



**Business
School**

MONETIZACIÓN DE DATOS PARA ENTORNOS
INDUSTRIALES Y DE GENERACIÓN
ELÉCTRICA, MEDIANTE MACHINE LEARNING
E INTELIGENCIA DE NEGOCIO.

TRABAJO FIN DE MASTER
Curso: 2018 / 19

Realizado por: Jesús Ponce Gómez

RESUMEN.

En este TFM haremos uso de diferentes herramientas análisis de negocio, así como diferentes técnicas de aprendizaje automático (machine learning), con objeto de establecer un modelo matemático que nos permita realizar predicciones e identificar patrones de comportamiento de una turbina de gas de 14.0 MWe, empleada en una industria de cogeneración. Igualmente emplearemos herramientas de almacenamiento de datos, como MongoDB y de visualización, como Tableau, de tal manera que podamos identificar fácilmente esas tendencias y comportamientos.

En el caso de estudio de este TFM, emplearemos datos procedentes de un Scada, donde se registran miles de señales electrónicas, que aseguran el correcto funcionamiento de la máquina, en términos de seguridad y eficiencia productiva. Nuestro reto será determinar "*Qué variables y cómo determinan la Potencia Máxima que la máquina es capaz de generar?*". Esta será la pregunta de negocio que nos ocupará en este trabajo.

Es extensa la bibliografía sobre máquinas rotativas y ciclos termodinámicos que podrían dar respuesta a esta pregunta en un marco teórico. Esto supone un problema, la imposibilidad de encontrar un caso que describa las particularidades precisas de nuestra máquina e instalación. Por ello, proponemos utilizar la ciencia de datos para describir el comportamiento de la máquina para modelizar **la máxima potencia** que es capaz de desarrollar y ser capaces de optimizar nuestro proceso de toma decisiones, allí donde nuestros conocimientos técnicos y experiencia no llegan.

INDICE GENERAL.

1. Introducción
 - 1.1. Descripción del sistema y proceso objeto de estudio
 - 1.2. Modelo Machine Learning
2. Adquisición de datos
 - 2.1. Adquisición de datos vía OPC Server
 - 2.2. Adquisición de datos desde el Sistema de registro de Históricos
3. Fases de creación del modelo
4. Descriptiva del Proceso de Machine Learning
5. Ingeniería de Características. Selección de variables
 - 5.1. Selección de características con SelectKBest
 - 5.2. Selección de características con método Forward de mejor r2
 - 5.3. Conclusiones: comparativa de ambos métodos de selección
6. Análisis de sensibilidad para la detección de patologías
 - 6.1. Obtención de los datasets de la simulación de control diario.
 - 6.2. Entrenamiento y chequeo del modelo
 - 6.3. Almacenamiento y visualización
 - 6.3.1. Instalación de MongoDB
 - 6.3.2. Instalación de Tableau
 - 6.3.3. Conexión de MongoDB con Tableau
7. Análisis visual en tiempo real con Tableau
 - 7.1. Visualización r2 train y r2 test
 - 7.2. Visualización de desvíos en la simulación de control diario
 - 7.3. Visualización de Características frente a la variable objetivo
8. Predicción basada en las condiciones ambientales

- 8.1. Sensorización de las condiciones ambientales
- 8.2. Entrenamiento del modelo
- 8.3. Uso de series temporales
9. Análisis basado en los estadísticos del modelo
10. Conclusiones

11. ANEXOS

- ANEXO I. Comunicación vía OPC
- ANEXO II. Adecuación datos capturados desde el Scada de Registros Históricos.
- ANEXO III. Desarrollo del Modelo de Machine Learning
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO IV. Selección de variables con SelectKBest
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO V. Selección de variables. Método Forward Selection
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO VI. Simulación Control Diario
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO VII. Desarrollo de los modelos diarios
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO VIII. Código Función Almacenamiento en MongoDB
- ANEXO IX. Código cálculo iterativo de la temperatura de bulbo húmedo
- ANEXO X. Modelo basado en Condiciones Ambientales
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones
- ANEXO XI. Análisis mediante Series temporales
 - i. NoteBook de funciones
 - ii. Scripts con el código de funciones

1. Introducción.

En este apartado realizaremos una breve descripción del proceso de generación eléctrica mediante una turbina de gas, basada en un ciclo termodinámico de Brayton. Posteriormente realizaremos una descripción de cómo fueron adquiridos los datos, de las diferentes fases para la creación del modelo para la predicción de nuestra variable objetivo y por último haremos una introducción de los modelos de regresión lineal múltiple, que han sido empleados en este caso de estudio.

1.1. Descripción del sistema y proceso objeto de estudio.

El modelo termodinámico de las turbinas de gas se fundamenta en el ciclo de Brayton, a pesar de que se generaliza como ciclo termodinámico, en realidad el fluido de trabajo no cumple un ciclo completo en las turbinas de gas ya que este finaliza en un estado diferente al que tenía cuando inició el proceso. Se podría decir que es un ciclo abierto. Las turbinas de gas de ciclo abierto simple utilizan una cámara de combustión interna para suministrar calor al fluido de trabajo y las turbinas de gas de ciclo cerrado simple utilizan un proceso de transferencia para agregar o remover calor del fluido de trabajo.

El ciclo básico de Brayton en condiciones ideales está compuesto por cuatro procesos:

- 1-2. Compresión isentrópica en un compresor.
- 2-3. Adición de calor al fluido de trabajo a presión constante en un intercambiador de calor o una cámara de combustión.
- 3-4. Expansión isentrópica en una turbina.
- 4-1. Remoción de calor del fluido de trabajo a presión constante en un intercambiador de calor o en la atmósfera.

En el ciclo Brayton, el trabajo neto realizado por unidad de masa es la diferencia entre el trabajo obtenido en la expansión y el trabajo invertido en la compresión, es decir: $W_{net} = W_t - W_c$

En este proceso el aire de combustión, antes de ser comprimido pasa por una filtración de partículas y por un enfriamiento mediante una humectación que lleva el aire a su temperatura de bulbo húmedo (que es la mínima teórica que se puede alcanzar por humectación), al objeto de maximizar la unidad de masa por unidad de volumen e incrementar el trabajo de la máquina. Una vez comprimimos el aire en un compresor, pasa a la cámara de combustión con gas natural, donde se produce el aporte de calor al ciclo, y por último, en la turbina de potencia se realiza la transmisión de la energía térmica a energía mecánica, en forma de rotación de un eje acoplado al eje en un generador eléctrico, donde transfiere la energía mecánica a energía eléctrica, produciendo la electricidad que llega a nuestros hogares a través de las diferentes redes de distribución y transporte.

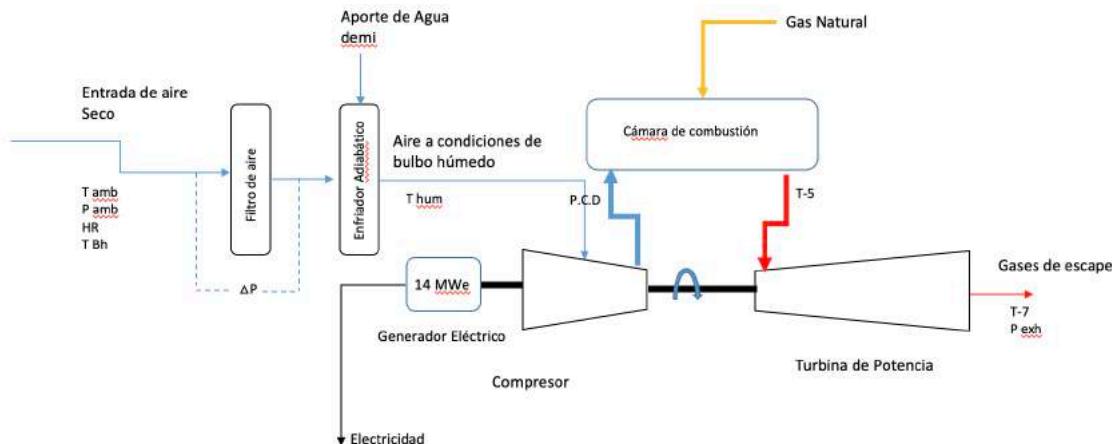


Fig. 1

La eficiencia térmica del ciclo Brayton ideal depende de la compresión. Si se aumenta la relación de compresión en el ciclo será necesario suministrar más calor al sistema debido a que las líneas de presión constante divergen hacia arriba y hacia la derecha en el diagrama T-s y la temperatura máxima del ciclo será mayor. Como el calor suministrado es mayor, la eficiencia térmica aumentará con el ratio de compresión.

Sin embargo la temperatura máxima del ciclo está limitada por los materiales en los cuales están construidos los componentes y por lo tanto se requerirán sistemas de refrigeración más eficientes.

La eficiencia del ciclo también se ve afectada por las pérdidas en el compresor, en la turbina y en las caídas de presión en la cámara de combustión y otros pasajes. Podemos verlo en el diagrama que representa estas condiciones en el ciclo, disminuyendo en consecuencia la eficiencia del ciclo.

Intuimos la importancia de las variables de presión y temperatura del proceso, ya que el rendimiento de la máquina dependerá de qué presión y temperatura conseguimos en la combustión y qué presión y temperatura obtenemos tras la expansión del fluido.

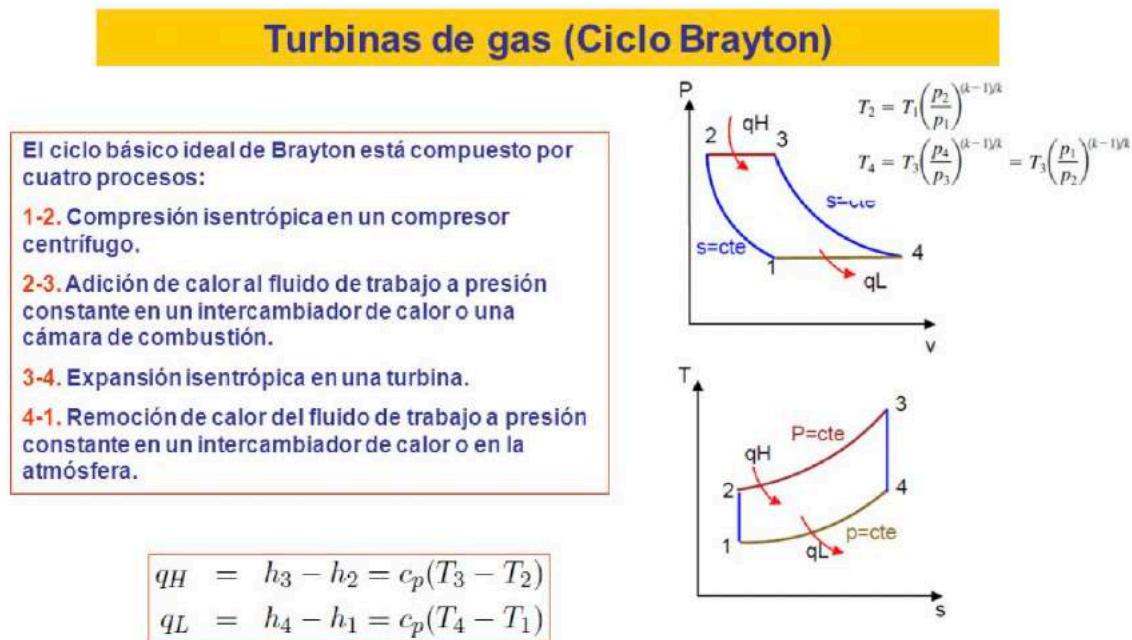


Fig. 2

1.2. Modelo Machine Learning.

Este será el esquema básico que seguiremos para obtener un modelo predictivo en nuestra industria:

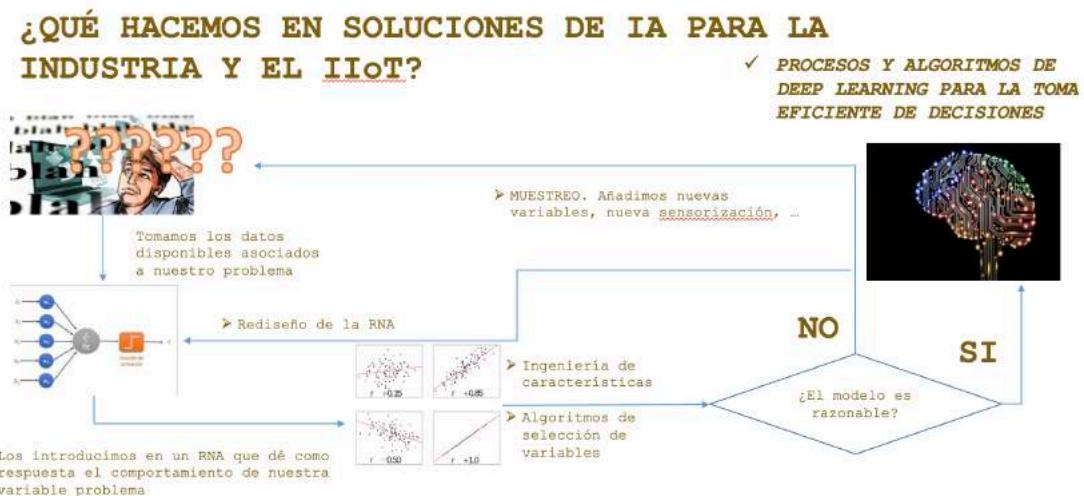


Fig. 3

- Nos enfrentamos un problema.

Ante un problema o cuestión, que mediante nuestra experiencia o conocimientos técnicos no sepamos dar respuesta, recurrimos a la ciencia de datos para encontrar respuesta.

- Modelo de regresión lineal Múltiple.

Para ello utilizaremos los datos disponibles para representar la dependencia de una variable Y (variable dependiente o variable respuesta) con respecto a tras variables X₁, X₂, X₃, ..., X_n (variables independientes o variables explicativas). Para determinar dicha dependencia o respuesta, emplearemos modelo de regresión, con dos objetivos: describir la forma de dependencia, es decir, conocer como la variable Y depende de las variables X, y ser capaces de predecir el valor de Y, conocidos los valores de las X.

iii. Librerías Scikit-learn de Python.

Python es uno de los lenguajes más populares en la actualidad para implementación de modelos de aprendizaje automático debido a su sencillez y potencia. Se utilizarán las librerías scikit-learn y Jupyter Notebook, como editor para ejecutar el código Python.

Scikit-learn es una librería de open source para el lenguaje de programación Python que implementa diferentes métodos de machine learning. Además cuenta con diversos algoritmos para la implementación de los modelos de regresión y reducción de la dimensionalidad y selección de variables.

iv. Determinar la razonabilidad de un modelo.

Una vez obtenido el modelo es necesario conocer la calidad del mismo a la hora de predecir resultados, para lo cual existen diferentes estimadores:

- MSE (Mean Squared Error): El valor medio de la diferencia entre la variable independiente y la predicción al cuadrado:

$$MSE = \frac{1}{n} \sum (y_i - y(x_i))^2$$

- MAE (Mean Absolut Error): Este estimador mide los errores al cuadrado, penalizando los puntos en los que la discrepancia entre la variable independiente y la predicción es mayor.

$$MAE = \frac{1}{n} \sum |(y_i - y(x_i))|$$

- MedAE (Median Absolut Error): Este estimador evita los problemas de valores atípicos, en los cuales el valor estimado por el modelo muestra una gran discrepancia con los originales.

$$\text{MedAE} = \text{median}(|y_1 - y(x_1)|, |y_2 - y(x_2)|, \dots, |y_n - y(x_n)|)$$

- Coeficiente de determinación.

Uno de los estimadores más utilizados es el coeficiente de determinación o R^2 . Este coeficiente es un estadístico que se define como 1 menos el cociente entre la varianza de los residuos, partido por la varianza de la variable dependiente, es decir:

$$R^2 = 1 - \frac{\sigma_r^2}{\sigma^2}$$

con

$$\sigma_r^2 = \sum (y_i - \hat{y}(x_i))^2$$

y

$$\sigma^2 = \sum (y_i - \bar{y})^2$$

Aquí \bar{y} representa la media de los valores de la variable dependiente.

El R^2 se puede interpretar como el porcentaje de la variabilidad total de la variable dependiente respecto a la media que se puede explicar con el modelo. Idealmente su valor debería ser 1, lo que indicaría que el modelo reproduce perfectamente los datos del conjunto de entrenamiento. En el caso de un modelo constante, que reproduce la media de la variable dependiente, el valor de R^2 será 0.

Estos cuatro estimadores se encuentran en la librería scikit-learn

v. Ingeniería y Selección de Características.

El proceso de selección de las características que han de ser incluidas en un modelo es fundamental. Antes de la creación de un modelo no se puede saber cuáles son las más significativas, las menos significativas ni tampoco las que simplemente son ruido. Una selección poco exigente puede terminar en un modelo con demasiadas características que puede acabar en sobreajuste. Por otro lado, una eliminación excesiva puede llevar a que no se consigan obtener modelos útiles.

Una vez cargado el conjunto de datos, se pueden ir eliminando características en función de un conjunto de métodos que se pueden enumerar en base a su complejidad, de menor a mayor:

- Eliminar características con poca capacidad predictiva.
- Características con baja varianza: constantes.
- Identificadores: nombres o ID.
- Análisis univariante.
- Eliminación recursiva de características.

En caso de que el modelo no sea razonable, bien por el número de variables, o bien por unos estimadores de bajo valor, deberemos iniciar el proceso, bien seleccionando un nuevo modelo de regresión, o transformación de las variables mediante métodos de normalización o discretización.

vi. Análisis de sensibilidad para la detección de “patologías”.

Una vez tengamos un modelo razonable, ejecutaremos algoritmos de Deep Learning, que permitan ver la sensibilidad y variación de los principales estadísticos y coeficientes del modelo. Así, nuestro objetivo será identificar y cuantificar la pérdida respecto de unas condiciones básicas del proceso y la causa de dicha pérdida. Este conocimiento nos permitirá ser más eficiente en nuestra gestión de la O&M de la instalación, mediante una toma de decisiones más eficiente.

2. Adquisición de Datos.

Este es el proceso crítico y más importante, identificar las correctas fuentes de datos y disponer la tecnología adecuada para su almacenamiento, y posterior extracción, transformación y carga en nuestras herramientas de análisis.

Para la adquisición de datos hemos desarrollado dos métodos, por un lado empleamos la librería OpenOPC de Python, para conexión con el Server OPC, y por otro lado emplearemos la aplicación del Scada WinCC de Siemens, para la gestión de Registros Históricos, la cual no permite obtener ficheros tipo csv dados unos requerimientos.

2.1. Adquisición de datos vía OPC Server.

En este apartado explicaremos cómo y de dónde obtenemos los datasets para este TFM, qué herramientas se han empleado y que ampliaciones en la sensorización de la instalación se ha llevado a cabo, al objeto de obtener un mejor modelo.

Partimos de que los datos de proceso son comunicados vía OPC a un sistema Scada, a tal efecto instalaremos Python 2.7 en una de las estaciones de supervisión y emplearemos la librería OpenOPC, para la captura de los datos. En el Anexo I, mostramos los pasos para la instalación de la librería OpenOPC y los scripts con los códigos en Python para el establecimiento de la comunicación y captura de datos, los cuales serán almacenados en ficheros .csv, para su posterior uso en los modelos de regresión.

2.2. Adquisición de datos desde el Sistema de Registro de Históricos.

Otra forma de adquirir datos del Scada es mediante la programación de una tarea en el gestor de registros históricos del Scada, que nos devuelve un fichero .csv con las variables seleccionadas y con la periodicidad de tiempo definida.

Los valores históricos o su evolución están guardados en archivos de valores del proceso. Además de valores de proceso, WinCC también archiva avisos y datos de usuario. Los datos se guardan en los denominados archivos, en la base de datos de Microsoft SQL Server operando con alto rendimiento: Para el servidor central de archivos no supone ningún problema almacenar hasta 10.000 valores medidos y 100 avisos por segundo en modo permanente (con una avalancha de hasta 15.000 avisos en 10 segundos). La capacidad de memoria necesaria es muy baja gracias a las potentes funciones de compresión sin pérdidas. Los valores de proceso se pueden archivar de forma cíclica (continua), controlada por evento o proceso (p. ej. en caso de sobrepasar determinados límites), comprimidos (p. ej. promediados) e incluso integrados.

Esta forma de adquirir datos presenta el inconveniente de que los datasets son irregulares en tamaño y en la periodicidad de adquisición y puede presentar gran cantidad de valores nulos o perdidos. Los datos son adquiridos en pares valor variable – instante de captura. Todo ello implicó la necesidad de realizar un Script para la adecuación de los datos y éstos puedan ser tratados posteriormente en los modelos de regresión.

En el ANEXO II se presenta el código de dicho Script. Así pasamos de un fichero de las características mostradas en la imagen donde cada variable Y, viene asociada a su variable Time:

A1	B	C	D	E	F	G	
	Temperatura ambiente TG Time	Temperatura ambiente TG ValueY	Temperatura Bulbo H medo Time	Temperatura Bulbo H medo ValueY	PresiÓN Atmosferica Time	PresiÓN Atmosferica ValueY	Temperatura T
1	Temperatura ambiente TG Time	9,557292938	7/12/18 1:26	7,499995232	7/12/18 1:26	974,2592529	7/12/
2	7/12/18 1:26	9,557292938	7/12/18 1:26	7,499995232	7/12/18 1:26	973,7036865	7/12/
3	7/12/18 1:26	9,557292938	7/12/18 1:26	7,499995232	7/12/18 1:26	975,8333496	7/12/
4	7/12/18 1:26	9,557292938	7/12/18 1:26	7,499995232	7/12/18 1:26	975,0000244	7/12/
5	7/12/18 1:27	9,557292938	7/12/18 1:27	7,499995232	7/12/18 1:26	975,8333496	7/12/
6	7/12/18 1:27	9,557292938	7/12/18 1:27	7,499995232	7/12/18 1:26	975,8333496	7/12/
7	7/12/18 1:27	9,557292938	7/12/18 1:27	7,499995232	7/12/18 1:26	974,9073975	7/12/
8	7/12/18 1:27	9,557292938	7/12/18 1:27	7,499995232	7/12/18 1:27	975,4629761	7/12/
9	7/12/18 1:27	9,557292938	7/12/18 1:27	7,499995232	7/12/18 1:27	974,6296143	7/12/
10	7/12/18 1:27	9,557292938	7/12/18 1:27	7,399995327	7/12/18 1:27	975,2777832	7/12/
11	7/12/18 1:28	9,557292938	7/12/18 1:28	7,399995327	7/12/18 1:27	975,2777832	7/12/
12	7/12/18 1:28	9,557292938	7/12/18 1:28	7,399995327	7/12/18 1:27	974,3518188	7/12/
13	7/12/18 1:28	9,557292938	7/12/18 1:28	7,399995327	7/12/18 1:27	973,5184937	7/12/
14	7/12/18 1:28	9,557292938	7/12/18 1:28	7,399995327	7/12/18 1:27	973,7962769	7/12/
15	7/12/18 1:28	9,602142334	7/12/18 1:28	7,399995327	7/12/18 1:27	973,9814697	7/12/
16	7/12/18 1:28	9,602142334	7/12/18 1:28	7,399995327	7/12/18 1:27	973,8888916	7/12/
17	7/12/18 1:29	9,602142334	7/12/18 1:29	7,399995327	7/12/18 1:27	971,0184937	7/12/
18	7/12/18 1:29	9,602142334	7/12/18 1:29	7,399995327	7/12/18 1:27	973,9814453	7/12/
19	7/12/18 1:29	9,602142334	7/12/18 1:29	7,399995327	7/12/18 1:27	973,4258911	7/12/
20	7/12/18 1:29	9,602142334	7/12/18 1:29	7,399995327	7/12/18 1:28	973,7036865	7/12/
21	7/12/18 1:29	9,64699173	7/12/18 1:29	7,499995232	7/12/18 1:28	972,7777466	7/12/
22	7/12/18 1:29	9,64699173	7/12/18 1:29	7,499995232	7/12/18 1:28	975,4629883	7/12/
23	7/12/18 1:30	9,64699173	7/12/18 1:30	7,499995232	7/12/18 1:28	973,8888672	7/12/
24	7/12/18 1:30	9,64699173	7/12/18 1:30	7,499995232	7/12/18 1:28	974,0740845	7/12/
25	7/12/18 1:30	9,64699173	7/12/18 1:30	7,499995232	7/12/18 1:28	974,0740601	7/12/
26	7/12/18 1:30	9,64699173	7/12/18 1:30	7,499995232	7/12/18 1:28	973,2407104	7/12/
27	7/12/18 1:30	9,691841125	7/12/18 1:30	7,499995232	7/12/18 1:28	973,9814575	7/12/
28	7/12/18 1:30	9,691841125	7/12/18 1:30	7,499995232	7/12/18 1:28	973,6110962	7/12/
29	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:28	973,7962891	7/12/
30	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:28	973,0555298	7/12/
31	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:28	974,6296143	7/12/
32	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:29	974,7222412	7/12/
33	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:29	973,9814453	7/12/
34	7/12/18 1:31	9,691841125	7/12/18 1:31	7,499995232	7/12/18 1:29	974,6296265	7/12/
35	7/12/18 1:32	9,691841125	7/12/18 1:32	7,499995232	7/12/18 1:29	974,4444336	7/12/
36	7/12/18 1:32	9,691841125	7/12/18 1:32	7,499995232	7/12/18 1:29	972,499939	7/12/
37	7/12/18 1:32	9,691841125	7/12/18 1:32	7,499995232	7/12/18 1:29	972,8703613	7/12/
38	7/12/18 1:32	9,691841125	7/12/18 1:32	7,499995232	7/12/18 1:29	974,8148438	7/12/

Fig. 4

A un fichero de la siguiente estructura mostrada en la Figura 5, donde se han eliminado las variables Time, se han eliminado los valores nulos o perdidos, y además las variables que han sido capturadas con diferente frecuencia han sido combinadas para que todas las variables resultantes coincidan en el mismo instante de captura:

B1									
A	B	C	D	E	F	G	H	T5 #1	T5 #2
1	Temperatura ambiente TG ValueY	Temperatura Bulbo Humedo ValueY	Presión Atmosférica ValueY	Temperatura T1 ValueY	Presión P.C.D. ValueY	Factor corrección potencia TG ValueY	-0,40290001	744,2999878	711,2
2	0,12,74160767	11,40000725	965,3703613	12,53556824	15,35188866		-0,40290001	746,2000122	
3	1,13,5489006	12,00000954	966,7592773	13,26523209	15,30862427		-0,40290001	745,7999878	713,4
4	2,13,81799698	12,00000954	965,7407349	13,19226456	15,27978039		-0,40290001	745,7999878	713,2
5	3,13,27980423	11,60000801	965,2777832	12,97336578	15,27978039		-0,40290001	745,7999878	713,2
6	4,13,72829819	11,60000801	965,1851929	12,75446701	15,32304573		-0,40290001	747,0999756	711,7
7	5,13,77314758	11,50000763	965,4630005	12,8639183	15,34467793		-0,40290001	745,5	711,9
8	6,12,69675827	11,20000648	966,0185547	12,64502335	15,32304573		-0,40290001	746	711,5
9	7,12,51736069	10,90000534	964,1666992	12,13425827	15,315835		-0,40290001	745,2000122	712,5
10	8,12,11371613	10,60000042	964,2592651	11,84239197	15,35910034		-0,40290001	745,0999756	711,4
11	9,11,30642319	10,20000267	963,9815063	11,33163452	15,40957546		-0,40290001	745	710,5
12	10,11,79977036	10,50000381	964,16666748	11,40459442	15,38073254		-0,40290001	746	
13	11,11,21567244	10,10000229	964,6296387	11,14921188	15,38794327		-0,40290001	746,5	
14	12,11,39612198	10,30000305	963,0555542	11,44107819	15,35188866		-0,40290001	746,2000122	711,7
15	13,12,06886673	10,50000381	964,722229	11,65997696	15,35188866		-0,40290001	744,7000122	710,7
16	14,12,56221008	10,90000534	964,2592773	12,24370956	15,315835		-0,40290001	746,0999756	711,0
17	15,12,74160767	11,00000572	965,00000122	12,42612457	15,28699112		-0,40290001	747,0999756	
18	16,13,5489006	11,40000725	964,0740967	12,97336578	15,27256966		-0,40290001	744,0999756	712,0
19	17,13,99739456	11,70000839	966,4814941	13,08281708	15,30141354		-0,40290001	745,9000244	712,0
20	18,12,96585464	11,30000687	966,2037109	12,97336578	15,38794327		-0,40290001	746	712,0
21	19,15,29803085	12,50001144	966,944458	14,46917343	15,27256966		-0,40290001	747	714,4
22	20,16,32957077	13,30000145	967,1296265	15,30828094	15,19325066		-0,40290001	746,5999756	713,2
23	21,16,28472137	13,10001373	968,7037109	15,70959473	15,22930431		-0,40290001	744,7000122	712,2
24	22,16,59866714	13,20001411	968,7962769	16,11090851	15,13556385		-0,40290001	745,4000244	712,4
25	23,16,91261673	13,20001411	968,2407349	15,89200974	15,15719604		-0,40290001	747,2000122	711,7
26	24,17,58535767	13,40001488	968,7963013	16,11090851	15,15719604		-0,40290001	746,4000244	
27	25,18,033385544	13,60001564	969,1666626	16,2203598	15,15719604		-0,40290001	745,5999756	713,5
28	26,17,76475525	13,60001564	968,7036865	16,2203598	15,14277458		-0,40290001	744,0999756	
29	27,18,21325302	13,50001526	969,3518066	16,402771	15,14277458		-0,40290001	744	
30	28,18,97569275	14,20001793	969,2592285	16,5122261	15,09950924		-0,40290001	743,5	
31	29,19,42419052	13,8000164	969,7222168	16,6946373	15,12114239		-0,40290001	746,7000122	712,5
32	30,19,82783508	14,20001793	969,8148071	17,02298355	15,06345558		-0,40290001	745,7999878	712,7
33	31,19,28964233	14,30001831	967,5925903	17,02298355	15,04903412		-0,40290001	744,7999878	
34	32,19,15509415	14,50001907	969,3518188	16,91353607	15,05624485		-0,40290001	747,0999756	
35	33,19,46903992	14,60001945	970,1851807	17,02298355	15,05624485		-0,40290001	746,9000244	713,5
36	34,19,33449173	14,00001717	970,1851807	16,5122261	15,13556385		-0,40290001	744,7999878	711,5
37	35,18,61689758	14,20001793	968,6111084	16,402771	15,12114239		-0,40290001	745	712,0
38	36,18,3926506	14,40001869	969,2592407	16,402771	15,12114239		-0,40290001	746	

Fig. 5

3. Fases de creación del modelo.

Fase 1. Tras la obtención del dataset, el primer paso para la creación de modelos es definir la variable dependiente. Ésta será nuestra pregunta de negocio, el problema al que nos enfrentamos y que ni nuestros conocimientos técnicos, ni nuestra experiencia pueden dar respuesta. En el caso tratado en este TFM es: **“Qué variables y cómo, determinan la Potencia Máxima que la máquina es capaz de generar?”**. En nuestro caso será la variable “Active power”, que representa la potencia activa generada por la máquina.

Fase 2. Proceso ETL, hay que extraer los datos desde el dataset, transformarlos para que sean matemáticamente consistentes, es decir eliminar valores nulos, perdido, heterogeneidades, etc., y por último cargarlos en el sistema de tal manera que puedan ser procesados en nuestro entorno de ejecución de código (Jupyter Notebook).

Fase 3. Definir las variables independientes o descriptoras, de nuestro problema, extrayendo y tratando los datos para que sean matemáticamente consistentes. Así en nuestro caso, trataremos variables numéricas continuas y tomando como criterio únicamente aquellos datos en los que la máquina funciona a máxima potencia, ya que en ocasiones la potencia puede ser limitada por razones técnicas del mercado eléctrico.

Fase 4. Selección del modelo. Como ha sido comentado anteriormente se elige un modelo de regresión múltiple, utilizando la librería de Python scikit-learn

En la Figura 16, mostramos un cuadro de las diferentes fases en el desarrollo de nuestro trabajo:



Fig 6.

4. Descriptiva del proceso de Machine Learning.

Nuestro objetivo es entender qué variables afectan a la máxima capacidad de producción de la máquina y cómo estas variables afectan a dicha capacidad. Para ello, y según los pasos descritos en el apartado anterior, presentamos en el ANEXO III el contenido de un Notebook de Jupyter y el código de los scripts que contienen las funciones. en el que se desarrolla el código Python, que nos dará la siguiente información:

- i. Tamaño del dataset: 168 variables conteniendo 3.814 tuplas de datos. Se trata de datos horarios tomados en la primera mitad del año.
- ii. En el fichero “var_remove_TBM.csv”, realizamos un descarte previo de características, bien porque nos conste que no tengan influencia en la variable dependiente, o sean cálculos intermedios propios de la determinación de nuestra variable dependiente, y muy probablemente introduzcan ruido al modelo. En cualquier caso, una vez entrenemos nuestro modelo con las variables preseleccionadas, y atendiendo a la calidad de dicho entrenamiento, veremos si es o no necesario, modificar las variables preseleccionadas.
- iii. Estadísticos del ajuste, MSE, MAE, MedAE.
- iv. Con respecto al coeficiente de correlación r^2 , lo hemos calculado por un lado para un 75 % de la población de datos, y que corresponde al modelo de entrenamiento, y por otro al 25 % de la población de datos, que corresponde con un test, de esta manera podremos ver si existe sobreajuste. En el caso de que r^2 train y r^2 test presenten valores similares, podremos afirmar que el modelo no aparenta estar sobreajustado.
- v. Dos ficheros (“file_var.csv” y “file_model.csv”), que contienen respectivamente el nombre de todas las variables que han participado en el entrenamiento y los coeficientes y bias del modelo lineal de ajuste. Éste

servirá más adelante para verificar la evolución del sistema respecto a esta situación modelizada.

A	B	C	D	E	F	G
1	Variable	Coefficiente				
2	0 bias	-10748,399				
3	1 Inlet air filter differential pressure	-6,97644				
4	2 Presión diferencial primera etapa filtro aire	9,87268797				
5	3 Presión diferencial segunda etapa filtro aire	7,67271695				
6	4 Presión diferencial total filtro aire	-2,1714052				
7	5 Compressor outlet air pressure (PCD)	88,7001441				
8	6 BAM monitoring Torch pressure ***not wired***	-1,1E-11				
9	7 BAM monitoring : Rumble	0,8009162				
10	8 BAM monitoring: Oscillation	0,00515982				
11	9 Gas fuel pressure	-27,144672				
12	10 Main manifold gas fuel pressure	471,72841				
13	11 Outside ambient pressure (P0)	3,13491143				
14	12 Pilot manifold gas fuel pressure	104,817015				
15	13 Lube Oil pressure	-39,481965				
16	14 Vapour Tank Oil Pressure	1,99707962				
17	15 Turbine exhaust pressure	-14,352887				
18	16 Container temperature	-3,1642926				
19	17 Turbine temperature (T5) #1	0,98169267				
20	18 Turbine temperature (T5) #10	3,81506072				
21	19 Turbine temperature (T5) #11	4,97324414				
22	20 Turbine temperature (T5) #12	6,65171246				
23	21 Turbine temperature (T5) #2	0,79916444				
24	22 Turbine temperature (T5) #3	4,84871225				
25	23 Turbine temperature (T5) #4	2,55111436				
26	24 Turbine temperature (T5) #5	1,88119931				
27	25 Turbine temperature (T5) #6	7,77037884				
28	26 Turbine temperature (T5) #7	7,3362316				
29	27 Turbine temperature (T5) #8	5,48188297				
30	28 Turbine temperature (T5) #9	-4,1581338				
31	29 Lube Oil temperature	1,58197763				
32	30 Compressor inlet air temperature (T1)	-44,01951				
33	31 Turbine exhaust temperature (T7 #1)	24,3757577				
34	32 Turbine exhaust temperature (T7 #2)	8,56550859				
35	33 Turbine exhaust temperature (T7 #3)	20,6650734				
36	34 Temperatura entrada aire antes el enfriador	2,2225468				
37	35 Temperatura entrada aire después el enfriador	11,2786823				
38	36 Turbine exhaust temperature (T7)	-37,784948				
39	37 Compensated T5 average	-14,071409				
40	38 T7 average used for control	-32,665795				
41	39 Turbine Hours	-0,0809873				

Fig. 7

En la figura 7 presentamos la tabla con las cuarenta variables preseleccionadas, y su coeficiente W en el modelo, así como el bias del modelo.

```
*****
Tamaño DataSet: 3814 filas de datos y 168 variables registradas
*****
=====
R2 en training: 0.9906
R2 en test: 0.9910
Sobre Ajuste--->>> 0.0005
Error cuadrático medio: 2240.09
Error absoluto medio: 37.06
Mediana del error Absoluto: 30.71
-----
Iteraciones realizadas >>> 1
```

Fig. 8

En la figura 8 vemos el resultado de la regresión lineal. EL $r^2_{train} = 0.99$ y el $r^2_{test} = 0.99$. Estos coeficientes tienen un magnífico valor muy cercano a 1, con lo que podemos afirmar que el modelo reproduce de manera muy efectiva los datos del conjunto del entrenamiento y que aparentemente el modelo no está sobreajustado.

En el ANEXO III, se muestra el código de las funciones creadas en el Script “*ModReg_TG.py*”, para la ejecución de este modelo de entrenamiento.

5. Ingeniería de Características. Selección de variables.

Llegado a este punto hemos obtenido un modelo con 40 variables, con un $r^2 = 0.99$ y sin sobreajuste, el siguiente paso sería estudiar si todas las variables deberían entrar en el modelo de regresión, y en caso negativo, debemos determinar cuáles descartar.

A tal efecto vamos a probar dos métodos y compararlos entre sí,

- i. Qué variables selecciona cada método
- ii. Qué coeficiente de correlación se obtienen con cada método

El primer método será SelectKBest, incluido dentro de la librería scikit-learn de Python. En el segundo modelo se hará uso de un algoritmo Forward de adición de características, que irá seleccionando variables, según vayan mejorando el r^2 del modelo.

5.1. Selección de características con SelectKBest

Este método permite seleccionar las K mejores características, que mejor resultado den en la función *f_regression*, contenida también en la librería scikit-learn de Python.

La función *f_regression* nos devuelve un *f_score*, basado en la correlación existente entre cada variable y el objetivo mediante la siguiente expresión:

$$((X[:, i] - \text{mean}(X[:, i])) * (y - \text{mean}_y)) / (\text{std}(X[:, i]) * \text{std}(y))$$

En el ANEXO IV, mostramos el Notebook con el código de este método de selección, y de las funciones contenida en el Script *f_selection.py*, empleadas en el mismo.

Los resultados obtenidos con $k = 20$ son un data frame con las variables ordenadas por mejor *f_score*, y los parámetros de la regresión del nuevo modelo.

		Variables	f_score
9	Turbine temperature (T5) #9	54578.478664	
4	Main manifold gas fuel pressure	38820.930233	
11	Compressor Inlet air temperature (T1)	28972.732844	
15	Temperatura entrada aire antes el enfriador	11410.813868	
2	Compressor outlet air pressure (PCD)	2967.174190	
1	Presion diferencial segunda etapa filtro aire	1236.415061	
8	Turbine temperature (T5) #2	757.058309	
10	Lube Oil temperature	551.423696	
3	Gas fuel pressure	417.948046	
7	Container temperature	346.447936	
6	Pilot manifold gas fuel pressure	317.594062	
0	Presion diferencial primera etapa filtro aire	63.395638	
19	Turbine Hours	49.145121	
16	Temperatura entrada aire despues el enfriador	43.983561	
18	T7 average used for control	5.524941	
12	Turbine exhaust temperature (T7 #1)	5.152789	
14	Turbine exhaust temperature (T7 #3)	4.176453	
17	Turbine exhaust temperature (T7)	2.150355	
13	Turbine exhaust temperature (T7 #2)	1.104342	
5	Outside ambient pressure (P0)	NaN	

Fig. 9

```
=====
R2 en training: 0.9791
R2 en test: 0.9779
Sobre Ajuste--->>> 0.0013
Error cuadratico medio: 5112.91
Error absoluto medio: 55.08
Mediana del error Absoluto: 45.10
```

Fig. 10

Volviendo a lanzar el modelo de entrenamiento automático por regresión lineal, obtenemos un $r^2 = 0.9787$, sin aparente sobre ajuste.

5.2. Selección de características con método Forward de mejor r^2 .

Este método pertenece a la familia de métodos Stepwise para la selección de características de forma iterativa. En este caso seleccionamos la adición de características, lo que implica comenzar sin características en el modelo, poner a prueba la adición de cada característica y se prueba a agregar una a una el resto de características. Una vez hecho esto se comprueba la calidad del modelo, utilizando un criterio para realizar la comparación, en este caso usaremos el coeficiente r^2 .

En la tabla de la figura 11 obtenemos el resultado de este método:

	Variables	r^2
0	Main manifold gas fuel pressure	0.934715
1	Compressor Inlet air temperature (T1)	0.958987
2	Turbine temperature (T5) #12	0.973896
3	Presion diferencial primera etapa filtro aire	0.976168
4	Turbine Hours	0.979449
5	Inlet air filter differential pressure	0.983690
6	Compensated T5 average	0.985311
7	Pilot manifold gas fuel pressure	0.986740
8	Turbine temperature (T5) #9	0.987655
9	Turbine temperature (T5) #7	0.988450
10	Turbine exhaust temperature (T7 #2)	0.988994
11	BAM monitoring: Rumble	0.989348
12	Outside ambient pressure (P0)	0.989675
13	Container temperature	0.989850
14	Turbine temperature (T5) #2	0.989966
15	Turbine temperature (T5) #6	0.990129
16	Turbine temperature (T5) #1	0.990242
17	Compressor outlet air pressure (PCD)	0.990345
18	Turbine temperature (T5) #5	0.990448
19	Presion diferencial total filtro aire	0.990528

Fig. 11

En el gráfico de la figura 12, observamos como a medida que se van agregando variables al modelo, mejoramos el coeficiente de correlación r2

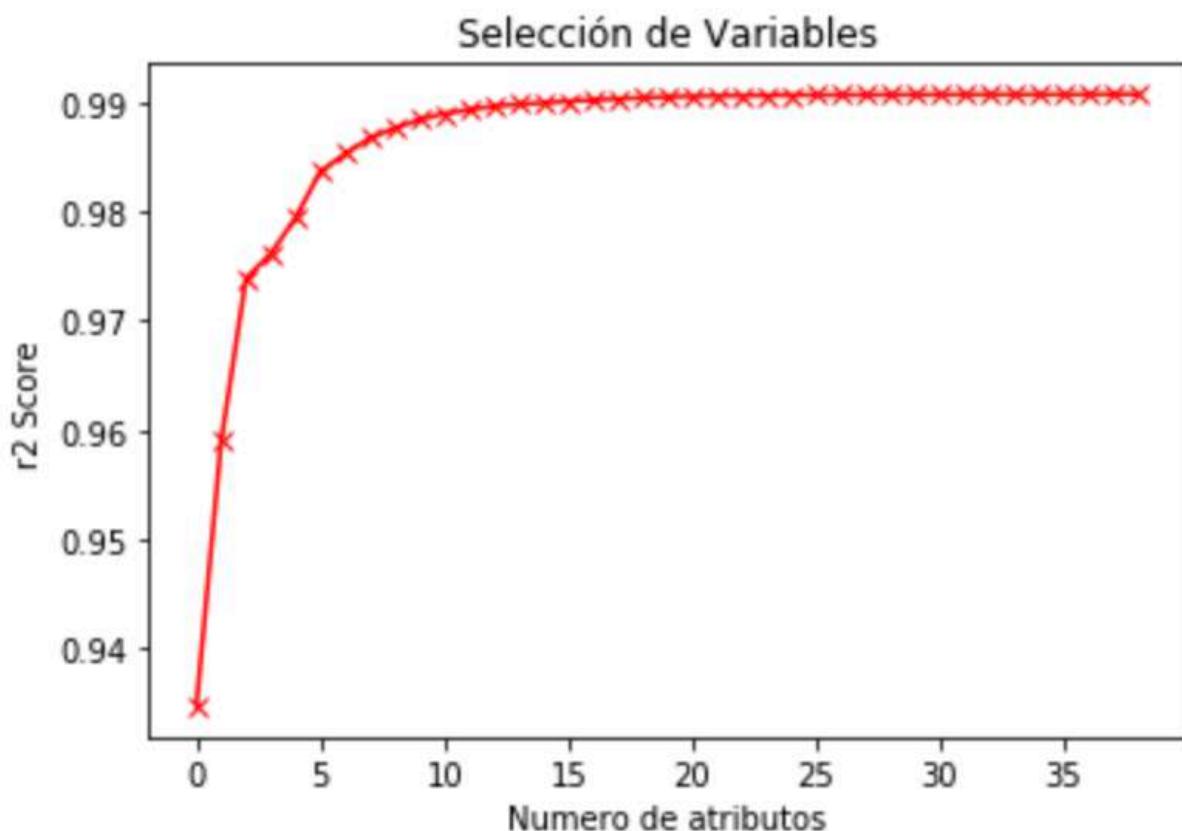


Fig. 12

En el ANEXO V, mostramos el Notebook y el código de la función empleada para el desarrollo de este algoritmo:

5.3. Conclusiones: comparativa de ambos métodos.

Parece claro a la vista de los resultados, que el método Forward se comporta mejor, ya que el r2 es de 0.99 con 20 variables, mientras que el de SelectKbest es de 0.97. De hecho con el método Forward somos capaces obtener un r2 = 0.97 con 5 variables.

	Variables	r2
0	Main manifold gas fuel pressure	0.934715
1	Compressor Inlet air temperature (T1)	0.958987
2	Turbine temperature (T5) #12	0.973896
3	Presion diferencial primera etapa filtro aire	0.976168
4	Turbine Hours	0.979449

Fig. 13

El resultado desde el punto de vista técnico, es muy interesante, ya que nos dice que somos capaces de predecir la potencia de la máquina según, la presión de gas, la Temperatura a la que alimentamos el aire de combustión, la temperatura de la combustión, tres variables críticas en el ciclo termodinámico de Brayton. La pérdida de carga en el filtro de aire afecta a la cantidad de masa que procesa la máquina lo cual hace evidente su importancia, y por último las hora de funcionamiento de la máquina, lo cual indica el hecho de la pérdida de prestaciones la máquina con el paso del tiempo, de hecho una turbina de gas debe ser “cambiada” cada 30.000 horas funcionamiento. Todo ello nos induce a pensar que el proceso de Machine Learning va “buen camino”.

Hemos obtenido un modelo con un $r^2=0.9908$ y sin sobre ajuste con 20 variables, basadas en el algoritmo Forward del mejor r^2 acumulado y el fichero “file_model.csv” con el modelo matemático, que nos permitirá dadas unas condiciones en el proceso predecir el comportamiento de la máquina, así como, y lo más importante, detectar desvíos de este modelo y causas, lo cual nos podrá llevar a plantear soluciones y evitar pérdidas de producción. Igualmente obtenemos el fichero “var_remove_DL_Xr2.csv”, con la lista de variables a eliminar para los siguientes estudios y análisis de estado de máquina, se trata de las variables no seleccionadas por el modelo de entrenamiento, y que no entrarán en los modelo de simulación que explicamos en el próximo apartado.

En la Figura 14, se muestra una tabla con el modelo resultante del proceso de entrenamiento anterior, y que emplearemos para evaluar la tendencia de las prestaciones de la máquina, y que llamaremos modelo técnico.

	A	B	C	D
1	Variable		Coeficiente	
2	0 bias		-14186,774	
3	1 Main manifold gas fuel pressure		464,268752	
4	2 Compressor Inlet air temperature (T1)		-31,447315	
5	3 Turbine temperature (T5) #12		2,49671848	
6	4 Presion diferencial primera etapa filtro aire		9,06403942	
7	5 Turbine Hours		-0,0813167	
8	6 Inlet air filter differential pressure		-7,1047021	
9	7 Compensated T5 average		38,4070688	
10	8 Pilot manifold gas fuel pressure		114,65769	
11	9 Turbine temperature (T5) #9		-8,1672355	
12	10 Turbine temperature (T5) #7		3,08973624	
13	11 Turbine exhaust temperature (T7 #2)		-16,3739	
14	12 BAM monitoring: Rumble		0,89847329	
15	13 Outside ambient pressure (P0)		2,82629666	
16	14 Container temperature		-1,850088	
17	15 Turbine temperature (T5) #2		-3,1556989	
18	16 Turbine temperature (T5) #6		3,95852602	
19	17 Turbine temperature (T5) #1		-3,2296978	
20	18 Compressor outlet air pressure (PCD)		103,884825	
21	19 Turbine temperature (T5) #5		-3,8993132	
22	20 Presion diferencial total filtro aire		-0,9707645	

Fig. 14

6. Análisis de Sensibilidad para la detección de patologías.

Una vez que hemos entrenado un modelo con los datos disponibles, hemos comprobado que no sólo es razonable, sino que es excelente, y hemos podido seleccionar las variables, y que dichas variables, son efectivamente críticas para nuestro proceso. Lo siguiente es desarrollar algoritmos que nos permitan detectar cambio en las condiciones de funcionamiento y que puedan indicar alguna falla, o pérdida de prestaciones, que nos lleve en última instancia, a una pérdida económica, y de esta manera actuar de manera predictiva para corregirla.

En este apartado simularemos una situación de chequeo de prestaciones de la máquina, para ello, partiremos de un modelo, que es razonable, sin aparente sobreajuste y con variables seleccionadas, siguiendo las siguientes pautas:

- i. Como fuente de datos tomaremos los datos de un año (recordar que el entrenamiento se ha realizado con los datos de la primera mitad).
- ii. Transformaremos esos datos en pequeños grupos de data sets con 48 tuplas de datos para que sean matemáticamente consistentes para un modelo de regresión lineal. De esta manera simulamos un chequeo cada 48 horas. En el algoritmo desarrollado y que se muestra en el ANEXO VI.
- iii. Cada dataset de 48 tuplas de datos, que a partir de ahora llamaremos “datos de diario”, será sometido a los criterios de concordancia con las condiciones de nuestra pregunta de negocio y a las variables seleccionadas en los pasos anteriores, posteriormente realizaremos una regresión lineal, obteniendo los parámetros del ajuste.
- iv. A continuación realizaremos un chequeo o test de las prestaciones del modelo y de la máquina, comparando los datos reales registrados, con las predicciones del modelo, tanto para el modelo obtenido con los datos de diario y la predicción según nuestro modelo base. El error entre la predicción de nuestro modelo base y el del modelo de diario será el desvío técnico. Damos por supuesto que el error entre la predicción del modelo de diario y el valor real debe ser despreciable, dada la calidad de la regresión del modelo base obtenido en el apartado anterior
- v. Almacenamiento y visualización de resultados. El código del ANEXO VI, dará como resultado para cada análisis, el cuadro mostrado en la Figura 15, con la siguiente estructura de datos:

Fecha Análisis	Nombre Variable	R2	Valor Medio	% desv Std	Coef Variable (W)
dd/mm/aa	Bias	-	-	Std Dev	-
dd/mm/aa	Variable 1				
dd/mm/aa
dd/mm/aa	Variable n				
dd/mm/aa	Desv Tecnico %	-	%	-	-
dd/mm/aa	R2 Train	-	0.xx	-	-
dd/mm/aa	R2 Test	-	0.xx	-	-

Fig. 15

Esta tabla será almacenada en un fichero .csv, cuyo nombre empezará por *fa_* y le seguirá la fecha a la que corresponden los datos analizados: “*fa_dd-mm-aa.csv*”.

Igualmente, este cuadro de datos pasará a formato .json y serán almacenados en una base de datos no relacional, orientada a documentos, como es MongoDB. Y para terminar usaremos Tableau como herramienta de visualización de datos, para el análisis inteligente de negocio y toma de decisiones.

6.1. Obtención de los datasets de la simulación de control diario.

En el ANEXO VI, se muestra el código para obtener los ficheros que servirán para la simulación. En nuestro caso de estudio habremos obtenido 211 ficheros tipo csv para la simulación, almacenamiento y visualización de datos.

 data_train_196.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_197.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_198.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_199.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_200.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_201.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_202.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_203.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_204.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_205.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_206.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_207.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_208.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_209.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_210.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train_211.csv	13 mar 2019 22:21	54 KB	Comm...et (.csv)
 data_train.csv	12 mar 2019 10:38	5,3 MB	Comm...et (.csv)

Fig. 16

6.2. Entrenamiento y chequeo del modelo.

En el ANEXO VII se muestra un Notebook con el código para la ejecución de nuestro algoritmo de simulación, en el que:

- i. Obtenidos los datasets de diario, realizaremos una regresión para obtener un modelo entrenado, pero esta vez no con las 169 variables del dataset inicial, si no con las 20 variables seleccionados por el método Fordward de mejor r2. Mostrando los parámetros propios de la regresión.

```
-----
R2 en training: 0.9964
R2 en test: 0.9888
Sobre Ajuste-->>> 0.0077
Error cuadratico medio: 183.90
Error absoluto medio: 11.14
Mediana del error Absoluto: 9.54
-----
Iteraciones realizadas >>> 1
```

Fig. 16

- ii. A continuación realizaremos un test para verificar las predicciones del modelo obtenido en el entrenamiento con los datos de diario. Posteriormente comprobaremos la predicción del modelo obtenido inicialmente con la actual predicción con los datos de diario. En la Figura 18, vemos como para los datos de un día en concreto, se obtiene un modelo apropiado $r^2 = 0.99$, sin sobreajuste, y además como tanto el

modelo de datos diarios, como el modelo inicial, tienen una similar predicción. Con un desvío técnico del 0,1 %.

```
#####
Pot Calculada según Modelo = 14098.807

Potencia Real alcanzada: 14097.75
Desvio Técnico: 0.114 %

Potencia Máxima esperada: 14081.747
#####
```

Fig. 17

En el apartado 7, analizaremos los resultados obtenidos de los 211 ficheros con los datasets de diario, y es aquí donde podremos sacar conclusiones sobre el estado del funcionamiento de la máquina.

- iii. Por último sólo queda almacenar estos datos en un datawarehouse y usar herramientas de visualización, que nos permita aplicar inteligencia de negocio.

6.3. Almacenamiento y visualización.

En el Anexo VIII se muestra el código desarrollado en la función para pasar los resultados en formato json y almacenarlos en MongoDB.

6.3.1. Instalación de MongoDB

- i. Instalaremos la versión de MongoDB 4.0.6, descargando el fichero desde la dirección:

<https://www.mongodb.com/download-center/enterprise?jmp=docs>

- ii. Por línea de comandos realizamos la extracción con del fichero descargado

```
tar -zxvf mongodb-osx-ssl-x86_64-enterprise-4.0.6.tgz
```

- iii. Creamos el directorio de datos, en cual los procesos de mongod serán almacenados. Por defecto los procesos de mongod usa el directorio data/db. Por línea de comandos introduciremos la instrucción para crear el directorio:

```
mkdir -p /data/db
```

- iv. Para lanzar mongod desde la terminal en primer plano:

```
mongod --config /usr/local/etc/mongod.conf
```

Por último solo queda lanzar el comando mongo y comprobar que se conecta a la Shell de comando mongo:

```

MacBook-Pro-de-Jesus:bin jpgmacbookpro$ mongo
MongoDB shell version v4.0.6
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("7596794c-b8db-4e85-9fa4-a2616c2dc121") }
MongoDB server version: 4.0.6
Server has startup warnings:
2019-03-12T12:20:21.383+0100 I CONTROL  [initandlisten]
2019-03-12T12:20:21.383+0100 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-03-12T12:20:21.383+0100 I CONTROL  [initandlisten] ** Read and write access to the database and configuration is unrestricted.
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** WARNING: You are running this process as the root user, which is not recommended.
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten]
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** Remote systems will be unable to connect to this server.
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning.
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten]
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten]
2019-03-12T12:20:21.384+0100 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> ■

```

Fig. 18

Para una más fácil gestión emplearemos Robo 3T, que es una herramienta que permite gestionar gráficamente bases de datos MongoDB. Para ello nos descargaremos el fichero de instalación en la dirección: <https://robomongo.org> y seguiremos las instrucciones del asistente de instalación.

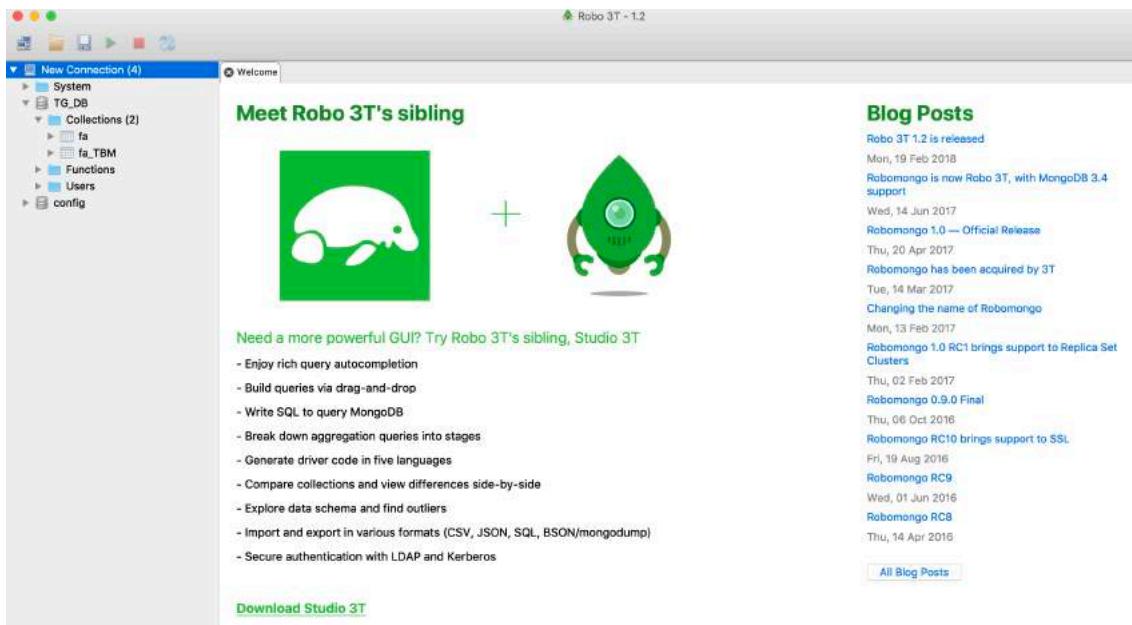


Fig. 19

6.3.2. Instalación Tableau.

Instalaremos la versión Desktop de Tableau gratuita para estudiantes y que será suficiente para los objetivos de este TFM. Para ello nos registraremos en la web y en un mail recibiremos las keys para acceder a la aplicación.

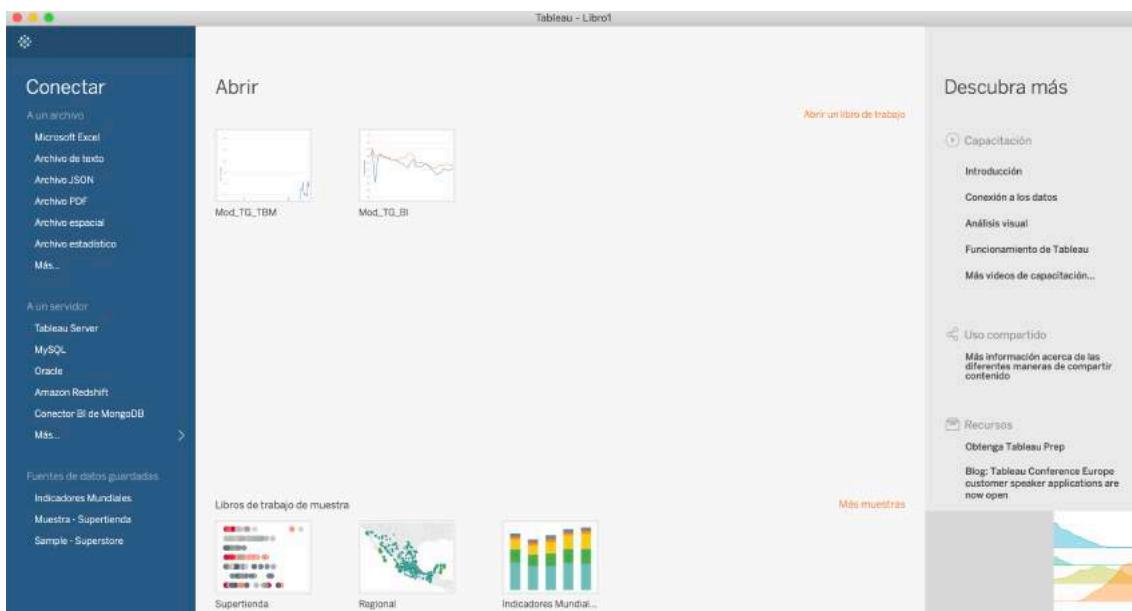


Fig. 20

6.3.3. Conexión de MongoDB con Tableau.

Una vez instaladas las herramientas de almacenamiento y visualización el siguiente paso es crear un esquema de conexión, para que toda aquella información que sea cargada en mongo, pueda ser reproducida en Tableau para su análisis. Para ello seguiremos los siguientes pasos:

- i. Descarga del conector para BI de MongoDB en el fichero: <https://www.mongodb.com/download-center/bi-connector>, una vez se descargado lo extraemos mediante el siguiente comando.

```
tar -xvzf mongodb-bi-osx-x86_64-v4.0-latest.tgz
```

- ii. Instalamos el programa en el PATH:

```
sudo install -m755 bin/mongo* /usr/local/bin/
```

- iii. Preparamos el esquema de conexión a la base de datos creada en MongoDB. En nuestro caso hemos creamos una base de datos llamada: TG_DB. Para ello tecleamos por línea de comandos la siguiente instrucción:

```
mongodrql --host localhost -d TG_DB -o schema2.drql
```

- iv. Para lanzar la conexión sólo hemos de introducir por línea de comandos:

```
mongosqld --schema schema2.drql --mongo-uri localhost
```

En este punto ya estaríamos en disposición para visualizar en Tableau, los datos generados en Python y almacenado en Mongo.

The screenshot shows the Tableau Data Source configuration window titled 'Tableau - Mod_TG_TBM'. It displays a connection to 'fa_TBM (TG_DB)' from 'localhost' using the 'MongoDB BI Connector'. The 'Base de datos' is set to 'TG_DB'. Under 'Tabla', there are two tables listed: 'fa' and 'fa_TBM'. A preview of the 'fa_TBM' table is shown with columns: 'Año', 'Id', 'Coef Model', 'Año', 'Fecha', 'Fecha_F', 'R2 Score', 'Año', 'Std Dev', 'Valor Medio', and 'Año', 'Variable'. The 'Actualizar ahora' button is visible at the bottom.

Fig. 21

7. Análisis visual en tiempo real con Tableau.

En este apartado realizaremos el análisis de los datos obtenidos en los procesos de Machine Learning descritos en los apartados anteriores. Tras ejecutar el Notebook del Anexo VII, en el que se obtiene un cuadro de datos con los indicadores descritos en la tabla de la Figura 15, éstos son almacenado en Mongo.

Así hemos cargado 3912 documentos en la base de dato TG_DB dentro de la colección que hemos llamado fa_TBM,

```
[> use TG_DB
switched to db TG_DB
[> db.fa_TBM.count({})
3912
> ]
```

Fig. 22

con la siguiente estructura de información: Fecha; Nombre de la variable seleccionada; r2_score, coeficiente de correlación de esa variable con la variable objetivo; Valor promedio de esa variable en el intervalo de tiempo en el que se han recogido los datos; Std Dev, % de la desviación estándar de los valores tomados por dicha variable en el intervalo de tiempo tratado; Coef Model, coeficiente de la variable en el modelo.

```
/* 1 */
{
    "_id" : ObjectId("5c899e0b244dce1fd0b6ea1a"),
    "Fecha" : "31/10/18",
    "Variable" : "Turbine temperature (T5) #5",
    "r2_score" : 0.269409823,
    "Valor Medio" : 716.8979166667,
    "Std Dev" : 0.0119419428,
    "Coef Model" : 9.5433602824
}

/* 2 */
{
    "_id" : ObjectId("5c899e0b244dce1fd0b6ea1b"),
    "Fecha" : "31/10/18",
    "Variable" : "Turbine temperature (T5) #6",
    "r2_score" : 0.0913229878,
    "Valor Medio" : 720.1625,
    "Std Dev" : 0.0054655071,
    "Coef Model" : -21.4049548243
}
```

Fig. 23

7.1. Visualización r2 train y r2 test.

Para la visualización hemos creado en Tableau un libro llamado “Mod_TG_TBM.twb”, el cual emplea como fuente de datos el documento de MongoDB, anteriormente mostrado.

La primera visualización en Tableau que realizaremos será comprobar que todos los r2 Test y r2 train son adecuados, es decir están en el orden del 0.95 – 1, y que no exista sobre ajuste, aquellos días en lo que el entrenamiento no haya sido adecuado, será eliminado.

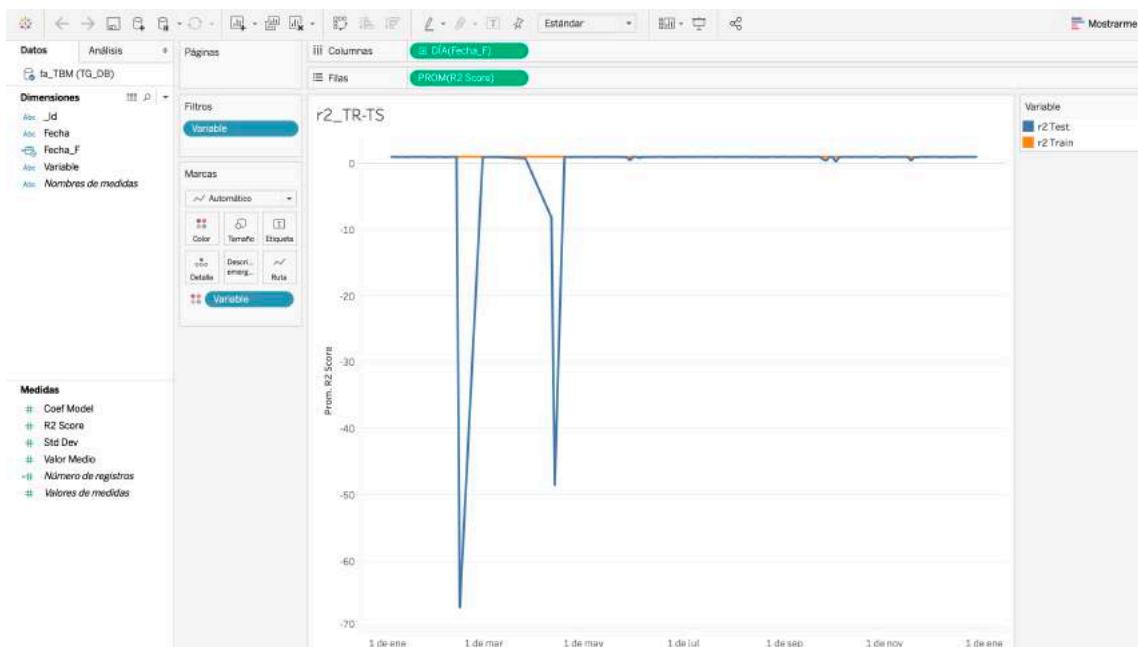


Fig. 24

Aquellos días en el que los modelos han presentado sobre ajuste o una mal r2 serán eliminados, mediante código en la consola de Mongo:

```
[> db.fa_TBM.remove({"Fecha" : "11/04/18"})
WriteResult({ "nRemoved" : 24 })
```

Tras el proceso de depuración de datos no entrenados convenientemente, obtenemos:

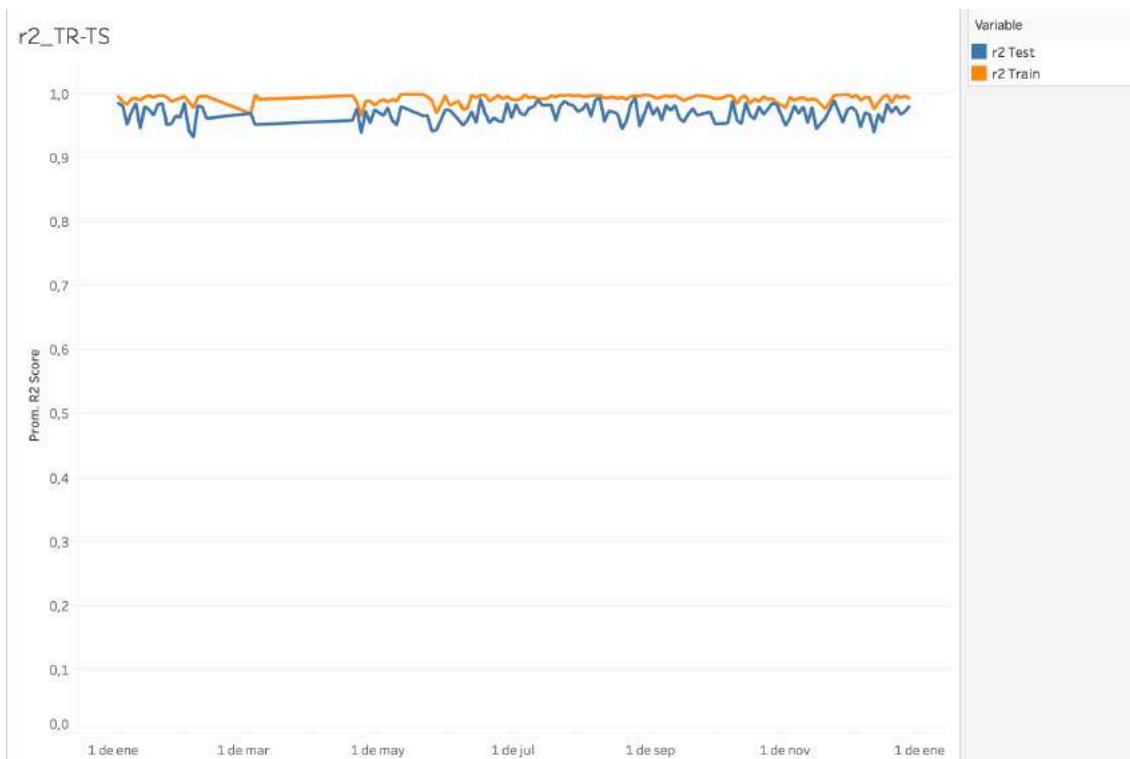


Fig. 25

Ahora todos los datos disponibles, provienen de modelos cuyo r2 train está en el entorno del 0.99 y podemos afirmar que en ninguno de ellos existe sobreajuste.

Notar que durante los meses de marzo a mayo todos los datos del análisis han sido descartados debido a que debido a exigencias del mercado eléctrico la turbina trabajó en un modo limitado de regulación, en este caso no tiene sentido analizar los datos, ya que no se trabaja en condiciones de máxima potencia.

7.2. Visualización de desvíos en la simulación de control diario

Esta visualización nos permitirá realizar un análisis de cómo ha evolucionado las prestaciones de la máquina a lo largo del año. La línea amarilla representa el Desvío Técnico, que representa el error entre la predicción del modelo con los datos de diario con la predicción del modelo empleado en el entrenamiento inicial.

En la Figura 26, vemos esta tendencia, además representamos la tendencia de nuestra variable objetivo (datos reales).



Fig. 26

Podemos observar como la primera mitad del año, el modelo base y el modelo de los datos de día coinciden (desvío técnico muy próximo a cero).



Fig. 27

Pero vemos que se producen tres eventos, que provocan que el modelo técnico se desvíe de manera positiva, casi 1% a finales de julio y otro en un 2,5% a finales de octubre, previo a una caída de dichas prestaciones tras un mantenimiento programado a finales de septiembre. Estos eventos se deben a mejoras técnicas que llevaron a mejorar las prestaciones de la máquina, gracias al uso de la ciencia de datos y que explicaremos brevemente cómo se detectaron y qué se hizo. El resultado ya lo conocemos una mejora promedio de 2,5 % en las prestaciones de la máquina. Lo que supone un incremento en el margen de la operación del sistema de unos 150.000 €.

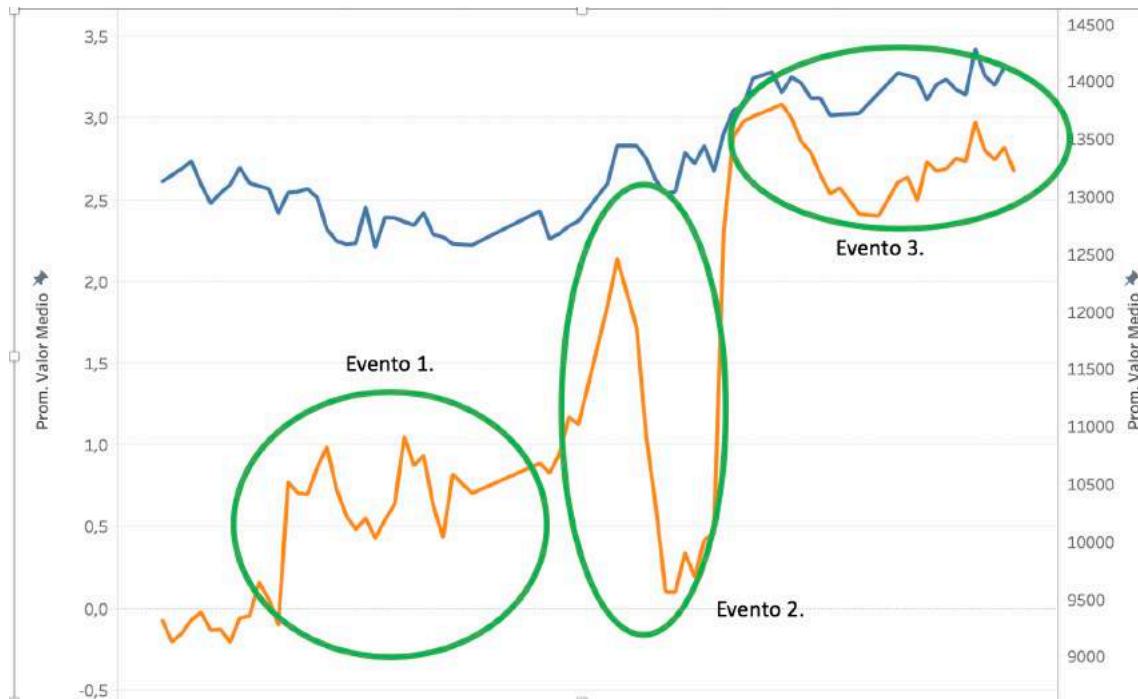


Fig. 28

7.3. Visualización de Características frente a la variable objetivo.

Una de las herramientas clásicas de análisis del estado de procesos industriales, es la visualización de nuestra variable objetivo con otras características, de esta manera ante eventos o comportamientos anómalos, los procesistas visualizan la variable objetiva junto a otras para inferir relación entre el comportamiento anómalo de la variable objetivo y el resto de característica. Por ello no podíamos dejar de crear este dashboard en nuestro libro Tableau, donde podremos ver la tendencia de las diferentes características. Así por ejemplo:

- i. Visualizamos, en la Figura 29, la Temperatura de combustión foco caliente y la temperatura de escape o foco frío del ciclo y su relación con la potencia.

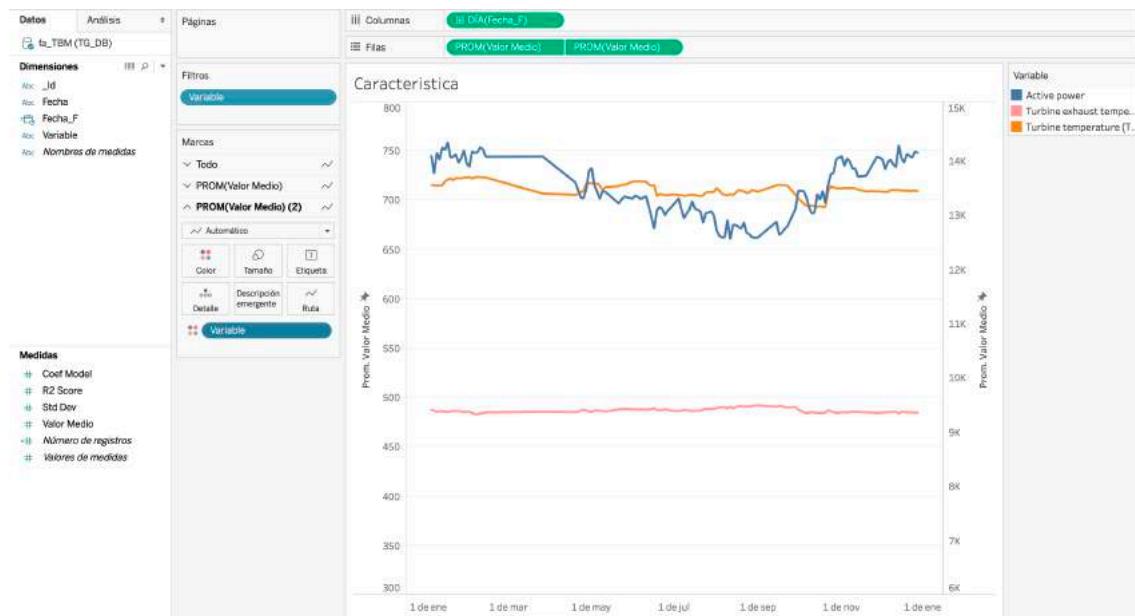


Fig. 29

- ii. Visualizamos en la Figura 30, la presión del aire combustión obtenido en compresor y la presión de gas obtenido tras la regulación de pre combustión y la principal de combustión. Vemos un total paralelismo entre nuestra variable objetivo y la presión de aire y gas, así como entre la de aire y gas.

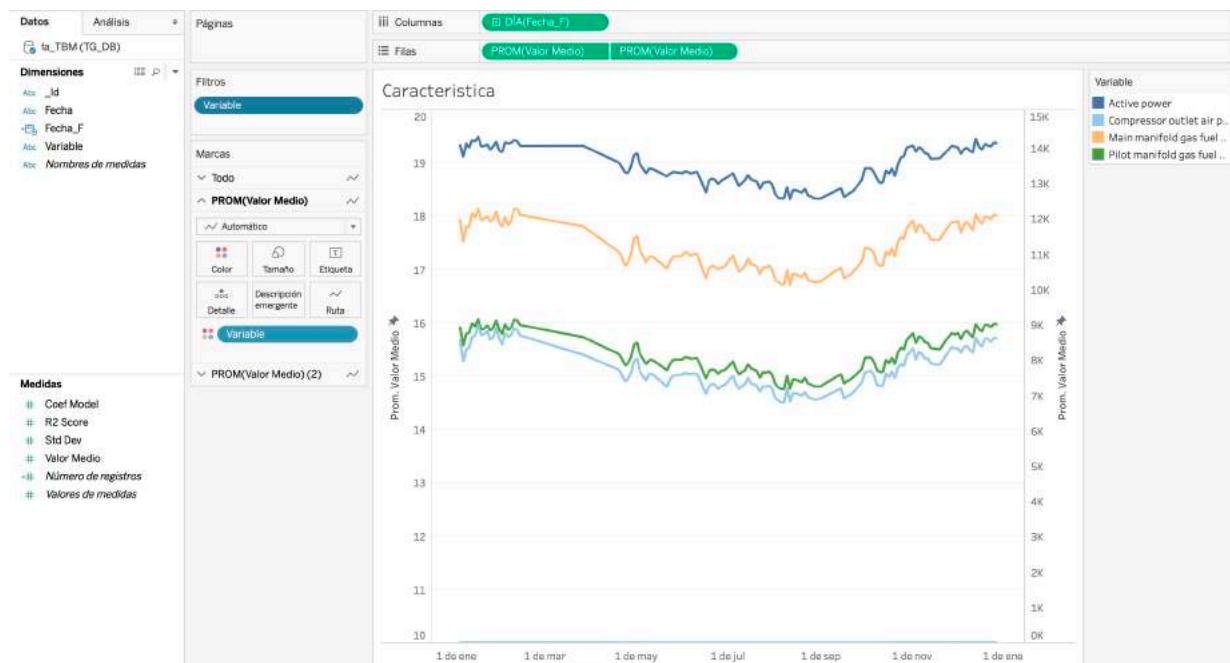


Fig. 30

- iii. Visualizamos en la Fig. 30.1, como la Presión de aire varia con la temperatura de entrada a la máquina. Vemos como hay un relación entre la presión que conseguimos con respecto a la Temperatura de entrada a la máquina, lo cual está acorde a nuestros conocimientos y experiencia.



Fig. 30.1

- iv. Visualizamos la temperatura de la zona de turbina y la temperatura del aire a la entrada al compresor.

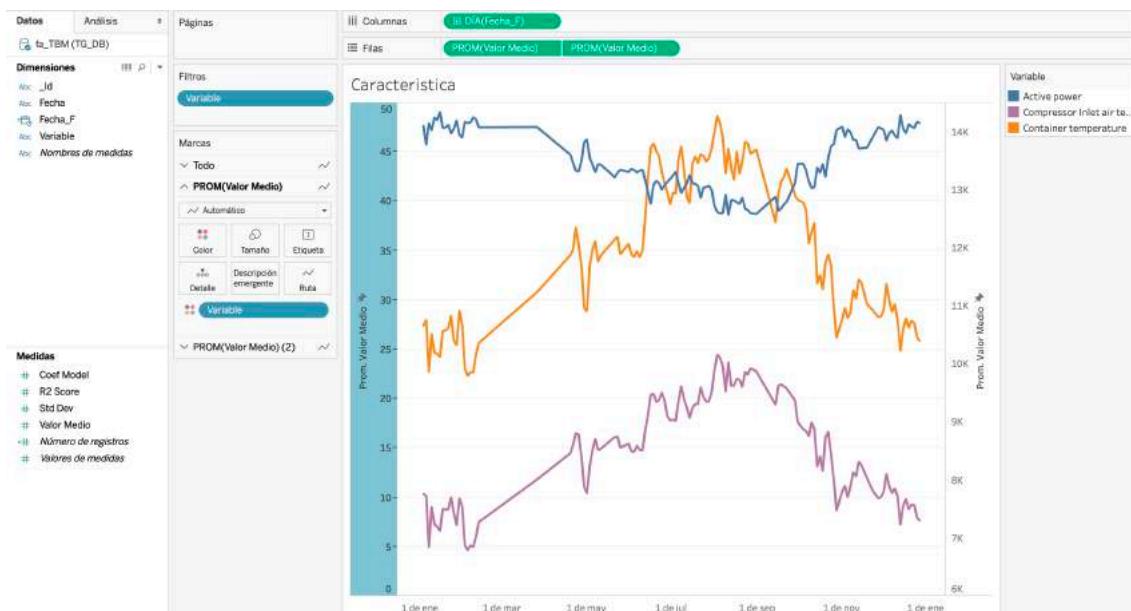


Fig. 31

v. Visualizamos el nivel de vibraciones de la llama

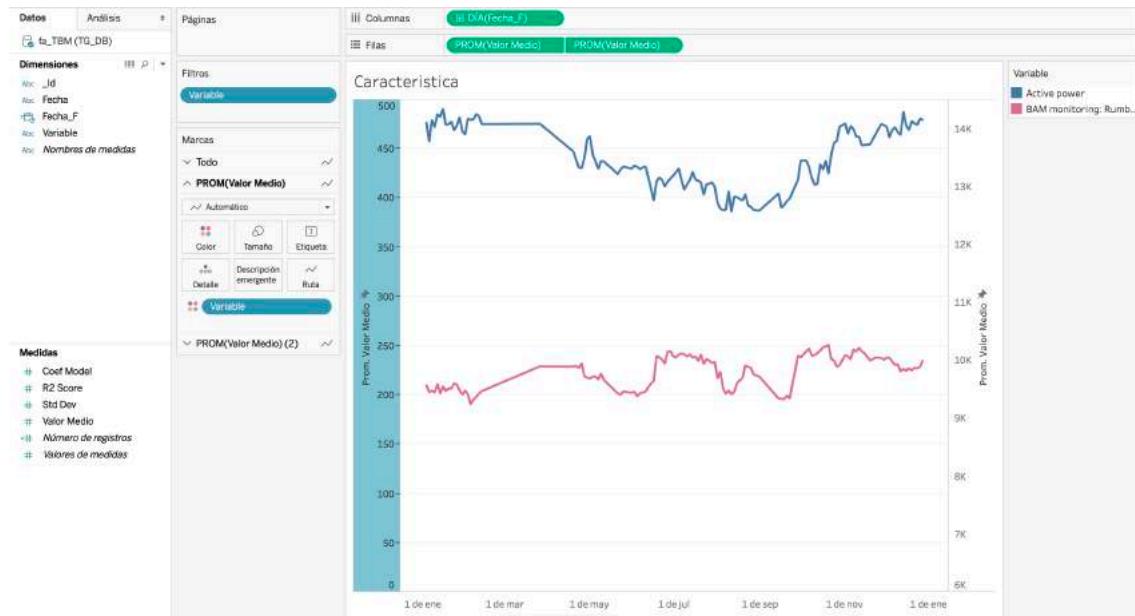


Fig. 32

vi. Visualizamos la influencia de la presión atmosférica.

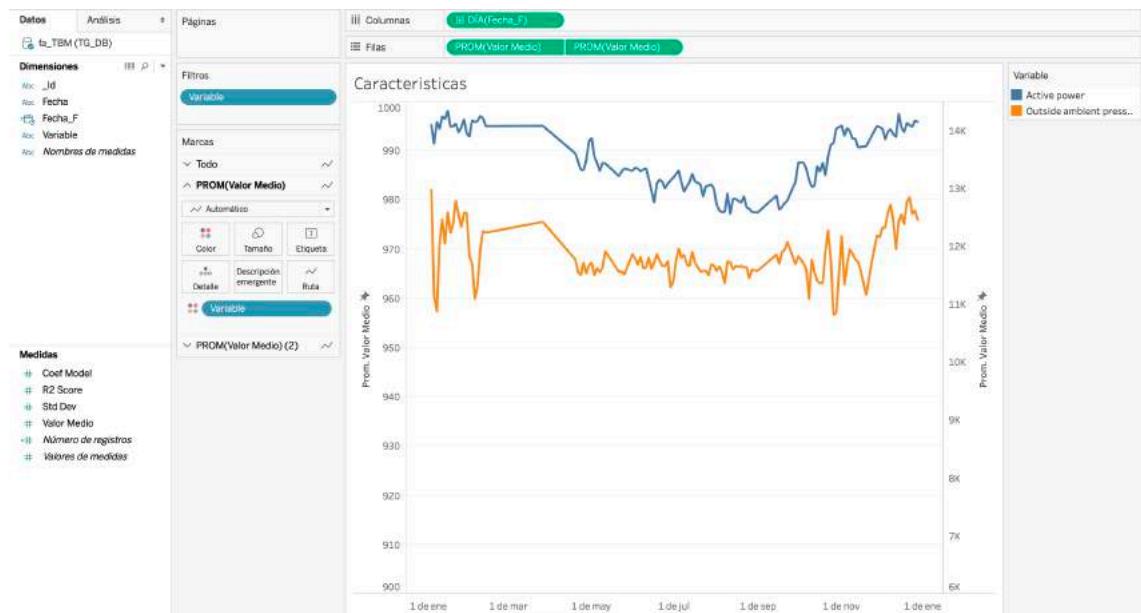


Fig. 33

vii. Visualización de los parámetros propios de la regresión.

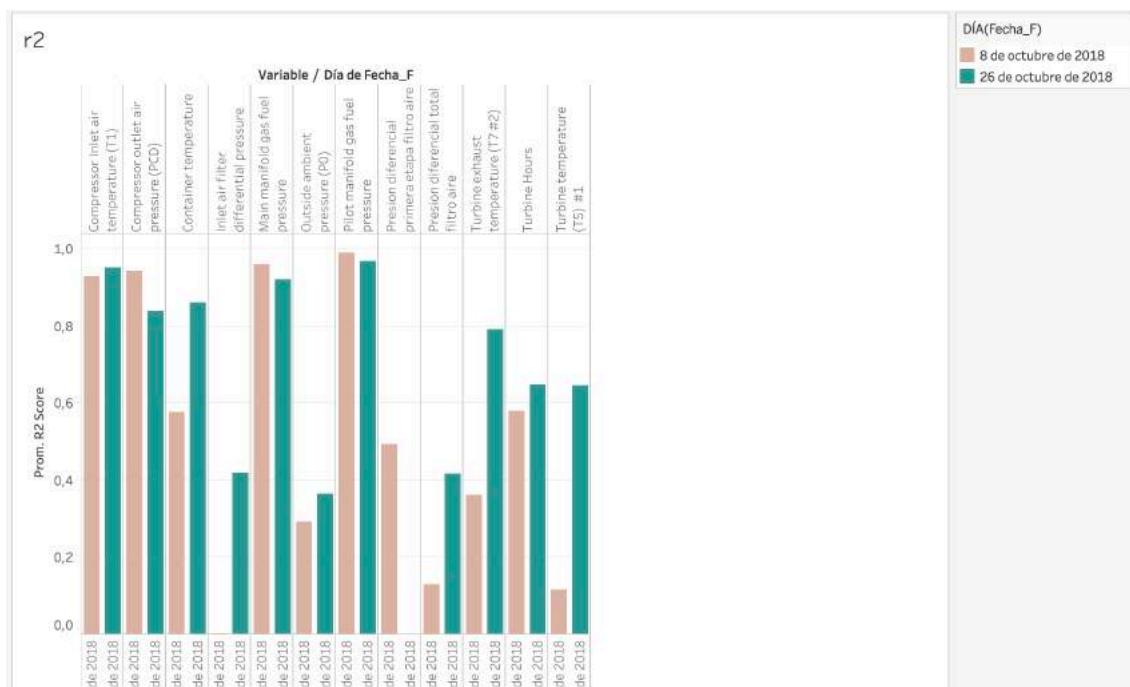


Fig. 34

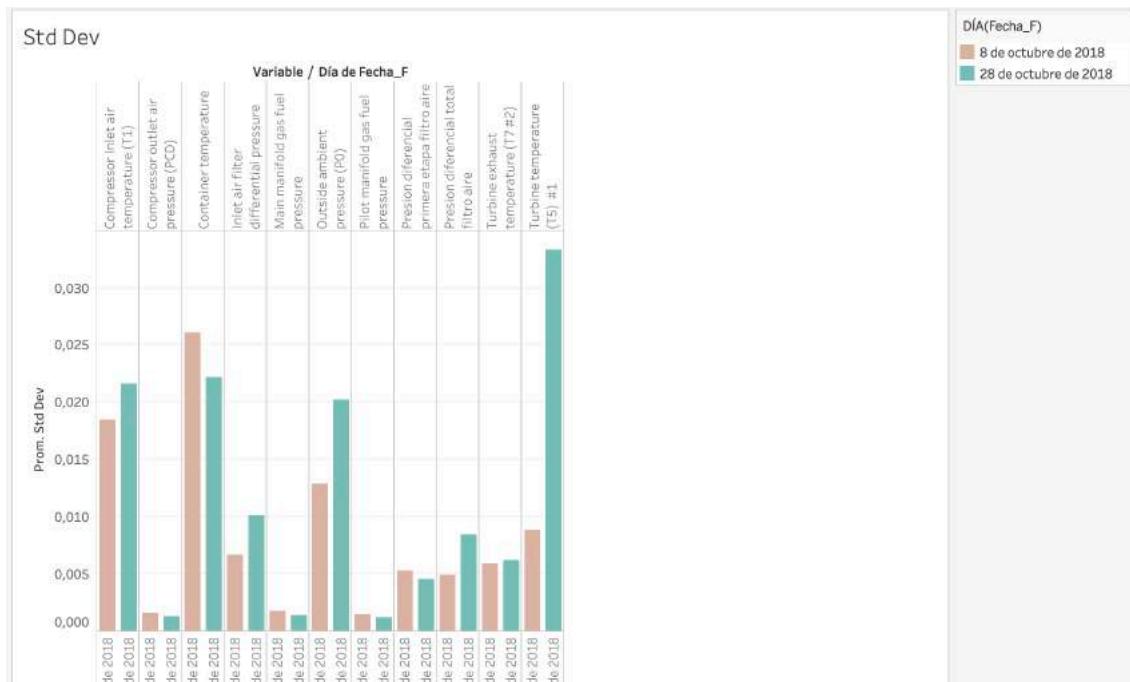


Fig. 35

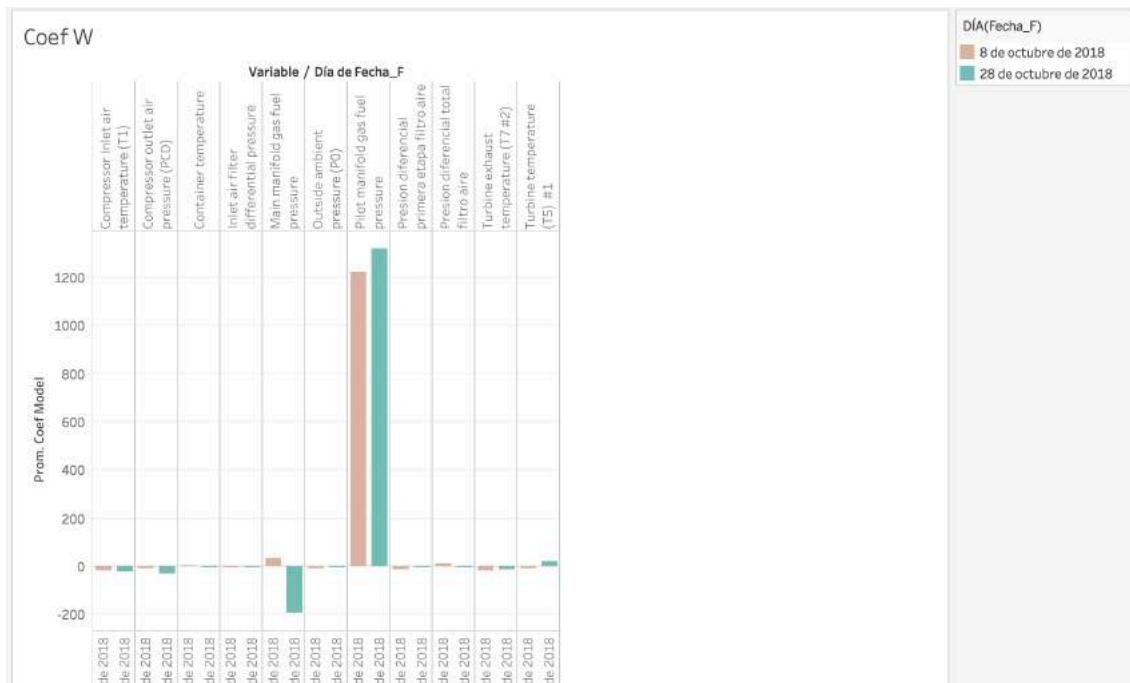


Fig 36

Con este análisis podremos ser capaces de discernir cambios en la sensibilidad de los modelos a las diferentes variables, y como estos cambios afectan a las prestaciones de la máquina.

De todo lo anterior, podemos concluir de manera general, la gran influencia que tiene la época del año en la potencia que la máquina es capaz de desarrollar, así vemos como en invierno la máquina da más potencia, mientras que en verano las prestaciones bajan, lo que nos lleva a concluir la importancia que tiene las condiciones ambientales. Por ello nos planteamos desarrollar un modelo de predicción basado en las condiciones ambientales. Trabajo que desarrollamos en el siguiente apartado.

8. Predicción basada en las condiciones ambientales.

Tres son las principales características o variables que definen las condiciones ambientales de cualquier clima, la **presión atmosférica**, la **temperatura ambiente** y la humedad relativa.

La **humedad relativa (RH)** es la relación entre la presión parcial del vapor de agua y la presión de vapor de equilibrio del agua a una temperatura dada. La humedad relativa depende de la temperatura y la presión del sistema de interés. La misma cantidad de vapor de agua produce una mayor humedad relativa en el aire frío que en el aire caliente.

Un parámetro relacionado con la humedad relativa es la **temperatura de bulbo húmedo**, que es el límite de enfriamiento que alcanza una corriente de aire al entrar en contacto con una masa de agua.

Así, y tras lo visto en el apartado anterior para incrementar la potencia de la máquina deberemos intentar disminuir la temperatura de aire al entrada del

ciclo, es decir del compresor. Para ello nos basaremos en los principios anteriormente comentados, llevando el aire de combustión a su punto más cercano posible de la temperatura de bulbo húmedo. A tal efecto todos los fabricantes de turbinas de gas instalan un sistema de enfriamiento adiabático, por lo que un buen diseño y mantenimiento de este sistema garantiza las máximas prestaciones de la máquina.

8.1. Sensorización de las condiciones ambientales.

A pesar de la gran cantidad de instrumentación que posee esta máquina, no disponemos de los sensores adecuados para medir las condiciones ambientales, para ello adquirimos una estación meteorológica, que nos permita obtener con máxima fiabilidad las características mencionadas anteriormente (temperatura ambiente, presión ambiente y humedad relativa). Las características del equipo y sensores adquiridos se muestran la Figura 37



Fig. 37

Además con estas tres variables podremos calcular de forma teórica la temperatura de bulbo húmedo, que como fue anteriormente mencionado, se trata del límite teórico de enfriamiento que podemos alcanzar en el aire.

En el Anexo IX, se presenta el algoritmo de cálculo de la Temperatura de bulbo húmedo, conocidas la temperatura ambiente, humedad relativa y presión atmosférica.

La Humedad relativa viene dada por la expresión:

$$HR = \frac{e(T)}{e_s(T)} \cdot 100\%$$

Donde:

HR: Humedad Relativa

e(T): presión parcial real de vapor de agua en aire húmedo, en Pa

$e_s(T)$: presión parcial de vapor de agua en aire húmedo saturado, en Pa, que expresa el hecho de que a una temperatura dada, existe un máximo en la cantidad de vapor de agua que puede estar presente, en otras palabras es la máxima presión parcial $e_s(T)$, que puede ejercer el vapor de agua a una temperatura (bulbo seco) particular.

$$e_s(T) = 1\text{Pa} \cdot e^{\left(A \cdot T^2 + B \cdot T + C + \frac{D}{T} \right)}$$

Donde:

$$A = 1,237884 \times 10^{-5}$$

$$B = 1,9121316 \times 10^{-2}$$

$$C = 33,93711047$$

$$D = -6,3431645 \times 10^3$$

T = temperatura ambiente de bulbo seco en grados Kelvin

La presión parcial real de vapor de agua se puede obtener mediante la ecuación experimental de Carrier:

$$e(T) = e_s(T_w) - \frac{[P - e_s(T_w)] \cdot (T - T_w)}{\Theta + \chi \cdot T_w}$$

Donde:

$E(t)$ = presión parcial real de vapor de agua en aire húmedo en Pa, a la temperatura de bulbo seco T

$e_s(T_w)$ = presión parcial de vapor de agua en aire húmedo saturado en Pa, a la temperatura de bulbo húmedo T_w

P = presión atmosférica local en Pa

T_w = temperatura de bulbo húmedo en K

$\Theta = 1940$

$\chi = -1,44$

Sustituyendo en la ecuación de la HR, tenemos:

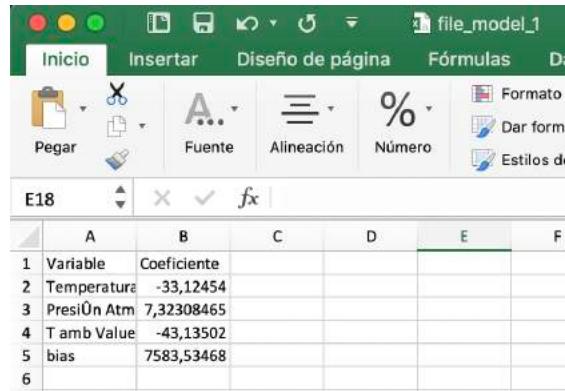
$$HR = \frac{e_s(Tw) - \frac{[P - e_s(Tw)] \cdot (T - Tw)}{\theta + \chi \cdot Tw}}{e_s(T)} \cdot 100\%$$

Para la solución de esta ecuación necesitaremos conocer la humedad relativa, la presión atmosférica y la temperatura ambiente, iremos iterando diferentes valores de Tw (T bulbo húmedo), hasta que ambos términos se igualen, con una tolerancia de error del 1%. El algoritmo de cálculo iterativo se presenta en el código del script del Anexo IX.

8.2. Entrenamiento del modelo.

En el Anexo X mostramos el código de un NoteBook que nos devuelve un modelo entrenado en base a las condiciones meteorológicas, recogidos en el dataset "data_CCAmb_1.csv".

En la Figura 38, se muestran Los coeficientes y el bias del modelo resultante se recogen en el fichero csv, "file_model_1.csv". El resultado del entrenamiento no es razonable, disponemos de un $r^2 = 0.75$, que consideramos malo, aparentemente no tenemos sobre ajuste., y que mostramos en la Figura 39.



The screenshot shows a Microsoft Excel spreadsheet titled "file_model_1". The table has columns A and B. Column A contains labels for variables: "Variable", "Temperatura", "PresiUn Atm", "T amb Value", and "bias". Column B contains their corresponding coefficients: "-33,12454", "7,32308465", "-43,13502", and "7583,53468".

A	B
1 Variable	Coeficiente
2 Temperatura	-33,12454
3 PresiUn Atm	7,32308465
4 T amb Value	-43,13502
5 bias	7583,53468
6	

Fig. 38

```
*****
Tamaño DataSet: 626 filas de datos y 4 variables registradas
*****
=====
R2 en training: 0.7469
R2 en test: 0.7881
Sobre Ajuste--->> 0.0551
Error cuadratico medio: 15943.45
Error absoluto medio: 84.17
Mediana del error Absoluto: 59.84
=====
=====
R2 en training: 0.7818
R2 en test: 0.6656
Sobre Ajuste--->> 0.1487
Error cuadratico medio: 14803.55
Error absoluto medio: 80.90
Mediana del error Absoluto: 55.88
=====
=====
R2 en training: 0.7394
R2 en test: 0.8190
Sobre Ajuste--->> 0.1076
Error cuadratico medio: 16984.59
Error absoluto medio: 86.81
Mediana del error Absoluto: 63.39
=====
=====
R2 en training: 0.7636
R2 en test: 0.7437
Sobre Ajuste--->> 0.0260
Error cuadratico medio: 14806.83
Error absoluto medio: 81.07
Mediana del error Absoluto: 59.05
-----
```

Fig. 39

8.3. Análisis del modelo basado en las condiciones ambientales mediante series temporales en Python.

Una serie de tiempo es una secuencia de datos, observaciones o valores, medidos en determinados momentos y ordenados cronológicamente. Los datos pueden estar espaciados a intervalos iguales o desiguales. Una vez que se captura una serie de tiempo, a menudo se realiza un análisis sobre ella para identificar patrones en los datos, en esencia, lo que se busca es entender que sucede a medida que el tiempo va avanzando. Ser capaz de procesar datos de series de tiempo es esencial para entender como las condiciones meteorológicas evolucionan a lo largo del día y del año y ver cómo éstas impactan en las prestaciones de nuestra máquina.

Así someteremos a test el modelo obtenido en el apartado anterior, esperando obtener conclusiones sobre las desviaciones del modelo y ser capaces de identificar causa raíz y la contramedida que anule o minimice dicha desviación.

Para ello trabajaremos con el fichero “ds_data.csv”, que contiene datos meteorológicos correspondiente a los meses de mayo, junio y comienzos de julio.

	A	B	C	D	E	F	G	H
1	Fecha	Temperatura Bulbo Humedad ValueY	Presión Atmosférica ValueY	Temperatura T1 ValueY	Potencia TG ValueY	T amb ValueY	Temperatura entrada aire después del enfriador	Temperatura entrada aire antes del enfriador
2	2/5/18 0:00	6,399996281	964,6396387	7,975193024	14184,80293	7,853012085	7,22	6,75
3	2/5/18 1:00	6,499996185	964,722229	7,792778015	14201,08848	7,269996125	7,16	6,71
4	2/5/18 2:00	5,799996853	965,2777832	7,17256546	14243,53018	6,642074585	7,01	6,55
5	2/5/18 3:00	5,699996948	963,7963379	7,099597931	14244,51719	6,552371979	6,41	5,96
6	2/5/18 4:00	5,699996948	963,7963135	6,8077311628	14257,84189	6,417823792	6,28	5,8
7	2/5/18 5:00	5,499997139	963,0555664	6,58883667	14272,64678	6,238426208	6,02	5,57
8	2/5/18 6:00	5,399997234	963,6111206	6,58883667	14285,47813	6,283275604	5,88	5,39
9	2/5/18 7:00	5,399997234	963,7963257	7,099597931	14240,07549	6,417823792	5,86	5,41
10	2/5/18 8:00	7,499995232	965,1852051	9,799343109	14067,34902	9,781539917	6,28	5,8
11	2/5/18 9:00	8,199995041	967,2222046	11,55709839	13811,19395	11,08217621	8,6	8,09
12	2/5/18 10:00	9,099998474	969,4443336	12,42612457	13887,19143	12,066886673	11,53	10,86
13	2/5/18 11:00	9,700000763	971,2036865	13,15578079	13816,54863	13,05045121	9,98	11,95
14	2/5/18 12:00	9,900001526	972,5	13,66654205	13767,29814	14,98408508	10,61	12,91
15	2/5/18 13:00	9,700000763	970,555298	13,48413086	13771,24609	15,85682983	10,94	14,15
16	2/5/18 14:00	10,500000381	970,1851563	14,06785583	13691,79189	16,015625	10,82	14,52
17	2/5/18 15:00	10,900000534	970,4629395	14,17730713	13687,35029	17,02315152	11,32	14,86
18	2/5/18 16:00	9,900001526	969,3518555	14,17730713	13709,55791	17,58535767	11,18	15,35
19	2/5/18 17:00	9,900001526	971,1110962	14,3962059	13676,49336	18,07870483	11,06	16,12
20	2/5/18 18:00	9,600000381	967,8703979	13,37467957	13737,19434	17,36111069	11,41	16,63
21	2/5/18 19:00	9,700000763	968,7963013	13,37467957	13712,02559	17,45080948	10,86	15,96
22	2/5/18 20:00	9,5	966,9444702	13,19216456	13738,6748	18,8307681	11	16,14
23	2/5/18 21:00	9,399999619	967,6852051	12,97336578	13714,49307	14,53559113	11	14,84
24	2/5/18 22:00	9,199998856	969,1666626	12,64502335	13800,45645	12,56221008	10,69	13,56
25	2/5/18 23:00	8,699996948	968,2407349	13,55709839	13722,38526	12,92100525	11,39	11,02
26	3/5/18 0:00	8,399995804	966,7592407	13,37467957	13780,12939	12,03041492	12,18	12,14
27	3/5/18 1:00	8,399995804	966,3888672	10,60197067	13979,01162	10,63368225	11,93	11,51
28	3/5/18 2:00	7,299995422	964,0740845	9,325056613	14066,85547	8,974246979	9,54	9,09
29	3/5/18 3:00	6,899995804	963,8889038	8,923751831	14085,6084	8,256652832	8,3	7,81
30	3/5/18 4:00	7,399995327	964,9074097	9,434513092	14057,97236	9,063394577	7,87	7,32
31	3/5/18 5:00	6,799995899	963,0555664	8,376502991	14089,52041	7,942710876	8,38	7,97
32	3/5/18 6:00	6,099996567	963,5185425	7,573879242	14171,97178	7,000865936	7,56	7,04
33	3/5/18 7:00	6,399996281	963,5185547	8,084640503	14169,01074	7,987560272	6,67	6,18
34	3/5/18 8:00	7,999994755	965,0000366	10,7118115	13968,54824	10,63368225	7,16	6,71

Fig. 40

En el ANEXO XI presentamos un notebook con el código para la obtención de gráficos de las series temporales, al objeto de observar tendencias y evoluciones. En la celda de inputs, tan sólo debemos elegir las fechas del intervalo que queremos representar y los variables.

i. Evolución de las condiciones de meteorológico.

En primer lugar visualizaremos como evolucionan la presión y temperatura atmosférica a lo largo del año, así como la temperatura de bulbo humedad, que como hemos visto anteriormente es un cálculo en función de la presión y temperatura atmosférica y de la humedad relativa.

Tomamos siete días del mes de mayo, Figura 41, y otros siete del mes de julio, Figura 42, y vemos como se comportan a lo largo de este periodo de tiempo. En ambas casos vemos una tendencia cíclica diaria, con un cierto retraso en la temperatura respecto de la presión. También inferimos una mayor peso en la temperatura ambiental, respecto de la presión atmosférica, en la humedad relativa resultante y en última instancia en la temperatura de bulbo húmedo. Hagamos un zoom y veamos esa evolución en 1 día.

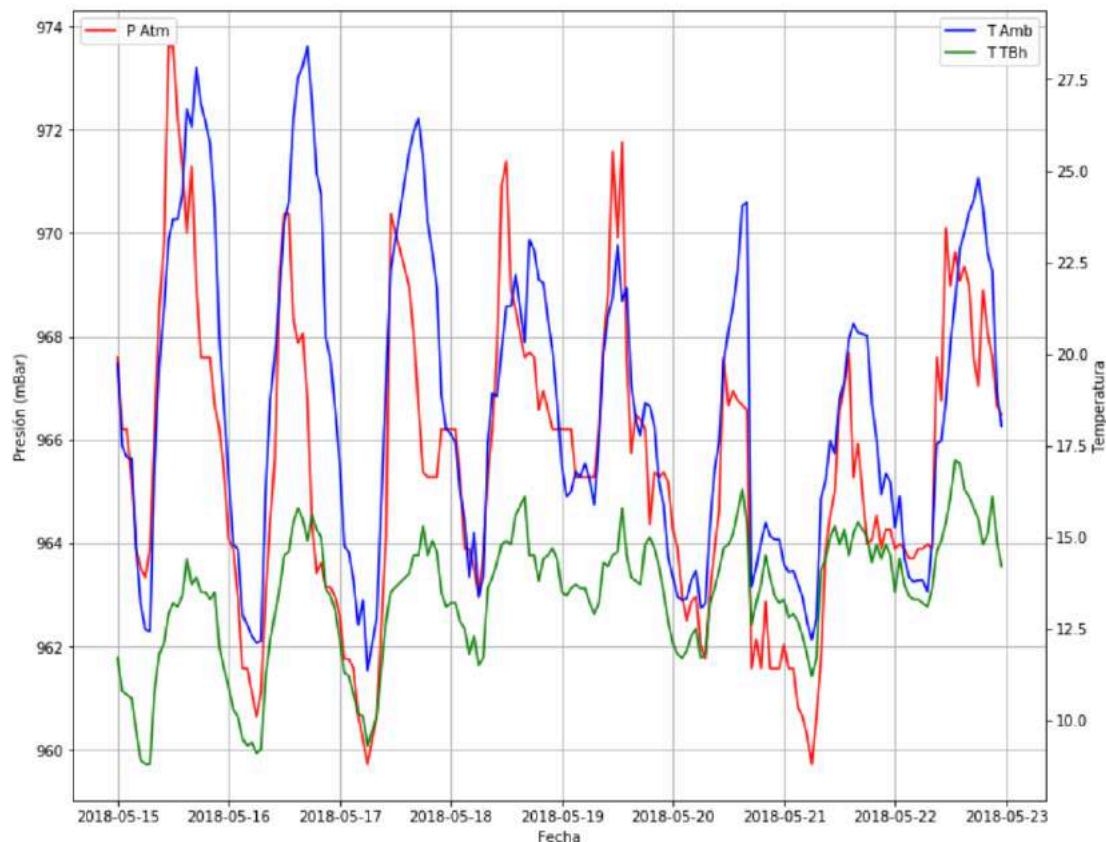


Fig. 41

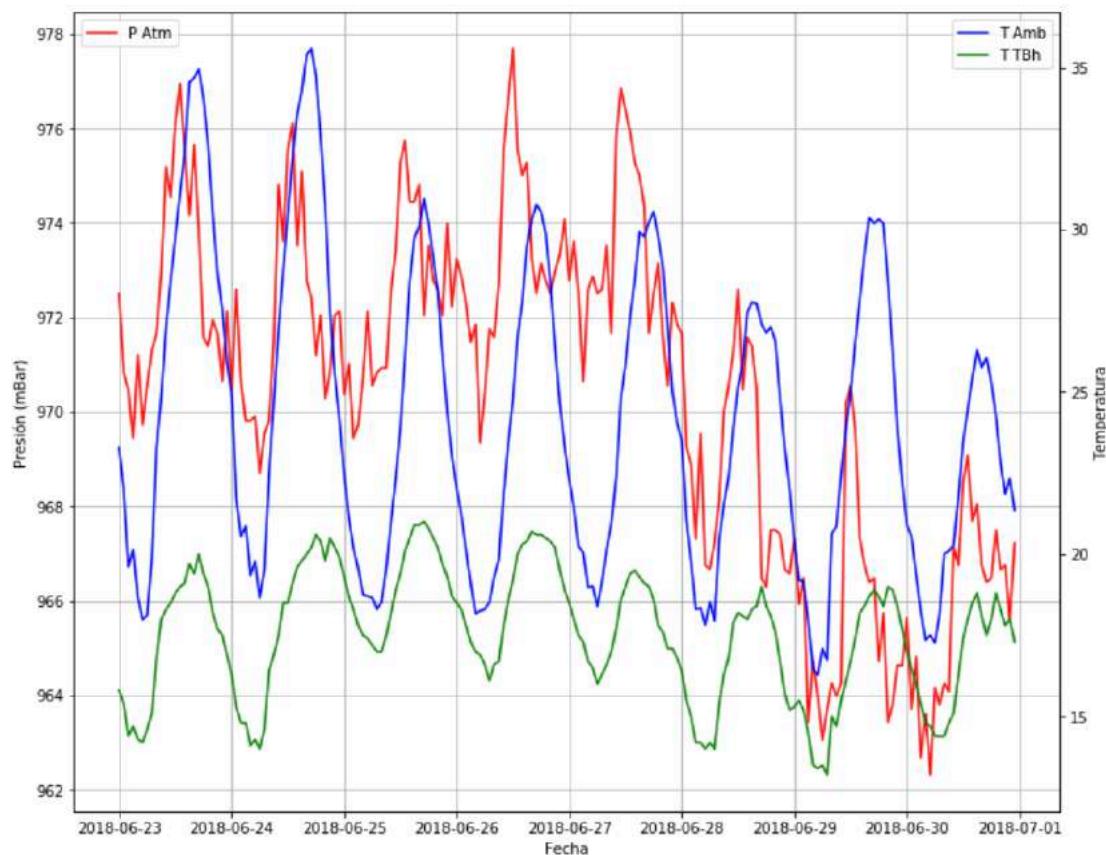


Fig. 42

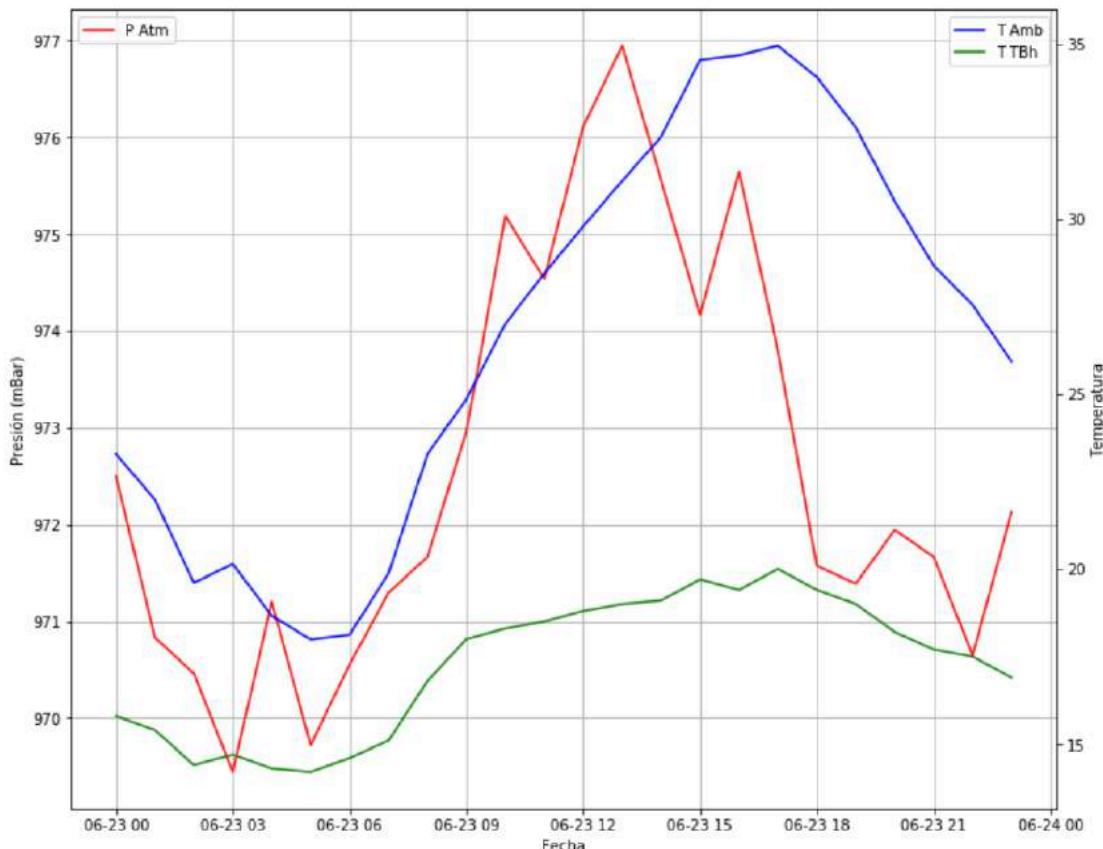


Fig. 43

Vemos como la presión atmosférica alcanza su máximo valor diario a primera hora de la mañana, estabilizándose hasta que comienza a descender a media tarde, alcanzando el mínimo, momentos previos al amanecer. La temperatura sigue un proceso similar , salvo que el máximo de temperatura lo tenemos a últimas hora de la tarde, con el ocaso del sol. Vemos como la temperatura de bulbo húmedo posee un paralelismo con la temperatura ambiente, aunque es más suavizada la pendiente.

ii. Cómo condiciona esta evolución el comportamiento del modelo.

En este punto vamos a comparar la desviación del modelo en términos de potencia real alcanzada y la predicción del modelo. Para ello vamos a analizar el comportamiento del modelo en cinco días del mes de mayo

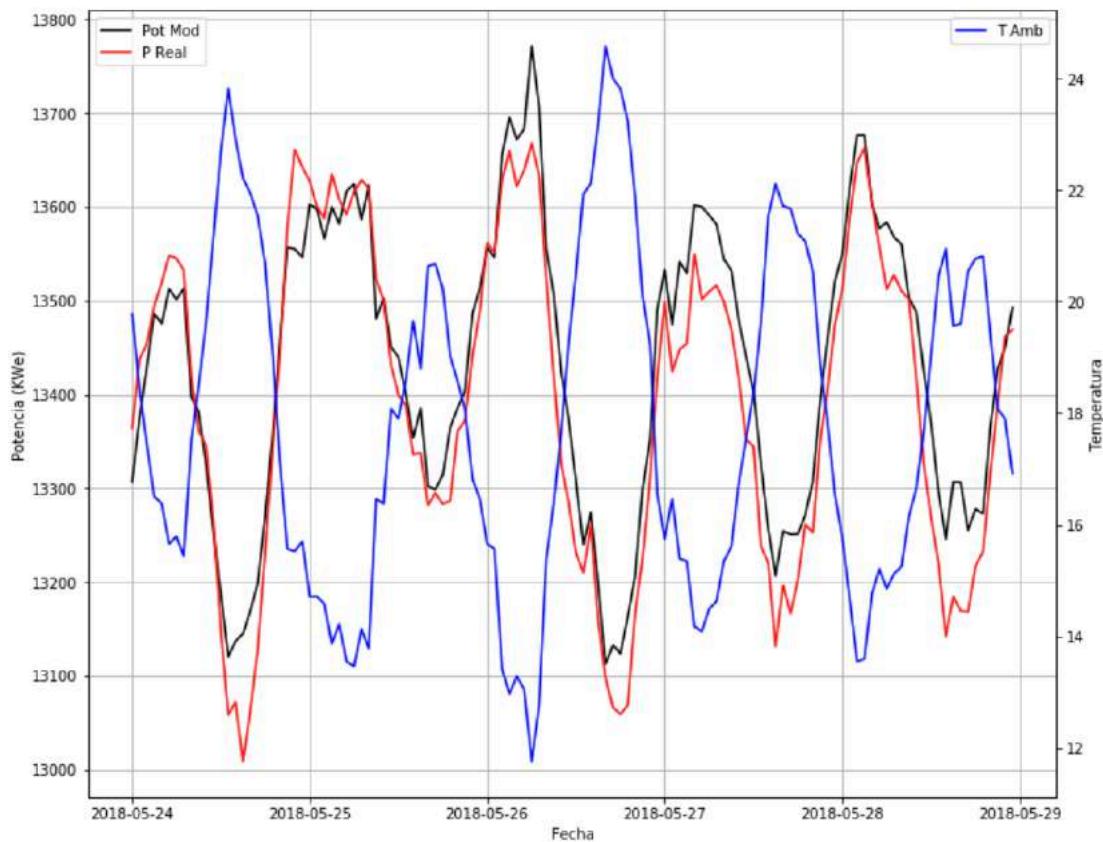


Fig. 44

Observamos una desviación de la potencia real a la predicción del modelo, y esta diferencia se acentúa en las horas centrales del día, cuando la temperatura es mayor, pasadas esas horas centrales la predicción coincide con la potencia real.

En las figuras 45 y 46 podemos observar, como además de lo comentado anteriormente, sobre la desviación de la predicción, vemos igualmente, que en días nublado y de menor temperatura la desviación baja y el comportamiento de dicha predicción mejora. Esto podría explicar que el modelo no sea tan bueno como cabría esperar, dados los antecedentes.

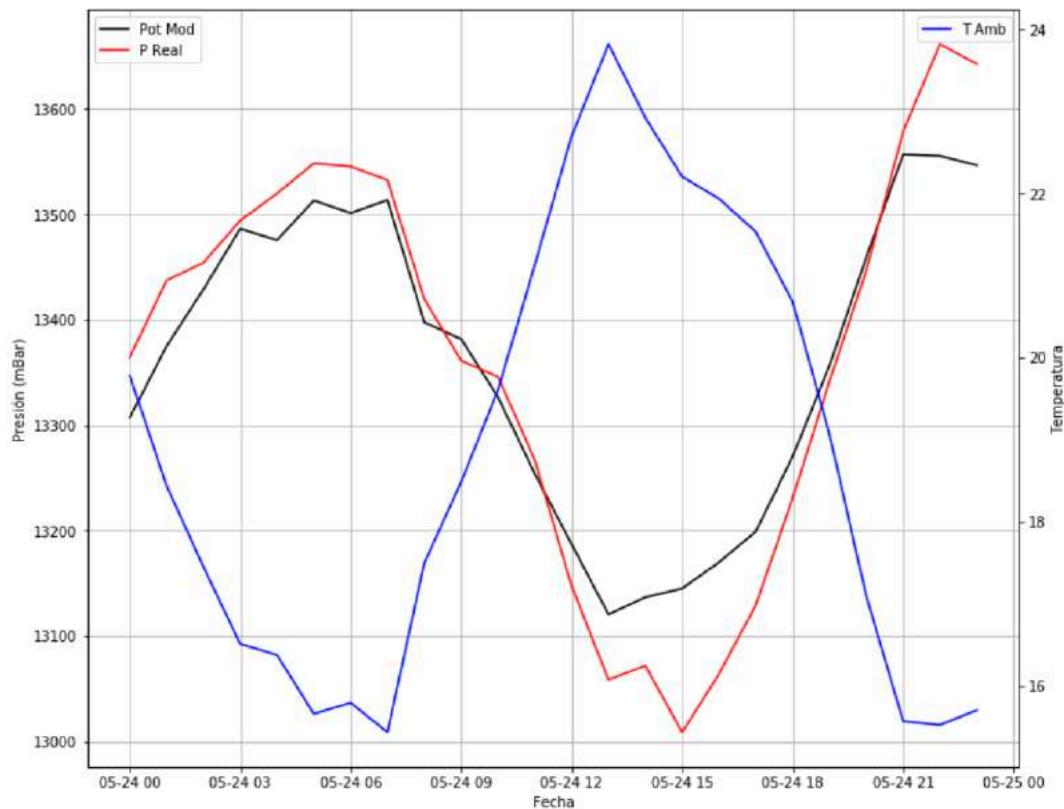


Fig. 45

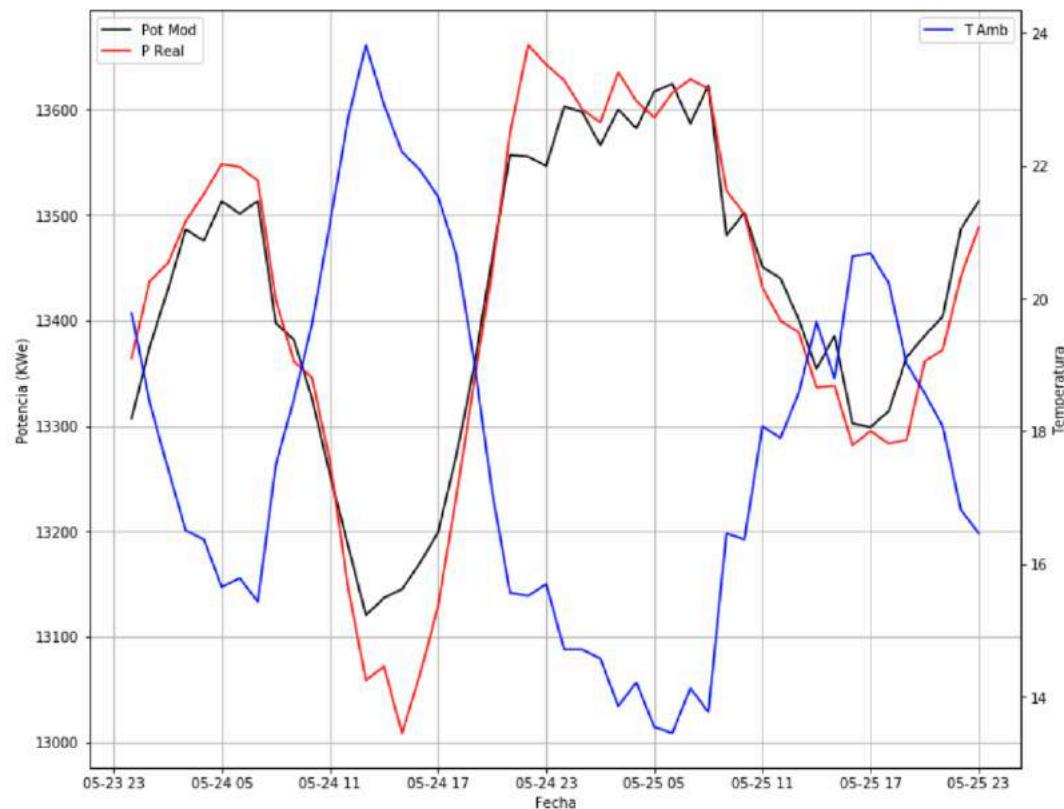


Fig. 46

iii. Análisis de Causa raíz.

Nos llama la atención el hecho de que haya una mayor desviación sistemática en las horas centrales del día y que el modelo no sea tan bueno como se podría esperar visto los resultado del entrenamiento anterior. Por ello vamos a presentar una breve descripción de la investigación de causa raíz, para hayar el motivo por el cual bajan las prestaciones de la máquina a unas horas determinadas del día. Para ello seguiremos los siguientes pasos:

a. Descripción del sistema.

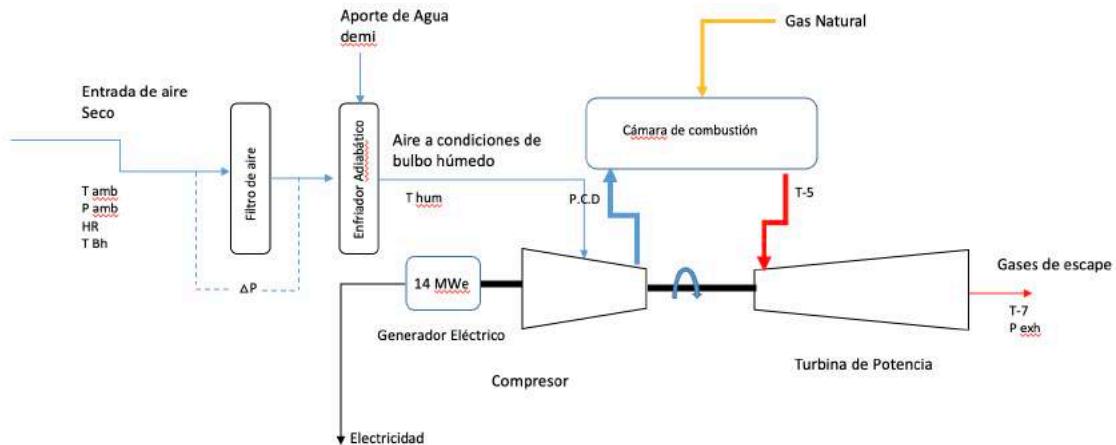


Fig. 47

Sabemos por el proceso de machine cuales son las principales variables (presión t temperatura), por ello veamos como evolucionan la P desde la atmósfera hasta el escape (Ver punto 1.1 de esta memoria) y la temperatura desde la admisión en la calle hasta el escape.

a.1. Presión vs Potencia.

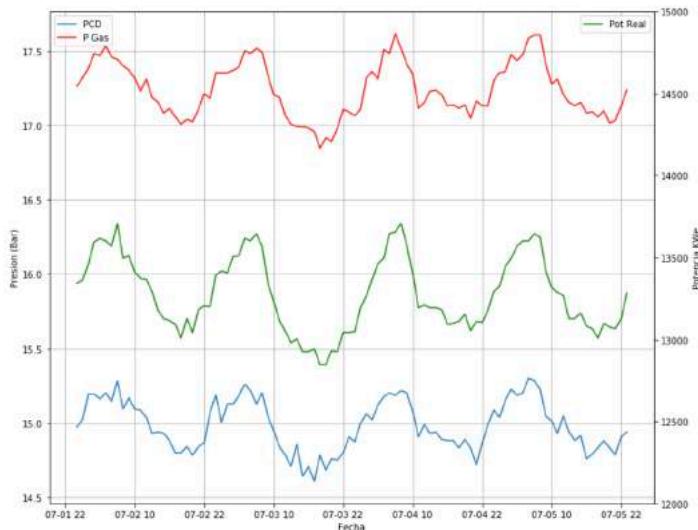


Fig. 48

a.2. Temperatura vs Potencia

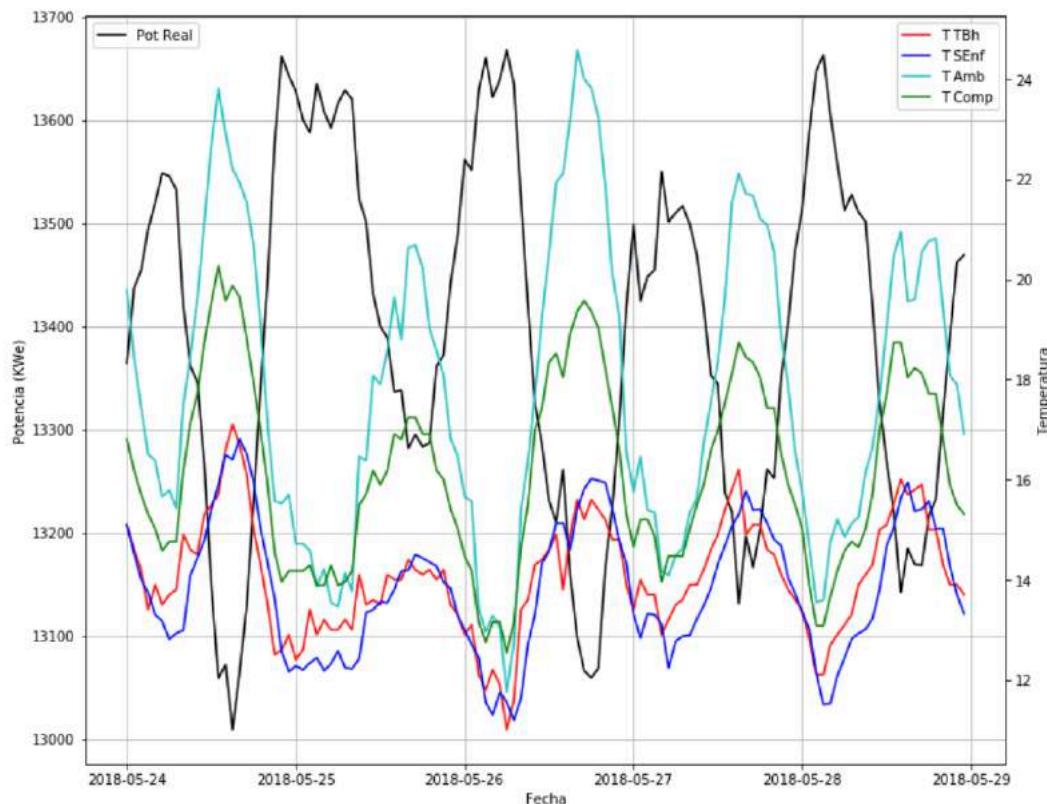


Fig. 49

Con respecto a la presión no se aprecia nada relevante, es decir sigue el comportamiento esperado tanto por nuestra experiencia, como por el predicho por el modelo.

En cuanto a la temperatura, obtenemos los siguiente hallazgos relevantes:

- La línea azul representa la Temperatura de bulbo húmedo que es la temperatura mínima teórica que podemos alcanzar. La línea roja es la temperatura real que alcanzamos en el equipo de enfriamiento adiabático. Siendo la línea celeste la Temperatura de la calle. Con todo esto, vemos el rendimiento del enfriador adiabático es adecuado, prácticamente se consigue la temperatura de bulbo húmedo. **Hallazgo: Equipo de enfriamiento adiabático funciona correctamente.**
- La línea verde representa la temperatura a la que el aire entra en el compresor. En este caso la T aumenta, en las horas centrales. **Hallazgo: Desde la salida del enfriador al compresor el aire de combustión se calienta siendo el motivo por el que se pierde prestaciones en la máquina.**

b. Investigación Física del fenómeno.

Mediante una cámara termográfica, hallamos la causa raíz del fenómeno: **falta de aislamiento en algunos tramos del conducto.**

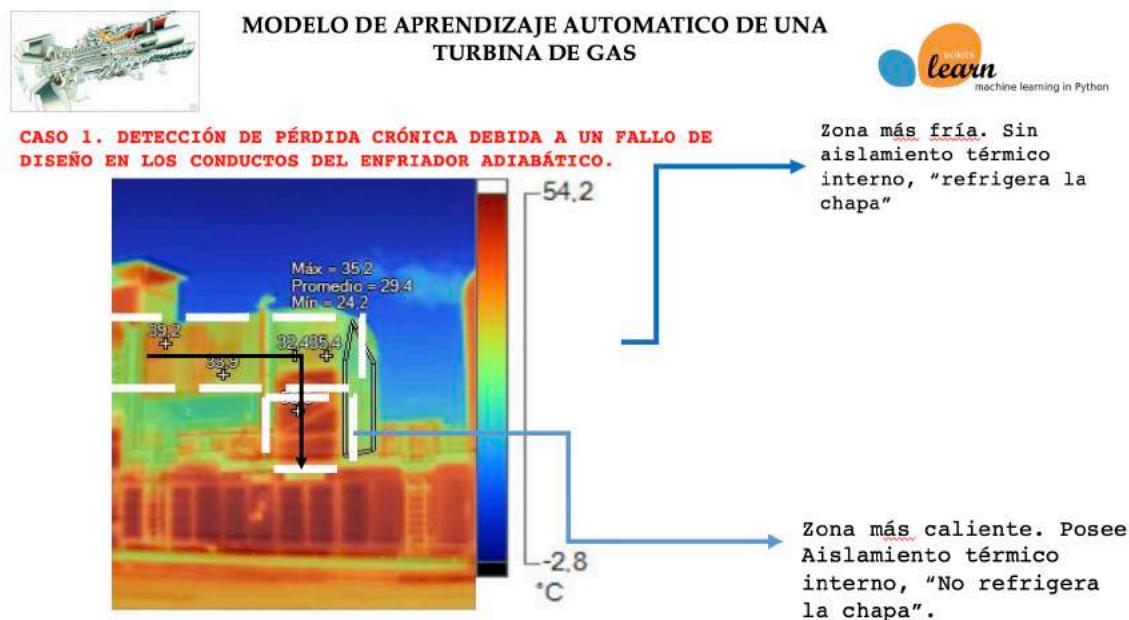


Fig. 50

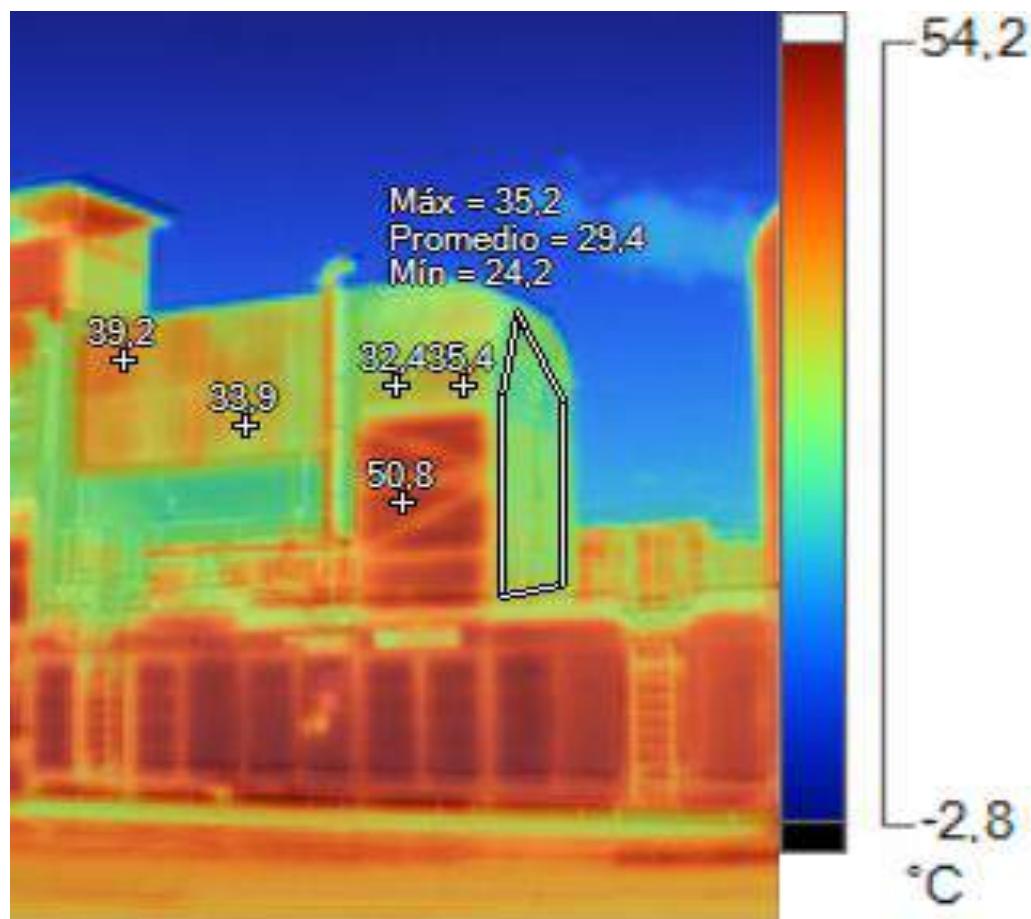


Fig. 51

Como podemos observar, en el conducto de salida del enfriador y entrada a la turbina, existen zonas de diferente temperatura, debido a que ciertas zonas fueron aisladas y otras no. Este aislamiento se coloca interiormente y su objetivo es atenuar el ruido del fluido circulado por el conducto, pero a su vez hace también de aislante térmico. Así en las horas centrales del día, cuando la insolación es mayor, la chapa de los conductos se calienta, el tramo que está aislado permanece más caliente, mientras que el tramo no aislado es refrigerado por el aire, previamente enfriado en el enfriador adiabático, a costa del calentamiento del aire, lo que provoca la pérdida de prestaciones, según hemos visto en las figuras 30.1 y 30.

c. Solución y beneficios.

Se hizo una pequeña inversión de 4.000 € en aislar el conducto, obteniendo una reducción de hasta máximos de 4°C en esas horas, lo que nos supuso en 2018, el incremento de producción de 1.100 MWhe al año, lo equivale a unos 50.000 € de pérdidas evitadas.

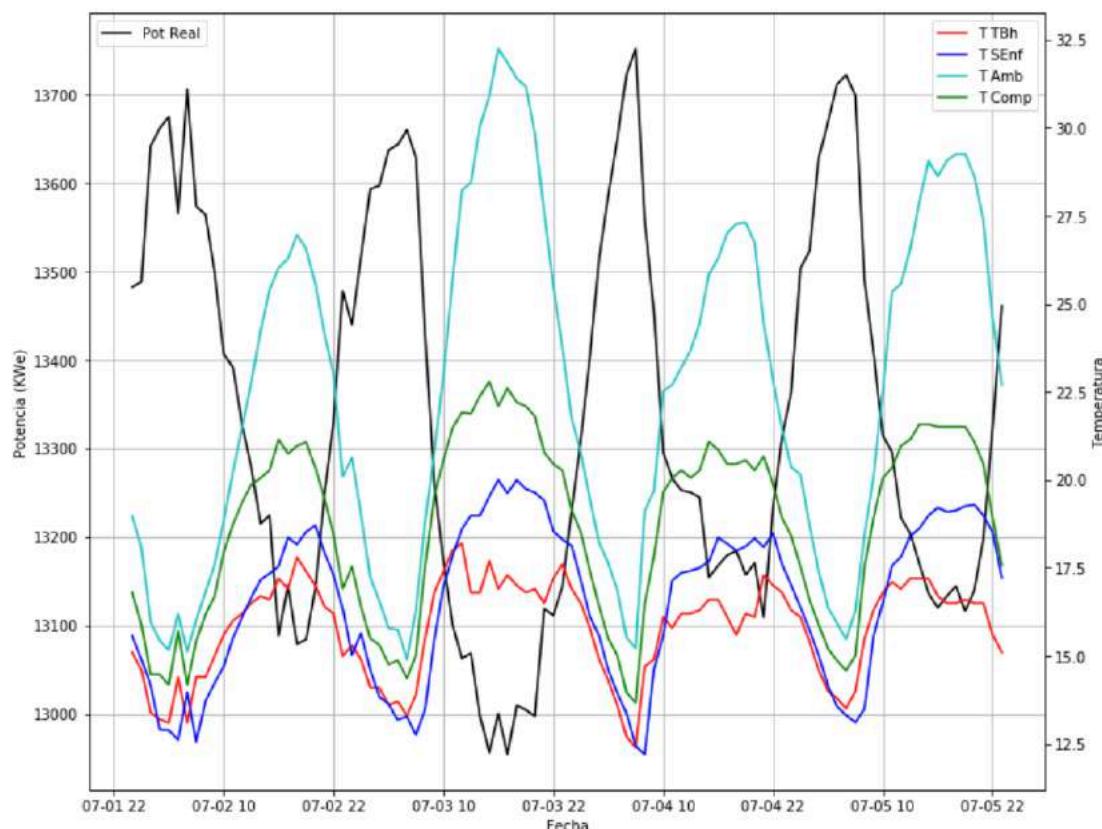


Fig. 52

La medida correctora fue tomada en mayo, en el gráfico de la Figura 52, observamos como las líneas azul y verde están próximas, es decir la diferencia entre la temperatura de salida del enfriador y la de alimentación al compresor se atenúan, pasando de 4°C en Mayo a 2 °C en Julio, a pesar del incremento de la radiación solar y de la temperatura ambiente, que en julio rondó lo 30 °C, en la ubicación de la estación meteorológica y 24 °C en Mayo.

En las Figuras 53 y 54 podemos observar el “ANTES Y DESPUES”, vemos como en Mayo las predicciones del modelo era superior a la real obtenida, y como tras el análisis y el hallazgo de la causa raíz, hemos conseguido invertir la situación y ahora el modelo da predicciones menores a la potencia real del modelo.

En consecuencia hemos mejorado las prestaciones de la máquina respecto al modelo.

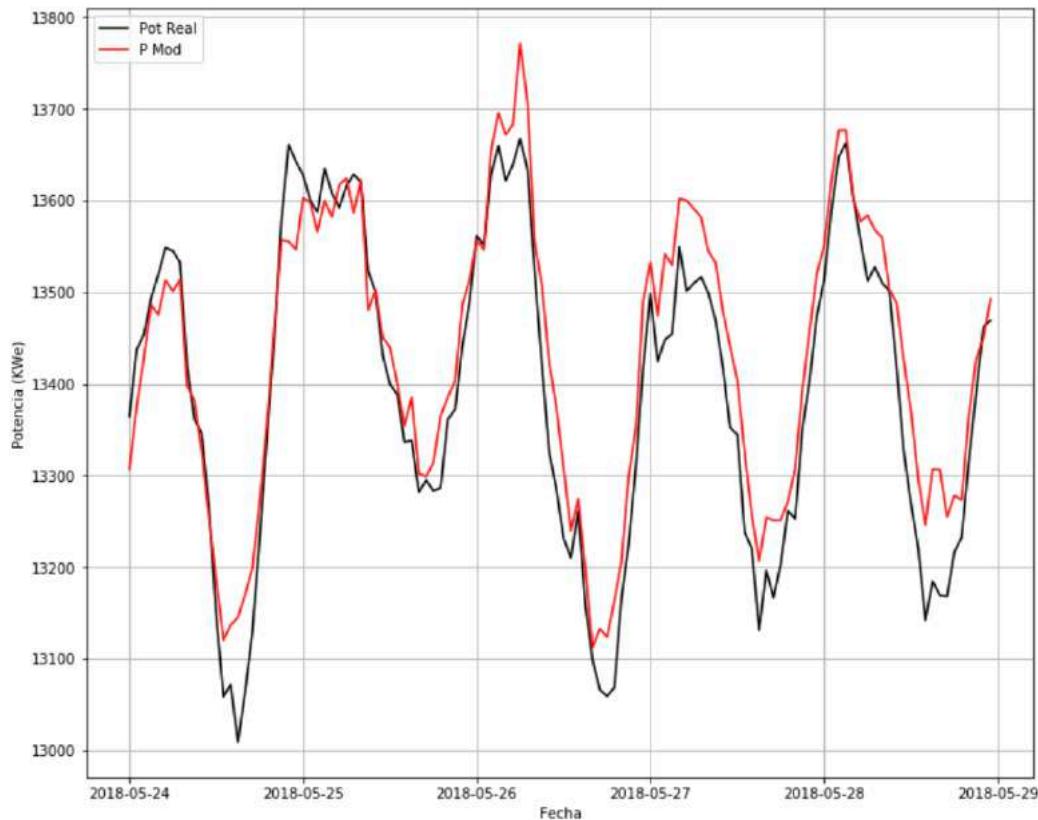


Fig. 53

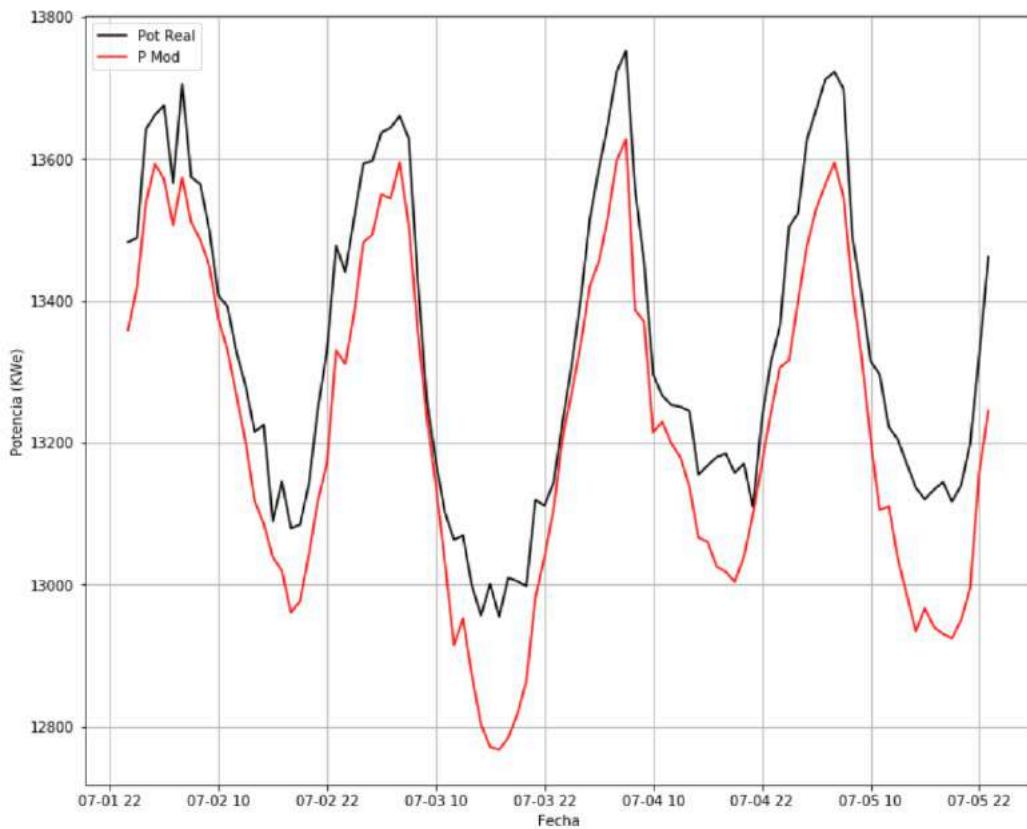


Fig. 54

8.4. Nuevo Modelo basado en Condiciones ambientales.

Un ejercicio interesante es ver como cambia el modelo, dadas una mejoras en el sistema. Para ello utlizamos el mismo notebook, pero cambiamos el dataset, que ahora será “*data_CCAmb_2.csv*”.

El resultado obtenido se muestra en la Figura 55.

```
*****
Tamaño DataSet: 669 filas de datos y 4 variables registradas
*****
=====
R2 en training: 0.9774
R2 en test: 0.9788
Sobre Ajuste--->> 0.0014
Error cuadratico medio: 2889.58
Error absoluto medio: 41.29
Mediana del error Absoluto: 33.14
-----
Iteraciones realizadas >>> 1
```

Fig. 55

Efectivamente ahora el modelo mejora sensiblemente su comportamiento, desaparecida la anomalía. Pasamos de un $r^2 = 0.75$ a un $r^2 = 0.97$. Respecto a los parámetros de la regresión los recogemos en el fichero “file_model_2.csv”

A	B
1 Variable	Coeficiente
2 Temperatura ambiente ValueY	-24,803105
3 Temperatura Bulbo Húmedo ValueY	-69,368693
4 Presión Atmosférica ValueY	11,986888
5 bias	3451,49606
6	

Fig. 56

Variable	Coeficiente Modelo 1	Coeficiente Modelo 2
Temperatura Bulbo Húmedo ValueY	-33,12454	-69,368693
Presión Atmosférica ValueY	7,32308465	11,986888
T amb ValueY	-43,13502	-24,803105
bias	7583,53468	3451,49606

Fig. 57

Nuestro nuevo modelo, se ha hecho más sensible a la capacidad de enfriar de nuestro enfriador adiabático, es decir aumenta el coeficiente de las variables de Temperatura de bulbo húmedo al doble, mientras que de la temperatura ambiente se reduce a la mitad. Lo cual es bueno, como se ha visto en la visualización, ya que las prestaciones de nuestra máquina han aumentado su grado de dependencia de una variable técnica, sobre la que nosotros podemos incluir y no una atmosférica sobre la que nada podemos hacer.

En la Figura 57, vemos otro indicador del modelo, el coeficiente de correlación r^2 , que nos dirige a la misma conclusión.

Variable	R2 Modelo 1	R2 Modelo 2
Temperatura Bulbo Húmedo ValueY	0.513	0.933
Presión Atmosférica ValueY	0.181	0.270
T amb ValueY	0.708	0.847

Fig. 58

Efectivamente vemos como el r^2 del modelo 2 para cada variable mejora, y como ahora el mejor coeficiente r^2 , el que explica el comportamiento del modelo es la T de bulbo húmedo.

9. Análisis del modelo basado en variaciones de los estadísticos.

Si recordamos la Figura 28, veímos tres eventos que provocaban la mejora de las prestaciones frente a un modelo obtenido con los datos de la primera mitad del año.

El evento 1, ha sido tratado en el punto 8.3 y supuso una mejora de un 1%. En el evento 2, observamos una subida hasta el 2%, provocado por una parada programada de mantenimiento para llevar la máquina a condiciones básicas, y tras la parada y pasados unos pocos días observamos una caída brusca de 2 puntos porcentuales, llegando a niveles del modelo inicial. Esto hizo saltar las alarmas del procesista, que analizando el modelo y los parámetros del mismo, llegó a una conclusión: El compresor estaba sucio, se organizó una parada y efectivamente se comprobó la existencia de un error humano, que provocó el rápido ensuciamiento del compresor. Como podemos observar en el evento 3, tras la limpieza exhaustiva, se recuperó la máquina a niveles de 2,5 % mejor de los que era a primeros de año.

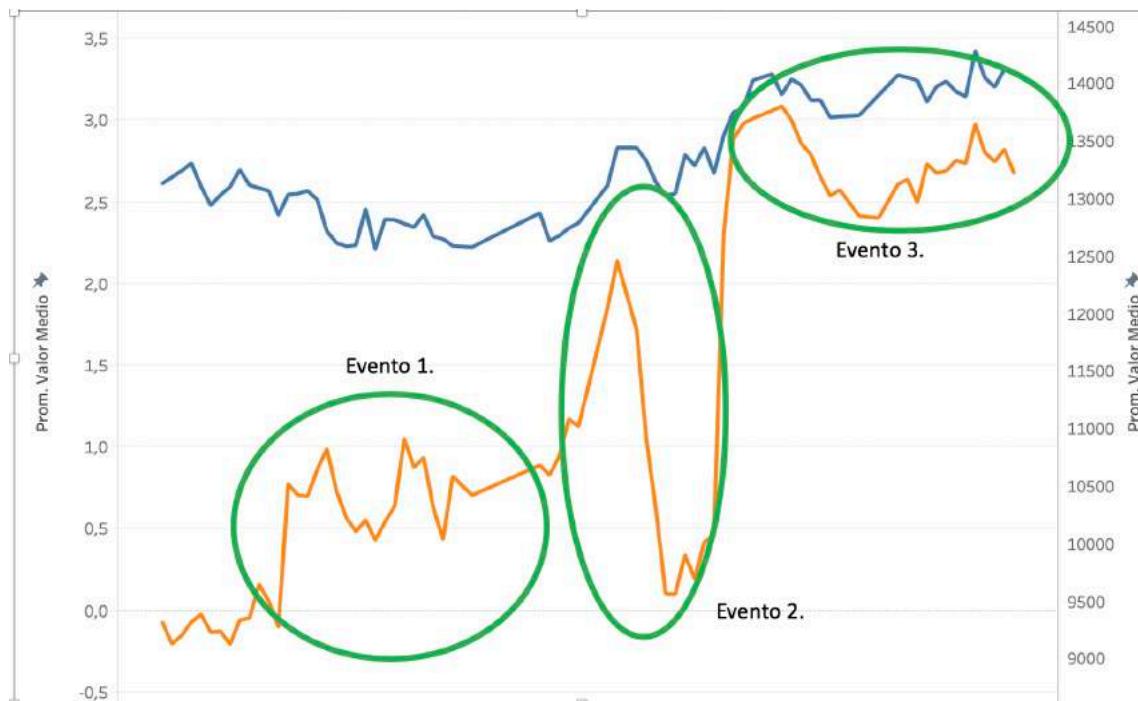
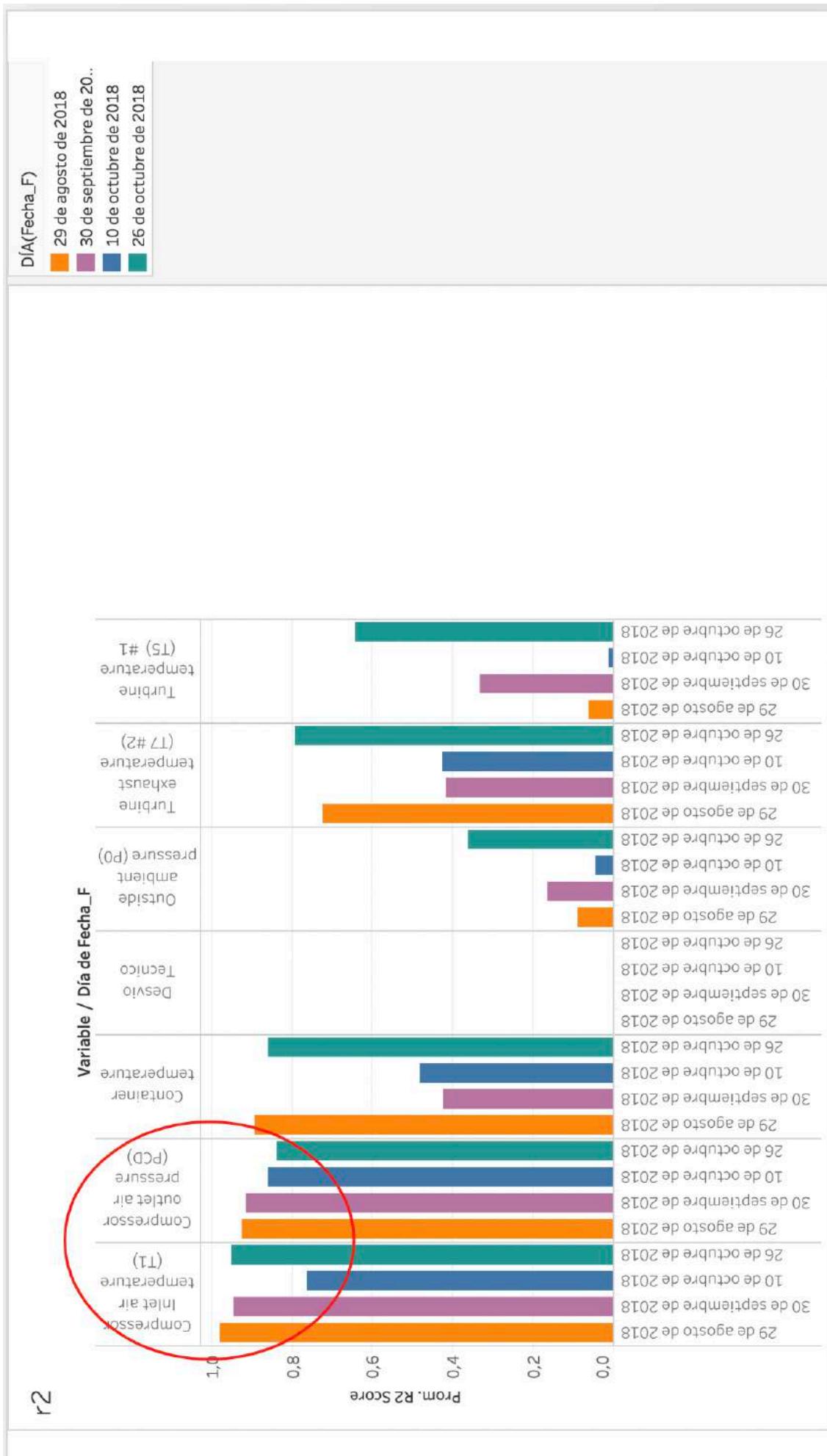


Fig. 28

En este apartado veremos como el procesista llegó a la conclusión de que el compresor estaba sucio, analizando los estadísticos del modelo. Para este análisis se empleará nuevamente Tableau.



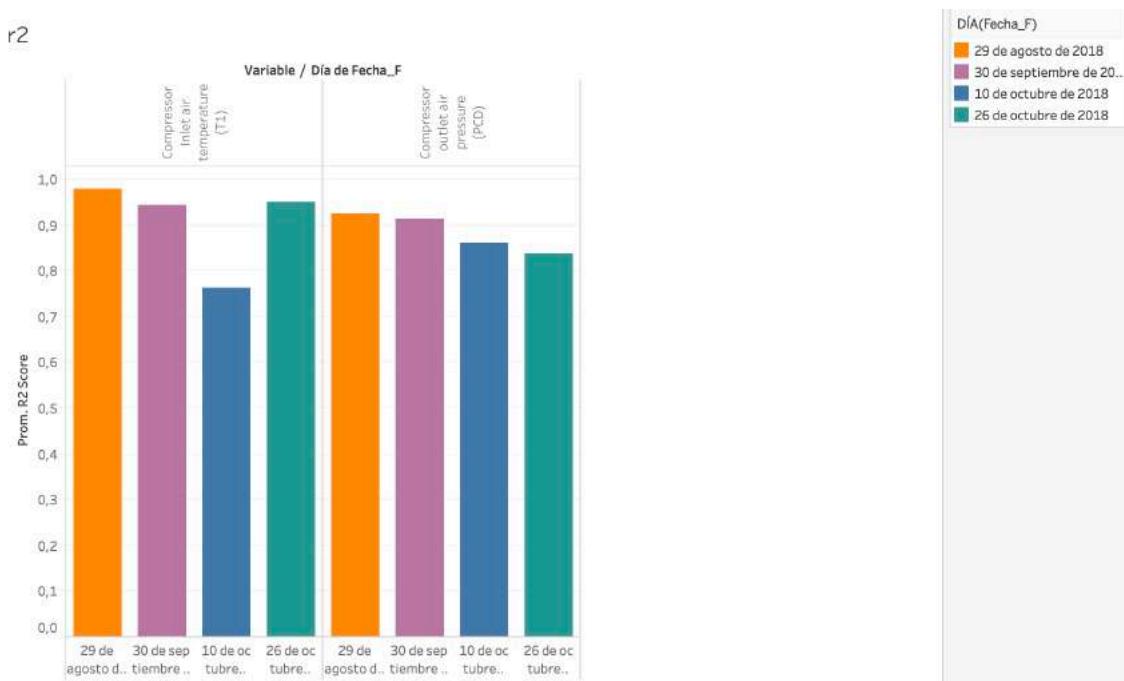


Fig. 60

Observamos como los días 29 y 30 de septiembre donde la máquina estaba recién arrancada tras el mantenimiento preventivo y puesta a condiciones básicas, el r2 de la variable T1 (T de alimentación al compresor), era la variable dominante del modelo, tras el evento, la variable dominante pasa a ser la PCD (presión del compresor). Esto indujo al procesista a pensar que el problema estaba en el compresor por pérdida de capacidad (ensuciamiento prematuro). El día 26 de Octubre se paró se limpió y vemos como el r2 de la T1, pasa a ser de nuevo la variable predominante y como se consigue la mejora sostenida durante varios meses de un 2,5 %.

10. Conclusiones

En este TFM, hemos conseguido desarrollar una herramienta para el análisis de un proceso industrial, basado en la generación eléctrica mediante una turbina de gas. Para ello hemos realizado los siguientes:

- ✓ Identificación de la Fuente de Datos, WinCC V7.4

- ✓ Extracción de los datos y construcción de un dataset
 - Librería de Python OpenOPC, para conexiones vía OPC Server
 - Adquisición desde el sistema de registros históricos del Scada, desde una base de datos con motor SQL Server

- ✓ Creación del modelo en cuatro fases:

- Establecer pregunta de negocio: *Qué variables y cómo, determinan la Potencia Máxima que la máquina es capaz de generar?* Y definir la variable dependiente, para ello empleamos el código del Script “ModReg_TG.py”.
 - Proceso ETL, Acondicionamos el dataset para que sea matemáticamente consistente, eliminando valores perdidos, nulos, variables descriptivas.
 - Determinar qué variables independientes serán tratadas, y los criterios de selección de datos. En nuestro caso, los datos deben corresponder a momentos de funcionamiento sin limitaciones técnicas. Para ello empleamos el código del Script “ETL_Scada.py”
 - Selección del modelo para el entrenamiento, hemos empleado la librería scikit-learn de Python, y la clase Linearregression.
-
- ✓ Ejecución del modelo: para ello hemos empleado el Notebook MLconSelección.ipynb y el código del Script “ModReg_TG.py”.

 - ✓ Selección de características, para lo que hemos empleado el Notebook MLconSelección.ipynb y el Script con el código “f_selection.py” en el que se han utilizado dos métodos:
 - Método SelectKBest
 - Método Forward Selection por agrgación de variables atendiendo al mejor r²
 - Hallazgo: El método Forward Selection mejora las correlación de la proporcionado por SelectKBest

 - ✓ Proceso de simulación de un control diario
 - Obtención de 211 datasets, para ello usamos el Notebook “Obtencion_Ficheros.ipynb” y el código del Script “get_files.py”
 - Realizamos el entrenamiento de estos datos
 - Hacemos un test del modelo obtenido con el obtenido en el proceso inicial, para ello empleamos el Notebook “OYMconDL.ipynb” y el código del Script “ModDL.py”
 - Almacenamos en MongoDB
 - Realizamos visualización de inteligencia de negocio mediante Tableau

 - ✓ Detección de patologías en la máquina, basadas en una modelización de las condiciones meteorológicas. Usando el Notebook SeriesCCAmb.ipynb
 - Uso de series temporales para ver la evolución de las características
 - Hallazgo: Se detecta un fallo de diseño, que nos permite mejorar un 1% las prestaciones de la máquina
 - Nuevo modelo con mejores parámetros.

- ✓ Detección de patologías basada en los estadísticos de los modelos de entrenamiento
 - Uso de visualización inteligente de negocio, mediante Tableau.
 - Hallazgo: error humano que provocó el ensuciamiento prematuro del compresor de la máquina. Mejora de las prestaciones en un 2,5%

En definitiva hemos creado una herramienta, extensible a múltiples procesos, de la que esperamos potencie nuestro conocimiento sobre nuestros procesos industriales, y nos permita una mejor y más eficiente toma de decisiones, basadas en datos y métodos científicos y vayamos dejando a un lado las opiniones e intuiciones en los diferentes procesos de toma de decisiones.

11. ANEXOS.

ANEXO I. Comunicación vía OPC.

Pasos a seguir:

1. Descarga de la librería OpenOPC en el link <https://sourceforge.net/projects/openopc/>
2. Por línea de comando instalaremos pywin32:
`pip install pywin32`
3. Por línea de comando instalaremos pyro 3.16:
`pip install pyro`
4. Por último verificamos la correcta conexión con el servidor OPC. A modo de ejemplo mostramos la conexión con el server de simulación que el gestor de conexiones OPC Server, Matrikon incluye.

```
C:\Users\jpgmacbookpro>python
Python 2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import OpenOPC
>>> opc = OpenOPC.client()
>>> opc.servers()
[u'Matrikon.OPC.Simulation.1']
>>> opc.connect('Matrikon.OPC.Simulation')
>>> opc.read('Random.Int4')
(2220, 'Good', '03/18/19 18:11:16')
>>> -
```

Las variables que serán capturadas vienen listadas en el fichero “SC_TBM.csv”

jupyter SC_TBM.csv ✓ Last Wednesday at 11:00 AM

	File	Edit	View	Language
1	Date			
2	APPARENT POWER			
3	CURRENT (Average)			
4	CURRENT R			
5	CURRENT S			
6	CURRENT T			
7	FREQUENCY			
8	ACTIVE POWER (3710 ACM)			
9	Factor de Potencia del Generador			
10	REACTIVE POWER			

....

```
151 T5 #3 compensated
152 T5 #4 compensated
153 T5 #5 compensated
154 T5 #6 compensated
155 T5 #7 compensated
156 T5 #8 compensated
157 T5 #9 compensated
158 T7 average used for control
159 Final control fuel demand
160 Gas fuel fired hours
161 Gearbox hours
162 Liquid fuel fired hours
163 Package hours
164 Turbine Hours
165 Water injection hours
166 REACTIV ENERGY
```

Código del script para la conexión al OPC Server y captura de datos.

```
1 import OpenOPC
2 import pandas as pd
3 import time
4
5 def DataOPC(n,timer):
6     opc = OpenOPC.open_client('localhost')
7     opc.connect('Matrikon.OPC.Server')
8     i=0
9     list_val=[]
10    list_time=[]
11    f=open('SC_TBM.csv','r')
12    sc_var=f.read().split('\n')
13    while i <= n:
14        val = opc.read(sc_var)
15        df=pd.DataFrame(val)
16        list_val.append(list(df[1]))
17        list_time_all=(list(df[3]))
18        list_time.append(list_time_all[0])
19        print('Dato It',i,'>>>',list_val[i])
20        list_var=list(df[0])
21        i+=1
22        time.sleep(timer)
23    return list_val, list_var, list_time
24
```

Código del script para la captura de datos vía OPC en un fichero .csv

```
1 from DataOPC import DataOPC
2 import pandas as pd
3 import time
4 import datetime
5
6 #Código para crear un data frame con los datos obtenidos desde el OPC.
7
8 #Se van a crear cuatro ficheros con cuatro data frames procedente del OPC
9 #Tomamos 10 datos durante 1 sg
10 for iter in range(4):
11     print('Iniciamos Iteración ##',iter)
12     list_var=DataOPC(24, 3600)
13     data=pd.DataFrame()
14     k=0
15     for i in list_var[1]:
16         data_l=[]
17         for j in range(len(list_var[0])):
18             data_n=list_var[0][j][k]
19             data_l.append(data_n)
20         data[i]=data_l
21         k+=1
22     data['Time'] = list_var[2]
23     data=data.set_index('Time')
24     print(data)
25     #Escribimos un nuevo .csv con los datos listo para ser trabajados.
26     #Nombre será la fecha del día y hora y el número de la iteración
27     #Obtenemos la fecha
28     date=datetime.datetime.now()
29     #Generamos el nombre del fichero
30     y=date.year
31     m=date.month
32     d=date.day
33     h=date.hour
34     data.to_csv('df_%d_%d_%d_%d.csv'%(iter,y,m,d,h),
35                 encoding='ISO-8859-1',decimal=',',sep=';')
```

ANEXO II. Adecuación datos capturados desde el Sistema de Registros Históricos.

```

1 def ETL_Scada(file,file_exit):
2     import pandas as pd
3     import numpy as np
4     import shutil
5     #Exportamos el fichero con los datos del modelo
6     df=pd.read_csv(file,sep=';',decimal=',',encoding='ISO-8859-1',low_memory=False)
7     lista=list(df.columns)
8     #Tamaño del dataset
9     print('Tamaño DataSet:',df.shape[0],'filas de datos y ',df.shape[1],'variables
10 registradas')
11     #Número de valores válidos en cada variable
12     len_lista=[]
13     j=0
14     for i in lista:
15         len_lista.append(len(df[i])-df[i].isnull().values.sum())
16         j+=1
17
18     #Lista de valores de la variable de referencia
19     #con el que se construirá el dataset final
20     df1=df[lista[len_lista.index(min(len_lista))]]
21     df1=df1.dropna()
22     list_df1=list(df1)
23     #Generamos las diferentes listas de las variable con los valores
24     #válidos y depurados
25     df_aux=pd.DataFrame()
26     s_list=[]
27     for i in range(int(df.shape[1])-1):
28         df_aux[lista[i]]=list(df[lista[i]])
29         df_aux[lista[i+1]]=list(df[lista[i+1]])
30         df_aux_1=df_aux[np.in1d(list(df_aux[lista[i]]),list_df1)]
31         df_aux_1.drop_duplicates(lista[i],keep="last",inplace=True)
32         s_list.append(list(df_aux_1[lista[i+1]]))
33     for i in range(int(len(s_list)/2)):
34         i+=1
35         s_list.remove(s_list[i])
36     #Construimos el data frame final
37     data=pd.DataFrame()
38     j=0
39     data['Fecha']=list(df_aux_1.iloc[:,0])
40     for i in range(len(s_list)):
41         data[lista[j+1]]=s_list[i]
42         j+=2
43     # Eliminamos datos en parada o en regulación de secundaria
44     data=data.drop(data[data['Potencia TG ValueY'].astype(float)<12500].index)
45     data.set_index('Fecha')
46     #Escribimos un nuevo .csv con los datos listos para ser trabajados en la RNA.
47     data.to_csv(file_exit,encoding='iso-8859-1',decimal=',',sep=';')
48     #Escribimos un nuevo .csv con los datos listos para ser almacenado en MongoDB
49     f='data'+file_exit
50     data.to_csv(f,sep=',',decimal='.',encoding='UTF-8')
51     ruta_origen =
52     '/Users/jpgmacbookpro/Documents/Modelizaciones_Plantas/Cabra/Modelo_TG/Machine_Learn
53     g/Modelo_2018/'+f
54     ruta_destino =
55     '/Users/jpgmacbookpro/Documents/Modelizaciones_Plantas/Cabra/Modelo_TG/Machine_Learn
56     g/Modelo_2018/'+f
57     ruta=shutil.move(ruta_origen, ruta_destino)

```

ANEXO III. Código del Notebook donde se desarrolla la Obtención del Modelo.

```
In [1]: 1 ##### INPUTS DEL SISTEMA #####
2 file_data = 'data_train.csv' # Datos primera mitad del año.
3 var_remove = 'var_remove_TBM.csv' # variables descartadas
4 #inicialmente bajo criterios de tipo de dato, y conocimientos
5 #del proceso industrial
6 criteria = 20 # Numero de variables elegidas para
7 #la descripción del modelo
```



```
In [2]: 1 from ModReg_TG import imp_ds, Fit_Var, sense_var, f_model_reg, new
2 from f_selection import ForwardSelection_r2, ForwardSelection_r2_
3
4 # Cargamos el dataset con todas las variables de la máquina
5 #objetivo de estudio
6 file=imp_ds(file_data)
7 # devuelve lista de variable y el dataframe completo
8
9 #Función que elimina las variables no numéricas y establece
10 #la variable dependiente
11 df_var=Fit_Var(file,var_remove)
12 # devuelve lista de variables tratadas, la X y la Y para el modelo
13 # de entrenamiento
14
15 #Obtención de los primeros estadísticos.
16 #Ejecutamos una función que devuelve el valor medio
17 #y la desviación std
18 stat=sense_var(df_var[1],df_var[2])
19 # devuelve lista de variables y lista con las medias y
20 #lista con las descv std.
21
22 #Una vez que el dataset ha sido tratado, se lanza la función
23 #de regresión para el Modelo
24 #Se lanza tantas veces como sea necesario para asegurar que
25 #no haya sobreajuste
26 e=0.1
27 i=0
28 while (e >0.05 and i <10):
29     reg=f_model_reg(df_var[1],df_var[2])
30     e=reg[2]
31     if i >10:
32         print('Modelo Presenta Sobre ajuste.')
33     i+=1
34 print('-----\nIteraciones realizadas >>>',i)
35 # devuelve una lista con los coeficientes de la regresion,
36 # el bias de la regresion y la diferencia entre el r2
37 #del train y el test
```

Tamaño DataSet: 3814 filas de datos y 168 variables registradas

=====

===
R2 en training: 0.9909
R2 en test: 0.9903
Sobre Ajuste-->>> 0.0006
Error cuadratico medio: 2213.06
Error absoluto medio: 36.83
Mediana del error Absoluto: 30.50

Iteraciones realizadas >>> 1

Funciones contenidas en el Script ModReg_TG.py, empleadas en el proceso de entrenamiento automático:

i. Función de carga del dataset.

```
6 def imp_ds(file):
7     import pandas as pd
8     # Cargamos el fichero en un dataframe
9     df=pd.read_csv(file,sep=';',decimal=',', encoding='ISO-8859-
1',low_memory=False)
10    # Eliminamos las filas con valores nulos o períodos
11    df=df.dropna()
12
13    # Capturamos la fecha del dataser para su trazabilidad
14    Fecha=df.loc[0][0][0:-5]
15
16    # Condición lógica que descarta aquellos valores de potencia que están
17    # bajo procesos de regulación o limitación.
18    df=df[df['Active power']+200 < df['Power setpoint']]
19    df=df[df['Active power']>12000]
20    if df.shape[0] > 24:
21        # Creamos un fichero con la lista de todas las variables
22        var=list(df.columns)
23        f=open('file_var.csv','w')
24        for i in var:
25            f.write(i+'\n')
26        f.close()
27        print('*'*60)
28        print('Tamaño DataSet:',df.shape[0],'filas de datos y ',
29              df.shape[1],'variables registradas')
29        print('*'*60)
30    else:
31        print('Datos en modo regulación o limitación. No hay datos
32 suficientes para una buena simulación.')
32
32    return var,df,Fecha
```

- ii. Función de obtención del modelo matemático, con sus estadísticos y parámetros.

```

34 #Función Modelo de Regresión Lineal
35 def f_model_reg(x,y):
36     #Entrenamos el modelos Mediante Regresión Lineal
37     Multivariable
38     import numpy as np
39     import pandas as pd
40     from sklearn.model_selection import train_test_split
41     from sklearn.linear_model import LinearRegression
42     from sklearn.metrics import mean_squared_error
43     from sklearn.metrics import mean_absolute_error
44     from sklearn.metrics import median_absolute_error
45
46     #Generamos las variables de entrenamiento y test
47     x_train,x_test,y_train,y_test=train_test_split(x,y)
48
49     #Modelo de Regresión Lineal
50     model=LinearRegression(fit_intercept=True)
51     model.fit(x_train,y_train)
52     predict_train=model.predict(x_train)
53     predict_test=model.predict(x_test)
54
55     #Evaluamos los resultados de la Regresión
56     r2_training = model.score(x_train,y_train)
57     r2_test = model.score(x_test,y_test)
58     print('='*70)
59     print('R2 en training: %.4f' %
60           (model.score(x_train,y_train)))
61     print('R2 en test: %.4f' %
62           (model.score(x_test,y_test)))
63     e=abs(model.score(x_train,y_train)-
64           model.score(x_test,y_test))/model.score(x_train,y_train)
65     print('Sobre Ajuste--->> %.4f' % (e))
66
67     #Determinación de los estimadores de la calidad del modelo.
68     print("Error cuadrático medio: %.2f" %
69           (mean_squared_error(predict_train,y_train)))
70     print("Error absoluto medio: %.2f" %
71           (mean_absolute_error(predict_train,y_train)))
72     print("Mediana del error Absoluto: %.2f" %
73           (median_absolute_error(predict_train,y_train)))
74
75     #Obtenemos los coeficientes de la regresión y los llevamos a un
76     #fichero
77     list_model=[model.intercept_]
78     list_xvar=['bias']
79     #print('Termino Ind: %.2f' % (model.intercept_))
80     j=0
81     for i in list(x.columns):
82         list_xvar.append(i)
83         list_model.append(model.coef_[j])
84         #print(i,'-- %.2f' %(model.coef_[j]))
85         j+=1
86     df_fm=pd.DataFrame()
87     df_fm['Variable']=list_xvar
88     df_fm['Coeficiente']=list_model
89     df_fm.to_csv('file_model.csv',sep=';',decimal=',',encoding='ISO-8859-
90     1')
91
92     return model.coef_,model.intercept_,e,r2_training,r2_test

```

ANEXO IV. Selección de variables con SelectKBest.

- i. Continuando en el Notebook del Anexo III, añadimos el código para lanzar el algoritmo de selectKBest:

```
In [4]: 1 # Proceso de selección de características por algoritmo selectKBest
2 df_Kvar=selectKvar(df_var[1],df_var[2],criteria)
```

```
In [10]: 1 # Valoramos modelo KBest.
2 nm=new_model(list(df_Kvar['Variables']),df_var[1],df_var[2])
3 # Obtenemos en fichero .csv la lista de variables a descartar
4 # en el proceso de análisis
5 var_remove_DL(file_data,list(df_Kvar['Variables']))
```

```
R2 en training: 0.9787
R2 en test: 0.9791
Sobre Ajuste-->> 0.0004
Error cuadrático medio: 5142.41
Error absoluto medio: 55.37
Mediana del error Absoluto: 45.97
```

- ii. Script f_selection.py, con las funciones creadas para el algoritmo descrito en el Notebook

```
24 ## Selección de un número caracterisiticas en función de funciones
25 # función f_regression para modelos de regresión lineal
26 # función chi2 para modelos de clasificación
27
28 def selectKvar(x,y,k):
29     from sklearn.feature_selection import SelectKBest
30     from sklearn.feature_selection import f_regression
31     import numpy as np
32     import pandas as pd
33
34     var_sk = SelectKBest(f_regression, k)
35     x_sk = var_sk.fit_transform(x,y)
36
37     # Obtenemos los f_score de cada variable
38     values = f_regression(x,y)
39     f_score=list(values[0])
40
41     # Obtenemos las k variables con mejor f_score
42     var_s=np.asarray(list(x))[var_sk.get_support()]
43
44     # Obtenemos un data frame con las k mejores variables y sus f_score
45     df_s=pd.DataFrame()
46     df_s['Variables']=var_s
47     df_s['f_score']=f_score[0:k]
48     df_s=df_s.sort_values(by='f_score',ascending=False)
49
50     return df_s
```

```
172 # Función que lanza el nuevo modelo de regresion con las variables seleccionadas
173 def new_model(var_sel,x,y):
174     from ModReg_TG import f_model_reg
175     import pandas as pd
176     x_nm=pd.DataFrame()
177     for i in var_sel:
178         x_nm[i]=x[i]
179     model=f_model_reg(x_nm,y)
180     return model
181
182 # Función para obtener una lista de variables descartadas
183
184 def var_remove_DL(file,var_sel):
185     import pandas as pd
186     df=pd.read_csv(file,sep=';',decimal=',', encoding='ISO-8859-1')
187     l=list(df.columns)
188     ll=[]
189     f=open('var_remove_DL_XXXXXX.csv','w')
190     for i in l:
191         if i not in var_sel:
192             f.write(i+'\n')
193     f.close()
194
195
196
```

ANEXO V. Selección de variables. Método Forward Selection.

- i. Continuando en el Notebook del Anexo III. Lanzamos el algoritmo Forward de agregación de variables.

```
In [6]: 1 # Proceso de selección de características por orden de r2.
2 var_r2= ForwardSelection_r2_Ac(list(df_var[1].columns),
3                               df_var[1],df_var[2])
```

```
In [5]: 1 # Valoramos modelo r2.
2 nm=new_model(var_r2[0:criterio]['Variables'],df_var[1],df_var[2])
3 # Obtenemos en fichero .csv la lista de variables a descartar en el
4 # proceso de análisis
5 var_remove_DL(file_data,var_r2[0:criterio]['Variables'])
6 # Obtenemos un fichero con el modelo matemático de las variables
7 # seleccionadas.
```

```
=====
R2 en training: 0.9905
R2 en test: 0.9906
Sobre Ajuste--->> 0.0001
Error cuadrático medio: 2285.62
Error absoluto medio: 37.54
Mediana del error Absoluto: 32.26
```

- ii. Script f_selection.py, con las funciones creadas para el algoritmo descrito en el Notebook

```
169 def ForwardSelection_r2_Ac(var,x,y):
170     from sklearn.linear_model import LinearRegression
171     import numpy as np
172     import pandas as pd
173     import matplotlib.pyplot as plt
174
175     # Modelo para realizar los ajustes
176     model = LinearRegression()
177
178     # Variable para almacena los índices de la lista de atributos usados
179     var_order = []
180     var_r2 = []
181
182     # Iteración sobre todas las variables
183     for i in range(len(var)):
184         idx_try = [val for val in range(len(var)) if val not in var_order]
185         iter_r2 = []
186
187         for i_try in idx_try:
188             useRow = var_order[:]
189             useRow.append(i_try)
190
191             use = x[x.columns[useRow]]
192
193             model.fit(use, y)
194             r2=model.score(use,y)
195             iter_r2.append(r2)
196
197             pos_best = np.argmax(iter_r2)
198             var_order.append(idx_try[pos_best])
199             var_r2.append(iter_r2[pos_best])
```

```

200
201 # Obtenemos la lista de variables ordenadas por mejor r2
202 var_s=[]
203 for i in range(len(var)):
204     var_s.append(var[var_order[i]])
205 df_s=pd.DataFrame()
206 df_s['Variables']=var_s
207 df_s['r2']=var_r2
208
209 # Grafico en el que representamos r2 vs Variable Añadidas
210 plt.plot(range(len(var)),var_r2,'rx-',label='Error vs Nº de Atributo')
211 plt.title('Selección de Variables')
212 plt.xlabel("Número de atributos")
213 plt.ylabel("r2 Score")
214 plt.savefig("select_r2.png")
215 plt.show()
216
217 return df_s
218

```



```

172 # Función que lanza el nuevo modelo de regresion con las variables seleccionadas
173 def new_model(var_sel,x,y):
174     from ModReg_TG import f_model_reg
175     import pandas as pd
176     x_nm=pd.DataFrame()
177     for i in var_sel:
178         x_nm[i]=x[i]
179     model=f_model_reg(x_nm,y)
180     return model

```



```

283 # Función para obtener una lista de variables descartadas
284
285 def var_remove_DL(file,var_sel):
286     import pandas as pd
287     df=pd.read_csv(file,sep=';',decimal=',', encoding='ISO-8859-1')
288     l=list(df.columns)
289     ll=[]
290     f=open('var_remove_DL_XXXXXX.csv','w')
291     for i in l:
292         if i not in var_sel:
293             f.write(i+'\n')
294     f.close()
295
296

```

ANEXO VI. Obtención de los ficheros de simulación de los datos de día.

- i. Notebook para lanzar el código de obtención de los ficheros de día

```
In [2]: 1 from get_files import crit_date, crit_n
2
3 # Criterios de construcción de los ficheros entre fechas
4 #y cada cierto numero de horas
5 #####
6 date_i = '31/12/2017 23:00'
7 date_f = '30/06/2018 23:00'
8 n = 48 # creamos un fichero cada 48 horas
9 file='TBM_data.csv'
10 file_train='TBM_data.csv'
11 #####
12
13 #df_train=crit_date(date_i,date_f,file)
```

```
In [3]: 1 # Crea ficheros de forma masiva para el entrenamiento cada n horas.
2 crit_n(n,file_train)

===== Se han creado 211 ficheros =====
```

Nota: Estos ficheros se emplearán para verificar, por un lado la validez del modelo de entrenamiento, y por otro, el estado de la máquina.

- ii. Script get_files.py, con la función para la creación de ficheros.

```
32
33 def crit_n(n,file_train):
34     import pandas as pd
35
36     #Obtenemos el dataframe desde el fichero
37     df=pd.read_csv(file_train,sep=';',decimal=',',encoding='ISO-8859-
1',low_memory=False)
38
39     #Criterio de construcción de los ficheros: cada n horas
40     new_row=0
41     file_list=[]
42     for i in range(int(len(df)/n)):
43         df_1=df[new_row:new_row+n]
44         new_row += n
45         df_1=df_1.set_index('Date')
46         f='data_train_'+str(i)+'.csv'
47         file_list.append(f)
48         df_1.to_csv(f,sep=';',decimal=',',encoding='ISO-8859-1')
49     print('===== Se han creado %i ficheros ======%i)
```

ANEXO VII. Código desarrollado para la obtención del modelo entrenado con las variables seleccionadas en el modelo base.

- i. Notebook que lanza las funciones para el entrenamiento, test y almacenamiento del proceso de chequeo diario.

```
In [1]: 1 # Lanzamos el análisis.
2 # Incluimos sólo las 20 variables seleccionadas en el
3 # apartado anterior.
4
5 from ModDL import Data_Analysis, r2_Analysis, mn_Connect
6 l=range(0,211)
7 for i in l:
8     try:
9         test=Data_Analysis(i,'var_remove_DL_Xr2.csv')
10        df_summary=r2_Analysis(i,'var_remove_DL_Xr2.csv',test)
11        mn_Connect('TG_DB','fa_TBM',df_summary)
12    except:
13        print('No hay Suficientes datos')

Datos en modo regulación o limitación. No hay datos suficientes p
ara una buena simulación.
No hay Suficientes datos
*****
Tamaño DataSet: 48 filas de datos y 168 variables registradas
*****
=====
R2 en training: 0.9964
R2 en test: 0.9888
Sobre Ajuste--->>> 0.0077
Error cuadrático medio: 183.90
Error absoluto medio: 11.14
Mediana del error Absoluto: 9.54
-----
Iteraciones realizadas >>> 1
#####
Pot Calculada según Modelo = 14098.807

Potencia Real alcanzada: 14097.75
Desvio Técnico: 0.114 %

Potencia Máxima esperada: 14081.747
#####
```

ii. Funciones del Script ModDL.py para el entrenamiento del modelo y chequeo:

```

1  ### Función Verificación y comprobación de los modelos ###
2
3  # Error entre el modelo entrenado, el modelo del día y el dato promedio real
4
5  def f_test(x,y,reg):
6      import numpy as np
7      import pandas as pd
8
9
10     df_var_mod=pd.read_csv('file_model_r2.csv',sep=';',decimal=',',encoding='ISO
11         -8859-1')
12
13     array=[]
14     l=list(x.columns)
15     for i in l:
16         array.append(np.mean(x[i]))
17     prod=np.dot(reg[0],array)+reg[1]
18     suma=0
19
20     for i in range(len(df_var_mod)-1):
21         suma+=np.mean(x[list(df_var_mod.loc[i])
22 [0]])*list(df_var_mod.loc[i])[1]
23     T1 = suma+df_var_mod[df_var_mod['Variable']=='bias']['Coeficiente']
24
25     print('#'*50,'nPot Calculada según Modelo = %0.3f'% prod)
26     print('nPotencia Real alcanzada:',np.mean(y))
27     print('Desvio Técnico: %.3f'%.((np.mean(y)/T1-1)*100),'%')
28     print('nPotencia Máxima esperada: %.3f'%(T1))
29     print('#'*50)
30     res_t=['Pot Calculada según Modelo','Potencia Real alcanzada','Desvio
31 Técnica(%),'Potencia Máxima esperada']
32     res_n=[prod,np.mean(y),float(np.mean(y)/T1-1)*100,float(T1)]
33
34     return float(np.mean(y)/T1-1)*100

```

```

32
33 # Función para la obtención de los ficheros con los resultados de los
34 # análisis.
35
36 def Data_Analysis(i,var_remove):
37
38     # Importamos funciones auxiliares creadas en los Scripts
39     from ModReg_TG import imp_ds, Fit_Var, sense_var, f_model_reg, plot_var,
40     f_score
41     from f_selection import ForwardSelection_r2, ForwardSelection_r2_Ac
42     from ModDL import f_test
43     from datetime import datetime
44     import pandas as pd
45     import shutil
46
47     file_data = 'data_train_'+str(i)+'.csv'
48     #Importamos el dataset con todas las variables de la máquina objetivo de
49     #estudio
50     file=imp_ds(file_data)
51     # devuelve lista de variable, el dataframe completo, y fecha de los datos
52     #sometidos a análisis.
53     #Función que elimina las variables no numéricas y establece la variable
54     #dependiente
55     df_var=Fit_Var(file,var_remove)
56     # devuelve lista de variables tratadas, la X y la Y para el modelo de
57     #entrenamiento
58
59     #Obtención de los primeros estadísticos.
60     #Ejecutamos una función que devuelve el valor medio y la desviación std
61     stat=sense_var(df_var[1],df_var[2])
62     # devuelve lista de variables y lista con las medias y lista con las
63     #descv std.
64
65     #Una vez que el dataset ha sido tratado, se lanza la función de regresión
66     #para el Modelo
67     #Se lanza tantas veces como sea necesario para asegurar que no haya
68     #sobreajuste
69     e=0.1
70     i=0
71     while (e >0.05 and i <10):
72         reg=f_model_reg(df_var[1],df_var[2])
73         e=reg[2]
74         if i >10:
75             print('Modelo Presenta Sobre ajuste.')
76             i+=1
77         print('-----\nIteraciones realizadas >>',i)
78         #devuelve una lista con los coeficientes de la regresion, el bias de la
79         #regresion y la diferencia entre el r2 del train y el test para valorar un
80         #posible sobre ajuste
81
82         # Función de selección de características por orden decreciente de r2
83         var_s =
84         ForwardSelection_r2_Ac(list(df_var[1].columns),df_var[1],df_var[2])
85         var_r2 = ForwardSelection_r2(df_var)
86         #La función nos devuelve la lista de variables ordenadas por r2 y la
87         #lista con los r2
88
89         # Verificamos el rendimiento del modelo con los datos reales
90         test = f_test(df_var[1],df_var[2],reg)
91
92         return test
93
94 # Función de análisis de sensibilidad de r2
95
96

```

```

83 def r2_Analysis(i,var_remove,test):
84
85     # Importamos funciones auxiliares creadas en los Scripts
86     from ModReg_TG import imp_ds, Fit_Var, sense_var, f_model_reg, plot_var,
87     f_score
88     from f_selection import ForwardSelection_r2, ForwardSelection_r2_Ac
89     from ModDL import f_test
90     from datetime import datetime
91     import pandas as pd
92     import shutil
93
94     file_data = 'data_train_'+str(i)+'.csv'
95     #Importamos el dataset con todas las variables de la máquina objetivo de
96     estudio
97     file=imp_ds(file_data)
98     # devuelve lista de variable, el dataframe completo, y fecha de los datos
99     #sometidos a análisis.
100    #Función que elimina las variables no numéricas y establece la variable
101    dependiente
102    df_var=Fit_Var(file,var_remove)
103    # devuelve lista de variables tratadas, la X y la Y para el modelo de
104    #entrenamiento
105
106    #Obtención de los primeros estadísticos.
107    #Ejecutamos una función que devuelve el valor medio y la desviación std
108    stat=sense_var(df_var[1],df_var[2])
109    # devuelve lista de variables y lista con las medias y lista con las
110    #descv std.
111
112    #Una vez que el dataset ha sido tratado, se lanza la función de regresión
113    #para el Modelo
114    #Se lanza tantas veces como sea necesario para asegurar que no haya
115    #sobreajuste
116    e=0.1
117    i=0
118    while (e >0.05 and i <10):
119        reg=f_model_reg(df_var[1],df_var[2])
120        e=reg[2]
121        if i >10:
122            print('Modelo Presenta Sobre ajuste.')
123            i+=1
124        print('-----\nIteraciones realizadas >>>',i)
125        #devuelve una lista con los coeficientes de la regresion, el bias de la
126        #regresion y la diferencia entre el r2 del train y el test para valorar un
127        #posible sobre ajuste
128
129        ### Creamos un Data Frame Resumen con los Estadísticos principales.
130        ## Con ellos realizaremos el análisis de sensibilidad para el
131        #diagnóstico del sistema
132        df_summary=pd.DataFrame()
133
134        # Creamos la lista con todas las variables
135        Variable=list(stat.index)
136
137        # Creamos la lista con los coeficientes del modelo
138        Coef=[0]
139        for i in list(reg[0]):
140            Coef.append(i)

```

```

131 # Creamos la lista con los valores promedios
132 Media=list(stat['Media'])
133
134 # Creamos la lista con los valores de desviación Std.
135 Dev_Std=list(stat['Std Dev'])
136
137 # Creamos la lista con los valores de r2
138 r2_score=[reg[1]]
139 r2_list=f_score(df_var[1],df_var[2],list(df_var[1].columns))
140 for i in r2_list:
141     r2_score.append(i)
142
143 # Creamos el campo fecha con formato
144 fecha_f=datetime.strptime(file[2].strip(' \t\r\n'), '%d/%m/%y')
145 fecha_fs=datetime.strftime(fecha_f, '%d/%m/%y')
146
147 list_fecha=[fecha_fs]*len(Variable)
148
149 # Creamos el dataframe
150 df_summary['Fecha']=list_fecha
151 df_summary['Variable']=Variable
152 df_summary['r2_score']=r2_score
153 df_summary['Valor Medio']=Media
154 df_summary['Std Dev']=Dev_Std
155 df_summary['Coef Model']=Coef
156 df_summary.loc[len(Variable)+1]=[fecha_fs,'Desvio Tecnico',0,test,0,0]
157 #####df_summary.loc[16]=[fecha_fs,'Desvio CC Amb',0,test[0][3],0,0]
158 df_summary.loc[len(Variable)+2]=[fecha_fs,'r2 Train',reg[3],0,0,0]
159 df_summary.loc[len(Variable)+3]=[fecha_fs,'r2 Test',reg[4],0,0,0]
160 print(df_summary.sort_values(by='r2_score',ascending=False))
161
162 # Creamos un fichero con los datos del resumen
163 f='fa_'+file[2].replace('/','-')+'.csv'
164 df_summary.to_csv(f,sep=';',decimal=',',encoding='ISO-8859-1')
165
166 # Movemos el fichero creado a la carpeta Salida_Analisis
167 ruta_origen = '/Users/jpgmacbookpro/Google Drive/Master Big Data/TRABAJO
FIN DE MASTER/Codigo_TFM/TBM DATA/'+f
168 ruta_destino = '/Users/jpgmacbookpro/Google Drive/Master Big Data/TRABAJO
FIN DE MASTER/Codigo_TFM/TBM DATA/Salida_Analisis/'+f
169 ruta=shutil.move(ruta_origen, ruta_destino)
170 print('*75')
171 print("Archivo copiado a",ruta)
172 print('*75')
173
174 return df_summary

```

ANEXO VIII. Almacenamiento y visualización.

```
177 # Función para el volcado de datos de análisis en MongoDB
178
179 def mn_Connect(DB,col_name,df):
180     import pymongo, json
181     # Habilitamos la conexión a la Base Datos MongoDB
182     mng_client = pymongo.MongoClient('localhost', 27017)
183
184     # Direccionamos a la bases y colecciones de MongoDB
185     mng_db=mng_client[DB]
186     collection_name=col_name
187     db_cm=mng_db[collection_name]
188
189     # Creamos los objetos json para exportar a Mongo
190     data_json = json.loads(df.to_json(orient='records'))
191     db_cm.insert_many(data_json)
192
```

ANEXO IX. Código de cálculo iterativo de la tempera de bulbo húmedo.

```
1  ### Función de cálculo de la Temperatura de Bulbo húmedo ###
2
3  def TempBH(Tamb,HR,Patm):
4
5      import math
6      # Lista de Constantes del método de cálculo de Carrier.
7      A = 1.2378847e-5
8      B = -1.9121316e-2
9      C = 33.93711047
10     D = -6343.1645
11     THETA = 1940
12     JI = -1.44
13
14     # Método de Carrier
15     e = 0 # Variable comparativa del error en cada iteraciones
16
17     Ps = math.exp(A * (Tamb + 273.15) ** 2 + B * (Tamb + 273.15) + C + D / (Tamb +
18     273.15))
19     P = Ps * HR
20     Tbh0 = 0
21     Tbh = []
22     i = 0 # Contador de iteraciones
23     Tbh.append(Tbh0)
24     # Iteramos hasta que el error entre ambos término sea de 0.01
25     while (e < 0.99):
26         Psw = math.exp(A * (Tbh[i] + 273.15) ** 2 + B *
27             (Tbh[i] + 273.15) + C + D / (Tbh[i] + 273.15))
28         Pw = Psw - ((Patm * 100 - Psw) * (Tamb - Tbh[i])) / (THETA + JI * (Tbh[i] +
29             273.15))
30         T = Tbh[i] + 0.1
31         i = i + 1
32         e = Pw / P
33         Tbh.append(T)
34     print("La Temperatura de bulbo humedo es: %.1f°C en %d iteraciones realizadas" %
35     (T,i))
36
37     return T
```

ANEXO X. Código para la obtención de un modelo basado en las condiciones meteorológicas.

- i. Notebook con las funciones de carga, adecuación y entrenamiento de datos

Obtención de un modelo basado en condiciones meteorológicas.

```
In [3]: 1 from ModReg_TG import *
2 # Cargamos el dataset con todas las variables de la máquina
3 #objetivo de estudio
4 file=imp_ds('data_model_1.csv')
5 # devuelve lista de variable y el dataframe completo
6
7 #Función que elimina las variables no numéricas y establece
8 #la variable dependiente
9 df_var=Fit_Var(file,'Potencia TG ValueY')
10 # devuelve lista de variables tratadas, la X y la Y para el mo
11 # de entrenamiento
12
13 #Obtención de los primeros estadísticos.
14 #Ejecutamos una función que devuelve el valor medio
15 #y la desviación std
16 stat=sense_var(df_var[1],df_var[2])
17 # devuelve lista de variables y lista con las medias y
18 #lista con las descv std.
19 #Una vez que el dataset ha sido tratado, se lanza la función
20 #de regresión para el Modelo
21 #Se lanza tantas veces como sea necesario para asegurar que
22 #no haya sobreajuste
23 e=0.1
24 i=0
25 while (e >0.05 and i <10):
26     reg=f_model_reg(df_var[1],df_var[2])
27     e=reg[2]
28     if i >10:
29         print('Modelo Presenta Sobre ajuste.')
30         i+=1
31     print('-----\nIteraciones realizadas >>>',i)
32     # devuelve una lista con los coeficientes de la regresion,
33     # el bias de la regresion y la diferencia entre el r2
34     #del train y el test
```

```
*****
Tamaño DataSet: 626 filas de datos y 4 variables registradas
*****
=====
R2 en training: 0.7469
R2 en test: 0.7881
Sobre Ajuste--->>> 0.0551
Error caudratico medio: 15943.45
Error absoluto medio: 84.17
Mediana del error Absoluto: 59.84
=====
=====
R2 en training: 0.7818
R2 en test: 0.6656
Sobre Ajuste--->>> 0.1487
Error caudratico medio: 14803.55
Error absoluto medio: 80.90
Mediana del error Absoluto: 55.88
=====
=====
R2 en training: 0.7394
R2 en test: 0.8190
Sobre Ajuste--->>> 0.1076
Error caudratico medio: 16984.59
Error absoluto medio: 86.81
Mediana del error Absoluto: 63.39
=====
=====
R2 en training: 0.7636
R2 en test: 0.7437
Sobre Ajuste--->>> 0.0260
Error caudratico medio: 14806.83
Error absoluto medio: 81.07
Mediana del error Absoluto: 59.05
-----
```

ii. Script “*ModReg_TG.py*” con el código de las funciones empleadas en el Notebook

```
1 #####  
2 # SCRIPT DE FUNCIONES PARA EL TRATAMIENTO, CARGA Y ENTRENAMIENTO DE DATOS  
3 #####  
4  
5 # Función para carga del fichero con dataset con todas las variables de la  
máquina objetivo de estudio  
6 def imp_ds(file):  
7     import pandas as pd  
8     # Cargamos el fichero en un dataframe  
9     df=pd.read_csv(file,sep=';',decimal=',', encoding='ISO-8859-  
1',low_memory=False)  
10    # Eliminamos las filas con valores nulos o períodos  
11    df=df.dropna()  
12  
13    # Condición lógica que descarta aquellos valores de potencia que están  
bajo procesos de regulación o limitación.  
14    #df=df[df['Active power']+200 < df['Power setpoint']]  
15    df=df[df['Potencia TG ValueY']>13000]  
16    if df.shape[0] > 24:  
17        # Creamos un fichero con la lista de todas las variables  
18        var=list(df.columns)  
19        f=open('file_var.csv','w')  
20        for i in var:  
21            f.write(i+'\n')  
22        f.close()  
23        print('*'*60)  
24        print('Tamaño DataSet:',df.shape[0], 'filas de datos y ',  
df.shape[1], 'variables registradas')  
25        print('*'*60)  
26    else:  
27        print('Datos en modo regulación o limitación. No hay datos  
suficientes para una buena simulación.')  
28  
29    return var,df  
30
```

```

30
31 #Función Modelo de Regresión Lineal
32 def f_model_reg(x,y):
33     #Entrenamos el modelos Mediante Regresión Lineal Multivariable
34     import numpy as np
35     import pandas as pd
36     from sklearn.model_selection import train_test_split
37     from sklearn.linear_model import LinearRegression
38     from sklearn.metrics import mean_squared_error
39     from sklearn.metrics import mean_absolute_error
40     from sklearn.metrics import median_absolute_error
41
42     #Generamos las variables de entrenamiento y test
43     x_train,x_test,y_train,y_test=train_test_split(x,y)
44
45     #Modelo de Regresión Lineal
46     model=LinearRegression(fit_intercept=True)
47     model.fit(x_train,y_train)
48     predict_train=model.predict(x_train)
49     predict_test=model.predict(x_test)
50
51     #Evaluamos los resultados de la Regresión
52     r2_training = model.score(x_train,y_train)
53     r2_test = model.score(x_test,y_test)
54     print('*70)
55     print('R2 en training: %.4f' %(model.score(x_train,y_train)))
56     print('R2 en test: %.4f' %(model.score(x_test,y_test)))
57     e=abs(model.score(x_train,y_train)-
58           model.score(x_test,y_test))/model.score(x_train,y_train)
59     print('Sobre Ajuste--->> %.4f' %(e))
60     #Determinación de los estimadores de la calidad del modelo.
61     print("Error cuadrático medio: %.2f" %
62           (mean_squared_error(predict_train,y_train)))
63     print("Error absoluto medio: %.2f" %
64           (mean_absolute_error(predict_train,y_train)))
65     print("Mediana del error Absoluto: %.2f" %
66           (median_absolute_error(predict_train,y_train)))
67
68     #Obtenemos los coeficientes de la regresión y los llevamos a un fichero
69     list_model=[]
70     list_xvar=[]
71     #print('Termino Ind: %.2f' % (model.intercept_))
72     j=0
73     for i in list(x.columns):
74         list_xvar.append(i)
75         list_model.append(model.coef_[j])
76         #print(i,'-- %.2f' %(model.coef_[j]))
77         j+=1
78     list_xvar.append('bias')
79     list_model.append(model.intercept_)
80     df_fm=pd.DataFrame()
81     df_fm['Variable']=list_xvar
82     df_fm['Coeficiente']=list_model
83     df_fm=df_fm.set_index('Variable')
84     df_fm.to_csv('file_model.csv',sep=';',decimal=',',encoding='ISO-8859-1')

return model.coef_,model.intercept_,e,r2_training,r2_test

```

```
132 #Función que elimina las variables no numéricas y establece la variable
133 dependiente
134 def Fit_Var(file,var_remove):
135     import pandas as pd
136     var=file[0]
137     df=file[1]
138     #y=input('Escribe el nombre de la variables dependiente >>> ')
139     y=var_remove
140     var.remove(var_remove)
141     x=df[var].astype(float)
142     y=df[y].astype(float)
143     return var,x,y
144 #Función que nos devuelve un dataframe con los valores medios y desviación
145 estandar de cada variable
146 def sense_var(x,y):
147     import numpy as np
148     import pandas as pd
149     var_n=['Active power']
150     for i in list(x.columns):
151         var_n.append(i)
152     val_mean=[np.mean(y)]
153     val_std=[np.std(y)/np.mean(y)*100]
154     #Creamos las tres listas con las variables y sus estadisticos
155     j=0
156
157     for i in list(x.columns):
158         val_mean.append(np.mean(x[i]))
159         val_std.append(np.std(x[i])/np.mean(y)*100)
160         j+=1
161     df = pd.DataFrame()
162     df['Feat']=var_n
163     df['Media']=val_mean
164     df['Std Dev']=val_std
165     df=df.set_index('Feat')
166
167     return df
```

ANEXO XI. Series Temporales de las Condiciones Ambientales.

- i. Código para carga de fichero con los datos de la serie temporal.

Series Temporales. Análisis de la evolución de las condiciones temporales a lo largo del tiempo.

```
In [1]: 1 # Descarga del fichero de trabajo
2 import pandas as pd
3 df=pd.read_csv('ds_data.csv',sep=';',decimal=',', encoding='ISO-8859-1')

In [2]: 1 # Creamos una función para transformar las fechas
2 # a formato válido para trabajar con series temporales
3
4 # Inputs de la función:
5 # lista --> Campo de la Fecha en el fichero
6 # formato --> formato de la fecha que posee el fichero
7 # inter --> nº de caracteres de cola de la fecha que no se van
8 # a compilar. En este caso eliminamos sg y mins: -6
9
10 def date_f(lista,formato,inter):
11     from datetime import datetime
12     date=[]
13     for i in lista:
14         date.append(datetime.strptime(i[:inter], formato))
15     return date

In [3]: 1 # Obtenemos el Data Frame preparado para trabajar con series temporales
2 dates=date_f(df['Fecha'],'%d/%m/%y %H',-3)
3 df['Fecha']=dates
4 df=df.set_index('Fecha')

In [4]: 1 df.head()

Out[4]:
Temperatura    Presión    Temperatura    Potencia    Temperatura    Temperatura
Bulbo        Húmedo    Atmosférica    T1 ValueY    TG ValueY    salida enfriador    Ambiente Value Y
ValueY          ValueY      ValueY          ValueY          ValueY          ValueY          ValueY          ValueY
Fecha
2018-04-25 00:00:00  14.200018  966.851880  15.198830  13567.42861  13.87  14.48
2018-04-25 01:00:00  13.800016  966.574085  14.761036  13594.07813  13.52  13.95
2018-04-25 02:00:00  13.700016  966.203687  14.469173  13620.72686  13.15  13.46
2018-04-25 03:00:00  13.500015  966.111096  14.177307  13680.44121  12.89  13.54
2018-04-25 04:00:00  12.900013  965.648132  13.958408  13681.42842  12.56  13.17
```

ii. Código para la representación gráfica.

Representaciones Graficas

```
In [30]: 1 # Esta celda recoge los inputs de datos al grafico
2 #Input para el eje principal
3 input1=df[['Temperatura Ambiente Value Y','Temperatura Bulbo Hume
4 #Input para el eje secundario
5 input2=df['Presion Atmosferica ValueY']['2018-05-01':'2018-05-07']

In [31]: 1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 fig=plt.figure(figsize=(12,10))
5 ax=fig.add_subplot(111)
6 lt1 = ax.plot(input1,label=['Tamb','TBh'])
7 ax2=ax.twinx()
8 lt2 = ax2.plot(input2,'r',label='Patm')
9 lts=lt1+lt2
10 labs=[i.get_label() for i in lts]
11 ax.legend(lts,labs,loc=0)
12 #ax2.legend(loc=0)
13 ax.grid()
14 ax.set_xlabel('Fecha')
15 ax.set_ylabel('Temperatura ºC')
16 ax2.set_ylabel('P. Atm (mBar)')
17 ax2.set_ylim(900, 1000)
```