEE Transactions on Sustainable Energy,* vol. 4, no. 1, p. 219–228, 2013.

[51] D. A. Tejada-Arango, "GitHub," 11 March 2021. [Online]. Available: https://github.com/datejada/modified-ieee-118-bus-system-data.

[52] I. Staffell and S. Pfenninger, "Using Bias-Corrected Reanalysis to Simulate Current and Future Wind Power Output," *Energy,* vol. 114, pp. 1224-1239, 2016.

[53] United Nations, "Transforming our world: the 2030 Agenda for Sustainable Development," New York, 2015.

[54] Unted Nations, "The Sustainable Development Goals Report," New York, 2017.

[55] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, "Data Preprocessing for Supervised Leaning," *International Journal of Computer and Information Engineering,* vol. 1, no. 12, pp. 4104-4109, 2007.

[56] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing,* vol. 97, no. B, 2020.

[57] J. Snoek, H. Larochelle, and R. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems 25,* p. 2960–2968, 2012.

[58] O. Kisi, J. Shiri, and V. Demir, "Hydrological Time Series Forecasting Using Three Different Heuristic Regression Techniques," in *Handbook of Neural Computation,* Academic Press, 2017, pp. 45-65.

[59] A. Wong, "Building Model Trees - GitHub," 2020. [Online]. Available: https://github.com/ankonzoid/LearningX/tree/master/advanced_ML/model_tree.

[60] Z. Huang and M. K. Ng, "A fuzzy k-modes algorithm for clustering categorical data," *IEEE Transactions on Fuzzy Systems,* vol. 7, no. 4, pp. 446-452, 1999.

[61] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, methods, and applications in interpretable machine learning," *Proceedings of tje*

National Academy of Sciences of the United States of America, vol. 116, no. 44, pp. 22071-22080, 2019.