Proyecto Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Técnicas de predicción aplicadas al mercado español de la energía eléctrica

Autor: Albert Jakob Liñán Maho
Tutor: Teodoro Álamo Cantarero

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Grado
Grado en Ingeniería en las Tecnologías Industriales

Técnicas de predicción aplicadas al mercado español de la energía eléctrica

Autor:
Albert Jakob Liñán Maho

Tutor:
Teodoro Álamo Cantarero
Catedrático de Universidad

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo fin de Grado: Técnicas de predicción aplicadas al mercado español de la energía eléctrica

Autor: Albert Jakob Liñán Maho

Tutor: Teodoro Álamo Cantarero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El secretario del Tribunal:

Fecha:

A mi familia

A mis amigos

Agradecimientos

Agradezco a mi hermana, la persona más importante de mi mundo, no sería el mismo sin ella; a mis padres y mi tata por la educación que he recibido, apoyarme incondicionalmente, confiar siempre en mí y dar todo por mi hermana y por mí. También agradecer al resto de la familia por transmitirme muchos ánimos a lo largo de toda la carrera y ayudarme siempre que lo he necesitado.

Agradecer tambien a todos los buenos amigos, que son mi otra familia y con los que he vivido los mejores momentos de mi vida.

Finalmente agradecer a mi profesor por la buena atención que he recibido en todo momento y por guiarme en el desarrollo del proyecto.

Resumen

El presente documento presenta un estudio sobre el mercado diario español de la energía eléctrica mediante técnicas de *machine learning.* Para este estudio se han utilizado *mínimos cuadrados, Random Forest, Máquinas de soporte vectorial y Clustering.* Para realizar el estudio se han probado cada una de las técnicas en tres conjuntos de datos diferentes del año 2015.

Para programar los algoritmos para predecir el precio de la energía se han utilizado Matlab y Python. Se han desarrollado una serie de librerías en dichos lenguajes.

Los objetivos del proyecto son el estudio de todos estos métodos y su aplicación al mercado eléctrico.

Abstract

This document presents a study on the Spanish daily market electricity through *machine learning techniques*. For this study were used *least squares, Random Forest, Support Vector Machines and Clustering*. To conduct the study were tested each of the techniques in three different data sets 2015.

To program algorithms to predict the price of energy have been used Matlab and Python

The project objectives are the study of all these methods and their application to the electricity market.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

M.L	Machine Learning
RF	Random Forest
SVM	Máquinas de soporte vectorial
SVR	Máquinas de soporte vectorial para regresión

1 INTRODUCCIÓN

A lo largo de todo el grado se han ido abarcando muchas doctrinas dentro del campo de la ingeniería. De esta manera decidí buscar una nueva doctrina a priori desconocida. Elegí el aprendizaje automático o machine learning, ya que además de parecerme bastante interesante, es un tema actual y con mucha aplicación.

Se ha decidido hacer un estudio mediante regresiones del precio de la energía.El precio se establece todos los dias para las 24 horas del dia siguiente (mercado diario). El objetivo de nuestro proyecto es predecirlo con distintas técnicas y comparar las prediccines obtenidas; asi como conocer los fundamentos teóricos de cada una de las técnicas utilizadas.

1.1 Organización memoria

En el capítulo 2 se explica el funcionamiento del mercado de la energía eléctrica y los factores que afectan al precio. También se presentan los errores para valorar las predicciones y las variables que vamos a utilizar para intentar predecir el precio. Finalmente se exponen los objetivos alcanzados.

En el capitulo 3 se explica qué es el aprendizaje automático. Se explican sus tipos: apredizaje supervisado y no supervisado.Y al final del capítulo se explica la regresión.

Los capítulos 4, 5, 6 y 7 presentan respectivamente los métodos de mínimos cuadrados, RF, SVM y Clustering. Cada uno de ellos explica teóricamente el método, como se ha programado y los resultados que se han obtenido mediante tablas de errores y gráficas.

El capítulo 8 es una comparación de los métodos que se han utilizado utilizando histogramas del error. Se presentan las conclusiones obtenidas y se explican las posibles líneas futuras.

El capítulo 9 recoge los códigos de los algoritmos programados en Matlab y Python.

2 DESCRIPCIÓN DEL PROBLEMA Y OBJETIVOS

2.1 Mercado de la energia eléctrica

La electricidad tiene una gran importancia en nuestras vidas. La Agencia Internacional de la Energía aboga por un futuro más eléctrico, debido a las nuevas aplicaciones de la misma, que entre otras ventajas, permiten reducir la contaminación. El precio de la misma es tan importante para consumidores domésticos como empresas e industrias.

Con el objetivo de hacer llegar a los ciudadanos y a las empresas los beneficios de la liberalización del sector eléctrico, a mediados de los 90 se lanzó la construcción del Mercado Interior de la Electricidad en la UE. Así se crearon mercados organizados en las distintas zonas de Europa. En nuestro caso, la OMIE gestiona el mercado de la Península Ibérica (desde julio 2007, empezó en enero de 1998 con el mercado español).

2.1.1 Mercado diario

En Europa el precio de la electricidad se establece diariamente a las 12:00 horas para las 24 horas del día siguiente; a esto lo llamamos Mercado Diario. El precio y la cantidad de energía en cada una de las horas se obtienen del cruce entre la oferta y la demanda. Este modelo, denominado marginalista es el adoptado por toda la UE y basa su funcionamiento en ul algoritmo EUPHEMIA.

Los agentes compradores y vendedores pueden acudir a nuestro mercado sin depender de que estén en España o Portugal. Las ofertas de compra y venta se van atendiendo hasta que la interconexión entre España y Portugal se ocupe totalmente. Si en una hora del día, la capacidad de interconexión es suficiente, el precio será el mismo para los dos países; por lo contrario, si esta interconexión entre países está totalmente ocupada, los precios serán distintos para cada país. El año 2014 el 90% del tiempo el precio de la electricidad fue el mismo para España y Portugal, lo que demuestra un correcto funcionamiento de la interconexión en el mercado ibérico.

Los resultados obtenidos a partir de la libre contratación entre agentes compradores y vendedores constituyen la mejor solución económica, pero esta solución también tiene que ser viable físicamente. Una vez obtenidos los resultados, estos se remiten al operador del Sistema para su validación técnica; así aseguramos que los resultados sean fiables en la red de transporte. Los resultados del mercado sufren leves variaciones, del orden de 4-5% de la energía.

La función del mercado diario es llevar a cabo las transacciones de energía eléctrica para el día siguiente por medio de ofertas de venta y adquisición. Los vendedores de este mercado están obligados a ceñirse a las Reglas de Funcionamiento del Mercado de Producción de Energía Eléctrica. Las ofertas de cada agente se presentan al operador de mercado y se incluye en el procedimiento de casación.

Los compradores en este mercado son los comercializadores, los consumidores directos y los comercializadores de referencia. La oferta y la demanda se realizarán teniendo en cuenta los 24 tramos de cada día. El precio será creciente en el tramo de ventas y decreciente en el de las compras.

El algoritmo Euphemia utiliza curvas agregadas en escalón. Existe una curva de oferta y otra de demanda, el punto donde estas dos cortan se denomina precio de casación(o marginal). La precisión establecida para el mercado ibérico es de dos decimales para los precios (€/MWh) y de un decimal para las energías (MWh)

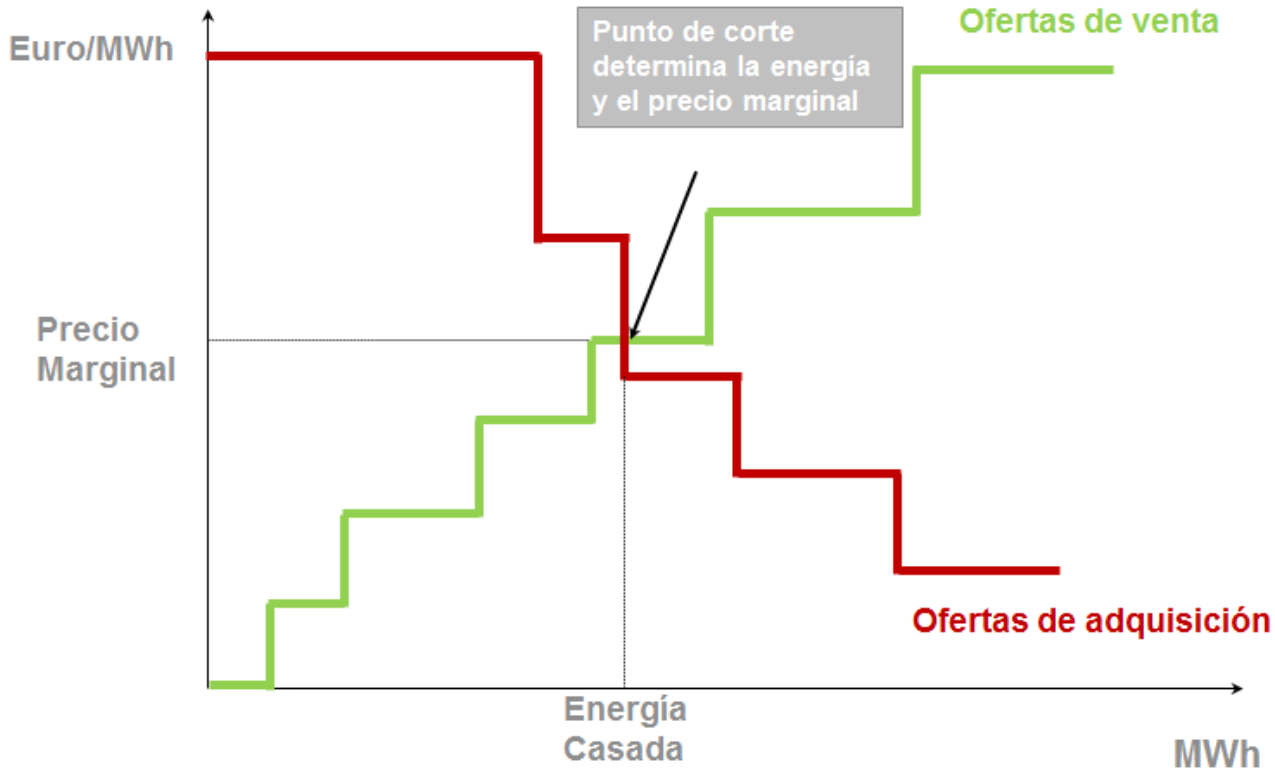


Figura 2-1. Energía casada.

2.2 Selección de datos

Debido a la volatilidad del precio de casación, poder prever este precio se hace muy importante para las comercializadoras. Hay diversos factores que afectan al precio de la energía eléctrica: demanda, climatología (generación de energías renovables), estación del año, día de la semana, días festivos...

Esta capacidad de previsión supondrá siempre una ventaja frente al resto de competidores; por esto, la utilización y desarrollo de estos modelos predictivos está en auge. Este tipo de modelos se respaldan en la matemática y la estadística.

Para realizar las predicciones es necesario determinar la calidad de las mismas. Esta calidad nos vendrá dada por varias medidas del error de predicción. Se utilizaran las siguientes:

- MAE (mean absolute error o error medio absoluto).

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t|$$

Donde *et=ft-yt,* siendo *yt* el valor real y *ft* el valor predicho.

- MAPE (mean absolute percentage error o error porcentual absoluto medio). Es un indicador que mide el tamaño del error absoluto en términos de porcentajes:

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right|$$

- RMSE (root mean square error o error medio cuadrático). Este error nos da la medida de las diferencias en promedio entre los valores pronosticados:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}{e_t}^2}$$

Para la elección de las variables independientes que nos ayudaran a predecir el valor de la energía. Analizaremos un poco las curvas de oferta y demanda del mercado.

- Curva de oferta: está formada por las ofertas de venta por parte de las unidades de generación. Estas unidades producen energía eléctrica de diversas formas: aerogeneradores (eólica), ciclos combinados, reactores nucleares,... Dependiendo de la tecnología utilizada para producir la energía, el coste para producir cada MWh irá cambiando. Así las ofertas irán desde precio cero hasta aquellas que lo hagan al precio máximo:

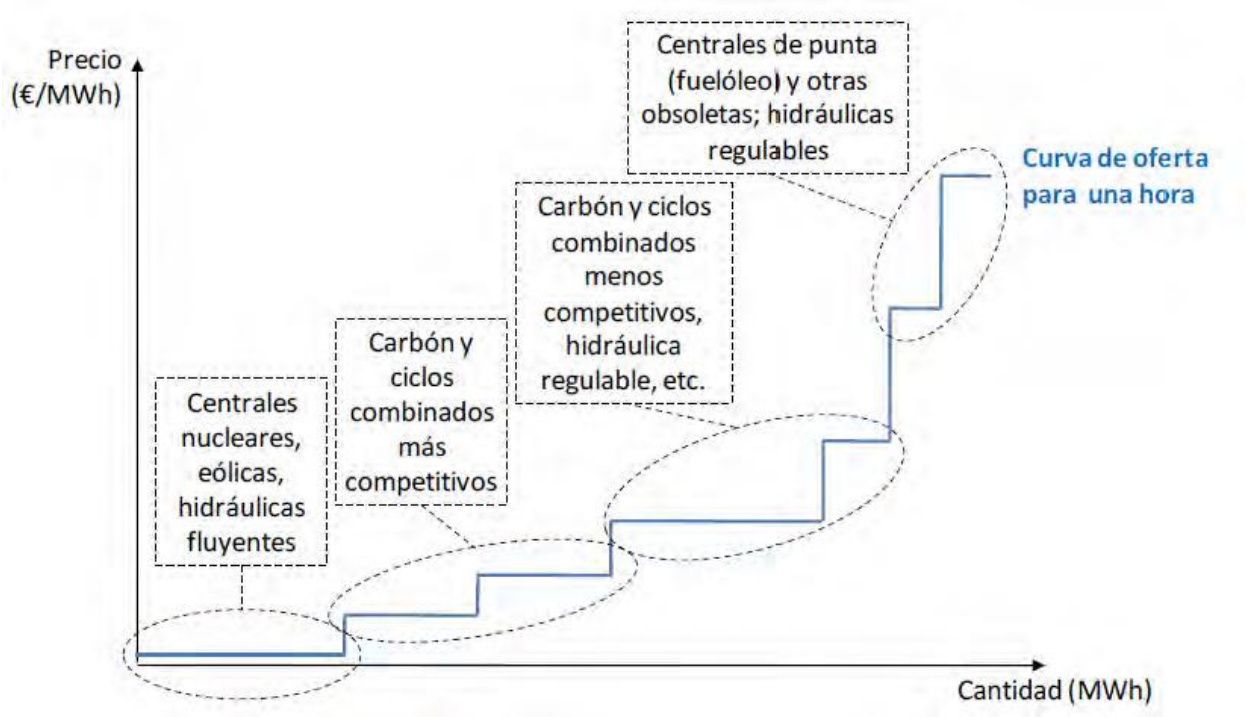


Figura 2-2. Curva oferta.

Es importante destacar de la figura que los precios más bajos son los que ofrecen las renovables y la nuclear. Como la capacidad de generación de las renovables está limitada por la climatología, se ofrece siempre a precio cero para poder asegurar su venta. Por otra parte la energía nuclear necesita asegurar continuidad (debido al alto coste de energía que resulta encender el reactor) por lo que ofertará a bajo precio también.

- Curva de demanda: el sistema intenta cubrir la demanda de todos los consumidores. Generación y consumo han de ser iguales en todo momento. Normalmente comercializadoras las que compran la energía para sus clientes, pero también existen consumidores propios. Estos consumidores propios tienen que ceñirse a una serie de requisitos para poder participar en el mercado. A diferencia que la energía demandada, el precio máximo al que se puede pagar el MWh está fijado en 180€, siendo el mínimo 0€.

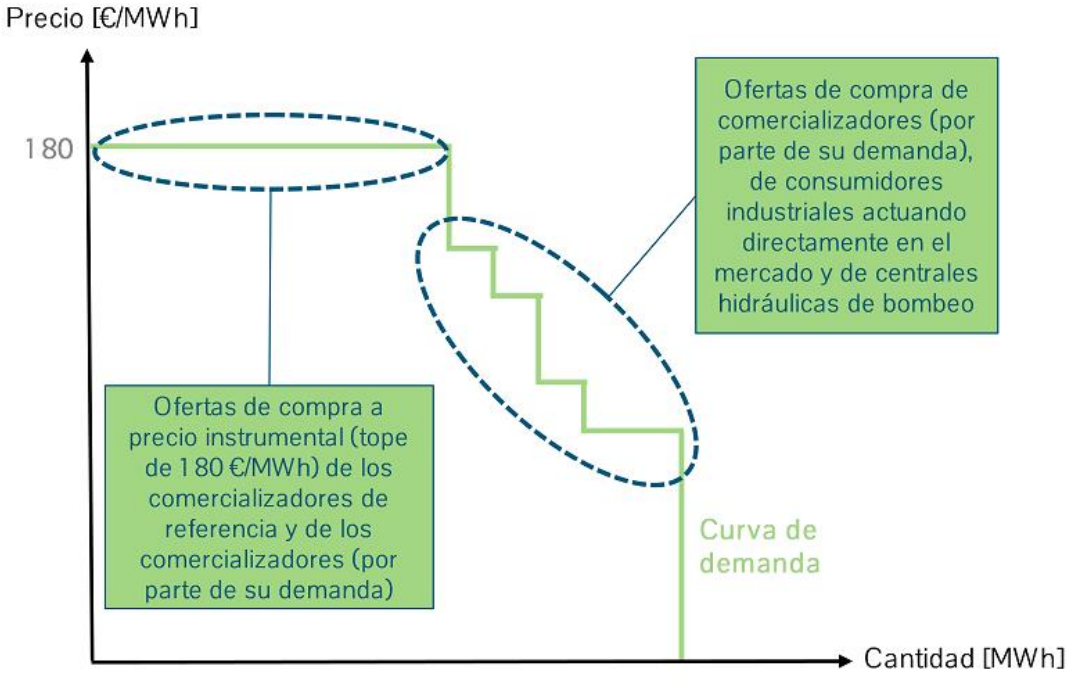


Figura 2-3. Curva de demanda.

La demanda sufre cambios intradiarios ya que dependiendo de la hora del día, por ejemplo, la demanda será superior para las primeras horas de la mañana (donde se inicia la actividad industrial) que para la noche (donde la actividad industrial es claramente inferior). Estas variaciones se producen también debidas a la estación del año (climatología) y también dependen de si el día de la semana es laborable o festivo.

Para predecir el precio de la energía en la hora h, finalmente se han tomado solo 4 variables que son lo suficientemente significativas para obtener una buena predicción y además simplifican el modelo evitando complicaciones y altos tiempos de ejecución. Las variables son:

- Producción de energía eólica en la hora h
- Precio de la electricidad el día anterior en la hora h
- Precio de la electricidad la semana anterior en la hora h
- Demanda eléctrica prevista en la hora h

Todos los históricos de estas variables se pueden descargar en la página www.esios.ree.es . Para el estudio se han tomado todos los datos del año 2015(desde 00:00h del día 01-01-2015 hasta las 23:50h del día 31-12-2015).

Se han decidido 3 casos diferentes para ser estudiados por los métodos de predicción citados. Las fechas para

los conjuntos de datos son las siguientes:

	Conjunto de entrenamiento	Conjunto de prueba
Caso1	10/01/2015 a 20/01/2015	20/12/2015 a 30/12/2015
Caso2	10/01/2015 a 09/02/2015	01/12/2015 a 31/12/2015
Caso3	10/01/2015 a 17/01/2015	18/01/2015 a 25/01/2015

Tabla 2–1. Conjuntos datos

2.3 Objetivos

.Los objetivos se muestran en orden cronológico, tal y como se han ido alcanzando:

1. Adquisición de conocimientos sobre *Machine Learning* y búsqueda de un problema de interés para aplicar nuestro estudio.
2. Tratamiento de datos. Los datos se obtienen directamente de la página de www.esios.ree.es y se descargan en Excel.
3. Decidido el estudio sobre el precio de la energía eléctrica se empezaron a realizar regresiones con *mínimos cuadrados* en Matlab. Partiendo de una variable predictora, se fueron eligiendo más hasta llegar a cuatro.
4. Ampliación de herramientas de programación. Se deciden realizar también regresiones mediante *Random Forest y Máquinas de soporte vectorial* con Python. Finalmente se introduce un método de clustering (kmeans) que intentara mejorar aún más la predicción.
5. Programados los resultados nos centramos en 3 casos específicos para el estudio dentro del marco del precio en 2015 y se realizan las simulaciones. Se analizan los resultados y se comparan los métodos entre ellos obteniendo las conclusiones del estudio. También se exponen las posibles aplicaciones o mejoras futuRAS.

3 APRENDIZAJE AUTOMÁTICO

3.1 Concepto machine learning (o aprendizaje automático)

El *Machine learning* o *aprendizaje automático* es una rama de la inteligencia artificial cuyo objetivo es programar algoritmos capaces de aprender o generalizar comportamientos a partir de información previa obtenida de ejemplos. El objetivo principal es utilizar la evidencia conocida, creando así una hipótesis y pudiendo así responder ante situaciones desconocidas a priori. El aprendizaje automático se centra en el estudio de la complejidad computacional de los problemas. Podríamos decir que esta disciplina intenta automatizar algunas partes del método científico mediante las matemáticas.

Hoy en día el aprendizaje automático tiene diversas aplicaciones:

- Procesamiento del lenguaje natural: análisis morfológico, sintáctico y semántico de textos.
- Sistemas de recuperación de información: los motores de búsqueda utilizan técnicas ML para confeccionar búsquedas personalizadas para cada perfil.
- Diagnóstico médico: hay algoritmos que miden el riesgo de vida de un accidentado. También se utilizan para asistir a médicos en el diagnostico partiendo de la historia clínica y los síntomas.
- Ciencias biológicas: clasificación de especies, patrones de movilidad, detección de tumores,...
- Finanzas e industria bancaria: para calcular los riesgos de impagos en cuotas bancarias. También se pueden utilizar para predecir comportamientos en el mercado de valores.
- Análisis de imágenes: Para identificar patrones en los dibujos, reconocer rostros, escritura manuscrita,...
- Juegos: existen muchos algoritmos programados para resolver puzles y muchos juegos como el ajedrez o las damas, donde computadoras programadas mediante ML superan a campeones del mundo.
- Robótica: para planificación de trayectorias, generación de movimientos,…

El ML se divide en dos áreas principales: aprendizaje supervisado y no supervisado.

3.1.1 Aprendizaje supervisado

El aprendizaje supervisado es un proceso de aprendizaje donde el entrenamiento está controlado por un agente externo, es decir, las predicciones se realizan en base a comportamientos o características observadas en datos almacenados. De esta manera, el aprendizaje supervisado busca patrones en datos históricos relacionando todos los datos con la función objetivo. Un buen ejemplo de esto es el etiquetado de correos como “spam” o “seguro” por parte de los usuarios. El proceso de predicción se basa en el análisis de las características que tienen los correos previamente ya etiquetados. Así determinamos, que existen una serie de direcciones IP, con una relación de texto/imagen o el uso de ciertas palabras que se clasifican como “spam”. Una vez determinados todos los patrones (proceso de aprendizaje), los nuevos correos recibidos se compararan con estos patrones y se clasificaran en “spam” o “seguros” en función de sus características.

Los principales métodos de aprendizaje supervisado son la regresión y la clasificación.

Un ejemplo de clasificación es el anteriormente mencionado del spam. Los correos se "categorizan" como "spam" o como "seguros". La tarea de una regresión es asignar una clase, es decir, predecir a que clase pertenece un conjunto de datos. El problema de clasificación trabaja con valores discretos

La regresión por otro lado predice valores continuos. Por ejemplo podemos utilizarla para calcular el precio de venta de una casa en función de las características de la misma (metros cuadrados, barrio, baños, habitaciones,...). En nuestro caso a estudio utilizaremos métodos de regresión.

3.1.2 Aprendizaje no supervisado

Por otro lado, el aprendizaje no supervisado usa datos históricos que no están etiquetados. Se distingue del supervisado por el hecho de que no hay conocimiento a priori. Este tipo de aprendizaje trata las entradas como un conjunto de variables aleatorias, construyendo así un modelo de densidad para el conjunto de datos.

El fin es explorar los datos para encontrar alguna estructura o forma de organizarlos. Por ejemplo, es frecuente su uso para agrupar clientes con características o comportamientos similares. Para hacer campañas de marketing altamente segmentadas se suelen utilizar métodos de agrupamiento como el Clustering.

3.2 Regresión

En el campo de la estadística, utilizamos la regresión como proceso para estimar la relación que existe entre diversas variables. La regresión incluye técnicas para el modelado y el análisis de variables, centrando la atención en la relación entre la variable objetivo o dependiente y las variables independientes o predictoras. Básicamente lo que hace el análisis de regresión es ayudar a comprender como cambia el valor de la variable dependiente en función de cómo varían cada una de las variables independientes, manteniendo el resto fijas. Se pude decir también que la regresión estima el valor promedio de la variable objetivo cuando se fijan las predictoras.

Este método es muy utilizado para la previsión y la predicción, donde se solapa totalmente con el campo del ML con el de la estadística. Su uso ayuda también a conocer cuál de las variables independientes tienen más relación con cada variable dependiente permitiéndonos así poder mejorar el algoritmo, seleccionando las variables más afines.

En los últimos año se han desarrollado muchas técnicas para llevar a cabo análisis de regresión. En nuestro caso realizaremos egresiones mediante mínimos cuadrados, mínimos cuadrados recursivos (en línea), máquinas de soporte vectorial y Random Forest.

4 MÍNIMOS CUADRADOS

4.1.1 Minimos cuadrados fuera de linea

Los mínimos cuadrados pretenden buscar la mejor función posible que relacione la variable dependiente con las independientes, es decir, la que mejor se aproxime a los datos con el criterio de mínimo error cuadrático.

Partiendo de la suposición de que los escalares y_K, $y_K = 0, 1, \ldots$ representan la salida (muestreada) de un sistema dinámico. Podemos suponer que aproximadamente el valor de la salida y_K puede ser calculado de la siguiente manera:

$$y_k \approx m_k\theta,\ k = 0, 1, \ldots \tag{1}$$

Donde $m_K \in R^{1\times nm}$ es un vector fila al que llamamos regresor, el regresor está formado por valores pasados de las entradas y salidas del sistema. θ es un vector columna que contiene los coeficientes que relacionan a las salidas y las entradas. También se pueden considerar las perturbaciones medibles y las acciones de control como entradas.

Vamos a suponer un sistema con salida y_K, una entrada u; la relación entre las dos vendría dada por la siguiente ecuación

$$y_k = a_1 y_k - 1 + b_1 u_k - 1 + b_2 u_k - 2. \tag{2}$$

El regresor para un instante k del tiempo de muestreo seria el siguiente:

$$m_k = [y_k - 1\ u_k - 1\ u_k - 2]. \tag{3}$$

El vector paramétrico θ sería:

$$\theta = [a_1\ b_1\ b_2] \tag{4}$$

Ahora se trata de calcular el vector de parámetros θ a partir de los valores de salida y_K y su regresor correspondiente m_K. Siempre no vamos a poder obtener un regresor, en nuestro ejemplo sería imposible construir el regresor m_0, ya que este depende de los valores pasados y_{K-1}, u_{k-1} y u_{K-2} valores imposibles de calcular. Por lo tanto el primer regresor que podríamos calcular en nuestro caso sería $m_2 = [y_1\ u_1\ u_0]$. De esta manera, podemos asumir que el primer regresor será (yn, mn), donde n será en la mayoría de los casos mayor que 0.

Para el cálculo del error cuadrático para cada instante k:

$$e_k = y_k - m_k\theta \tag{5}$$

Partiendo ahora de N pares (y (k), m (k)), se definen las siguientes matrices:

$$M(N) = \begin{bmatrix} m(n) \\ \vdots \\ m(N) \end{bmatrix} \tag{6}$$

$$E(N,\theta) = [e(n,\theta) \dots e(N,\theta)]^T \quad (7)$$

$$Y(N) = [y(n) \dots y(N)]^T \quad (8)$$

$$E(N,\theta) = Y(N) - M(N)\,\theta \quad (9)$$

La función a reducir será la suma de todos los errores al cuadrado:

$$J(\theta) = \|E(N,\theta)\|^2 = \sum_{k=n}^{N} e^2(k,\theta) \quad (10)$$

J (θ) también puede definirse de manera matricial:

$$J(\theta) = (Y(N) - M(N)\theta)^T (Y(N) - M(N)\theta) \quad (11)$$

Para obtener el mínimo, buscamos el valor de θ que anule la derivada de la función coste:

$$\frac{dJ(\theta)}{d\theta} = 0$$

Calculada la derivada e igualada a 0:

$$2\big(M(N)\theta - Y(N)\big)^T M(N) = 0 \quad (12)$$

El valor óptimo será el siguiente:

$$\theta^* = [M^T(N)M(N)]^{-1} M^T(N)Y(N) \quad (13)$$

Y es lo que denominamos estimador de mínimos cuadrados. Para que se cumpla la ecuación anterior el producto $M(N)^T M(N)$ tiene que ser invertible. A este primer método se le denomina fuera de línea.

4.1.2 Minimos cuadrados recursivos

La aplicación de mínimos cuadrados a sistemas variantes con el tiempo es de gran interés. Los parámetros que relacionan las salidas con las entradas también variaran con el tiempo. Se denomina θk la estimación del vector paramétrico en el tiempo de muestreo k.

En los mínimos cuadrados se obtiene θ_K para minimizar la función:

$$J_k(\theta) = \sum_{i=n}^{k} \lambda^{k-i}(y_i - m_i\theta)^2 \quad (1)$$

Donde λ ϵ (0; 1] se denomina factor de olvido. Si este fuera igual que 1, tendríamos lo mismo que en los mínimos cuadrados fuera de línea anteriormente explicados; afectando todos los errores de la misma

manera. A medida que el valor λ va alejándose del 1, los valores cercanos temporalmente (muestras cercanas a k) cobran más importancia. La elección de λ es importante para obtener buenos resultados. Definimos las siguientes matrices:

$$Y_k = \begin{bmatrix} y_n \\ y_{n+1} \\ \vdots \\ y_k \end{bmatrix}, \; M_k = \begin{bmatrix} m_n \\ m_{n+1} \\ \vdots \\ m_k \end{bmatrix}, \; W_k = \begin{bmatrix} \lambda^{k-n} & & & \\ & \lambda^{k-n-1} & & \\ & & \ddots & \\ & & & \lambda \\ & & & & 1 \end{bmatrix}$$

La función a minimizar es la siguiente

$$J_k(\theta) = (Y_k - M_k\theta)^\top W_k (Y_k - M_k\theta). \tag{2}$$

Para obtener el valor óptimo de θk analizamos la función para una perturbación $\theta k+\Delta\theta$ respecto del óptimo:

$$\begin{aligned} J_k(\theta_k + \Delta\theta) &= (Y_k - M_k(\theta_k + \Delta\theta))^\top W_k (Y_k - M_k(\theta_k + \Delta\theta)) \\ &= (Y_k - M_k\theta_k)^\top W_k (Y_k - M_k\theta_k) + \Delta^\top\theta M_k^\top W_k M_k \Delta\theta_k \\ &\quad -(Y_k - M_k\theta_k)^\top W_k M_k \Delta\theta - \Delta\theta^\top M_k^\top W_k (Y_k - M_k\theta_k) \\ &= J_k(\theta_k) + \Delta\theta^\top M_k^\top W_k M_k \Delta\theta_k - 2\Delta\theta^\top M_k^\top W_k (Y_k - M_k\theta_k). \end{aligned}$$

Eligiendo un valor de θ tal que todos los términos $\Delta\theta$ se eliminen (esto es equivalente a hacer cero el gradiente respecto de θk):

$$M_k^\top W_k (Y_k - M_k\theta_k) = 0 \tag{3}$$

De aquí extraemos que

$$M_k^\top W_k Y_k = M_k^\top W_k M_k \theta_k \tag{4}$$

O equivalentemente:

$$\theta_k = (M_k^\top W_k M_k)^{-1} M_k^\top W_k Y_k. \tag{5}$$

Por lo que la ecuación del coste para $\theta k+\Delta\theta$ es:

$$J_k(\theta_k + \Delta\theta) = J_k(\theta_k) + \Delta\theta M_k^\top W_k M_k \Delta\theta_k \geq J_k(\theta_k), \; \forall\Delta\theta. \tag{6}$$

Garantizando que

$$\theta_k = (M_k^\top W_k M_k)^{-1} M_k^\top W_k Y_k$$

Es óptima en el sentido de los mínimos cuadrados ponderados.

Ahora vamos a presentar una forma recursiva con la que podremos obtener valores de θk en cada instante k en función de θ_{k-1}. De esta manera definimos la siguiente matriz:

$$P_k = \left(\sum_{i=n}^{k} \lambda^{k-i} m_i^\top m_i \right)^{-1}.$$

(7)

P_k es una matriz simétrica ya que está definida como la inversa de una matriz simétrica. Para obtener P_k de manera recursiva utilizaremos P_{k-1}:

$$P_{k-1} = \left(\sum_{i=n}^{k-1} \lambda^{k-1-i} m_i^\top m_i \right)^{-1}$$

(8)

Ahora obteneos la relación entre Pk y Pk-1:

$$\begin{aligned} P_k^{-1} &= \sum_{i=n}^{k} \lambda^{k-i} m_i^\top m_i \\ &= m_k^\top m_k + \sum_{i=1}^{k-1} \lambda^{k-i} m_i^\top m_i \\ &= m_k^\top m_k + \lambda \sum_{i=1}^{k-1} \lambda^{k-1-i} m_i^\top m_i \\ &= m_k^\top m_k + \lambda P_{k-1}^{-1}. \end{aligned}$$

Aplicando el lema de inversión (no se explica) nos permitimos reescribir la relación entre Pk y Pk-1 sin realizar la inversa. La ecuación queda de la siguiente manera:

$$P_k = \frac{1}{\lambda} \left(P_{k-1} - \frac{(m_k P_{k-1})^\top (m_k P_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \right).$$

(9)

Se puede demostrar que P_{K-1} es igual a la matriz $Mk^T WkMk$:

$$
\begin{aligned}
M_k^\top W_k M_k &= \begin{bmatrix} m_n^\top & \dots & m_{k-1}^\top & m_k^\top \end{bmatrix} \begin{bmatrix} \lambda^{k-n} & & & \\ & \ddots & & \\ & & \lambda & \\ & & & 1 \end{bmatrix} \begin{bmatrix} m_n \\ \vdots \\ m_{k-1} \\ m_k \end{bmatrix} \\
&= \begin{bmatrix} m_n^\top & \dots & m_{k-1}^\top & m_k^\top \end{bmatrix} \begin{bmatrix} \lambda^{k-n} m_n \\ \vdots \\ \lambda m_{k-1} \\ m_k \end{bmatrix} \\
&= \sum_{i=n}^{k} \lambda^{k-i} m_i^\top m_i = P_k^{-1}.
\end{aligned}
$$

Sustituyendo en la ecuación (5) obtenemos:

El valor de θk-1 será el siguiente:

$$
\begin{aligned}
\theta_k &= (M_k^\top W_k M_k)^{-1} M_k^\top W_k Y_k \\
&= P_k M \\
&\qquad \theta_{k-1} = P_{k-1} \left(\sum_{i=n}^{k-1} \lambda^{k-1-i} m_i^\top y_i \right) . \\
&= P_k \begin{bmatrix} \cdots_n & \dots & \cdots_{k-1} & \cdots_k \end{bmatrix} \begin{bmatrix} & & & \\ & \ddots & & \\ & & \lambda & \\ & & & 1 \end{bmatrix} \begin{bmatrix} y_n \\ \vdots \\ y_{k-1} \\ y_k \end{bmatrix} \\
&= P_k \left(\sum_{i=n}^{k} \lambda^{k-i} m_i^\top y_i \right) .
\end{aligned}
$$

(10)

Y el de θk será:

$$
\begin{aligned}
\theta_k &= P_k \left(\sum_{i=n}^{k} \lambda^{k-i} m_i^\top y_i \right) \\
&= P_k \left(m_k^\top y_k + \sum_{i=n}^{k-1} \lambda^{k-i} m_i^\top y_i \right)
\end{aligned}
$$

(11)

Sustituyendo con la ecuación (9) obtenemos θk en función de θk-1:

$$\begin{aligned}
\theta_k &= \frac{1}{\lambda}\left(P_{k-1} - \frac{(m_k P_{k-1})^\top (m_k P_{k-1})}{\lambda + m_k P_{k-1} m_k^\top}\right) m_k^\top y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{1}{\lambda}\left(P_{k-1} m_k^\top - \frac{P_{k-1} m_k^\top (m_k P_{k-1} m_k^\top)}{\lambda + m_k P_{k-1} m_k^\top}\right) y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{1}{\lambda} P_{k-1} m_k^\top \left(1 - \frac{m_k P_{k-1} m_k^\top}{\lambda + m_k P_{k-1} m_k^\top}\right) y_k \\
&\quad + \theta_{k-1} - \frac{(m_k P_{k-1})^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \frac{P_{k-1} m_k^\top y_k}{\lambda + m_k P_{k-1} m_k^\top} + \theta_{k-1} - \frac{P_{k-1} m_k^\top (m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top} \\
&= \theta_{k-1} + \frac{P_{k-1} m_k^\top (y_k - m_k \theta_{k-1})}{\lambda + m_k P_{k-1} m_k^\top}.
\end{aligned}$$

(12)

Finalmente definimos la ganancia de estimación como:

$$K_k = \frac{P_{k-1} m_k^\top}{\lambda + m_k P_{k-1} m_k^\top}.$$

Con la ganancia de estimación sustituimos términos en Pk y θk. Obtenemos las siguientes ecuaciones para el cálculo recursivo de θk:

$$\begin{aligned}
K_k &= \frac{P_{k-1} m_k^\top}{\lambda + m_k P_{k-1} m_k^\top} \\
\theta_k &= \theta_{k-1} + K_k (y_k - m_k \theta_{k-1}) \\
P_k &= \frac{1}{\lambda}(\mathrm{I} - K_k m_k) P_{k-1}.
\end{aligned}$$

(13) (14) (15)

4.2 Explicacion de algoritmos

- **Mínimos cuadrados**

  Este algoritmo ha sido programado con Matlab. Las variables de entrada y objetivo las lee el algoritmo directamente desde cada uno de sus archivos de Excel (demanda, generación y precio). Después introducimos el intervalo de tiempo para el conjunto de entrenamiento y el de prueba (días totales, mes de inicio, día de inicio); hay que tener en cuenta que nuestro set de datos solo abarca 2015, por lo que debido a la variable *precio semana anterior en h* predeciremos con fechas a partir del 8-1-2015. Finalmente, tras

realizar los mínimos cuadrados obtenemos los errores y una gráfica con el valor real y el predicho.

- **Mínimos cuadrados recursivos**

  En este algoritmo también realizado en Matlab hemos adquirido los datos de la misma manera que en el primero y la manera de funcionamiento es la misma. Para la identificación en línea de este método hemos utilizado un bucle for que ha ido calculando el vector de características θ a partir de la matriz de covarianzas P y la ganancia de estimación K.

4.3 Resultados

4.3.1 Mínimos cuadrados fuera de línea

En el caso de este algoritmo, se ha aplicado directamente el algoritmo programado con los diferentes conjuntos de datos.

- Caso 1:

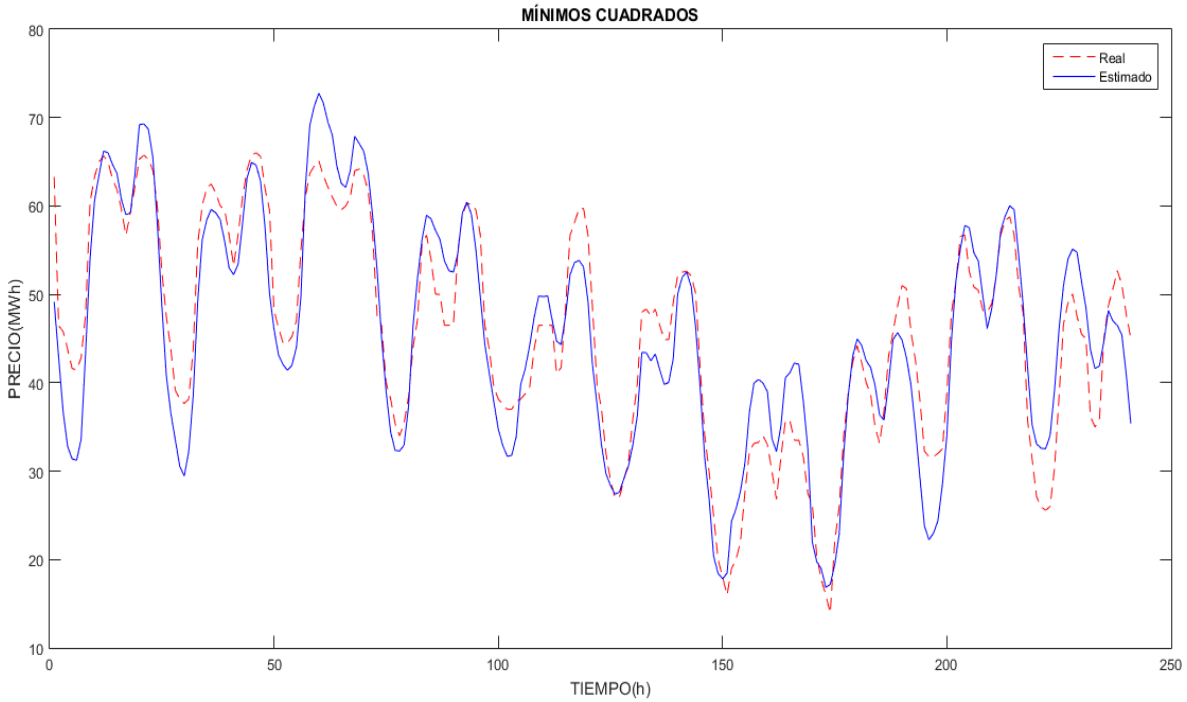


Figura 4-1. Caso 1 mínimos cuadrados.

- Caso 2

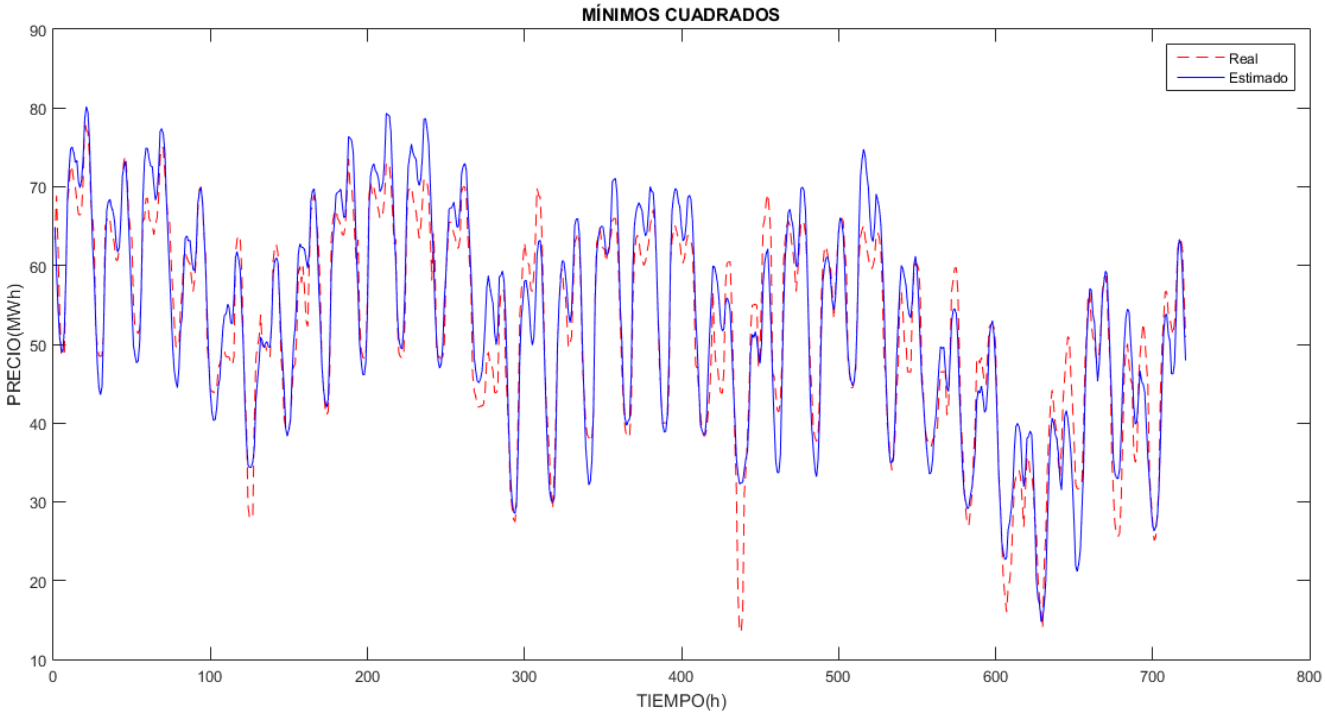


Figura 4-2. Caso 2 mínimos cuadrados.

- Caso 3

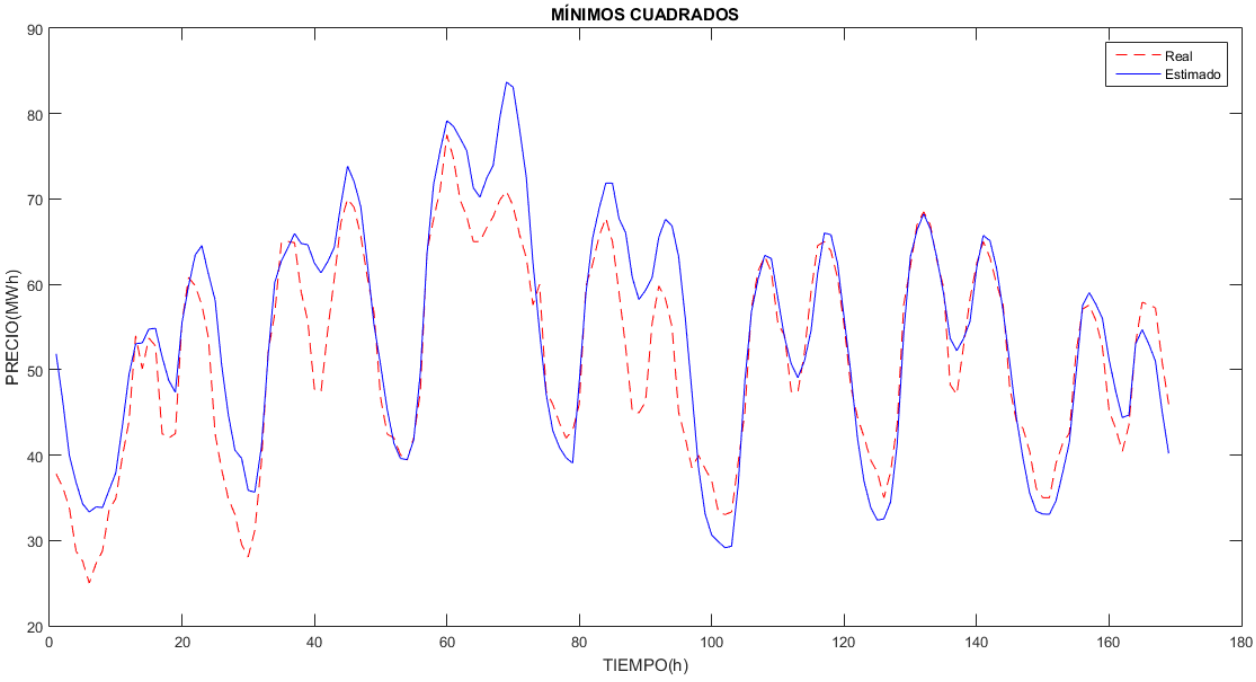


Figura 4-3. Caso 3 mínimos cuadrados.

Tabla de errores

	MAPE (%)	RMSE	MAE
Caso 1	9.65%	0,6199	3.9701
Caso 2	7.81%	0,1199	3.5556
Caso 3	9.82%	0,4456	4.5325

Tabla 4–1. Errores mínimos cuadrados fuera de linea

El caso que da los mejores resultados es el primero, todos sus errores son inferiores a los del resto de casos; siendo este el conjunto de datos más grande.

4.3.2 Mínimos cuadrados recursivos

Hemos realizado cada caso con tres factores de olvido diferentes: λ=0.97 λ=0,95 y λ=0.9

Mientras más se aleje el valor de λ de 1, mayor los valores más cercanos a la muestra cobraran más importancia que los demás.

- Caso 1:

Para λ=0.97

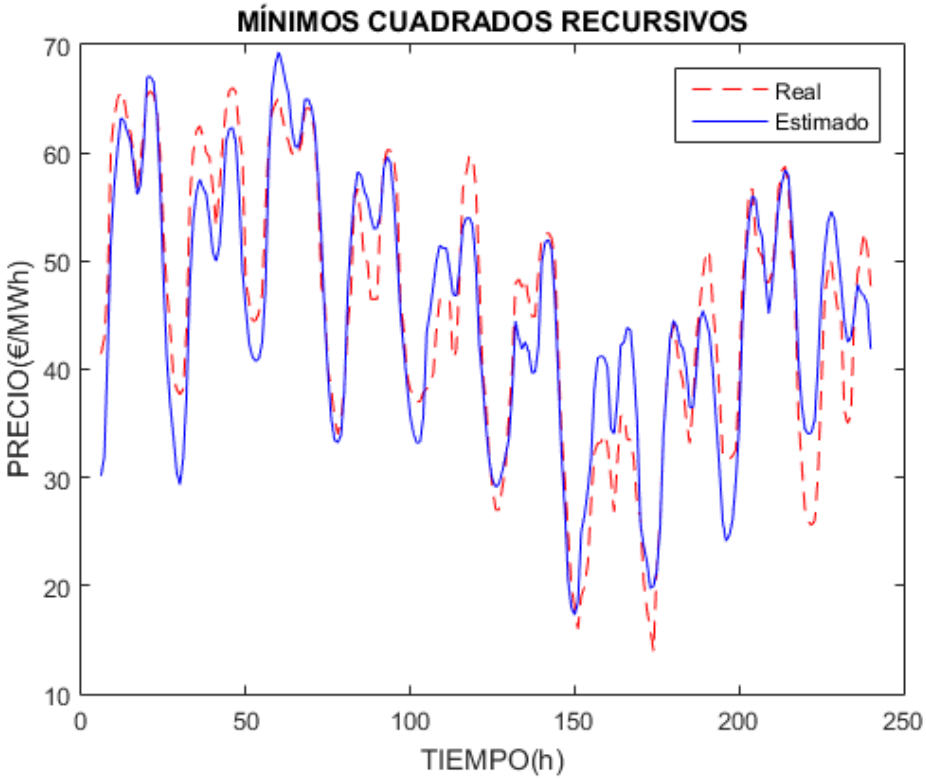


Figura 4-4. Caso 1 mínimos cuadrados recursivos, λ=0.97.

.Para λ=0.95

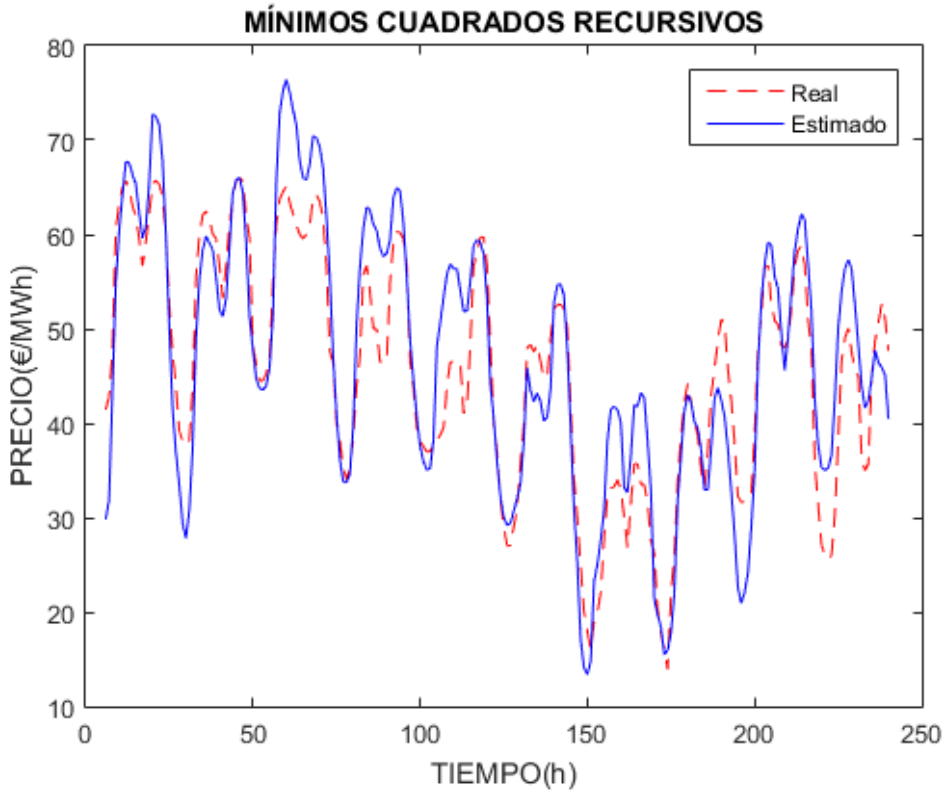


Figura 4-5. Caso 1 mínimos cuadrados recursivos, λ=0.95.

Para λ=0.90

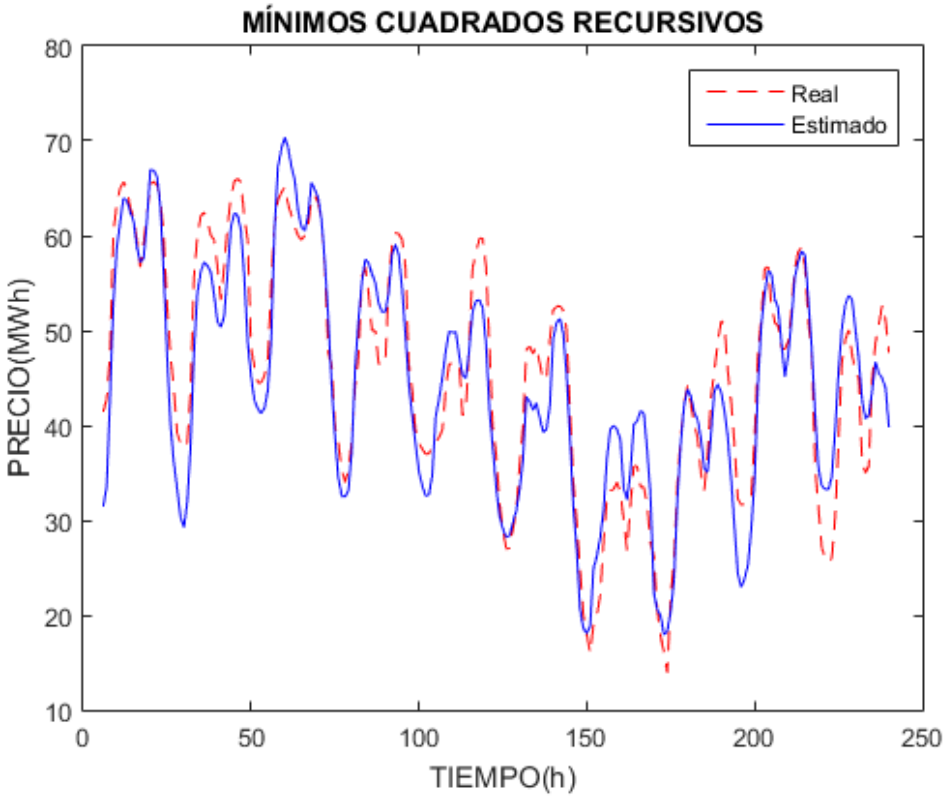


Figura 4-6. Caso 1 mínimos cuadrados recursivos, λ=0.90.

Tabla de errores

	MAPE (%)	RMSE	MAE
λ=0.97	9.84%	0,376	3.9161
λ=0.95	9.80%	0.4019	3.9168
λ=0.90	11.41%	0.4688	4.7349

Tabla 4–2. Errores mínimos cuadrados en línea Caso 1

En este primer caso, las respuestas dadas por los 3 factores de olvido son bastante parecidas. Especialmente en para λ=0.95 y 0.97 la respuesta es muy parecida a la de los mínimos cuadrados del apartado anterior. La respuesta para λ=0.9 empeora, todos sus errores son superiores a los de los obtenidos con los otros dos factores de olvido.

- Caso 2

  Para λ=0.97

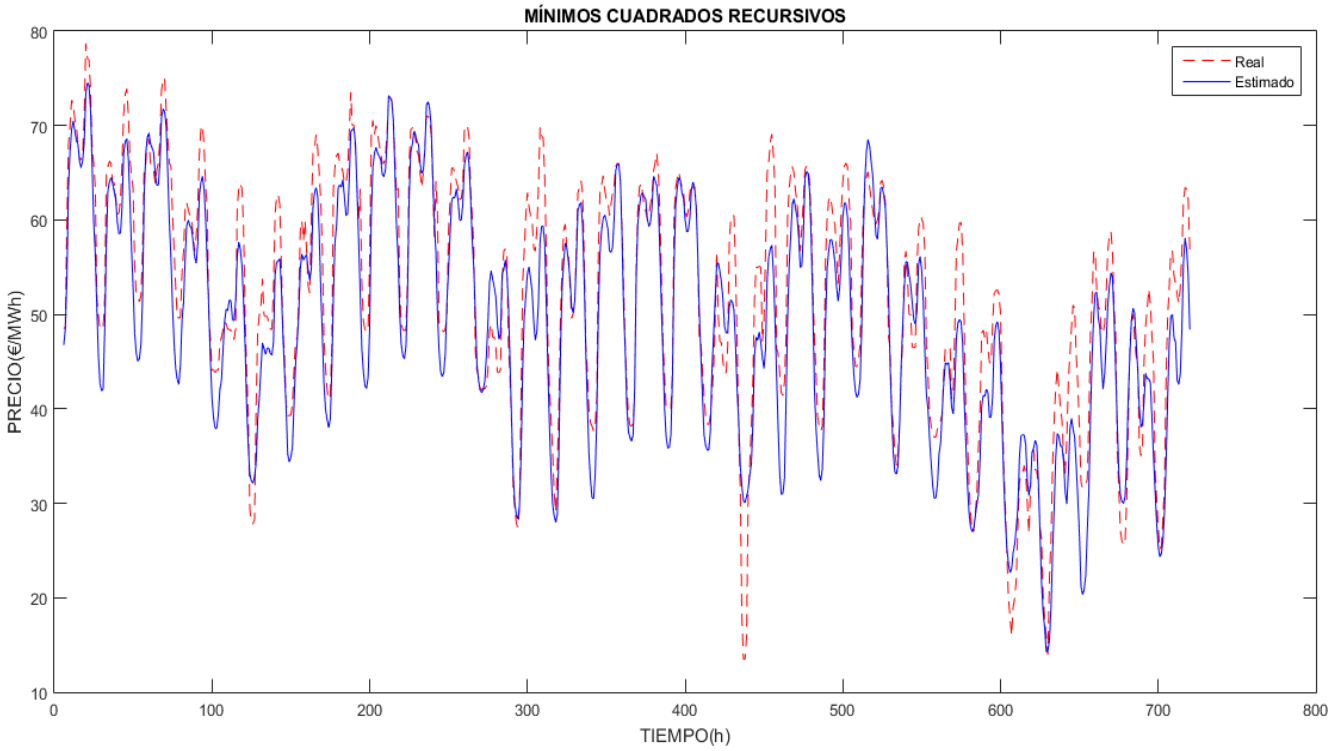


Figura 4-7. Caso 2 mínimos cuadrados recursivos, λ=0.97.

Para λ=0.95

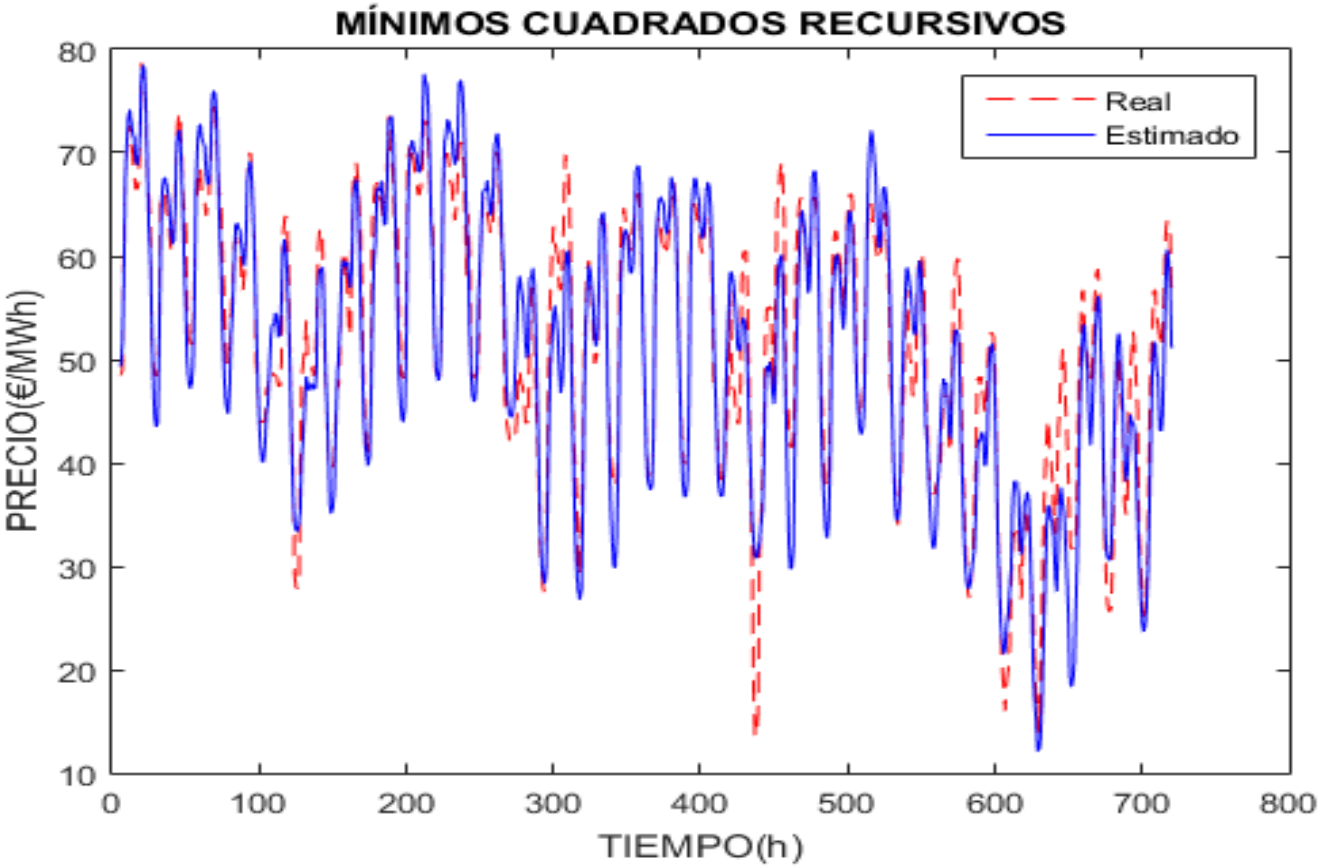


Figura 4-8. Caso 2 mínimos cuadrados recursivos, λ=0.95.

Para λ=0.90

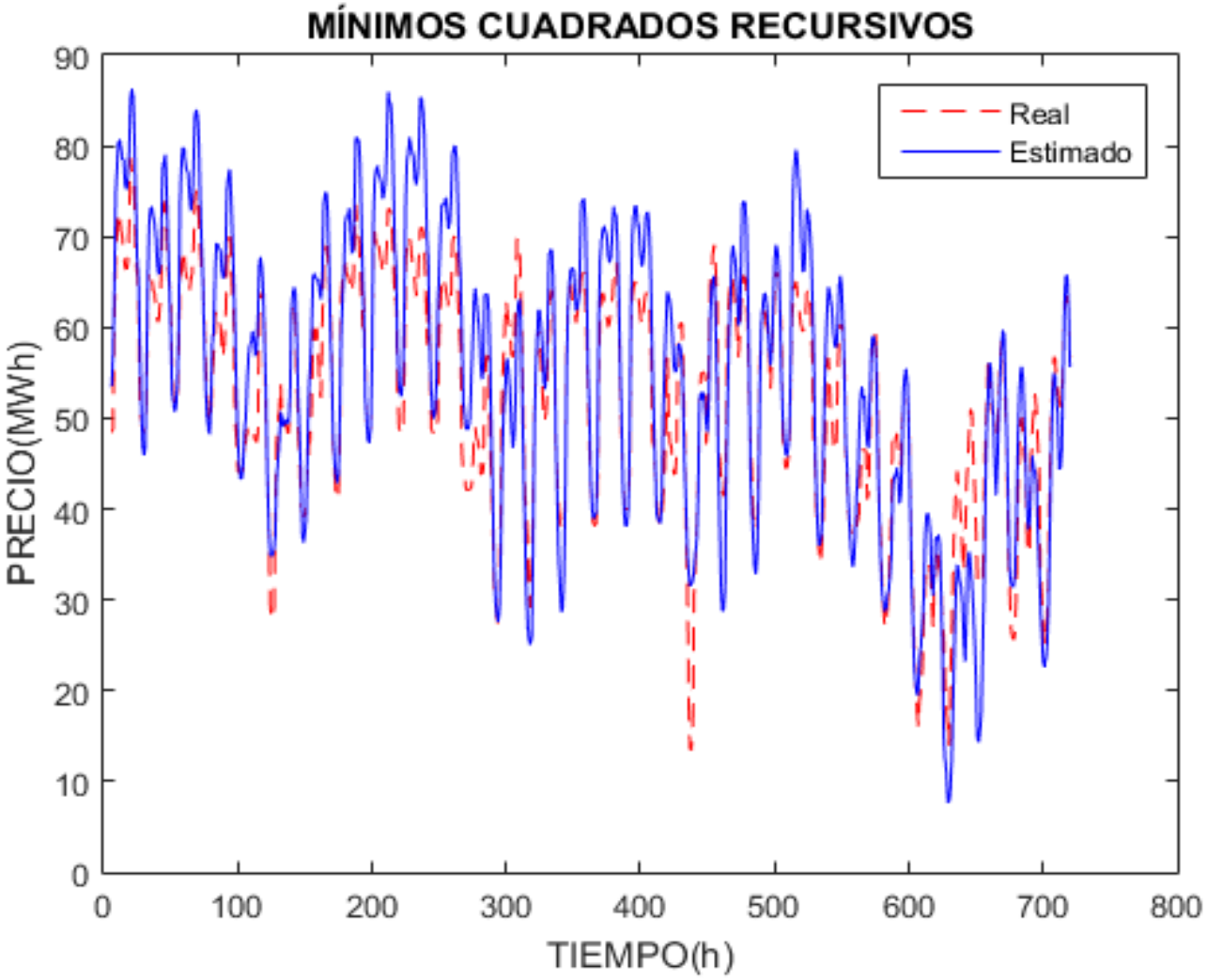


Figura 4-9. Caso 2 mínimos cuadrados recursivos, λ=0.90.

Tabla de errores:

	MAPE (%)	RMSE	MAE
λ=0.97	8.52%	0.3097	3.9767
λ=0.95	8.13%	0.2098	3.652
λ=0.90	12.67%	0.0176	6.1715

Tabla 4–3. Errores mínimos cuadrados en línea Caso 2

Para los dos primeros valores tenemos una respuesta muy parecida a la de los mínimos cuadrados fuera de línea. Para λ=0.90 el MAPE y el MAE aumentan; el RMSE se reduce drásticamente, siendo el valor mucho mejor que el obtenido con el resto de factores de olvido.

- Caso 3

  Para λ=0.97

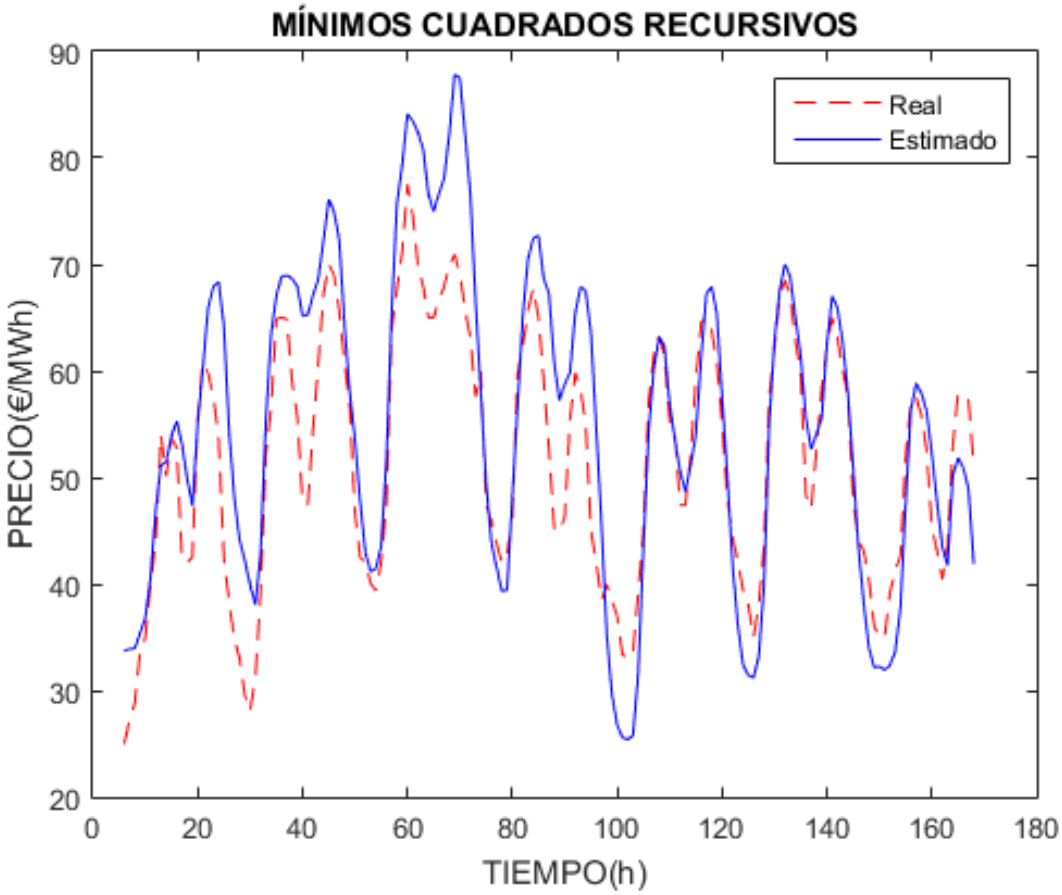


Figura 4-10. Caso 3 mínimos cuadrados recursivos, λ=0.97.

Para λ=0.95

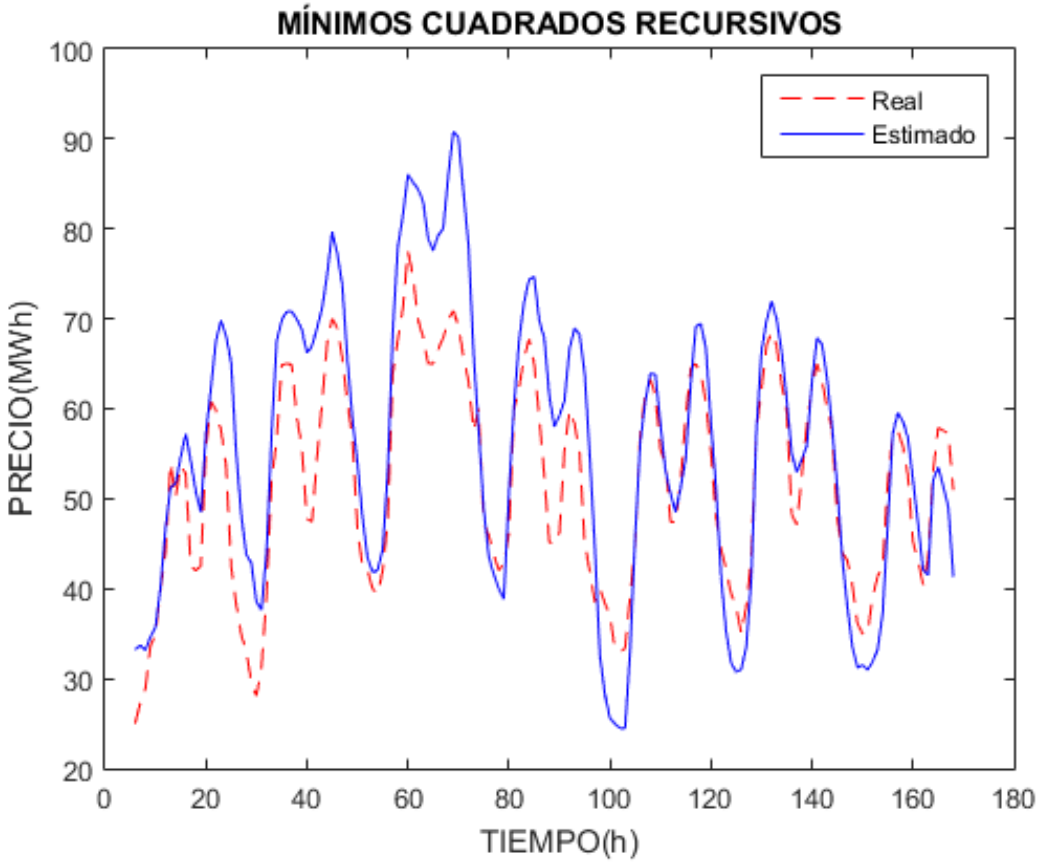


Figura 4-11. Caso 3 mínimos cuadrados recursivos, λ=0.95.

Para λ=0.90

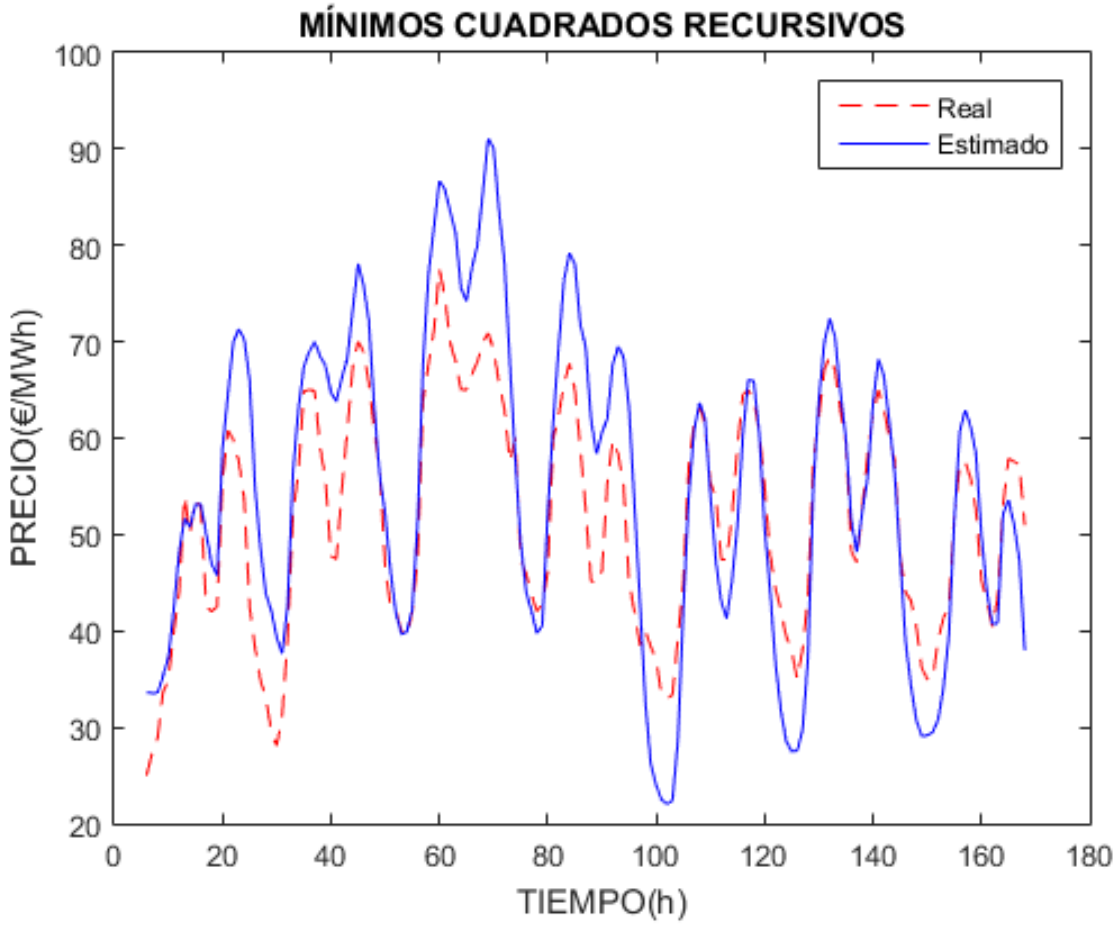


Figura 4-12. Caso 3 mínimos cuadrados recursivos, λ=0.90.

Tabla de errores

	MAPE (%)	RMSE	MAE
λ=0.97	11.96%	0.7021	5.7602
λ=0.95	13.63%	0.8481	6.6377
λ=0.90	13.66%	1.0024	6.636

Tabla 4–4. Errores mínimos cuadrados en línea Caso 3

Para el tercer caso, a medida que disminuimos el factor de olvido, aumentan los errores; como ha pasado en los dos casos anteriores. En este tercer caso la predicción es bastante mala en comparación con los otros 2.

Como hemos observado, el factor de olvido puede ser útil para mejorar la predicción en algunos casos. En nuestros casos la mejora no ha sido muy significativa; el factor de olvido tendrá mejor respuesta para conjuntos de datos más grandes.

5 RANDOM FOREST

5.1 Concepto

El concepto de Random Forest fue introducido por Leo Breiman en 2001. Para explicarlo, partiremos de los árboles de decisión. Los árboles de decisión son métodos comúnmente utilizados en tareas de clasificación y regresión. La predicción se realiza sobre el modelo (árbol) construido, el cual está formado por nodos y ramas. La estructura comienza con un nodo raíz (no tiene ramas de entrada), los nodos interiores se corresponden con una variable de entrada, de tal modo que las instancias se derivan a cada uno de sus nodos hijos dependiendo del valor que estas tengan en dicha variable. Los nodos que no tienen hijos se denominan hojas y realizan una predicción utilizando la información de la variable a predecir de las instancias que acaban en dicho nodo.

Para determinar la calidad de cada nodo utilizamos la función de impureza. Existen diversas medidas de la impureza (criterios de particionamineto): coeficiente de Gini, entropía…

Normalmente el árbol obtenido esta sobre ajustado, así que tiene que ser "podado", es decir, eliminar nodos y ramas hasta encontrar el tamaño óptimo.

Figura 5-1. Estructura árbol decisión.

Los árboles solo son recomendables para cuando el número de acciones es pequeño y no son posibles todas las combinaciones.En la elección de un modelo, existe una cantidad muy limitada y dificulta para elegir el árbol óptimo. Presenta inconvenientes cuando la cantidad de alternativas es grande y cuanto las decisiones no son racionales. Uniendo muchos árboles podemos llegar a un algoritmo mejor. ¿Cómo unificamos todos estos árboles?

Los métodos más utilizados para la agregación de modelos son el boosting y el bagging:

- Boosting: Se entrena una serie de clasificadores débiles(o regresores), así que en cada paso mejoramos el clasificador anterior, terminando en un clasificador fuerte.
- Bagging (Boostrap Agregating): Se entrena una serie de clasificadores(o regresores) independientes y son combinados para obtener un clasificador fuerte.

Boostrap Aggregating entrena a cada regresor con un conjunto de datos diferente. Los conjuntos de datos son generados por bootstrapping, esto significa generar nuevos conjuntos de datos siempre del mismo tamaño usando muestreo con reposición. El muestreo con reposición toma una muestra de manera aleatoria de la población y la devuelve, para que pueda ser reelegida por el resto de las muestras. En promedio una muestra de Boostrap contiene 65% de las muestras originales.

Los árboles son los candidatos ideales para el bagging, dado que ellos pueden registrar estructuras de interacción compleja en los datos, y si crecen suficientemente profundo, tienen relativamente baja parcialidad. Producto de que los árboles son notoriamente ruidosos, ellos se benefician grandemente al promediar.

Para aplicar Random Forest seguimos el siguiente algoritmo:

1. Sea *N* el número de casos de prueba, *M* es el número de variables en el clasificador.
2. Sea *m* el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; *m* debe ser mucho menor que *M*
3. Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.
4. Para cada nodo del árbol, elegir aleatoriamente *m* variables en las cuales basar la decisión. Calcular la mejor partición a partir de las *m* variables del conjunto de entrenamiento. Para realizar la mejor partición se debe buscar una función objetivo; habitualmente utilizamos la función de gini o la entropía.
5. Los anteriores procesos han de ser repetidos varias veces, de manera que se obtengan conjuntos de árboles de decisión entrenados sobre diferentes conjuntos de datos o atributos

Para la predicción un nuevo caso es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal donde termina. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción.

El conjunto de prueba se utiliza para determinar la impureza de los nodos, la suma de estas será la impureza total del árbol.

Podemos observar en la siguiente figura, en la cual para el mismo conjunto de datos, tenemos ejemplos de clasificación con árboles de decisión (izquierda) y con Random Forest (derecha).

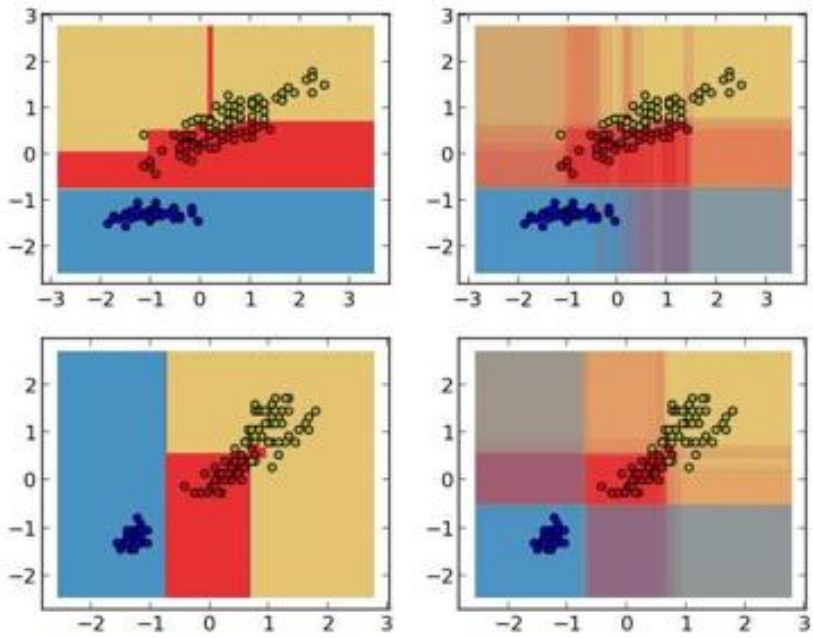


Figura 5-2. Ejemplo árbol decisión y RF

En la figura se aprecia como el algoritmo RF tiene más matices en la predicción que los árboles de decisión. Por otra parte, también se observa que los árboles de decisión son mejores en las áreas con pocos datos.

Los *bosques aleatorios o Random Forest* combinan varios árboles de decisión con bagging son fuertes clasificadores con alta exactitud y poca varianza. Pueden dar estimaciones para el error e importancia de variables. Pueden ser lentos cuando hay muchas muestras o variables. Sus parámetros más importantes son el número y la profundidad de los árboles.

5.2 Resultados obtenidos

Este ha sido el primer algoritmo programado con Python, para ello utilizamos la librería *sklearn.ensemble.forest* y la función *RandomForestRegressor*, con la cual, ajustando sus distintos atributos obtenemos la predicción. Las variables se leen directamente del csv separado por comas. Utilizamos un conjunto de entrenamiento y otro de prueba que son elegidos previamente.

El algoritmo programado ha realizado la predicción con 2000 árboles, no se han limitado los valores máximos y mínimos de hojas, variables y longitud de los árboles.

- Caso 1

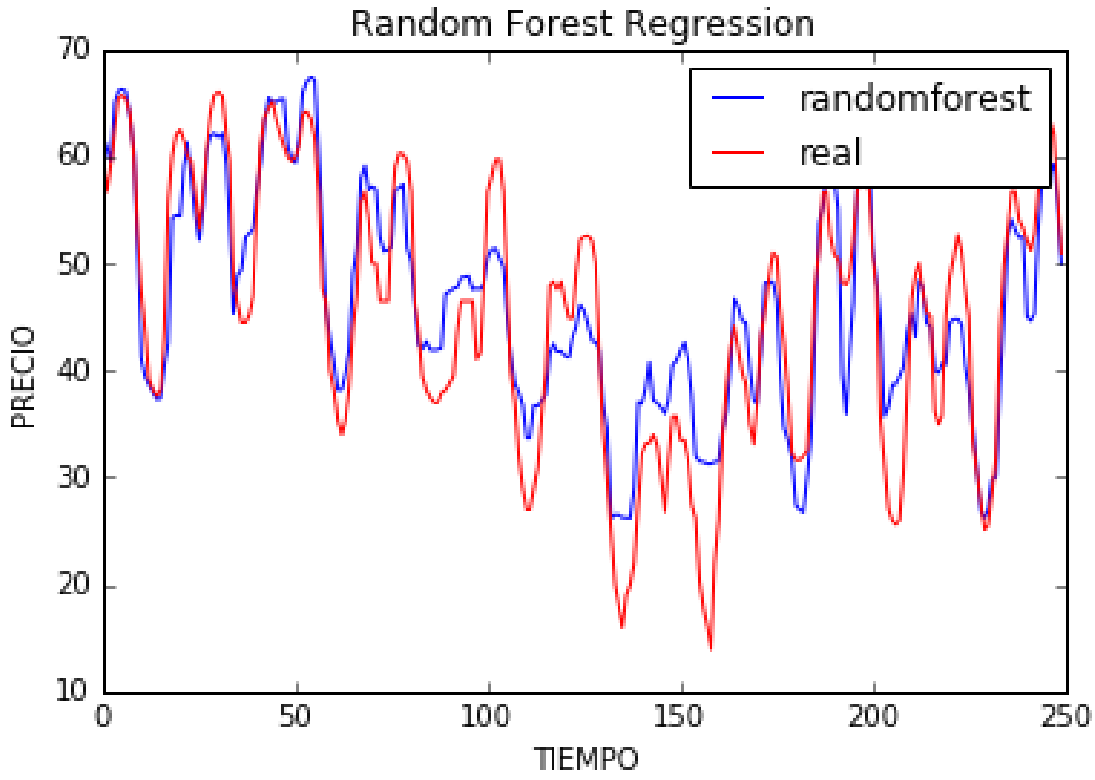


Figura 5-3. Caso 1 RF

- Caso 2

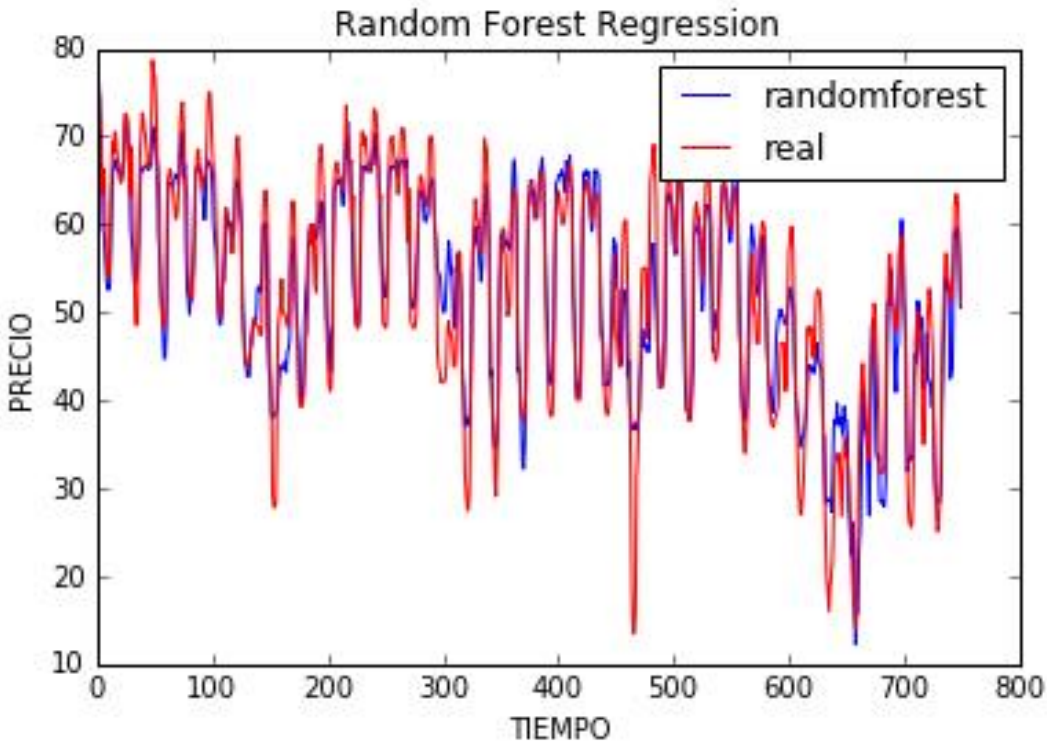


Figura 5-4. Caso 2 RF

- Caso 3

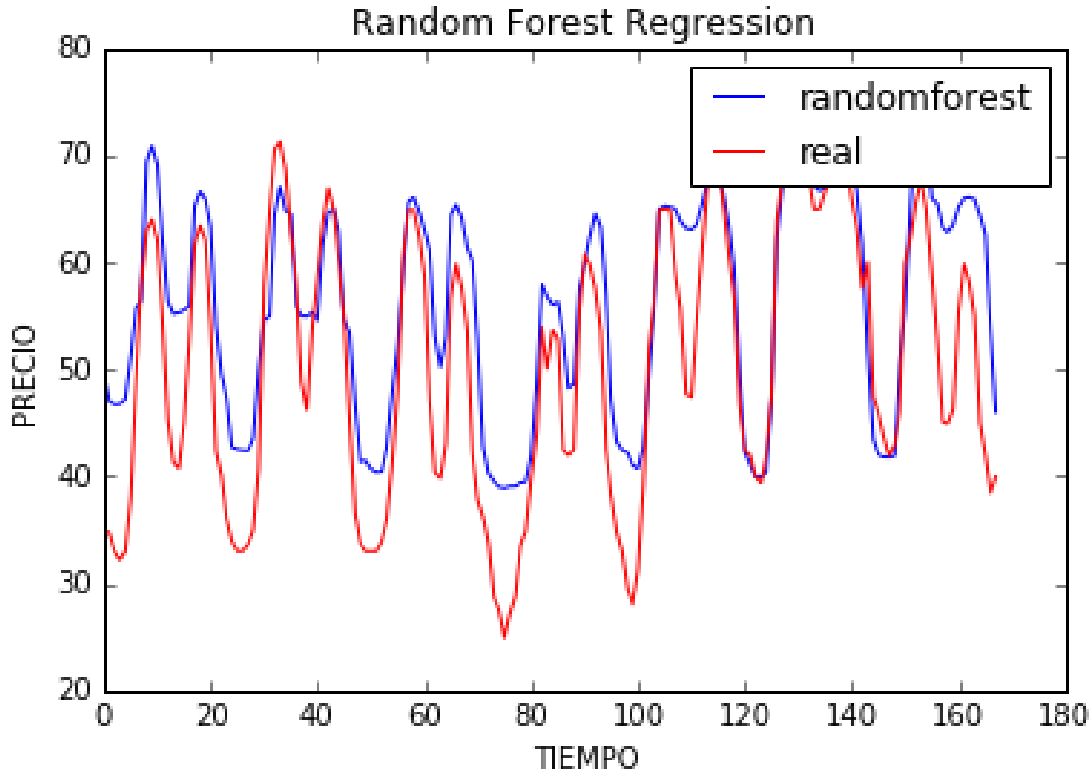


Figura 5-5. Caso 3 RF

Tabla de errores

	MAPE (%)	RMSE	MAE
Caso 1	10.73%	0,0677	4.481
Caso 2	8.11 %	0,017	3.76
Caso 3	12.34%	0,478	6.53

Tabla 5–1. Errores RF

La respuesta obtenida para los tres casos es bastante; los valores de MAPE no mejoran nada respecto a los mínimos cuadrados. Pero se disminuye drásticamente el RMSE y el MAE en los dos primeros casos. La respuesta en el tercer caso, siguiendo con la línea de los métodos anteriores, es la que tiene los errores más grandes.

6 MÁQUINAS DE SOPORTE VECTORIAL

6.1 Concepto

Las máquinas de vectores soporte (SVM, del inglés Support Vector Machines) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico y fueron introducidas en los años 90 por Vapnik y sus colaboradores. Aunque originariamente las SVMs fueron pensadas para resolver problemas de clasificación binaria, se pueden utilizar para resolver problemas de regresión, agrupamiento y multiclasificación entre otros. Han ido ganando mucho reconociendo debido a sus sólidos fundamentos teóricos

Las SVMs inducen separadores lineales o hiperplanos, ya sea en el espacio original (si el espacio es separable o cuasi-separable) o en un espacio transformado (si los espacios no son separables linealmente en el espacio original). Como ya veremos más adelante la búsqueda del hiperplanos de separación en los espacios transformados se hará mediante funciones kernel.

A diferencia que la mayoría de algoritmos de machine learning que se centran en reducir el error cometido por el modelo generado por el set de entrenamiento, las SVMs intentan reducir el riesgo estructural. La idea es seleccionar un hiperplanos de separación que equidista de los ejemplos más cercanos de cada clase para, de esta forma, consiguiendo un margen máximo a cada lado del hiperplanos. Para definir el hiperplanos solo utilizamos los valores que están en la frontera del margen. A estos ejemplos los llamamos vectores de soporte. Los hiperplanos separadores de máximo margen evitan en gran medida el sobreajuste a los valores de entrenamiento.

El problema a resolver es la optimización del margen geométrico, que es un problema de optimización cuadrático con restricciones lineales. Este problema se resuelve utilizando técnicas de programación cuadrática. La convexidad que exige su resolución nos garantiza una solución única.

Para comprender mejor el concepto vamos a exponer un ejemplo sencillo de clasificación para después introducir la regresión.

6.1.1 SVM para clasificación binaria

Partiendo de un conjunto separable *S= {(w1*x1),... (wn*xn)}*, donde *xi∈Rd e yi∈{+1,-1},* definimos el siguiente hiperplanos de separación:

$$D(\boldsymbol{x}) = (w_1 x_1 + \ldots + w_d x_d) + b = < \boldsymbol{w}, \boldsymbol{x} > + b \tag{1}$$

Donde *w* y *b* coeficientes reales. Para nuestro conjunto en concreto, se cumplirán las siguientes restricciones:

$$\begin{aligned} &< \boldsymbol{w}, \boldsymbol{x}_i > + b \geq 0 \quad \text{si} \quad y_i = +1 \\ &< \boldsymbol{w}, \boldsymbol{x}_i > + b \leq 0 \quad \text{si} \quad y_i = -1,\ i = 1, \ldots, n \end{aligned} \tag{2}$$

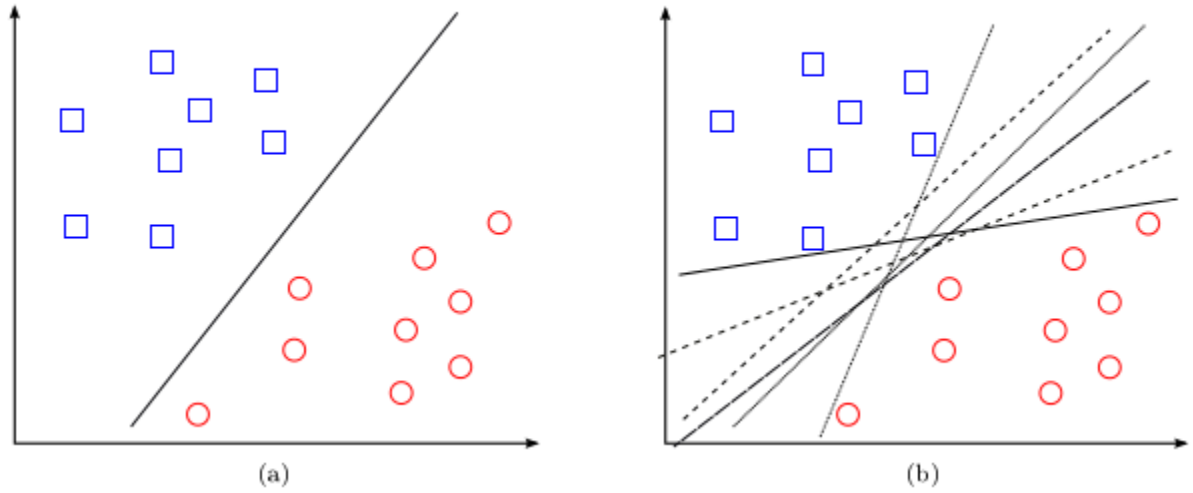
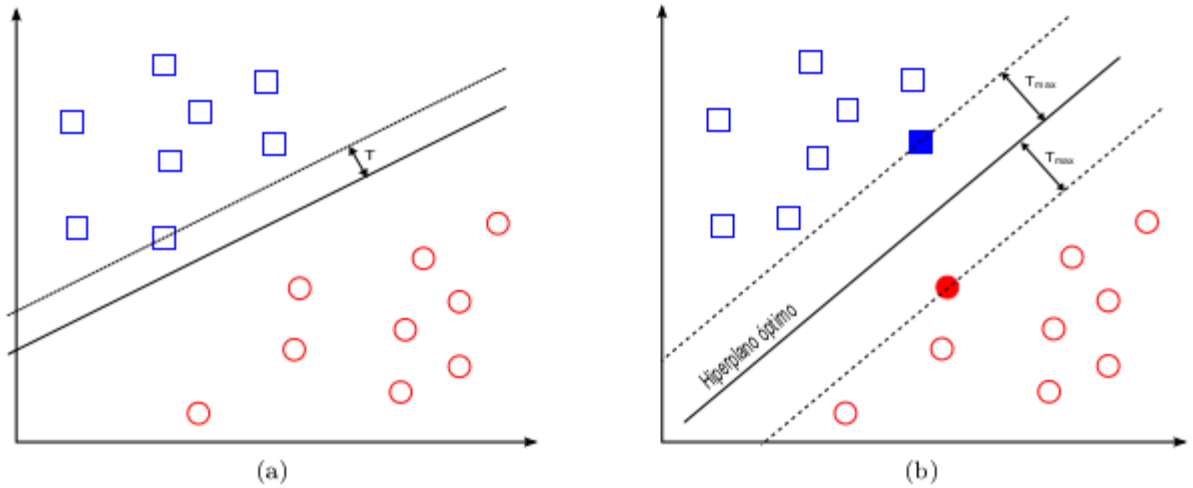


Figura 6-1. Clasificación binaria e hiperplanos

En la figura(a) observamos un ejemplo de hiperplanos que separa en dos clases. En (b) observamos otros hiperplanos de entre todos los infinitos posibles.

Surge la necesidad ahora de crear un criterio para elegir el hiperplanos de separación óptima (τ). Tomaremos como óptimo el hiperplanos de separación que su margen sea máximo, como observamos en esta figura:

Figura 6-2. Margen máximo



En la figura (a) observamos un plano de separación no optimo frente al de la figura (b) que si lo es (margen máximo).

Por geometría sabemos que la distancia entre un hiperplanos de separación D(x) y un ejemplo x' viene dada por:

$$\frac{|D(\boldsymbol{x}')|}{\|\boldsymbol{w}\|} \tag{3}$$

||w|| se define como la norma del vector w, que es vector que define el plano junto con b. Una manera más compacta de representar el hiperplanos es la siguiente:

$$y_i D(\boldsymbol{x}_i) \geq 0, \qquad i = 1, \ldots, n \tag{4}$$

Combinando las dos ecuaciones anteriores obtenemos la siguiente ecuación que cumplirán todos os ejemplos de entrenamiento:

$$\frac{y_i D(\boldsymbol{x}_i)}{\|\boldsymbol{w}\|} \geq \tau, \qquad i = 1, \ldots, n \tag{5}$$

O también la podemos expresar:

$$y_i D(\boldsymbol{x}_i) \geq \tau \|\boldsymbol{w}\|, \qquad i = 1, \ldots, n \tag{6}$$

Para limitar el número de soluciones a una sola, la escala del producto de τ y la norma de w se fija, de forma arbitraria como la unidad

$$\tau \|\boldsymbol{w}\| = 1 \tag{7}$$

Llegando a la conclusión de que para aumentar el margen hay que disminuir la norma de w. En la siguiente figura podemos observar como la distancia al hiperplanos optimo desde cualquier punto viene dada por |D (xi)| / ||w||:

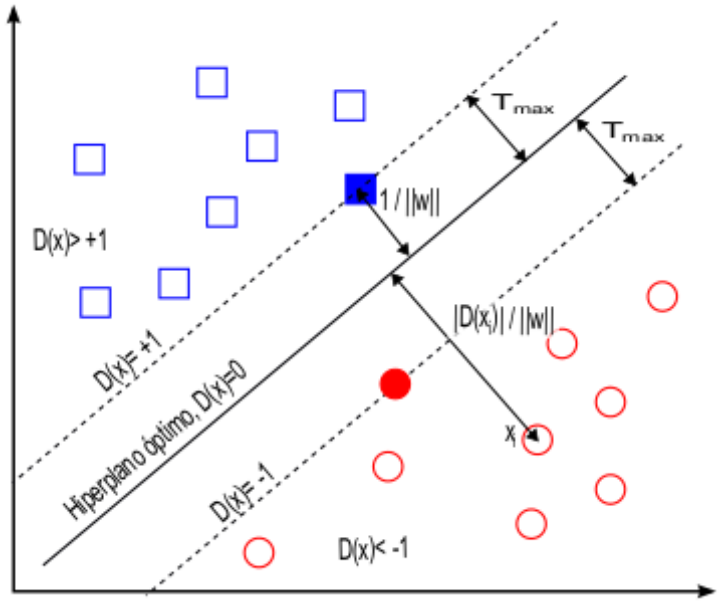


Figura 6-3. Vectores soporte.

Los ejemplos que están situados a ambos lados del hiperplanos óptimo y definen el margen, se denominan vectores de soporte. Para los vectores de soporte la distancia al hiperplanos será 1/||w|| ya que los vectores soporte siempre cumplen |D(x)|=1. El hiperplanos de separación óptima se define solo a partir de los vectores de soporte. Para un margen máximo, necesitaremos entonces un ||w|| mínimo sujeto a las restricciones antes impuestas. Siendo el funcional f (w)=||w||, nuestro problema es el siguiente:

$$\begin{aligned} \text{mín} \quad & f(\boldsymbol{w}) = \tfrac{1}{2}\|\boldsymbol{w}\|^2 = \tfrac{1}{2} < \boldsymbol{w}, \boldsymbol{w} > \\ \text{s.a.} \quad & y_i\left(< \boldsymbol{w}, \boldsymbol{x}_i > + b\right) - 1 \geq 0, \qquad i = 1, \ldots, n \end{aligned} \tag{8}$$

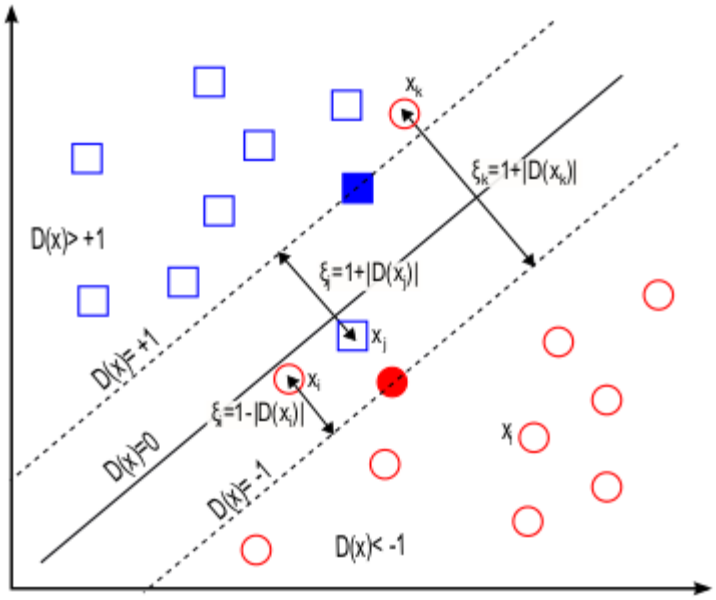
Este problema de optimización se resuelve mediante multiplicadores de Lagrange, como esto no es objetivo de nuestro estudio y es una explicación un tanto tediosa, la vamos a obviar. El hiperplanos de separación se construirá como combinación lineal de sólo dos vectores de soporte del conjunto de ejemplos (uno de cada clase).

El caso anterior es un caso idílico, ya que los problemas reales pueden tener ruidos y no ser perfectamente separables linealmente (cuasi-separable). Para resolver estos problemas reducimos el grado de separabilidad, permitiendo que haya errores de clasificación en algunos ejemplos y buscando un hiperplanos óptimo para el resto que si son separables. La idea es introducir en la ecuación del hiperplanos un conjunto de variables reales positivas, denominadas variables de holgura ξi, i = 1,. . . n, pudiendo así cuantificar los ejemplos no separables que se admitirán:

$$y_i\left(< \boldsymbol{w}, \boldsymbol{x}_i > + b\right) \geq 1 - \xi_i, \qquad \xi_i \geq 0,\ i = 1, \ldots, n \tag{9}$$

ξi representa la desviación del caso separable, medida desde el borde del margen de cada clase yi. Así la suma de todas las holguras nos permite medir el coste asociado a los ejemplos no separables.

Figura 6-4. Holguras



En este caso el funcional sería:

$$f(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{10}$$

Donde C es una constante elegida por el usuario, que permite cambiar la influencia del coste de los términos no separables. Finalmente nuestro problema de optimización con sus restricciones seria:

$$\begin{aligned} \text{mín} \quad & \tfrac{1}{2} < \boldsymbol{w}, \boldsymbol{w} > + C \sum_{i=1}^{n} \xi_i \\ \text{s.a.} \quad & y_i \left(< \boldsymbol{w}, \boldsymbol{x}_i > + b \right) + \xi_i - 1 \geq 0 \\ & \xi_i \geq 0, \ i = 1, \ldots, n \end{aligned} \tag{11}$$

Este hiperplanos recibe el nombre de hiperplanos de separación de "margen blando", mientras que el obtenido en el caso perfectamente separable se denomina de "margen duro".

Finalmente tenemos el caso de que no sean separables ni cuasi-separables por un hiperplanos. Para resolver estos problemas realizaremos transformaciones no lineales al conjunto inicialmente no separable y buscar hiperplanos de separación en esos espacios transformados. A estos espacios los llamaremos espacio de características para diferenciarlos del espacio de ejemplos de entrada.

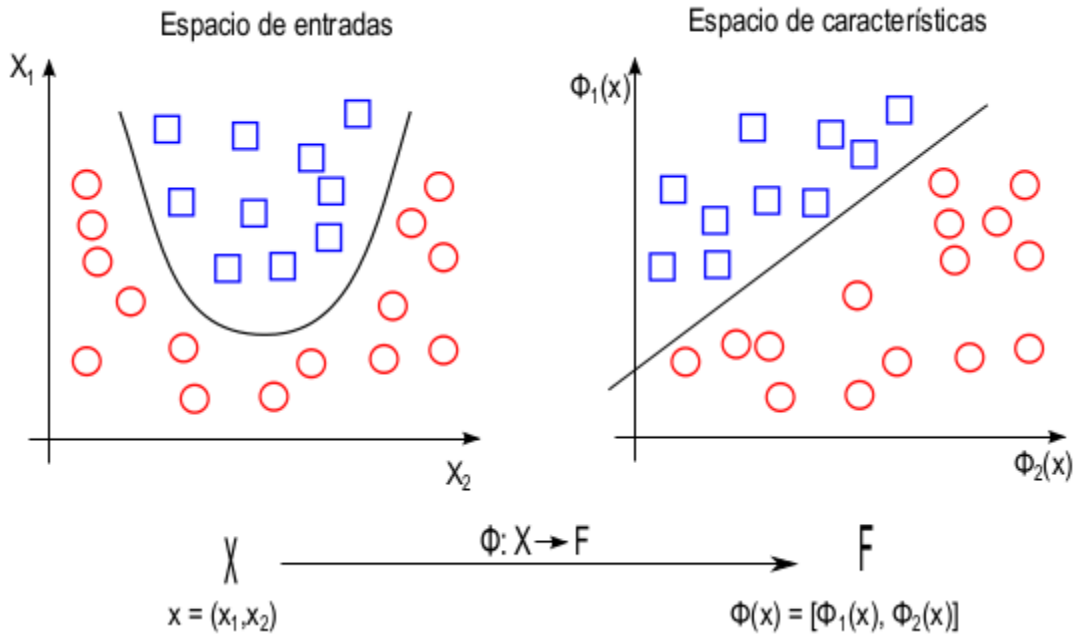


Figura 6-5. Aplicación de kernel.

Sea Φ: X → F la función de transformación que hace corresponder cada vector de entrada x con un punto en el espacio de características F, donde Φ(x) = [φ1(x),..., φm(x)] y ∃φ (x), i = 1,..., m, tal que φi(x) es una función no lineal. Nuestra función de decisión vendrá dada por:

$$D(x) \ = \ (w1 \ \varphi 1(x) + \ldots + wm \ \varphi m(x)) \ = < w, \Phi(x) > \tag{12}$$

La función de transformación se denomina kernel (K). El kernel es una función K: X×X→R que asigna a cada par de elementos de elementos de entrada, X, un valor real que se ha obtenido de la combinación de las imágenes de dichos elementos en el espacio de características. Donde Φ: X →F

$$K(x,x) \ = < \Phi(x), \Phi(x) > = \ \varphi 1(x) \ \varphi 1(x) + \ldots + \varphi m(x) \ \varphi m(x) \tag{13}$$

Existen diversas funciones Kernel:

- Kernel lineal: $K(x,x) = < x,x >$
- Kernel polinómico de grado p: $K \ p \ (x,x) \ = \ [\gamma < x,x > + \tau \ ]^{p}$
- Kernel gaussiano: $K(x,x) \ = \ exp \ (-\gamma \ ||x \ - \ x'||^2), y > 0$

- Kernel Sigmoideo: $K(x, x) = tanh(\gamma < x, x > +\tau)$

A los parámetros y, τ y p se les denomina parámetros del kernel.

6.1.2 SVM para regresión

Como ya dijimos al principio las SVM también se adaptan para problemas de regresión. Se suelen denominar con las siglas SVR.

Partiendo del conjunto de ejemplos de entrenamiento S = {(x1, y1),. . ., (x n, y n)}, donde xi ∈Rd, yi ∈R, en el que se asume que los valores yi de todos los ejemplos de S se pueden ajustar(o cuasi-ajustar) linealmente. El objetivo es encontrar los parámetros w que cumplan:

$$f(x) = (w\,1\,x\,1 + \ldots + w\,d\,x\,d) + b =< w, x > +b \qquad (14)$$

Como es cuasi-ajustable podemos relajar la condición de error. Utilizaremos la llamada función de pérdida e-insensible, Le caracterizada por una zona de anchura 2e donde el error es nulo:

$$Le(y, f(x)) = 0 \; si \; |y - f(x)| \leq e$$
$$Le(y, f(x)) = |y - f(x)| - e \; en\, otro\, caso \qquad (15)$$

Con esta función los ejemplos que quedan confinados en una región tubular definida por ±e no serán considerados vectores de soporte. De esta manera se verá reducido significativamente su número. Para resolverlo también utilizaremos un margen blando, que ya quedo definido anteriormente con el problema de clasificación. Definimos así dos variables de holgura $\xi i+$ y $\xi i-$.si $\xi i+>0$, la predicción, $f(xi)$, es mayor que su valor real, yi, en una cantidad superior a e, es decir, $f(xi) - yi > e$. En otro caso su valor sería 0. Por otro parte, si $\xi i->0$ el valor real es mayor que la predicción con una diferencia mayor que e, es decir $yi-f(xi)>e$. Para los demás casos su valor será 0. Como no puede ocurrir que un valor predicho sea simultáneamente mayor y menor que su valor real, afirmamos que: $\xi i- \cdot \xi i+ = 0$.

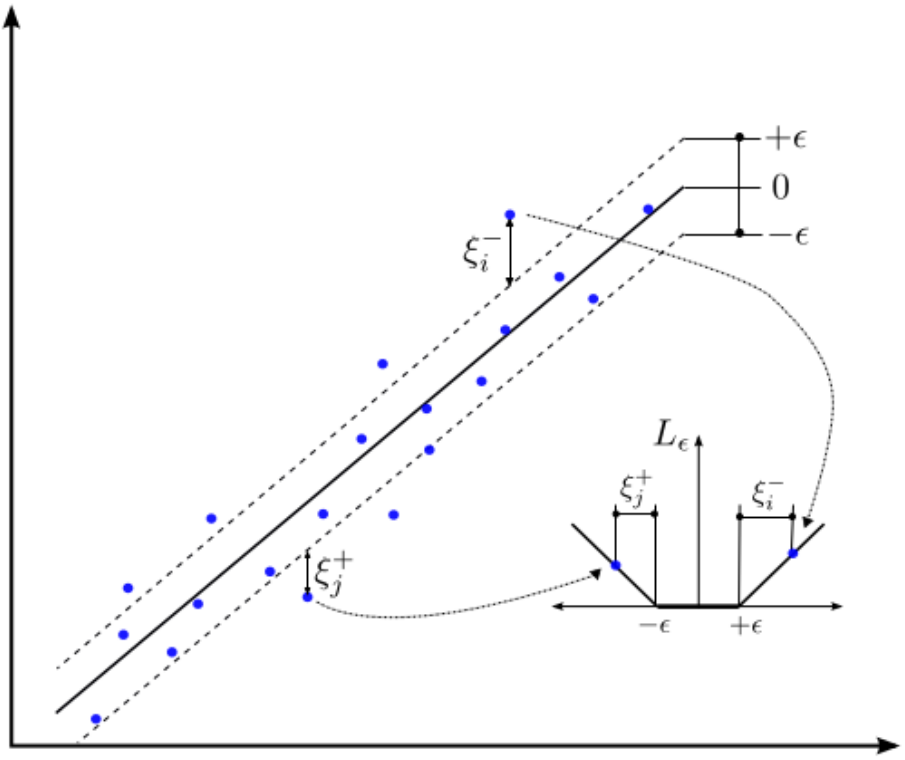


Figura 6-6. Zona insensible.

En la figura se muestra la relación entre las variables de holgura, asociadas a ejemplos situados fuera de la zona e-insensible y la función de pérdida Le.

El problema de optimización que habría que resolver, sería el siguiente:

$$\begin{aligned} \text{mín} \quad & \tfrac{1}{2} < \boldsymbol{w}, \boldsymbol{w} > + C \sum_{i=1}^{n} \left(\xi_i^+ + \xi_i^-\right) \\ \text{s.a.} \quad & (< \boldsymbol{w}, \boldsymbol{x}_i > + b) - y_i - \epsilon - \xi_i^+ \leq 0 \\ & y_i - (< \boldsymbol{w}, \boldsymbol{x}_i > + b) - \epsilon - \xi_i^- \leq 0 \\ & \xi_i^+, \xi_i^- \geq 0, \ i = 1, \ldots, n \end{aligned} \tag{16}$$

Al igual que para la clasificación, en regresión, cuando un problema no es separable linealmente aplicamos un kernel.

6.2 Resultados obtenidos

La programación se ha realizado con Python y el paquete que hemos utilizado es el *sklearn.svr* y la función *SVR*, la cual, ajustando sus atributos, obtiene la predicción. Se ha utilizado el kernel Gaussiano. Los datos se toman directamente del csv. Debido a como está programado la función *SVR* de Python, los variables de entrada se han normalizado (entre 0 y 1) y posteriormente se ha desnormalizado la salida para obtener la valores reales del precio.

- Caso 1

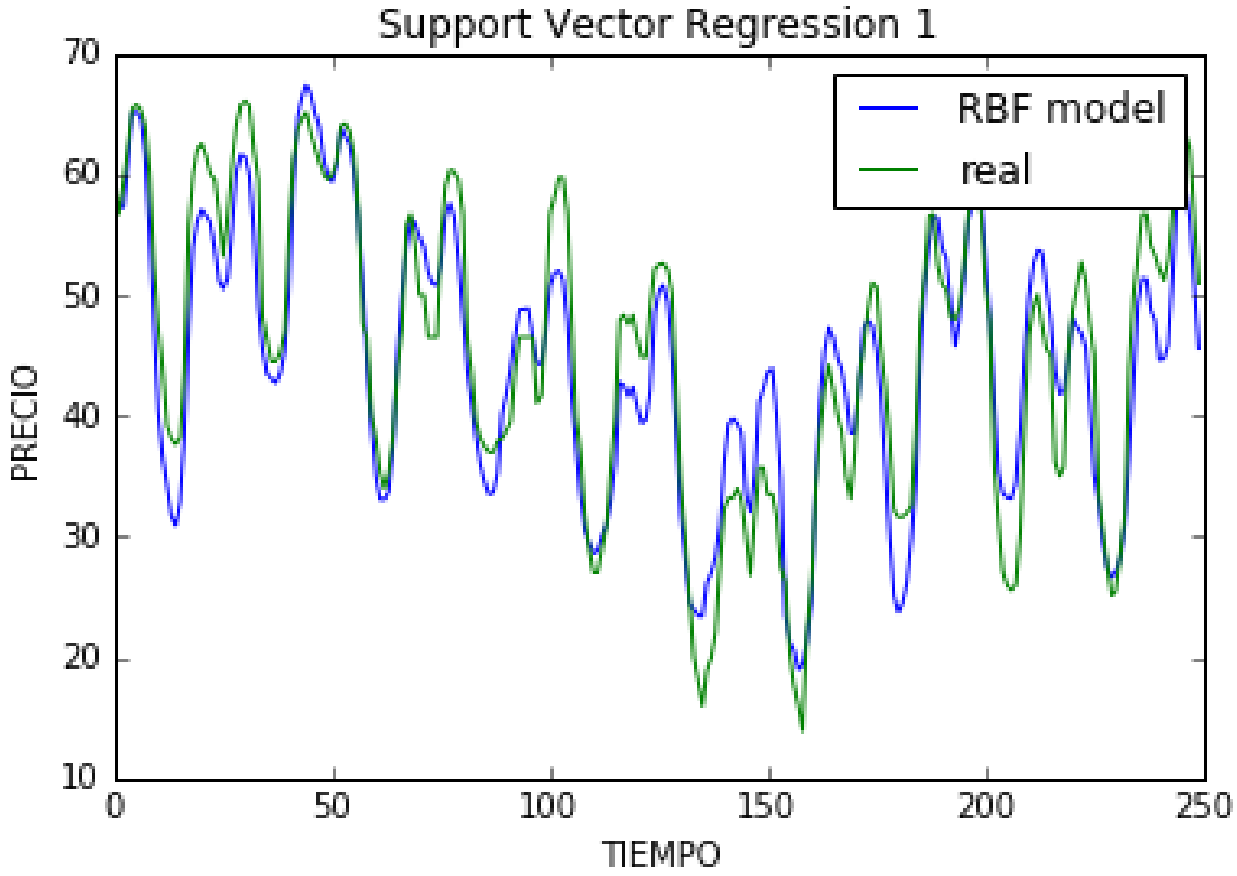


Figura 6-7. Caso 1 SVM

- Caso 2

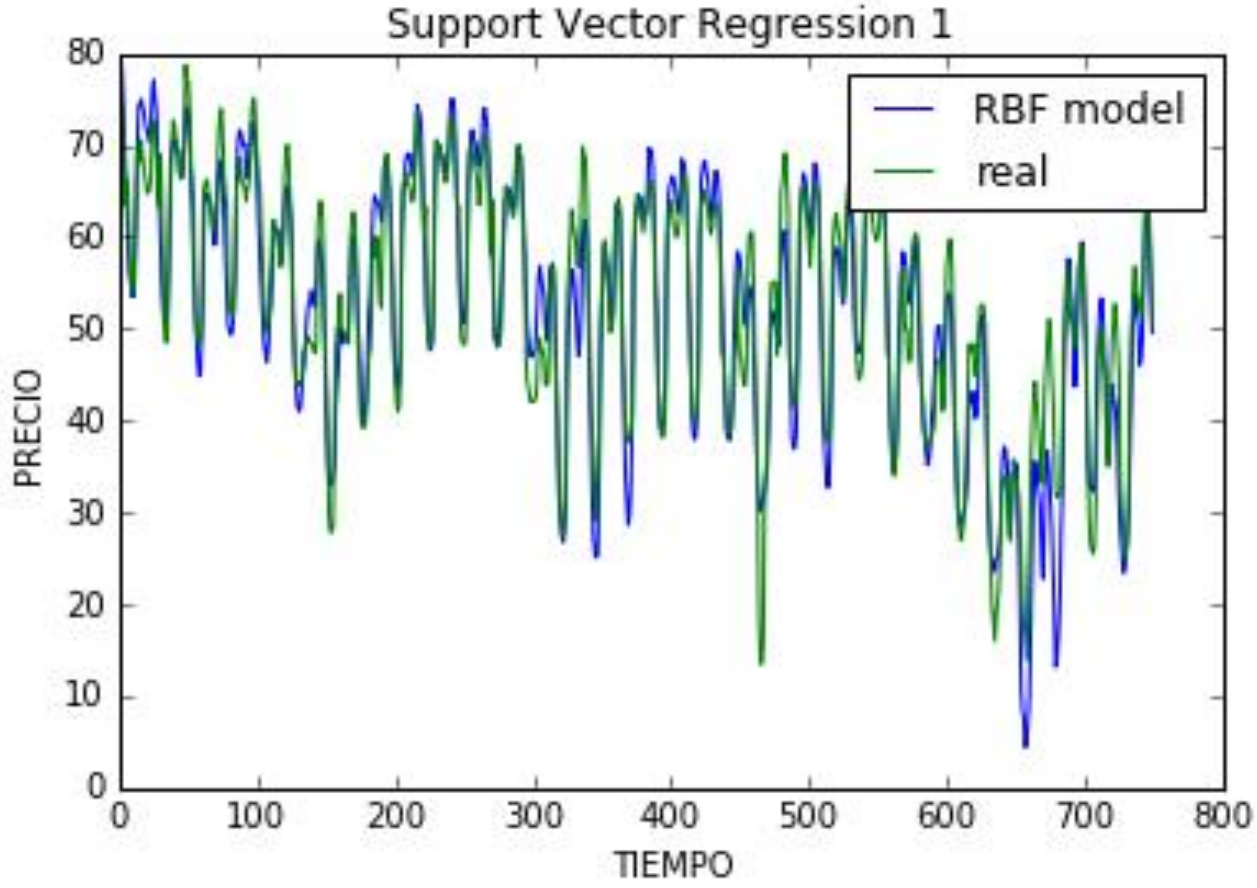


Figura 6-8. Caso 2 SVM

- Caso 3

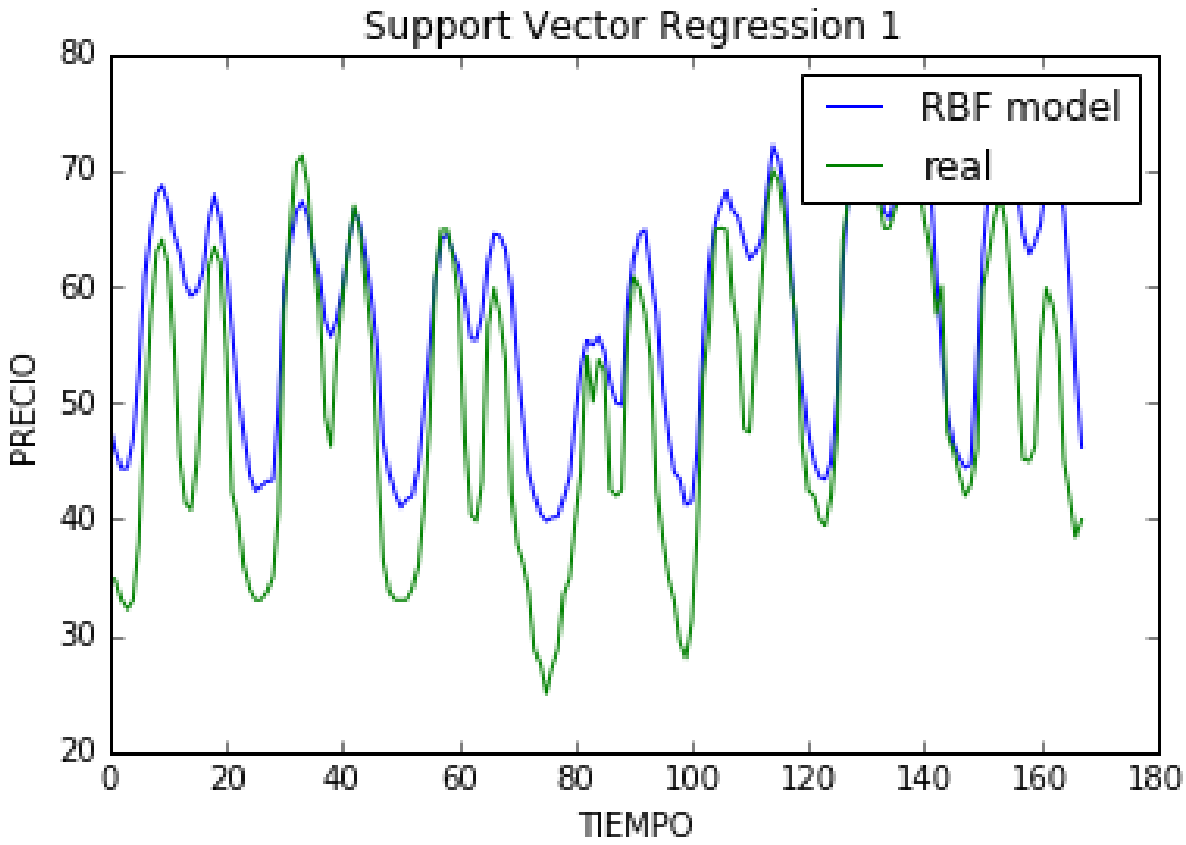


Figura 6-9. Caso 3 SVM

Tabla de errores

	MAPE (%)	RMSE	MAE
Caso 1	9.45%	0,343	3.85
Caso 2	9.86 %	0,0514	3.671
Caso 3	13.64%	0,486	7.37

Tabla 6–1. Errores SVR

El sistema tiene una respuesta muy buena para el segundo caso (RMSE=0.0514). El menor MAPE se obtiene en el caso 1.

7 CLUSTERING

7.1 Concepto

El clustering es una técnica de aprendizaje no supervisado, ya que pretendemos buscar la relación entre las variables descriptivas pero no la relación que guardan respecto a una variable objetivo.

El clustering se considera un proceso de agrupación, en el cual agruparemos mediante un criterio. De esta manera los objetos similares entre sí pertenecerán al mismo grupo y los diferentes a otro. A cada uno de estos grupos lo denominaremos cluster. Los criterios son por lo general distancia o similitud. Partiendo de que no existen clases predefinidas los resultados que obtendremos dependerán del algoritmo que utilicemos, el conjunto de datos disponible y de la medida de similitud que utilicemos para comparar objetos.

Para obtener la similitud entre dos ejemplos no tenemos por qué utilizar todos los atributos de los que disponemos y hay que tener cuidado con las magnitudes de cada variable; por esto segundo, para evitar que unas variables dominen sobre otras se suele normalizar. Las medidas de similitud se expresan en términos de distancias:

d (i, j) > d (i, k), esto indica que el objeto i es más parecido a k que a j

Las medidas de distancias verifican las siguientes propiedades:

- d(i,j)=0 si y solo si i=j
- d(i,j) = d(j,i)
- d(i,j) <= d(i,k)+d(k,j)

Métodos para calcular distancias

- Distancia de Minkowski:

$$d_r(x,y) = \left(\sum_{j=1}^{J} |x_j - y_j|^r\right)^{\frac{1}{r}}, r = >1$$

- Distancia de Manhattan(r=1):

$$d_1(x,y) = \sum_{j=1}^{J} |x_j - y_j|$$

- Distancia euclídea(r=2):

$$d_2(x,y) = \sqrt{\sum_{j=1}^{J}(x_j - y_j)^2}$$

- Distancia de Chebyshev(r→ ∞)

$$d_\infty(x,y) = máx_{j=1\ldots J}\,|x_j - y_j|$$

Existen muchas más medidas de similitud como el denominado "vecinos compartidos", los basados en medidas de correlación o los basados en la teoría de conjuntos. Existen los siguientes métodos de agrupamiento: agrupamiento por particiones, por densidad y clustering jerárquico.

7.1.1 Agrupamiento por particiones (kmeans)

En este tipo de agrupamiento, el número de clusters, k, se suele fijar a priori. Cada cluster tiene asociado un centroide (centro geométrico). Cada uno de los ejemplos se asociara al centroide que esté más cerca (utilizando cualquier método para la distancia). A medida que vamos asignando más puntos a cada cluster, los centroides de estos irán cambiando. El proceso de clustering llamado kmeans sigue el siguiente esquema:

1. Escoger k centroides aleatoriamente(evidentemente existen otros métodos que no eligen los primeros centroides aleatoriamente)
2. Formamos k grupos, asignando a cada punto su centroide más cercano.
3. Proceso iterativo(hasta que el valor de los centroides siga cambiando):
    - Cálculo de las distancias de los puntos a los centroides.
    - Formar k grupos, asignando a cada punto su centroide más cercano.
    - Cálculo de los nuevos centroides.

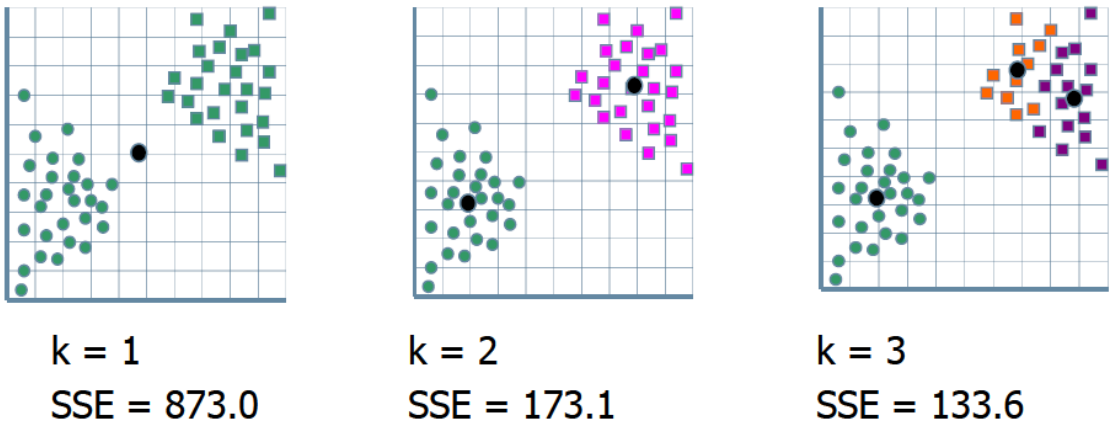


Figura 7-1. Kmeans clusters.

En la representación anterior se agrupan una serie de ejemplos con 1, 2 y 3 cluster respectivamente(los centroides son los puntos negros). En este caso con 3 cluster obtenemos el menor SSE.

Se podría decir entonces, que la complejidad de nuestro problema depende del número de puntos, clusters, iteraciones y de propiedades.

Los centroides se recalcularan partiendo de los nuevo grupos que se han hecho. A cada centroide se le denomina *mi*. Se elige una función objetivo y se escogen los valores de *mi* que minimizan dicha función:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} d^2(m_i, x)$$

En el caso de la distancia euclídea, el SSE se minimiza utilizado la media aritmética de cada atributo variable. La media es el criterio que se utiliza generalmente ya que da resultado con muchas métricas. Existen otros métodos para recalcular el centroide, como son la mediana o la moda.

7.1.2 Agrupamiento por densidad

Este tipo de clusters son útiles cuando existen formas irregulares, hay ruido o los datos están entrelazados; de esta manera podemos separar regiones densas frente a otras regiones menos densas, identificando clusters de formas arbitrarias:

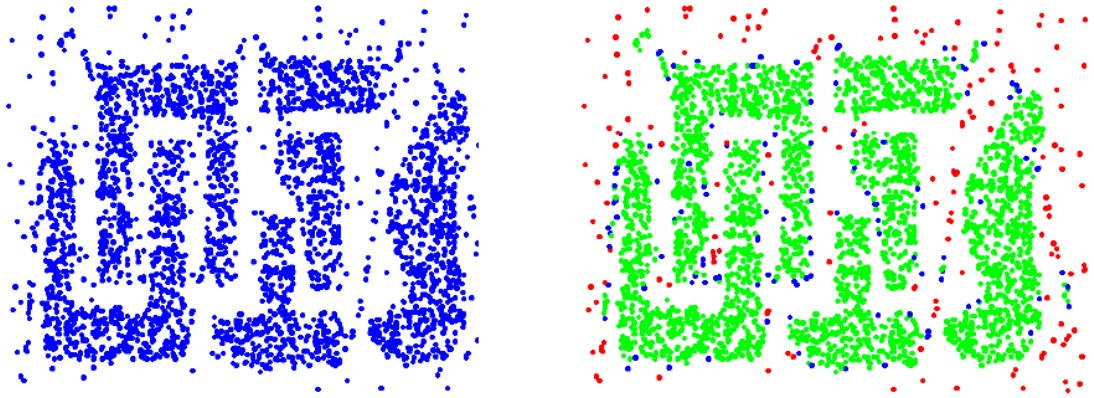


Figura 7-2. Clustering por densidad.

En este ejemplo observamos como originariamente teníamos la distribución de puntos de la izquierda y como tras la aplicación de clustering obtenemos puntos verdes (núcleo), azules (frontera) y rojos (ruido).

7.1.3 Agrupamiento jerárquico

En este tipo de clustering la similitud entre dos objetos viene dada el nodo común más cercano. El cluster jerárquico no elige a priori el número de clusters, se obtiene como resultado final un número de agrupamientos

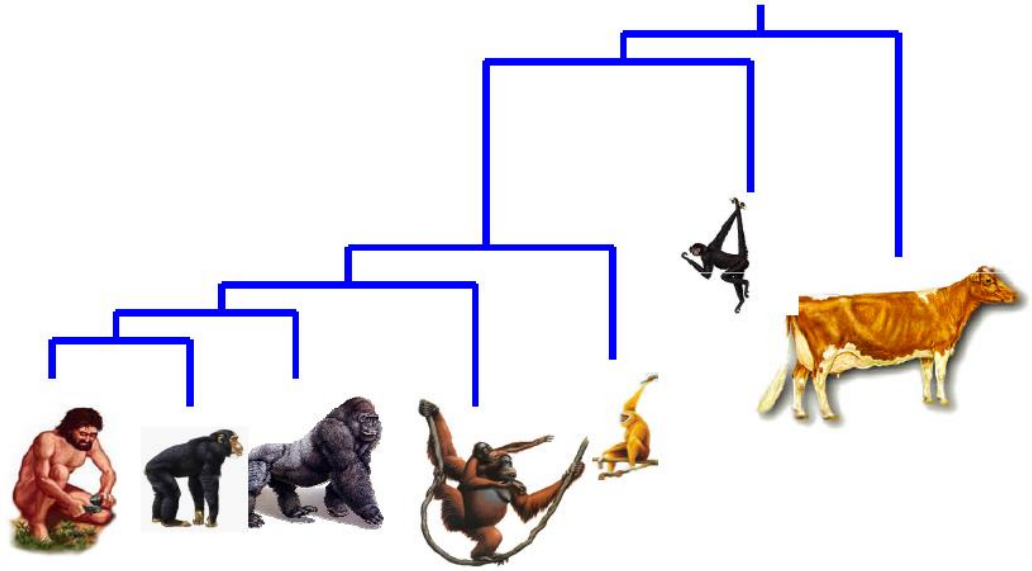


Figura 7-3. Clustering jerárquico.

Este tipo de gráfico se denomina dendograma que como observamos es una especie de árbol que organiza los datos en subcategorías que se van dividiendo en otros hasta llegar al nivel de detalle deseado.

Existen dos técnicas de clustering jerárquico:

- Técnicas aglomerativas: comenzamos con cada caso como cluster individual y a cada paso, combinar los pares de cluster más cercanos.
- Técnicas divisivas: comenzamos con un cluster que engloba a todos. En cada paso iremos partiendo clusters hasta que cada uno contenga un único caso.

El clustering jerárquico puede servir para solucionar el problema del número de cluster a priori que se toman en el clustering por particiones.

7.2 Resultados obtenidos

El algoritmo que se ha programado primero realizara clustering y después aplicará regresiones a los grupos obtenidos. La estructura es la siguiente:

1. Seleccionamos los datos(entrenamiento y validación)
2. Aplicamos agrupamiento por particiones (2 clusters) al conjunto de entrenamiento y al de prueba.
3. Para cada grupo separado:
    a. Entrenamos el mecanismo regresivo con el conjunto de entrenamiento
    b. Validamos el mecanismo regresivo con el conjunto de prueba
4. Obtenemos una regresión conjunta

Solamente se ha considerado el estudio del caso 2, ya que en los otros dos casos los conjuntos de datos son más pequeños y no tenemos suficiente información para obtener una buena regresión.

Para realizar el clustering se ha utilizado la función de Python *Kmeans* perteneciente a la librería *sklearn.cluster* y después se ha optado por aplicar RF y SVR. Solamente se ha aplicado sobre RF y SVR ya que estos también están programados en Python. El mejor resultado se ha obtenido aplicando solamente dos clusters y utilizando

solamente 2 variables de las 4 que tenemos (demanda prevista y generación energía eólica):

- Clustering con RF:

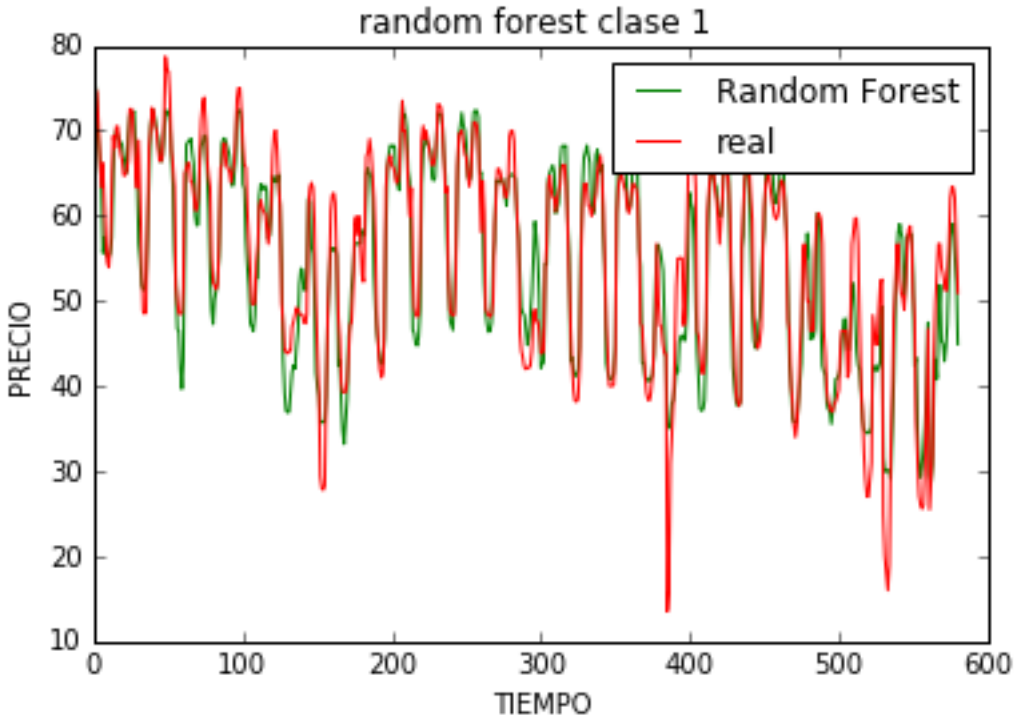


Figura 7-4. RF clase1

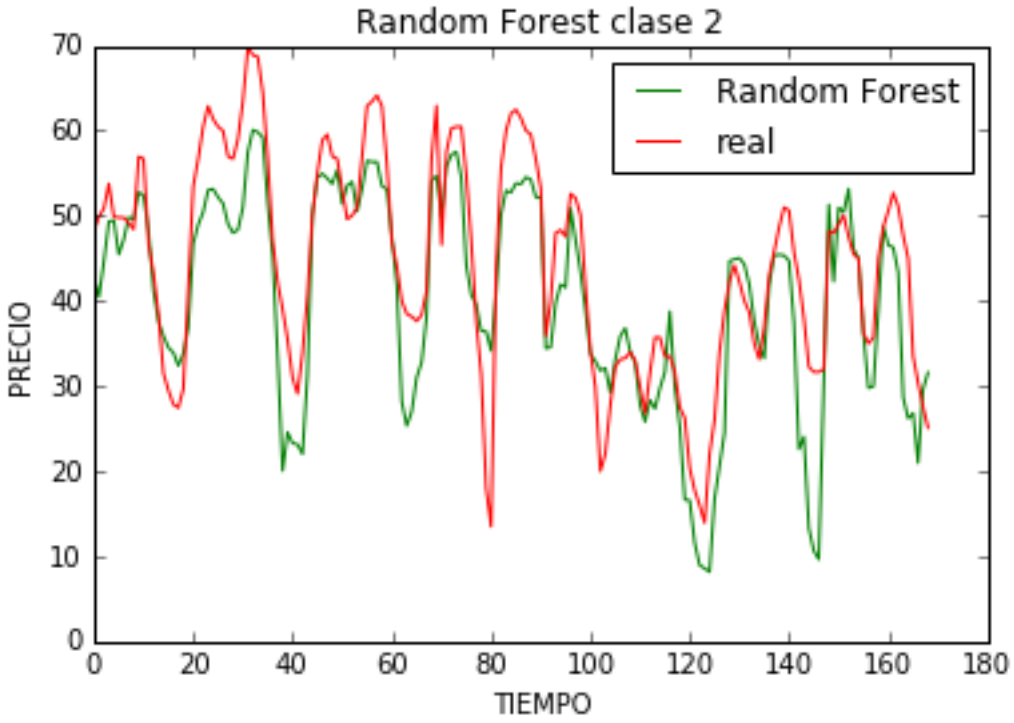


Figura 7-5. RF clase2

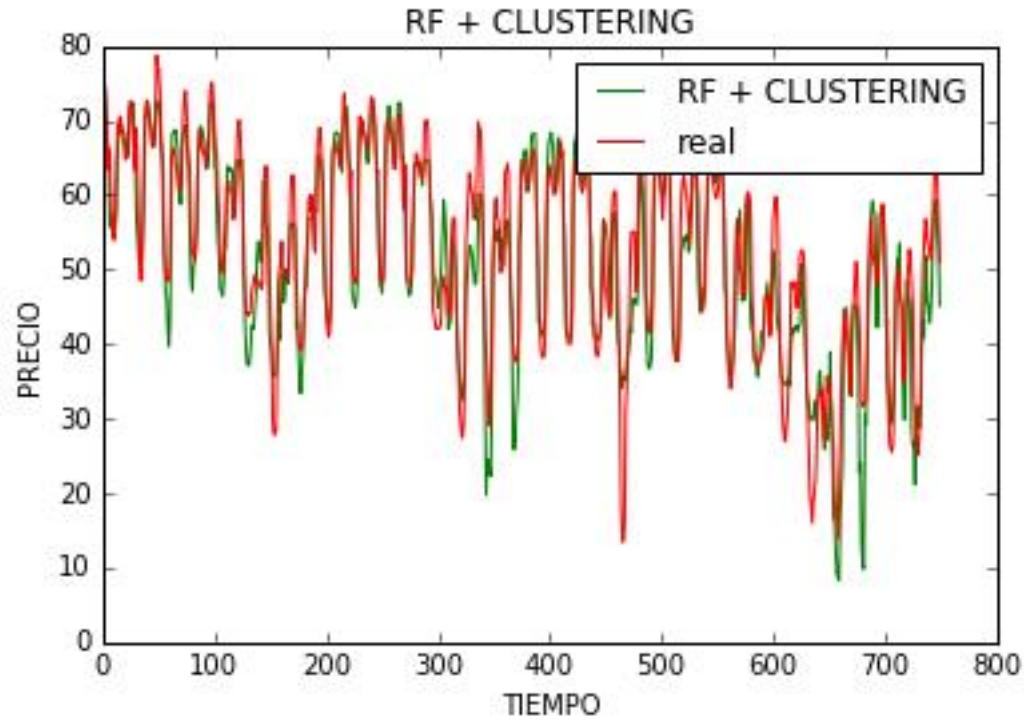


Figura 7-6. Clase1 +Clase2 (RF)

Tabla de errores:

RF	MAPE (%)	RMSE	MAE
Clase 1	6.78%	0.25	3.31
Clase 2	20.93 %	0,500	5.95
Clase1+ Clase2	9.04%	0,22	3.91

Tabla 7–1. Errores RF+Clustering

En la primera gráfica observamos una respuesta muy buena; esto es podría deberse a que la clase1 es un conjunto 3 veces más grande que la clase2, por lo que ha tenido más valores para entrenar. La regresión de la clase2 no es muy buena, tenemos unos valores de error bastante altos y además en la gráfica se perciben amplias desviaciones entre el valor real y el predicho. El resultado final será posteriormente comparado con el resultado de este caso aplicando solo RF.

- Clusterig con SVR

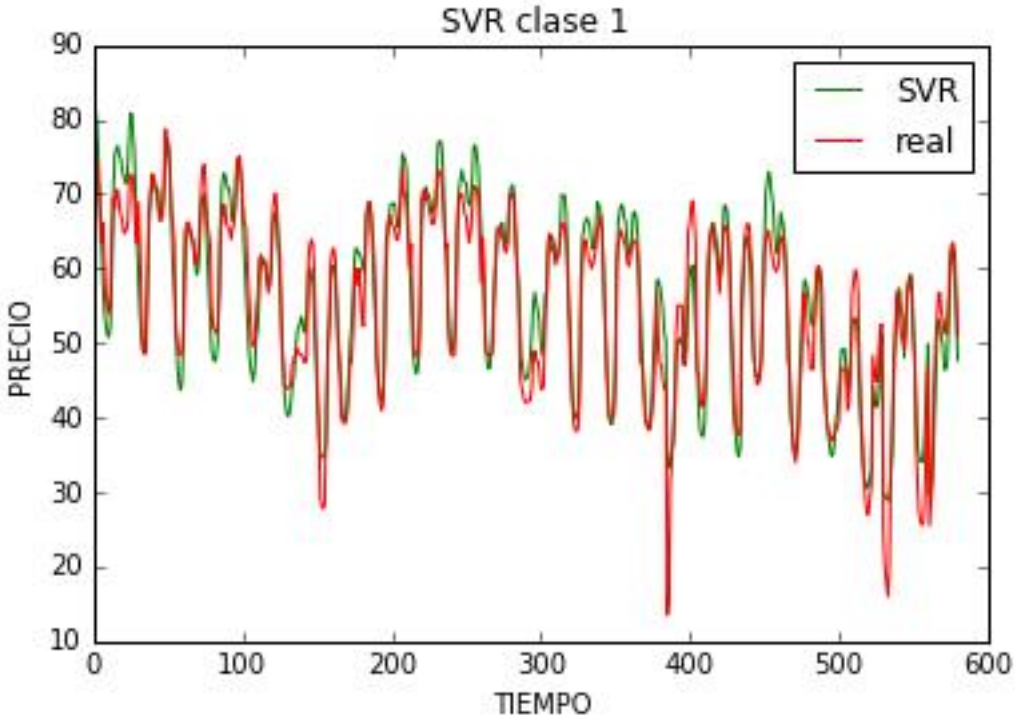


Figura 7-7. SVR clase1

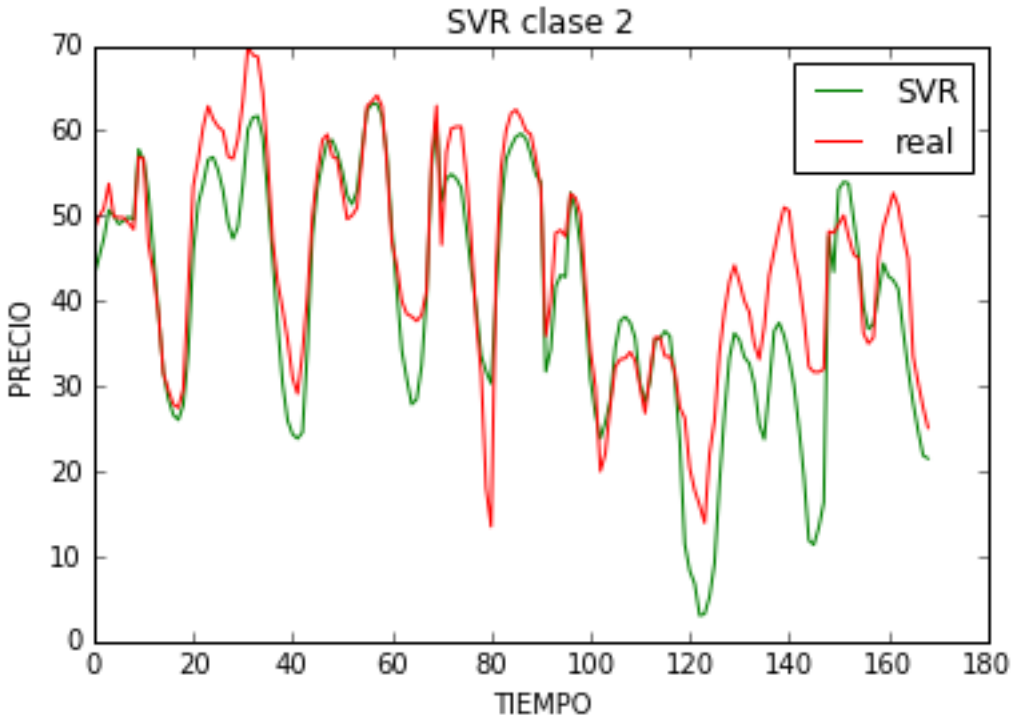


Figura 7-8. SVR Clase2

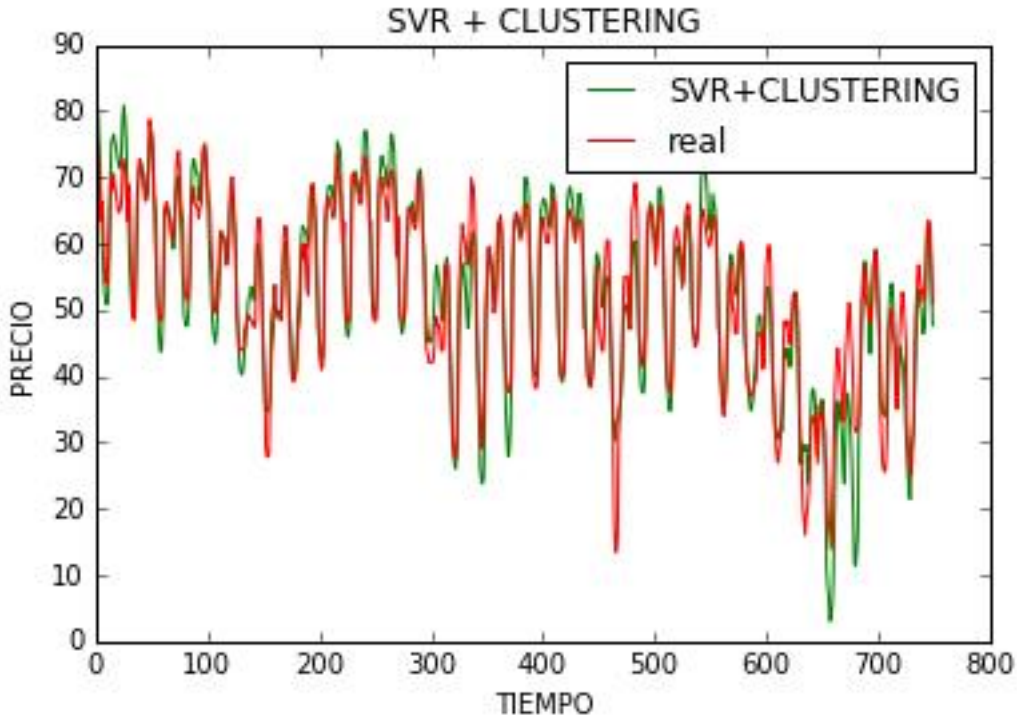


Figura 7-9. Clase1 + Clase2 (SVR)

SVR	MAPE (%)	RMSE	MAE
Clase 1	6.23	0.138	3.246
Clase 2	26.55	0.283	5.572
Clase1 + Clase2	8.74	0.121	3.770

Tabla 7–2. Errores RF+SVR

Al igual que las gráficas obtenidas para el RF con clusteringm tenemos un resultado muy bueno para una de las clases (clase que hace referencia al grupo más grande) y otra con un resultado muy malo. El resultado final será posteriormente comparado con el resultado de este caso aplicando solo SVR.

8 COMPARATIVA Y CONCLUSIONES

8.1 Comparativa

Para comparar cada uno de los métodos, vamos a representar el error de cada uno de ellos mediante un histograma. La comparación se ha realizado para cada uno de los tres casos de estudio:

- Caso 1

Mínimos cuadrados

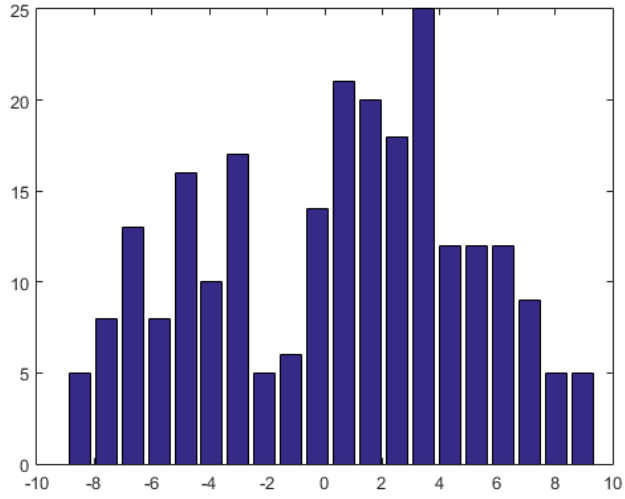


Figura 8-1. Hist. Min. Cuad. C1

Mínimos cuadrados recursivos λ=0.95

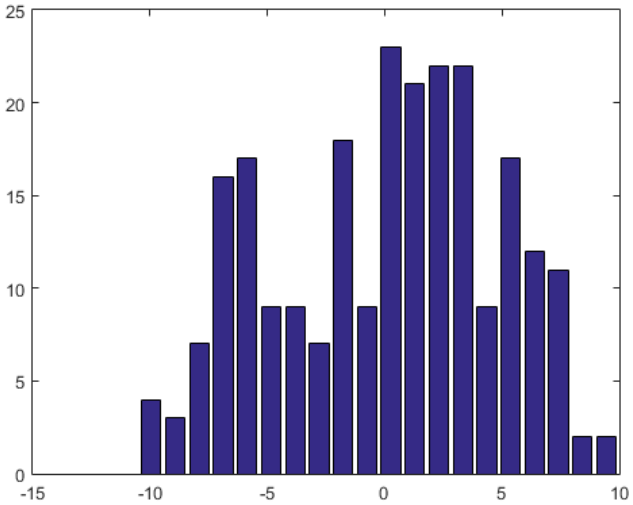
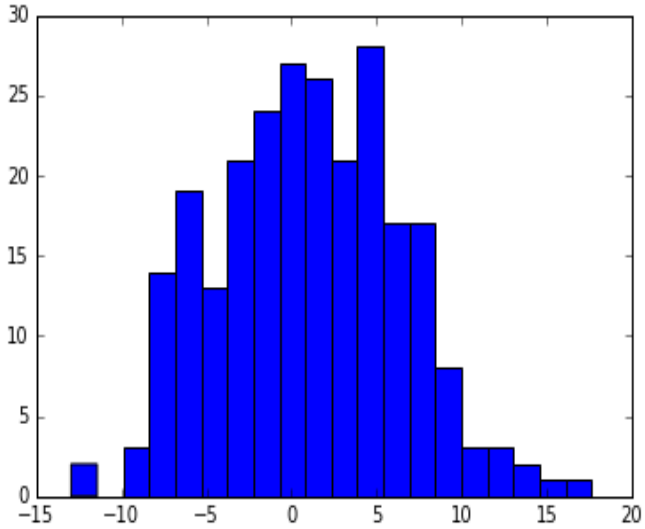


Figura 8-2. Hist. Min. Cuad. R. C1

Random Forest



Máquinas de soporte vectorial

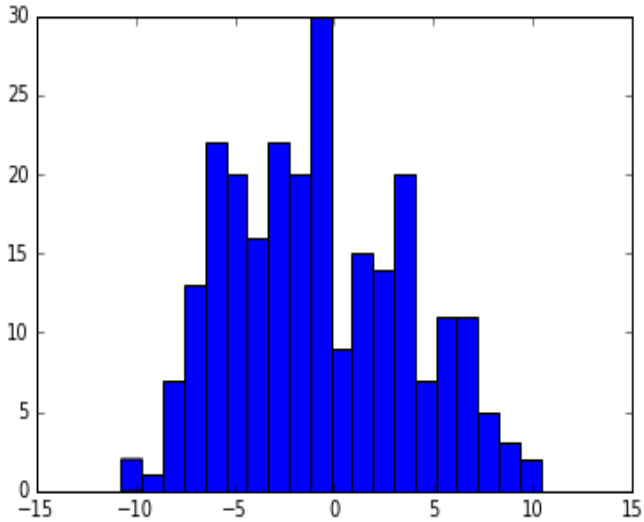


Figura 8-3. Hist. RF C1

Figura 8-4. Hist. SVR. C1

Para las cuatro figuras anteriores observamos que el error está acotado en el intervalo [-10,10] aproximadamente. Si observamos la tercera figura, obtenida con RF, se ve claramente que tiene la mejor distribución del error ya que se asemeja a la forma de la campana de Gauss a diferencia de las demás. Cabe también decir que el intervalo de valores de error es el de la primera figura, la obtenida con mínimos cuadrados.

- Caso 2

Mínimos cuadrados

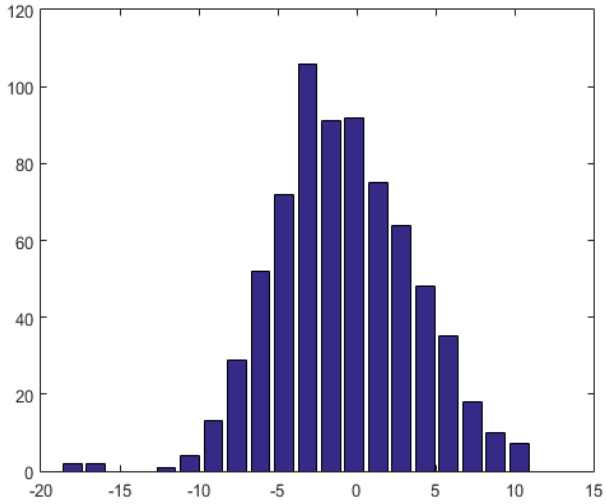


Figura 8-5. Hist. Min. Cuad. C2

Mínimos cuadrados recursivos λ=0.95

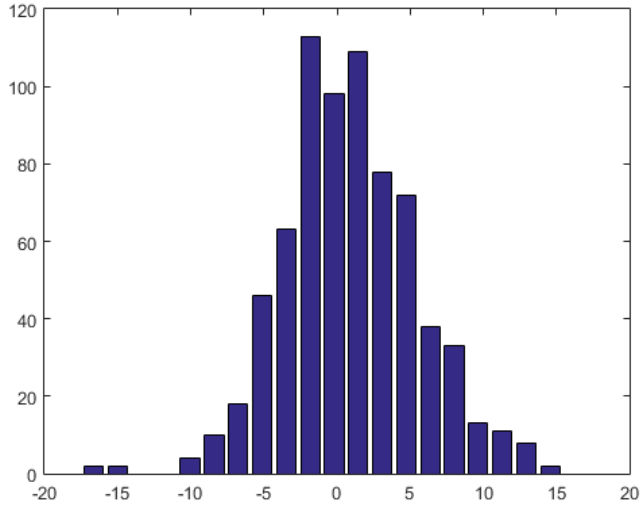


Figura 8-6. Hist. Min. Cuad. R. C2

Random Forest

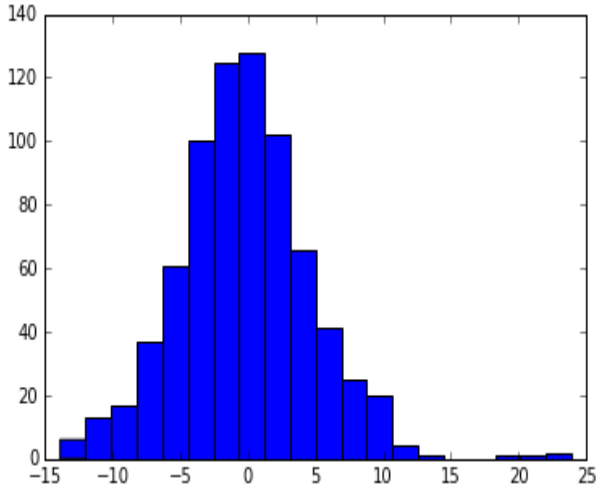


Figura 8-7. Hist. RF C2

Máquinas de soporte vectorial

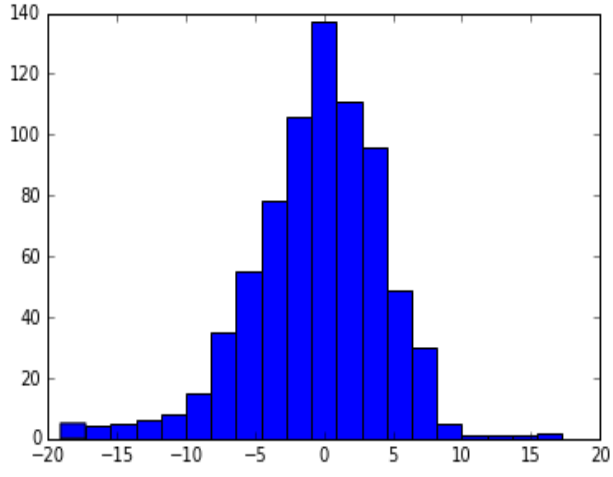


Figura 8-8. Hist. SVR. C2

Clusterng+Random Forest

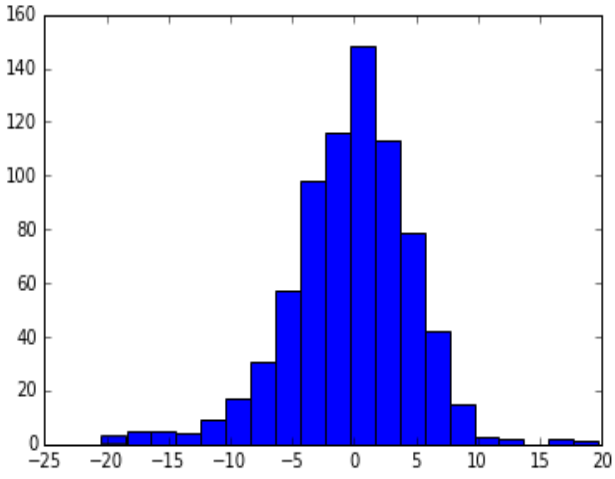


Figura 8-9. Hist. Clust. RF C2

Clusterng+SVR

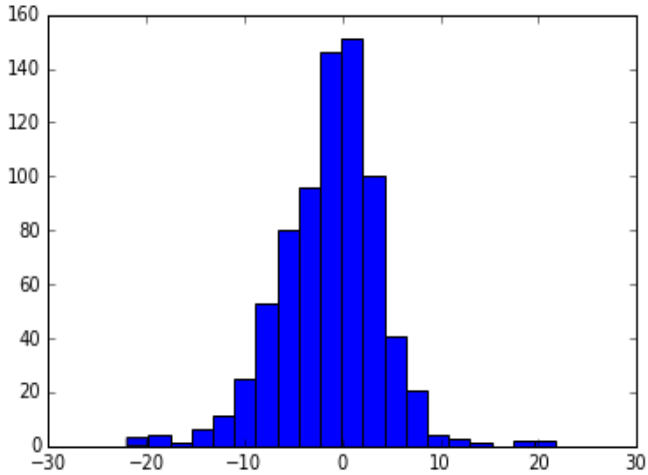


Figura 8-10. Hist. Clust SVR. C2

A simple vista se puede observar que este caso es el que mejor se ha predicho, pese a que los errores estén dentro de un intervalo más grande ya que esto es debido a que nuestro conjunto de datos es más grande. En cuanto a la distribución del error, es muy similar en las 6 gráficas.

Si comparamos las gráficas de RF y SVR con sus homologas pero con Clustering, observamos que las segundas tienen un intervalo de error mayor pero también tienen más valores cercanos al 0 que el resto. Esto se debe a que al separar el conjunto con los dos cluster, se obtiene una regresión muy buena pero también una muy mala; por lo tanto aumentan puntos con buena precisión pero también aumentan los puntos con poca. La aplicación de clustering no mejora prácticamente la predicción con respecto a la obtenida sin la aplicación del mismo.

Las gráficas de mínimos cuadrados tienen el intervalo de error más pequeño que el resto, especialmente la primera de las dos.

- Caso 3

Mínimos cuadrados

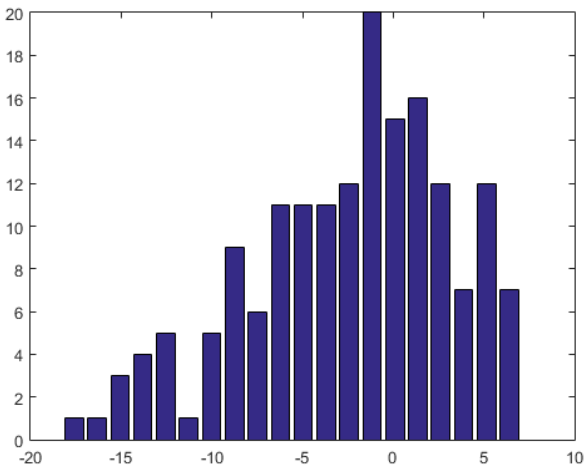


Figura 8-11. Hist. Min. Cuad. C3

Mínimos cuadrados recursivos λ=0.95

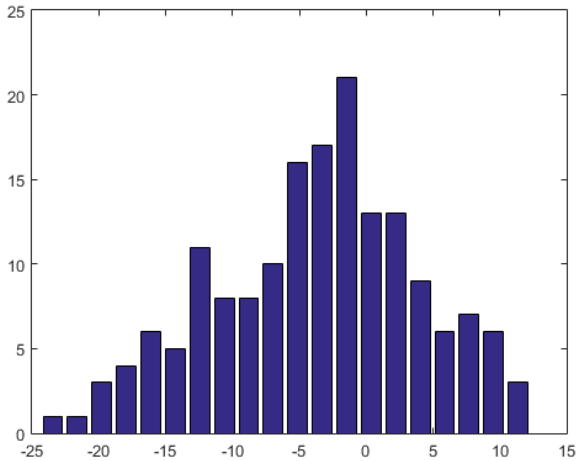


Figura 8-12. Hist. Min. Cuad. R. C3

Random Forest

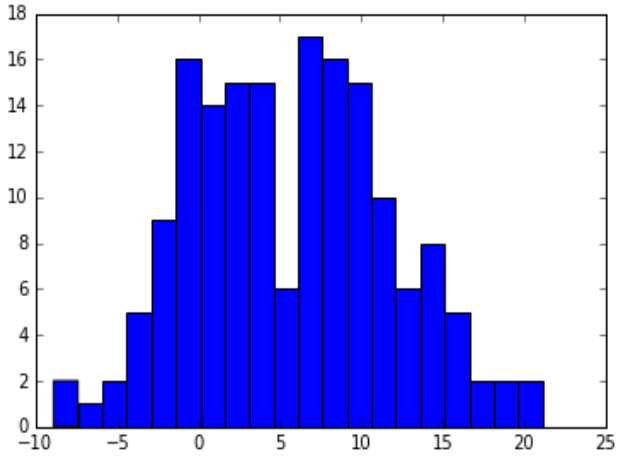


Figura 8-13. Hist. RF C3

Máquinas de soporte vectorial

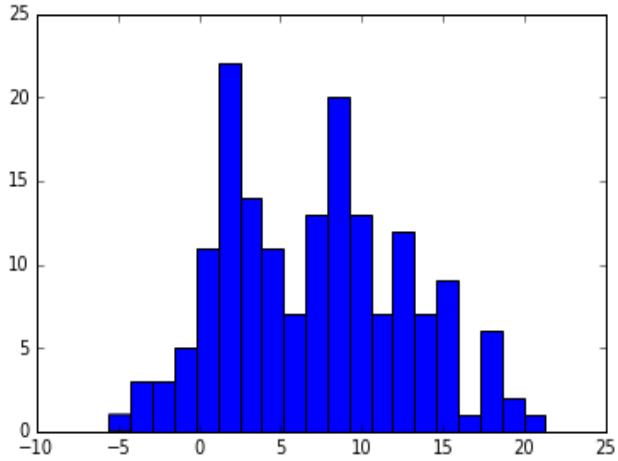


Figura 8-14. Hist. SVR. C3

La distribución del error en estas últimas gráficas es bastante irregular, a diferencia del caso anterior; aparentemente las que tienen más forma de campana son las de mínimos cuadrados. Hay que aclarar que el conjunto de datos de este caso es el más pequeño de los tres que tenemos, por lo que es normal que el error sea tan grande.

8.2 Conclusiones

Los resultados que se han obtenido con todos los metodos nos son muy dispares entre si. En el caso 2, que es el que tiene el conjunto de datos más grande se han obtenido los mejores resultados; los resultados de este caso para RF son mejores que los de el resto pero el tiempo de calculo es muy superior al resto debido a que para obtenerlo tenemos que utilizar 2000 arboles.

En todas las graficas de predicción precio/tiempo se observa como existen resultados con desvios muy grandes respecto su valor real y que no somos capaces de predecir con ninguno de los métodos.

El Clustering no consigue mejorar practicamente nada el resultado total de las predicciones, sin embargo da una respuesrta muy buena para los días que son de una clase asignada(por lo general la clase más numerosa). Las restricciones que hemos encontrado en el método *kmeans* son las siguientes:

- Sensible a elección inicial de centroides.
- El cálculo de centroides por media es sensible a valores anómalos.
- K-means no siempre da una buena respuesta para cluster de distinto tamaño, densidad diferente o no convexo.
- Para una variable como el precio de la enérgia el Clusterig necesitaría más datos para funcionar mejor.

8.3 Futuras líneas

Para mejorar los resultados en la predicción del precio, un paso lógico sería mejorar el conjunto de datos con el que estamos trabajando, se podría mejorar: adquiriendo datos de 2016 para poder predecir fechas con conjuntos de datos de las mismas fechas del año anterior, también podriamos aumentar el número de variables predictoras para aumentar más así los datos, realizar un pre-procesamiento de datos para detector valores anómalos y eliminarlos; también utilizando conjuntos más grandes, el clustering funcioanaría mejor. Otro manera de conseguir una mejor predicción sería intentando introducir en nuestro modelo de predicción el Mercado Intradiario que es el mercado que se abre para cada hora con el objetivo de cuadrar desvios imprevistos.

Con las herramientas de programación que se han desarrollado se podrían predecir muchas variables. Algo interesante de predecir serían por ejemplo los desvíos de en las predicciones del mercado de la energía eléctrica, también se podrían aplicar a otros campos de studio, como resultados deportivos o la bolsa.

9 CÓDIGO

9.1 Mínimos cuadrados

```
clear all
hold off
precio=xlsread('prec.xls');
generacion=xlsread('gene.xls');
demanda=xlsread('deman.xls');
precio=precio(:,5);
generacion=generacion(:,5);
demanda=demanda(:,5);
[m,n]=size(demanda);
%tiene que ser preveido a partir del 8 de enero
% dias=input('Introduce el numero de dias que quieres usar para el conjunto
de entrenamiento: ');
% dia(1)=input('Introduce mes:');
% dia(2)=input('Introduce dia:');
dias=7;
dia(1)=1;
dia(2)=10;
if dia(1)==1
    sup=0;
end
if dia(1)==2
    sup=24*31;
end
if dia(1)==3
    sup=24*31+24*28;
end
if dia(1)==4
    sup=24*31+24*28+24*31;
end
if dia(1)==5
    sup=24*31+24*28+24*31+24*30;
end
if dia(1)==6
    sup=24*31+24*28+24*31+24*30+24*31;
end
if dia(1)==7
    sup=24*31+24*28+24*31+24*30+24*31+24*30;
end
if dia(1)==8
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31;
end
if dia(1)==9
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31;
end
if dia(1)==10
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30;
end
if dia(1)==11
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31;
end
if dia(1)==12
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31+24*31;
end
localizador=sup+(dia(2)-1)*24;
s=1;
```

```
for t=localizador:(dias*24+localizador)
    x0(s)=precio(t-7*24);
    x1(s)=precio(t-24);
    x2(s)=demanda(t);
    x3(s)=generacion(t);

    sol(s)=precio(t);
    s=s+1;
end

x=[x0;x1;x2;x3;sol]';
intt=[ones(size(x0));x0;x1;x2;x3]';
[ndatos,nvar]=size(x);
m=nvar-1;           % Numero de variables dependientes (x1, x2,....)
valoresx=x(1:ndatos,1:m);
unos=ones(1,ndatos);
R=[unos;valoresx']; % Se introduce una fila de unos
C=R*R';             % Construimos la matriz C
y=x(:,nvar);        % la última columna es la variable independiente
b=R*y;              % Vector
[L,U,P]=lu(C);      % Factorizacion de la matriz C
% A partir de aqui se resuelve el sistema lineal a=C*b
zeta=P*b;
ygr=inv(L)*zeta;
a=inv(U)*ygr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% diax=input('Introduce el numero de dias que quieres usar para el conjunto
de prueba: ');
% di(1)=input('Introduce mes:');
% di(2)=input('Introduce dia:');
diax=7;
di(1)=1;
di(2)=18;
if di(1)==1
    su=0;
end
if di(1)==2
    su=24*31;
end
if di(1)==3
    su=24*31+24*28;
end
if di(1)==4
    su=24*31+24*28+24*31;
end
if di(1)==5
    su=24*31+24*28+24*31+24*30;
end
if di(1)==6
    su=24*31+24*28+24*31+24*30+24*31;
end
if di(1)==7
    su=24*31+24*28+24*31+24*30+24*31+24*30;
end
if di(1)==8
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31;
end
if di(1)==9
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31;
end
if di(1)==10
```

```
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30;
end
if di(1)==11
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31;
end
if di(1)==12
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31+24*31;
end
localizado=su+(di(2)-1)*24;
s=1;
for p=localizado:(diax*24+localizado)
    rp0(s)=precio(p-7*24);
    rp1(s)=precio(p-24);
    rp2(s)=demanda(p);
    rp3(s)=generacion(p);
    rol(s)=precio(p);
    s=s+1;
end



r=[rp0;rp1;rp2;rp3;rol]';
intr=[ones(size(rp0));rp0;rp1;rp2;rp3]';
yr=rol';
%una vez preparado, a calcular errores
[m,n]=size(yr);
mape=0;
rmse=0;
mae=0;
yest=intr*a;
errorp=ones(1,m);
for i=1:m
    errorp(1,i)=(yr(i)-yest(i));
    mape=mape+abs((yr(i)-yest(i))/yr(i)); %error porcentual absoluto medio
    rmse=sqrt((rmse+(yr(i)-yest(i))^2)/m); %root mean squared error
    mae=mae+(abs(yr(i)-yest(i))/m); %error absoluto medio
end
mape=mape/m;
errores=[mape rmse mae ]
time=1:m;
time=time';
plot(time,yr,'r--')
hold on
plot(time,yest,'b')
legend('Real','Estimado');
title('MÍNIMOS CUADRADOS');
ylabel('PRECIO(MWh)');
xlabel('TIEMPO(h)');
figure
[n,p] = hist(errorp,20);
bar(p,n);
```

9.2 Mínimos cuadrados recursivos

```
clear all
precio=xlsread('prec.xls');
generacion=xlsread('gene.xls');
demanda=xlsread('deman.xls');
precio=precio(:,5);
generacion=generacion(:,5);
demanda=demanda(:,5);
[m,n]=size(demanda);
%tiene que ser preveido a partir del 8 de enero
% dias=input('Introduce el numero de dias que quieres usar para el conjunto
de entrenamiento: ');
% dia(1)=input('Introduce mes:');
% dia(2)=input('Introduce dia:');
dias=7;
dia(1)=1;
dia(2)=10;
if dia(1)==1
    sup=0;
end
if dia(1)==2
    sup=24*31;
end
if dia(1)==3
    sup=24*31+24*28;
end
if dia(1)==4
    sup=24*31+24*28+24*31;
end
if dia(1)==5
    sup=24*31+24*28+24*31+24*30;
end
if dia(1)==6
    sup=24*31+24*28+24*31+24*30+24*31;
end
if dia(1)==7
    sup=24*31+24*28+24*31+24*30+24*31+24*30;
end
if dia(1)==8
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31;
end
if dia(1)==9
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31;
end
if dia(1)==10
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30;
end
if dia(1)==11
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31;
end
if dia(1)==12
    sup=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31+24*31;
end
localizador=sup+(dia(2)-1)*24;
s=1;
for t=localizador:(dias*24+localizador)
    x0(s)=precio(t-7*24);
    x1(s)=precio(t-24);
    x2(s)=demanda(t);
    x3(s)=generacion(t);
    sol(s)=precio(t);
```

```
    s=s+1;
end
x0=x0(:,1:s-2)';
x1=x1(:,1:s-2)';
x2=x2(:,1:s-2)';
x3=x3(:,1:s-2)';
sol=sol(:,1:s-2)';
% Calculo de la predicción SSN mes siguiente
horas=dias*24;
% precio_P=nan(horas,1); % Método Persistente
precio_est=nan(horas,1); % Método Mínimos Cuadrados (Least Squares)
dim_r=4; % Dimensión del regresor
lambda=0.95 % Factor de olvido
Set_Theta=nan(horas,dim_r);
theta=(1/dim_r)*ones(dim_r,1);
P_old=0.002*eye(dim_r);

for kk=1+dim_r:horas
%     precio_P(kk)=precio(kk-1);
    m_act=[x0(kk) x1(kk) x2(kk) x3(kk)];
    K=(P_old*m_act')/(lambda+m_act*P_old*m_act');
    precio_est(kk)=m_act*theta;
    if (isfinite(precio_est(kk))&&(isfinite(sol(kk))))
        error_ls=sol(kk)-precio_est(kk);
        theta=theta+K*(error_ls);
        P_old=(1/lambda)*(eye(dim_r)-K*m_act)*P_old;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% diax=input('Introduce el numero de dias que quieres usar para el conjunto
de prueba: ');
% di(1)=input('Introduce mes:');
% di(2)=input('Introduce dia:');
diax=7;
di(1)=1;
di(2)=18;
if di(1)==1
    su=0;
end
if di(1)==2
    su=24*31;
end
if di(1)==3
    su=24*31+24*28;
end
if di(1)==4
    su=24*31+24*28+24*31;
end
if di(1)==5
    su=24*31+24*28+24*31+24*30;
end
if di(1)==6
    su=24*31+24*28+24*31+24*30+24*31;
end
if di(1)==7
    su=24*31+24*28+24*31+24*30+24*31+24*30;
end
if di(1)==8
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31;
end
if di(1)==9
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31;
```

```
end
if di(1)==10
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30;
end
if di(1)==11
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31;
end
if di(1)==12
    su=24*31+24*28+24*31+24*30+24*31+24*30+24*31+24*31+24*30+24*31+24*31;
end
localizado=su+(di(2)-1)*24;
s=1;
for p=localizado:(diax*24+localizado)
    rp0(s)=precio(p-7*24);
    rp1(s)=precio(p-24);
    rp2(s)=demanda(p);
    rp3(s)=generacion(p);
    rol(s)=precio(p);
    s=s+1;
end
rol=rol(:,1:s-2)';
rp0=rp0(:,1:s-2)';
rp1=rp1(:,1:s-2)';
rp2=rp2(:,1:s-2)';
rp3=rp3(:,1:s-2)';
precio_estt=[rp0 rp1 rp2 rp3]*theta;
hor=(2+dim_r):diax*24;
close all
plot(hor,rol((2+dim_r):diax*24,:),'r--');
hold on;
% plot(hora,precio_P,'b-o');
plot(hor,precio_estt((2+dim_r):diax*24,:),'b')
legend('Real','Estimado');
title('MÍNIMOS CUADRADOS RECURSIVOS');
ylabel('PRECIO(€/MWh)');
xlabel('TIEMPO(h)');
mape=0;
rmse=0;
mae=0;
errorp=ones(1,diax*24);
for i=2+dim_r:horas
    errorp(1,i)=rol(i)-precio_estt(i);
    mape=mape+abs((rol(i)-precio_estt(i))/rol(i)); %error porcentual absoluto
medio
    rmse=sqrt((rmse+(rol(i)-precio_estt(i))^2)/(diax*24)); %root mean squared
error
    mae=mae+(abs(rol(i)-precio_estt(i))/(diax*24)); %error absoluto medio
end
mape=mape/(diax*24);
errores=[mape rmse mae]
figure
[n,p] = hist(errorp,20);
bar(p,n);
```

9.3 Preparación datos

```
clear all
precio=xlsread('precio.xls');
generacion=xlsread('generacion.xls');
```

```
demanda=xlsread('demanda.xls');
precio=precio(:,3);
generacion=generacion(:,3);
demanda=demanda(:,1);
s=1;
for p=(7*24+1):8760
    x0(s)=precio(p-7*24);
    x1(s)=precio(p-24);
    x2(s)=demanda(p);c
    x3(s)=generacion(p);
    sol(s)=precio(p);
    s=s+1;
end
x=[x0;x1;x2;x3;sol]';
xlswrite('lit.csv',x);
```

9.4 Random Forest

```
import numpy as np
from sklearn.ensemble.forest import RandomForestRegressor
import matplotlib.pyplot as plt
import csv
f = open('librut.csv')
lns = csv.reader(f)
s=0
tran=np.ones((8592,4))
sol=np.ones((8592,1))
for line in lns:
    tran[s,0]=line[0]
    tran[s,1]=line[1]
    tran[s,2]=line[2]
    tran[s,3]=line[3]
    sol[s]=line[4]
    s=s+1
ent=168
pru=168
x=tran[:ent,:]
xt=tran[168:336,:]
t=np.array(sol[:ent,:])
yr=np.array(sol[168:336,:])
sal=168
rf = RandomForestRegressor(n_estimators=2000, criterion='mse', max_depth=None,
min_samples_split=2,      min_samples_leaf=2,      min_weight_fraction_leaf=0.0,
max_features='auto',   max_leaf_nodes=None,   bootstrap=True,   oob_score=False,
n_jobs=1, random_state=None, verbose=0, warm_start=False)
rf.fit(x, t)
```

```
a=rf.get_params()
yt = rf.predict(xt)
error1=rf.score(xt,yr)
plt.figure()
plt.plot(range(sal), yt, c="b", label="randomforest" )
plt.plot(range(sal), yr, c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("Random Forest Regression")
plt.legend()
plt.show()
error=np.ones((len(t)))
rmse=0
mape=0
mae=0
e=np.ones((len(t)))
for i in range(len(yt)):
   error[i]=((yt[i]-yr[i]))
   mape=mape+abs((yt[i]-yr[i])/yt[i]) #error porcentual absoluto medio
   rmse=((rmse+((yt[i]-yr[i])**2))/len(yt))**(0.5) #root mean squared error
   mae=mae+(abs(yt[i]-yr[i])/(len(yt))) #error absoluto medio
mape=mape/(len(yt))
print mape
print rmse
print mae
plt.hist(error,20)
plt.show()
```

9.5 Máquinas soporte vectorial

```
import numpy as np
from sklearn.svm import SVR
import matplotlib.pyplot as plt
import numpy as np
import csv
f = open('librut.csv')
lns = csv.reader(f)
s=0
tran=np.ones((8592,4))
sol=np.ones((8592,1))
for line in lns:
```

```
    tran[s,0]=line[0]
    tran[s,1]=line[1]
    tran[s,2]=line[2]
    tran[s,3]=line[3]
    sol[s]=line[4]
    s=s+1
ent=168
sal=168
X=tran[:ent,:]
y=np.array(sol[:ent,:])
X_test=tran[168:336,:]
y_test =np.array(sol[168:336,:])
coefy=y.max(axis=0)
coefy_test=y_test.max(axis=0)
X = X / X.max(axis=0)
X_test = X_test / X_test.max(axis=0)
y = y / y.max(axis=0)
y_test = y_test / y_test.max(axis=0)
svr_rbf   =SVR(C=1.0,   cache_size=200,   coef0=0.0,   degree=3,   epsilon=0.01,
gamma='auto',
    kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=False)


a=svr_rbf.fit(X, y)
y_rbf = svr_rbf.predict(X_test)


y_rbf=y_rbf*coefy_test
y_test=y_test*coefy_test


plt.plot( range(ent),y_rbf, label='RBF model')
plt.plot( range(ent),y_test, label='real')
plt.plot()
plt.xlabel('TIEMPO')
plt.ylabel('PRECIO')
plt.title('Support Vector Regression')
plt.legend()
plt.show()


error=np.ones((len(y_rbf)))
rmse=0
mape=0
mae=0;
```

```
for i in range(len(y_rbf)):
   error[i]=((y_rbf[i]-y_test[i]))
   mape=mape+abs((y_rbf[i]-y_test[i])/y_rbf[i]) #error porcentual absoluto medio
   rmse=((rmse+((y_rbf[i]-y_test[i])**2))/len(y_rbf))**(0.5)  #root  mean  squared
error
   mae=mae+(abs(y_rbf[i]-y_test[i])/(len(y_rbf))) #error absoluto medio
mape=mape/(len(y_rbf))
print mape
print rmse
print mae
plt.hist(error,20)
plt.show()
```

9.6 Clustering

- RF

```
import numpy as np
from sklearn.ensemble.forest import RandomForestRegressor
import matplotlib.pyplot as plt
import csv
from sklearn.cluster import KMeans

f = open('librut.csv')
lns = csv.reader(f)
s=0
tran=np.ones((8592,4))
sol=np.ones((8592,1))
for line in lns:
    tran[s,0]=line[0]
    tran[s,1]=line[1]
    tran[s,2]=line[2]
    tran[s,3]=line[3]
    sol[s]=line[4]
    s=s+1
ent=750
pru=750
X=tran[:ent,:]
xt=tran[7842:8592,:]
t=np.array(sol[:ent,:])
```

```
y=np.array(sol[7842:8592,:])
coefy=y.max(axis=0)
coeft=t.max(axis=0)
X = X / X.max(axis=0)
xt = xt / xt.max(axis=0)
y = y / y.max(axis=0)
t = t / t.max(axis=0)
P=np.ones((ent,2))
Pt=np.ones((ent,2))
P[:,0]=X[:,2]
P[:,1]=X[:,3]
Pt[:,0]=xt[:,2]
Pt[:,1]=xt[:,3]



y_pred=np.ones((y.shape))
y_predp=np.ones((y.shape))
plt.figure(figsize=(12, 12))

n_samples = 1500
random_state = 170
plt.figure(figsize=(12, 12))

kk=KMeans(n_clusters=2,       random_state=None,       n_init=1000,init='k-
means++',tol=0.00001).fit(P)
y_pred=kk.predict(P)
y_predp=kk.predict(Pt)

clasificado1=np.ones((ent,4))
clasificado2=np.ones((ent,4))
sol1=np.ones((ent,1))
sol2=np.ones((ent,1))
cont1=0
cont2=0
for j in range(ent):
    if y_pred[j]==0:
        clasificado1[cont1,:]=X[j,:]
        sol1[cont1]=t[j]
        cont1=cont1+1
    if y_pred[j]==1:
        clasificado2[cont2,:]=X[j,:]
```

```
        sol2[cont2]=t[j]
        cont2=cont2+1


sol1=sol1[:cont1,:]
sol2=sol2[:cont2,:]
clasificado1=clasificado1[:cont1,:]
clasificado2=clasificado2[:cont2,:]
clasificado1p=np.ones((ent,4))
clasificado2p=np.ones((ent,4))
sol1p=np.ones((ent,1))
sol2p=np.ones((ent,1))
cont1p=0
cont2p=0
for j in range(ent):
    if y_predp[j]==0:
        clasificado1p[cont1p,:]=xt[j,:]
        sol1p[cont1p]=y[j]
        cont1p=cont1p+1
    if y_predp[j]==1:
        clasificado2p[cont2p,:]=xt[j,:]
        sol2p[cont2p]=y[j]
        cont2p=cont2p+1


sol1p=sol1p[:cont1p,:]
sol2p=sol2p[:cont2p,:]
clasificado1p=clasificado1p[:cont1p,:]
clasificado2p=clasificado2p[:cont2p,:]
rf     =     RandomForestRegressor(n_estimators=2000,     criterion='mse',
max_depth=None,          min_samples_split=1,          min_samples_leaf=1,
min_weight_fraction_leaf=0.0,   max_features='auto',   max_leaf_nodes=None,
bootstrap=True,  oob_score=False,  n_jobs=1,  random_state=None,  verbose=0,
warm_start=False)
rf.fit(clasificado1, sol1)
a=rf.get_params()
yt1 = rf.predict(clasificado1p)
error1=rf.score(clasificado1p,sol1p)
yt1=yt1*coefy
sol1p=sol1p*coefy
plt.figure()
plt.plot(range(cont1p), yt1, c="g", label="Random Forest" )
plt.plot(range(cont1p),sol1p, c="r", label="real")
plt.xlabel("TIEMPO")
```

```
plt.ylabel("PRECIO")
plt.title("random forest clase 1 ")
plt.legend()
plt.show()
rmse=0
mape=0
mae=0;
for i in range((cont1p)):
   mape=mape+abs((yt1[i]-sol1p[i])/yt1[i]) #error porcentual absoluto medio
   rmse=((rmse+((yt1[i]-sol1p[i])**2))/cont1p)**(0.5)  #root  mean  squared
error
   mae=mae+(abs(yt1[i]-sol1p[i])/(cont1p)) #error absoluto medio
mape=mape/cont1p
print mape
print rmse
print mae


rf1     =     RandomForestRegressor(n_estimators=2000,     criterion='mse',
max_depth=None,          min_samples_split=1,          min_samples_leaf=1,
min_weight_fraction_leaf=0.0,   max_features='auto',   max_leaf_nodes=None,
bootstrap=True,  oob_score=False,  n_jobs=1,  random_state=None,  verbose=0,
warm_start=False)
rf1.fit(clasificado2, sol2)
a=rf1.get_params()
# Predict
yt2 = rf1.predict(clasificado2p)
error2=rf1.score(clasificado2p,sol2p)
yt2=yt2*coefy
sol2p=sol2p*coefy
# Plot the results
plt.figure()
plt.plot(range(cont2p), yt2, c="g", label="Random Forest" )
plt.plot(range(cont2p),sol2p, c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("Random Forest clase 2")
plt.legend()
plt.show()


rmse=0
mape=0
mae=0;
```

```
for i in range((cont2p)):
   mape=mape+abs((yt2[i]-sol2p[i])/yt2[i]) #error porcentual absoluto medio
   rmse=((rmse+((yt2[i]-sol2p[i])**2))/cont2p)**(0.5)  #root  mean  squared
error
   mae=mae+(abs(yt2[i]-sol2p[i])/(cont2p)) #error absoluto medio
mape=mape/(cont2p)
print mape
print rmse
print mae
yfin=np.ones((len(y)))
tam1=0
tam2=0
for j in range(ent):
    if y_predp[j]==0:
        yfin[j]=yt1[tam1]
        tam1=tam1+1
    if y_predp[j]==1:
        yfin[j]=yt2[tam2]
        tam2=tam2+1
y=y*coefy
plt.figure()
plt.plot(range(ent), yfin, c="g", label="RF + CLUSTERING" )
plt.plot(range(ent),y , c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("RF + CLUSTERING")
plt.legend()
plt.show()
rmse=0
mape=0
mae=0
error=np.ones((len(t)))
for i in range((ent)):
   error[i]=((yfin[i]-y[i]))
   mape=mape+abs((yfin[i]-y[i])/y[i]) #error porcentual absoluto medio
   rmse=((rmse+((yfin[i]-y[i])**2))/ent)**(0.5) #root mean squared error
   mae=mae+(abs(yfin[i]-y[i])/(ent)) #error absoluto medio
mape=mape/(ent)
print mape
print rmse
print mae
```

```
plt.hist(error,20)
plt.show()
```

- SVR

```
import numpy as np
from sklearn.ensemble.forest import RandomForestRegressor
import matplotlib.pyplot as plt
import csv
from sklearn.cluster import KMeans

f = open('librut.csv')
lns = csv.reader(f)
s=0
tran=np.ones((8592,4))
sol=np.ones((8592,1))
for line in lns:
    tran[s,0]=line[0]
    tran[s,1]=line[1]
    tran[s,2]=line[2]
    tran[s,3]=line[3]
    sol[s]=line[4]
    s=s+1
ent=750
pru=750
X=tran[:ent,:]
xt=tran[7842:8592,:]
t=np.array(sol[:ent,:])
y=np.array(sol[7842:8592,:])
coefy=y.max(axis=0)
coeft=t.max(axis=0)
X = X / X.max(axis=0)
xt = xt / xt.max(axis=0)
y = y / y.max(axis=0)
t = t / t.max(axis=0)
P=np.ones((ent,2))
Pt=np.ones((ent,2))
P[:,0]=X[:,2]
P[:,1]=X[:,3]
```

```
Pt[:,0]=xt[:,2]
Pt[:,1]=xt[:,3]
#Pt=xt[:,2:3]

y_pred=np.ones((y.shape))
y_predp=np.ones((y.shape))
plt.figure(figsize=(12, 12))

n_samples = 1500
random_state = 170
plt.figure(figsize=(12, 12))

kk=KMeans(n_clusters=2,       random_state=None,       n_init=1000,init='k-
means++',tol=0.00001).fit(P)
y_pred=kk.predict(P)
y_predp=kk.predict(Pt)

clasificado1=np.ones((ent,4))
clasificado2=np.ones((ent,4))
sol1=np.ones((ent,1))
sol2=np.ones((ent,1))
#sol3=np.ones((ent,1))
cont1=0
cont2=0
#cont3=0
for j in range(ent):
    if y_pred[j]==0:
        clasificado1[cont1,:]=X[j,:]
        sol1[cont1]=t[j]
        cont1=cont1+1
    if y_pred[j]==1:
        clasificado2[cont2,:]=X[j,:]
        sol2[cont2]=t[j]
        cont2=cont2+1


sol1=sol1[:cont1,:]
sol2=sol2[:cont2,:]

clasificado1=clasificado1[:cont1,:]
clasificado2=clasificado2[:cont2,:]
```

```
clasificado1p=np.ones((ent,4))
clasificado2p=np.ones((ent,4))
sol1p=np.ones((ent,1))
sol2p=np.ones((ent,1))
cont1p=0
cont2p=0
for j in range(ent):
    if y_predp[j]==0:
        clasificado1p[cont1p,:]=xt[j,:]
        sol1p[cont1p]=y[j]
        cont1p=cont1p+1
    if y_predp[j]==1:
        clasificado2p[cont2p,:]=xt[j,:]
        sol2p[cont2p]=y[j]
        cont2p=cont2p+1

sol1p=sol1p[:cont1p,:]
sol2p=sol2p[:cont2p,:]
clasificado1p=clasificado1p[:cont1p,:]
clasificado2p=clasificado2p[:cont2p,:]
rf   =SVR(C=1.0,   cache_size=200,   coef0=0.0,   degree=3,   epsilon=0.01,
gamma='auto',
    kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=False)
rf.fit(clasificado1, sol1)
a=rf.get_params()
yt1 = rf.predict(clasificado1p)
error1=rf.score(clasificado1p,sol1p)
yt1=yt1*coefy
sol1p=sol1p*coefy
plt.figure()
plt.plot(range(cont1p), yt1, c="g", label="SVR" )
plt.plot(range(cont1p),sol1p, c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("SVR clase 1 ")
plt.legend()
plt.show()

rmse=0
mape=0
mae=0;
```

```
for i in range((cont1p)):
   mape=mape+abs((yt1[i]-sol1p[i])/yt1[i]) #error porcentual absoluto medio
   rmse=((rmse+((yt1[i]-sol1p[i])**2))/cont1p)**(0.5)  #root  mean  squared
error
   mae=mae+(abs(yt1[i]-sol1p[i])/(cont1p)) #error absoluto medio
mape=mape/cont1p
print mape
print rmse
print mae


rf1  =  SVR(C=1.0,  cache_size=200,  coef0=0.0,  degree=3,  epsilon=0.01,
gamma='auto',
    kernel='rbf', max_iter=-1, shrinking=False, tol=0.001, verbose=False)
rf1.fit(clasificado2, sol2)
a=rf1.get_params()
yt2 = rf1.predict(clasificado2p)
error2=rf1.score(clasificado2p,sol2p)
yt2=yt2*coefy
sol2p=sol2p*coefy
plt.figure()
plt.plot(range(cont2p), yt2, c="g", label="SVR" )
plt.plot(range(cont2p),sol2p, c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("SVR clase 2")
plt.legend()
plt.show()


rmse=0
mape=0
mae=0;
for i in range((cont2p)):
   mape=mape+abs((yt2[i]-sol2p[i])/yt2[i]) #error porcentual absoluto medio
   rmse=((rmse+((yt2[i]-sol2p[i])**2))/cont2p)**(0.5)  #root  mean  squared
error
   mae=mae+(abs(yt2[i]-sol2p[i])/(cont2p)) #error absoluto medio
mape=mape/(cont2p)
print mape
print rmse
print mae
yfin=np.ones((len(y)))
tam1=0
```

```
tam2=0
for j in range(ent):
    if y_predp[j]==0:
        yfin[j]=yt1[tam1]
        tam1=tam1+1
    if y_predp[j]==1:
        yfin[j]=yt2[tam2]
        tam2=tam2+1
y=y*coefy
plt.figure()
plt.plot(range(ent), yfin, c="g", label="SVR+CLUSTERING" )
plt.plot(range(ent),y , c="r", label="real")
plt.xlabel("TIEMPO")
plt.ylabel("PRECIO")
plt.title("SVR + CLUSTERING")
plt.legend()
plt.show()
rmse=0
mape=0
mae=0
error=np.ones((len(t)))
for i in range((ent)):
   error[i]=((yfin[i]-y[i]))
   mape=mape+abs((yfin[i]-y[i])/y[i]) #error porcentual absoluto medio
   rmse=((rmse+((yfin[i]-y[i])**2))/ent)**(0.5) #root mean squared error
   mae=mae+(abs(yfin[i]-y[i])/(ent)) #error absoluto medio
mape=mape/(ent)
print mape
print rmse
print mae
plt.hist(error,20)
plt.show()
```

REFERENCIAS

- Leo Breiman. Random Forests. *Machine Learning*. 2001.
- Teodoro Álamo. *Identificacion mediante el método de los mínimos cuadrados.* Ingeniría de control 3º GITI Escuela Superior Ingenieros. Sevilla.
- Teodoro Álamo. *Método de los Míınimos Cuadrados Recursivos. .* Ingeniría de control 3º GITI Escuela Superior Ingenieros. Sevilla.
- Enrique J. Carmona. *Tutorial sobre Máquinas de Vectores Soporte (SVM).* ETS Ingeniería Informatica UNED. 2014.
- Fernando Berzal. *Clustering.* DECSAI Univerisdad de Granada.
- Machine Learning. Wikipedia, la enciclopedia libre.

  https://en.wikipedia.org/wiki/Machine_learning
- Mercado de la electricidad.

  http://www.omel.es/inicio/mercados-y-productos/mercado-electricidad/nuestros-mercados-de-electricidad/diario-e-intradia
- Mercado diario.

  http://www.omel.es/inicio/mercados-y-productos/mercado-electricidad/nuestros-mercados-de-electricidad/mercado-diario