ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

RECONOCIMIENTO DE CARGAS DOMÉSTICAS POR LOS MÉTODOS DE MACHINE LEARNING

Autor: Clara Úbeda-Romero Arconada

Director: Trung Dung Le

Madrid

Agosto 2018

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

Reconocimiento de cargas domésticas por los métodos de *Machine Learning*

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2017/2018 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Clara Úbeda-Romero Arconada Fecha: 30 / 08 / 2018

Autorizada la entrega del proyecto

EL COORDINADOR DEL PROYECTO

Fdo.: Fernando de Cuadra García Fecha: 30 / 08 / 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor **D.ª Clara Úbeda-Romero Arconada** DECLARA ser el titular de los derechos de propiedad intelectual de la obra: ***Reconocimiento de cargas domésticas por los métodos de* Machine Learning**, que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

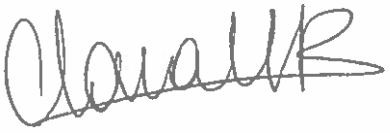
6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 30 de agosto de 2018

ACEPTA



Fdo. CLARA ÚBEDA-ROMERO ARCONADA

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA ELECTROMECÁNICA

RECONOCIMIENTO DE CARGAS DOMÉSTICAS POR LOS MÉTODOS DE MACHINE LEARNING

Autor: Clara Úbeda-Romero Arconada

Director: Trung Dung Le

Madrid

Agosto 2018

RECONOCIMIENTO DE CARGAS DOMÉSTICAS POR LOS MÉTODOS DE *MACHINE LEARNING*

Autor: ÚBEDA-ROMERO ARCONADA, Clara.

Director: LE, Trung Dung.

Entidades Colaboradoras: CentraleSupelec, ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Palabras clave: *Machine Learning*, clasificación, cargas domésticas, desagregación del consumo.

El desarrollo sostenible ha cobrado una especial importancia a nivel internacional desde finales del siglo pasado debido, entre otras causas, a las repercusiones directas que tiene sobre los ámbitos social, económico y ambiental.

En particular, la sostenibilidad energética se apoya sobre dos pilares principales: la diversificación de las energías y la racionalización de su consumo.

El inminente agotamiento de los combustibles fósiles exige la explotación de fuentes de energía renovables. De esta manera, la diversificación de las energías pretende hacer frente al ritmo desmesurado del aumento de la demanda energética que se viene produciendo ininterrumpidamente desde el comienzo de la Primera Revolución Industrial.

Además de la búsqueda de soluciones alternativas, la racionalización del consumo energético es indispensable para reducir el enorme malgasto de energía que se produce a diario, a nivel mundial.

La inestabilidad del actual modelo energético lleva a numerosas y diversas instituciones a comprometerse, mediante la determinación de objetivos y la elaboración de medidas destinadas a cumplirlos, a orientar al conjunto de las economías hacia un modelo más sostenible y respetuoso con el medio ambiente.

Entre estas instituciones destaca la Unión Europea que adopta, en 2008, un plan de acción con tres objetivos principales: la reducción de emisiones de CO_2, el aumento de la eficiencia energética y la introducción de las renovables en la producción de energía. La Comisión Europea publica en 2016 un paquete de medidas, llamado “Energía Limpia para todos”, que expone los tres elementos clave para la consecución de dichos objetivos y que aseguran la

transición energética. Estos elementos son la electrificación de la sociedad, la integración de las energías renovables y el aumento de la eficiencia energética. La primera propuesta es la más ambiciosa puesto que representa una renovación completa del sistema eléctrico. Además, incluye la integración de las energías renovables en la producción de electricidad y el aumento de la eficiencia energética, mediante la mejora de la red eléctrica y de las cargas consumidoras.

En la economía actual, todos los sectores hacen uso de alguna forma de energía, siendo los sectores del transporte y de la edificación los dos mayores consumidores. El sector de la edificación, que engloba los subsectores residencial y de servicios, representa un 20% de la energía final en España. Por tanto, a electrificación del subsector residencial puede ser especialmente interesante si se acompaña de un uso racional de la electricidad con el fin de alcanzar los citados objetivos marcados por la Comisión Europea.

En este sentido, conviene recordar que las nuevas tecnologías han mejorado exponencialmente en las últimas décadas. Esta nueva realidad ha permitido, entre otras, la robotización de la medicina, el reconocimiento vocal o facial en sistemas de seguridad o la informatización de una infinidad de servicios. Una tecnología que ha adquirido popularidad, aunque todavía se encuentre en fase de desarrollo, es la llamada Inteligencia Artificial. Ésta está compuesta por varias ramas, entre las cuales se encuentra el Aprendizaje Automático, o *Machine Learning* en inglés, que pretende desarrollar la capacidad de aprendizaje de las máquinas.

Pues bien, el presente Proyecto de Fin de Grado pretende aplicar los métodos de *Machine Learning* a la electrificación del sector residencial, mediante un estudio de las cargas domésticas.

Numerosos estudios demuestran que gran parte del gasto eléctrico en las viviendas se debe a la incapacidad de los usuarios de estimar correctamente su consumo, a pesar del reciente -pero significante- despliegue de medidores inteligentes en las instalaciones eléctricas residenciales. Estos sistemas de medición recopilan una gran variedad de información sobre el suministro eléctrico, pero, inconvenientemente, miden únicamente magnitudes referentes al consumo global de la vivienda. Un desglose de la factura eléctrica permitiría la optimización del uso energético de los usuarios. El objetivo de este proyecto es la elaboración de un sistema que permita dicho desglose.

La desagregación del consumo energético, también denominada *Non-Intrusive Load Monitoring* (NILM) o *Non-Intrusive Appliance Load Monitoring* (NIALM) en inglés, es el proceso que pretende realizar dicho desglose. La desagregación del consumo, originariamente propuesta por tres ingenieros del MIT, es, hoy en día, objeto de innumerables estudios. Mediante algoritmos, llamados de desagregación, se pueden determinar los aparatos que se están utilizando únicamente a partir de los datos de consumo total, también llamado consumo agregado. Para fomentar la investigación y el desarrollo la desagregación, se han creado bases de datos de libre acceso. Se ponen a disposición de todo el mundo porque su elaboración es costosa, ya que se necesita mucho material para cubrir la monitorización de todos los equipos de una vivienda. Para elaborar este proyecto, se trabaja con un conjunto de datos, llamado "*UK Domestic Appliance-Level Electricity*" (UK-DALE), elaborado a partir de registros obtenidos en varias viviendas del Reino Unido.

El método de *Machine Learning* seleccionado para cumplir con el objetivo de este proyecto es el clasificador de los k vecinos más cercanos. Se trata de un algoritmo de clasificación, que predice las etiquetas de objetos de clase desconocida. Realiza estas estimaciones a partir de una serie de ejemplos individuales, que le permiten crear un modelo generalizado que sirva para todos los nuevos objetos que intente clasificar.

El concepto de carga se puede definir de distintas maneras, por lo que el reconocimiento de cargas domésticas se puede plantear desde varios puntos de vista.

Un primer planteamiento sería reducir la carga a una señal temporal que defina una determinada magnitud eléctrica que le sea característica. Dado que el algoritmo escogido está limitado al cálculo de distancias y que, por lo tanto, no es capaz de reconocer las formas de las señales, se ha de plantear un estudio alternativo para las señales basado en sus características frecuenciales.

Un segundo planteamiento permitiría poner en práctica la teoría de la desagregación de consumos mediante el uso de UK-DALE. El algoritmo desarrollado para este proyecto es muy rudimentario, volviendo a evidenciar ciertas fragilidades debido a la enorme cantidad de datos disponibles en UK-DALE.

LOAD MONITORING BY MEANS OF MACHINE LEARNING METHODS

Author: ÚBEDA-ROMERO ARCONADA, Clara.

Supervisor: LE, Trung Dung.

Collaborating Entities: CentraleSupelec, ICAI – Universidad Pontificia Comillas.

ABSTRACT

Keywords: Machine Learning, classification, domestic loads, consumption disaggregation.

Sustainable development has been of great importance over the past two decades due its direct repercussions on the social, economic, and environmental fields.

Energetic sustainability leans on two fundamental pillars: energetic diversification and energetic consumption rationalization.

Firstly, the imminent depletion of fossil fuels demands the exploitation of other renewable energy sources. Thus, energy diversification pretends to address the overwhelming increase in energetic demand since the First Industrial Revolution.

Secondly, apart from trying to find alternatives, energetic consumption rationalization is crucial to reduce the world's wasteful attitude towards energy usage. Raising awareness of efficient energy use is imperative to ensure a reduction of current consumption levels.

The instability of the current energetic model has led numerous institutions to commit, by setting objectives and developing measures to achieve them, to guide economies across the world to adopt a more sustainable model.

Amongst these institutions, The European Union has taken on a leading role, adopting in 2008 a plan of action that sets three clearly defined objectives: reduction of CO_2 emissions, increase in energy efficiency, and the introduction of renewable energies in the production process.

The European Union publishes in 2016 a package of measures called Clean Energy for Everyone, through which it explains three key elements for the achievement of the proposed objectives that secure the energetic transition. These elements are the electrification of

society, the integration of renewable energies and the energy efficiency. The first proposal is the most ambitious, because it entails a complete renovation of the electric system. Furthermore, it encompasses the other two proposed aspects.

In today's economy, every sector makes use of some form of energy, being the transport and construction sector its heaviest users. The construction sector, that encompasses the residential and services subsectors, represents 20% of the final energy consumption in Spain. Subsequently, the electrification of the residential subsector can be of special interest if accompanied by a rational energy use to reach the stated objectives defined by the European Commission.

Likewise, it should be emphasized that the improvement rate of new technologies has been exponential. This new reality has enabled, among others, the robotization of the health sector, vocal and facial recognition applied to security systems, or the informatization of many services. A technology that has become popular over the last few years even though it is still in its earlier development phase, is the so called Artificial Intelligence. It is comprised by several branches, including Machine Learning, that pretends to develop the learning capabilities of machines.

That being said, this Thesis Paper tries to apply Machine Learning methods to the electrification of the residential sector, through the study of domestic loads.

Several studies have proven that most of the residential electricity consumption is due to the inability of the users to identify consumption despite the significant (although recent) deployment of intelligent meters inside residential installations. These measurement systems are able to gather useful information of the power supply but, inconveniently, they measure only information related to the total consumption of the residential unit. A breakdown of the power bill would increase energy efficiency. Thus, the objective of this project is the elaboration of a system that enables such breakdown.

The Non-Intrusive Load Monitoring (NILM) or Non-Intrusive Appliance Load Monitoring (NIALM) is a process by which the user is able to achieve the desired breakdown. It was first introduced by three MIT engineers and it is currently the subject of many academic studies. Through disaggregation algorithms, an individual can determine the device that is being used solely by analyzing total consumption information, also known as aggregate consumption. To promote disaggregation investigation and development, free-access

databases have been created. Exposing these databases is necessary to facilitate the monitoring process of every device in the owner's residence. To carry out this project, the so-called "UK Domestic Appliance-Level Electricity" (UK-DALE), has been used. It has been prepared from residential units across the United Kingdom.

The selected Machine Learning method is the k-Nearest Neighbors classifier. It is a classification algorithm, that predicts labels of unknown objects. It creates, from a series of individual examples, a generalized model that is useful to identify future objects.

The concept of the load can be defined in different ways, so the recognition of identical loads can be done from different perspectives.

A first approach would be to reduce the load to a temporary signal that defines the electricity magnitude as familiar. Given that the chosen algorithm is limited by the distance calculation and that, it is not capable of recognizing the forms of the signals, an alternative study of the signal's frequency characteristics becomes necessary.

A second approach would enable the implementation of the consumption dissagregation theory through the use of the UK-DALE. The algorithm developed for this project is very rudimentary, which translates in difficulties when processing the large quantity of data available in UK-DALE.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

La energía es, desde el punto social y económico, un recurso natural al que se le puede dar un uso industrial o económico. Se ha convertido en un pilar esencial del progreso de la humanidad por las actividades y servicios que permite desarrollar, existiendo una estrecha relación entre la demanda energética y el progreso económico. No se deben por tanto desatender las consecuencias que tiene el aumento de la demanda energética.

En efecto, la historia demuestra que el crecimiento económico viene necesariamente acompañado de un aumento de la demanda energética. Desde el inicio de la Primera Revolución Industrial en el siglo XVIII, el consumo de combustibles fósiles no ha cesado de incrementarse. Además, este crecimiento se ha producido a un ritmo muy superior al de regeneración de dichos combustibles, por lo que numerosas instituciones, de naturaleza, origen y posición política muy diferentes, prevén el inevitable agotamiento de estos recursos. Estas previsiones generan grandes desafíos que exigen la búsqueda de soluciones alternativas.

Sin embargo, encontrar otros recursos es una medida insuficiente si no se acompaña de otras. El enorme malgasto energético que se produce a diario, a nivel mundial, se debe limitar mediante la necesaria sensibilización de la sociedad, entre otras medidas. El uso responsable de la energía se tiene que convertir en un hábito que se debe inculcar a diferentes escalas con el fin de reducir en último extremo su consumo global.

En resumen, la diversificación de las energías y la racionalización de su uso son dos pilares fundamentales para asegurar el progreso de la humanidad, y son esenciales para aumentar la sostenibilidad energética del modelo social y económico actual.

1.1 EL DESARROLLO SOSTENIBLE

En primer lugar, se debe dar una definición explícita de lo que se considera oficialmente como desarrollo. La Asamblea General de las Naciones Unidas indicó, en la *Declaración sobre el derecho al desarrollo* de 1986, que el desarrollo es un "proceso global económico, social, cultural y político, que tiende a la mejora constante del bienestar de toda la población y de todos los individuos sobre la base de su participación, libre y significativa en este desarrollo y en la distribución justa de los beneficios que de él se derivan".

Poco después, en 1987, el informe presentado por la Comisión de Medio Ambiente y Desarrollo de Naciones Unidas, también conocido como el informe de Brundtland, formuló explícitamente el concepto de desarrollo sostenible como "el desarrollo que satisface las necesidades del presente sin poner en peligro la capacidad de las futuras generaciones para satisfacer sus propias necesidades". Esta definición considera que el desarrollo es posible y necesario, y que la sostenibilidad debe ser triple, es decir, social, medioambiental y económica.

El desarrollo sostenible se considera un elemento de gran trascendencia política desde la *Declaración de Río*, firmada durante la Conferencia de Naciones Unidas sobre el Medio Ambiente y el Desarrollo en 1992 y ratificada 10 años más tarde, en 2002, en la Cumbre de Johannesburgo. El desarrollo sostenible constituye actualmente un marco de orientación de políticas y estrategias para el desarrollo mundial, y se va incorporando gradualmente en la legislación a todos los niveles.

El modelo energético mundial, y especialmente el de las economías más desarrolladas, como es el caso de España, se considera insostenible en las tres dimensiones antes citadas. La gran preocupación que estos problemas llevan suscitando desde hace varias décadas, ha conducido al diseño de un nuevo modelo energético que presenta cambios profundos con respecto al actual. Este proceso es comúnmente conocido como transición energética, y numerosas instituciones se han sumado ya a la elaboración de medidas que pretenden posibilitar la transformación del modelo energético.

1.2 TRANSICIÓN HACIA UN NUEVO MODELO ENERGÉTICO

Para responder a la necesidad de un modelo energético más eficiente y sostenible, la Unión Europea se ha comprometido a cumplir los tres objetivos siguientes, también conocidos como el 20/20/20:

- 20% de reducción de las emisiones de gases de efecto invernadero (con respecto a los niveles de 1990).
- 20% de energías renovables en la Unión Europea.
- 20% de mejora de la eficiencia energética.

El paquete "Energía Limpia para todos", publicado en noviembre de 2016 por la Comisión Europea, explica los tres elementos clave para facilitar la transición hacia un nuevo modelo energético más sostenible. Estos tres elementos son: la electrificación de la economía, la integración de las energías renovables y el aumento de la eficiencia energética. Todas estas propuestas deben, además, garantizar la competitividad del sistema y la seguridad del suministro.

1.3 LA ELECTRIFICACIÓN DE LA SOCIEDAD Y SOSTENIBILIDAD

Los combustibles químicos y la electricidad han sido, desde siempre, las dos formas de energía más empleadas en todos los sectores. Sin embargo, la energía eléctrica tal y como se utiliza en la actualidad es casi siempre una forma secundaria de energía, que se obtiene a partir de otras formas de energía de naturaleza muy variada, como por ejemplo la química, térmica, cinética, nuclear, solar, etc.

De esta forma, se entiende por energía primaria aquella contenida en los combustibles antes de pasar las diferentes operaciones de tratamiento mientras que, por energía final, se entiende aquella utilizada por los consumidores. Se cumple por lo tanto la siguiente relación:

$$Energía\ primaria = Energía\ final + Pérdidas\ de\ transporte + Pérdidas\ de\ tratamiento$$

En efecto, la electricidad se obtiene mediante numerosas y diversas operaciones que requieren el uso de grandes cantidades de energía. Entre estas operaciones destacan la de tratamiento de la energía inicial, que se lleva a cabo en centrales eléctricas, y el transporte, tanto del yacimiento a la central, como desde la central hasta el consumidor.

La gran ventaja de la electricidad es que presenta una gran flexibilidad a la hora de introducir energías renovables en su proceso de producción. Por ello, la electrificación de la sociedad puede asegurar la transición hacia un mundo descarbonizado. El sector eléctrico tiene por tanto una nueva responsabilidad ambiental, que se suma a sus obligaciones en cuanto a calidad, seguridad y coste del suministro.

Para asegurar la consecución del objetivo de electrificación, se han propuesto numerosas herramientas, como son las nuevas tecnologías de almacenamiento de energía, el desarrollo de cargas inteligentes, la flexibilidad de la generación y del consumo, el desarrollo de las interconexiones y la implantación de redes eléctricas inteligentes. Dichas herramientas se pueden incluir, de una manera o de otra, en todos los sectores que hacen uso de electricidad para llevar a cabo sus actividades.

1.4 EL CONSUMO ENERGÉTICO POR SECTORES EN ESPAÑA

El Instituto para la Diversificación y Ahorro de la Energía (IDAE) es una entidad pública adscrita al Ministerio de Industria, Energía y Turismo que participa activamente en la consecución de los objetivos en materia de diversificación y ahorro energéticos.

Según el IDAE, los mayores consumidores de energía son los sectores del transporte y de la edificación. El transporte representa aproximadamente el 40% del consumo final de energía nacional, siendo el transporte por carretera el más destacado, puesto que representa el 80% del consumo del sector.

Por su parte, el sector de la edificación, que engloba los subsectores residencial y de servicios, representa el 20% del consumo de energía final en España. La relevancia de este sector se debe principalmente a que somos todos usuarios de más de un único edificio: para

empezar, se puede nombrar la vivienda y el lugar de trabajo, pero además están los servicios docentes, los sanitarios, los culturales, etc. Los principales consumos en este sector se deben a la satisfacción de las necesidades de calefacción, climatización, producción de agua caliente sanitaria e iluminación, entre otros.

Para realizar un seguimiento y aumentar el control del consumo en los edificios, se puede hacer uso de diversas nuevas tecnologías, que están actualmente a disposición de todos los usuarios de la red y que pueden contribuir al esfuerzo de ahorro eléctrico.

1.5 EDIFICIOS INTELIGENTES Y RECONOCIMIENTO DE CARGAS DOMÉSTICAS

Las redes eléctricas y cargas inteligentes son dos de los instrumentos de los cuales se pretende hacer uso para fomentar el ahorro y permitir una mayor eficiencia energética. Si se aplican estos dos instrumentos al sector residencial, aparece fácilmente la necesidad e interés de implantar en las viviendas sistemas inteligentes de gestión energética.

En efecto, un sistema inteligente de gestión energética minimizaría el consumo en las viviendas, puesto que permitiría a cada usuario adaptar su consumo según una serie de variables, tales como el precio, la comodidad o el impacto ambiental. Para optimizar este sistema de gestión es imprescindible evaluar el impacto que tiene la utilización de cada dispositivo conectado en la vivienda, en el consumo total de energía. El reconocimiento de las cargas domésticas parece ser, por lo tanto, una de las problemáticas fundamentales actuales en la gestión de la energía en edificios inteligentes.

Una vez expuestos los intereses y beneficios de implantar sistemas de reconocimiento de cargas domésticas en las viviendas, se pretende hacer uso para ello de la tecnología de Aprendizaje Automático, contribuyendo de esta forma al esfuerzo mundial de ahorro energético antes mencionado. En el siguiente capítulo, se describe dicha tecnología: su historia, sus principales métodos, etc.

Capítulo 2. EL APRENDIZAJE AUTOMÁTICO

La Inteligencia Artificial es una rama de las ciencias computacionales, cuyo objetivo es crear máquinas que presenten las mismas capacidades cognitivas que el ser humano. Las capacidades propias de los humanos que se desea transferir a estas máquinas son las de razonar, planificar, aprender y entender. Se trata de una tecnología en desarrollo, si bien ya existen numerosas aplicaciones en las que su utilización se ha estandarizado como por ejemplo en la detección facial de los móviles o sistemas de seguridad, los asistentes virtuales de voz como Siri de Apple o Alexa de Amazon, etc.

Para introducir el concepto objeto de este segundo capítulo, el Aprendizaje Automático, parece importante conocer la definición del verbo "aprender". Según la Real Academia Española, aprender significa "adquirir el conocimiento de algo por medio del estudio o de la experiencia".

El Aprendizaje Automático, o *Machine Learning*, es la rama de la Inteligencia Artificial que pretende desarrollar la capacidad de aprender por parte de las máquinas. Teniendo en cuenta la anterior definición de "aprender", y dado que una máquina no puede estudiar, ésta debe recibir una determinada experiencia para aprender. En la práctica, el *Machine Learning* consiste en crear algoritmos que permiten reconocer patrones y generalizar comportamientos a partir de un conjunto de informaciones con las que se entrena al algoritmo. La experiencia de la máquina se adquiere a través de estas informaciones, también llamadas ejemplos de entrenamiento o *training set*. El Aprendizaje Automático se fundamenta, por lo tanto, en un razonamiento inductivo, es decir, que obtiene conclusiones generales a partir de datos particulares o individuales.

Uno de los ejemplos más utilizados para explicar el concepto de Aprendizaje Automático es el reconocimiento de correos no deseados, o "spam", por parte del sistema de correo electrónico, aunque existen muchas otras aplicaciones, como es el procesamiento de lenguaje natural, las recomendaciones online o ciertas funciones de los coches inteligentes.

2.1 UNA BREVE HISTORIA SOBRE EL APRENDIZAJE AUTOMÁTICO

El ingeniero americano Arthur Lee Samuel (1901-1990), pionero en los campos de los juegos informáticos y de la Inteligencia Artificial, expone por primera vez el concepto de *Machine Learning* en 1959, en su artículo "*Some Studies of Machine Learning Using the Game of Checkers*". Samuel es principalmente conocido por haber desarrollado el primer programa capaz de jugar a las damas. A raíz del campeonato mundial de damas inglesas de 1948, celebrado en una localidad cercana a su residencia, se propone crear un programa capaz de ganar una partida de damas.

Figura 1. Arthur Lee Samuel jugando a las damas contra su programa

En 1949 empieza a trabajar en IBM, en el equipo encargado de crear lo que más tarde sería el primer ordenador comercial, el IBM 701. Su trabajo, principalmente de investigación, le lleva a intentar utilizar las instrucciones de la 701, aún inacabado, para la creación de su programa de damas, ver Figura 1. En 1952 acaba el primer programa de ordenador capaz de jugar a las damas. Este programa, destinado a utilizarse en una máquina inexistente, es una

hazaña por sí sola, por lo que a Arthur Lee Samuel se le considera también el pionero de la programación no numérica.

En 1955, IBM empieza a desarrollar el ordenador IBM 704 y Samuel tiene la posibilidad de ejecutar muchas horas seguidas su programa de damas, consiguiendo así acumular una gran cantidad de datos y estadísticas de esquemas de juegos. Partida tras partida, su programa desarrolla la capacidad de aprender a jugar a las damas hasta el punto de ganar a uno de los mejores jugadores americanos de damas.

Samuel incluye una definición del Aprendizaje Automático en su publicación de 1959:

> *"El Aprendizaje Automático es el conjunto de técnicas que permiten a una máquina aprender a realizar una tarea sin que haya sido programada para ese fin".*

En 1998, el informático y profesor americano Tom M. Mitchell da una segunda definición, matemática y relacional, del Aprendizaje Automático:

> *"Se dice que un programa de ordenador aprende de la experiencia (E) con respecto a alguna tarea (T) y alguna medida de rendimiento (P), si su rendimiento en (T), medido por (P), mejora con la experiencia (E)".*

2.2 LOS USOS DEL APRENDIZAJE AUTOMÁTICO

Los métodos de Aprendizaje Automático permiten resolver una gran variedad de problemas, todos ellos partiendo de la observación y análisis de casos particulares para terminar con la formulación de una regla o patrón generalizado, como se indicaba en la introducción de este capítulo.

Los diferentes problemas que se pueden resolver por métodos de *Machine Learning* suelen distinguirse según el tipo de objeto que pretenden predecir. Los más comunes son los problemas de regresión y clasificación, en los cuales el resultado de la cuestión es un valor numérico y una categoría, respectivamente.

2.2.1 REGRESIÓN

La regresión permite predecir valores continuos: la respuesta al problema no se encuentra en un conjunto finito de valores.

La regresión lineal es el modelo matemático de regresión más utilizado, aunque se suele subestimar por su simplicidad. Permite estudiar la relación de dependencia entre la variable dependiente Y, las variables independientes X_i y un término independiente β.

2.2.2 CLASIFICACIÓN

Los algoritmos de clasificación se utilizan cuando se desea obtener como resultado una etiqueta discreta. Esta vez, la respuesta de estos problemas se encuentra dentro de un conjunto finitos de resultados posibles.

La clasificación binaria es aquella que considera dos únicas clases, mientras que la multicategórica es la que agrupa todos los demás casos, es decir, que considera más de dos clases.

Por razones que se explicarán en el apartado 4.1.1, las definiciones y descripciones que se incluyen a continuación se particularizan para los algoritmos de clasificación, aunque todos estos conceptos sean aplicables a todos los problemas de Aprendizaje Automático.

2.3 CLASIFICACIÓN DE LOS ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

Los métodos de *Machine Learning* utilizan un razonamiento inductivo, como ya se ha mencionado anteriormente. Sin embargo, se puede ir más allá y clasificar los algoritmos según el tipo de aprendizaje que utilizan.

2.3.1 EL APRENDIZAJE AUTOMÁTICO SUPERVISADO

En el Aprendizaje Automático supervisado, los datos disponibles están etiquetados a priori, es decir, que se dispone de la clasificación correcta de estos objetos. El objetivo del algoritmo desarrollado es encontrar la función que establezca una relación entre entradas y salidas, siendo las entradas, objetos de clase desconocida y, las salidas, las etiquetas de dichos objetos.

Estos algoritmos trabajan en dos tiempos: primero, pasan por una fase de entrenamiento, o *training*, donde se "aprenden" los ejemplos de los que se dispone y, a continuación, durante la fase de prueba, o *test*, permiten predecir la clase de cualquier objeto nuevo. El modelo que permite la predicción de las clases se crea durante la fase de entrenamiento del algoritmo.

Matemáticamente, este método define una función de predicción f que permite asociar a cada nueva entrada x una salida $f(x)$, siendo esta salida la etiqueta más probablemente adecuada.

A menudo es más interesante predecir probabilidades de pertenencia a clases y no limitarse a una única clase: se habla entonces de aprendizaje supervisado probabilista.

Este tipo de aprendizaje se utiliza en problemas de clasificación y en problemas de regresión.

2.3.2 EL APRENDIZAJE AUTOMÁTICO NO SUPERVISADO

El Aprendizaje Automático no supervisado se utiliza cuando los datos disponibles no están etiquetados y, por lo tanto, no puede haber "entrenamiento" del algoritmo. Dado que no se conoce la clasificación correcta de los datos, sólo se puede describir su estructura, por lo que se dice que estos algoritmos tienen un carácter exploratorio.

El aprendizaje no supervisado sirve principalmente para entender y resumir datos, y se utiliza en problemas de *clustering*, de *profiling* y de agrupamientos de co-ocurrencia.

Los algoritmos de aprendizaje no supervisado pretenden agrupar el conjunto de datos que se debe clasificar en un número finito de subconjuntos, también llamados *clusters* para los

problemas de *clustering*. En cierto modo, el *clustering* es la versión no supervisada de la clasificación, ya que pretende agrupar los objetos, de clase desconocida, en subconjuntos. La definición de dichos subconjuntos, los *clusters*, es muy compleja, pues depende tanto de la naturaleza de los datos como de los resultados deseados. Se considera que son homogéneos, es decir que los objetos que los componen comparten una serie de características, que son justamente la razón por la que se asocian.

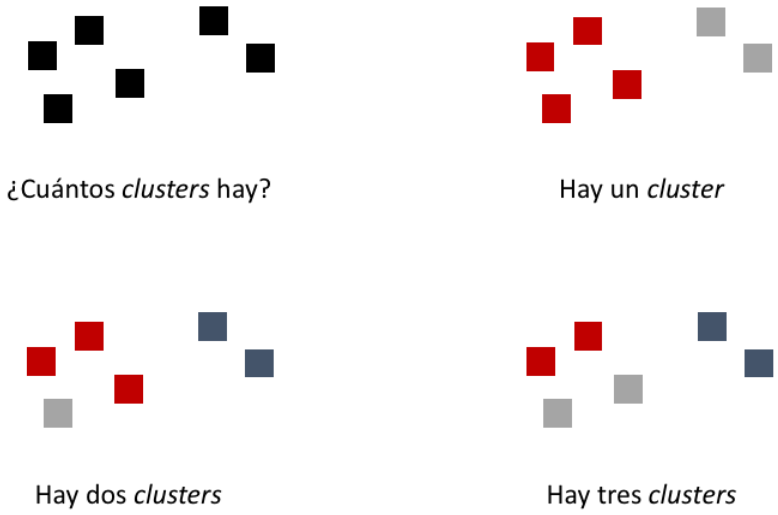


Figura 2. Ambigüedad de la definición de cluster.

Incluso si la definición de *cluster* es ambigua, como bien se muestra en la Figura 2, se puede afirmar que un buen agrupamiento es aquel que minimiza las distancias *intra-cluster*, es decir las diferencias entre objetos dentro de un mismo *cluster* y que maximiza las distancias *inter-cluster*, es decir las diferencias entre objetos de distintos *clusters*.

2.3.3 DIFERENCIA ENTRE LOS APRENDIZAJES AUTOMÁTICOS SUPERVISADO Y NO SUPERVISADO

La diferencia fundamental entre estos dos tipos de Aprendizaje Automático reside, por lo tanto, en el tipo de información disponible durante la fase de entrenamiento. Para distinguirlos hay que saber si a priori se conocen tanto las clases en las que se quiere clasificar los objetos como la clasificación real de los ejemplos de entrenamiento. Si se

dispone de toda esta información antes de lanzar el algoritmo, entonces estamos ante un caso de aprendizaje supervisado, y en el caso contrario, se trata de un caso de aprendizaje no supervisado.

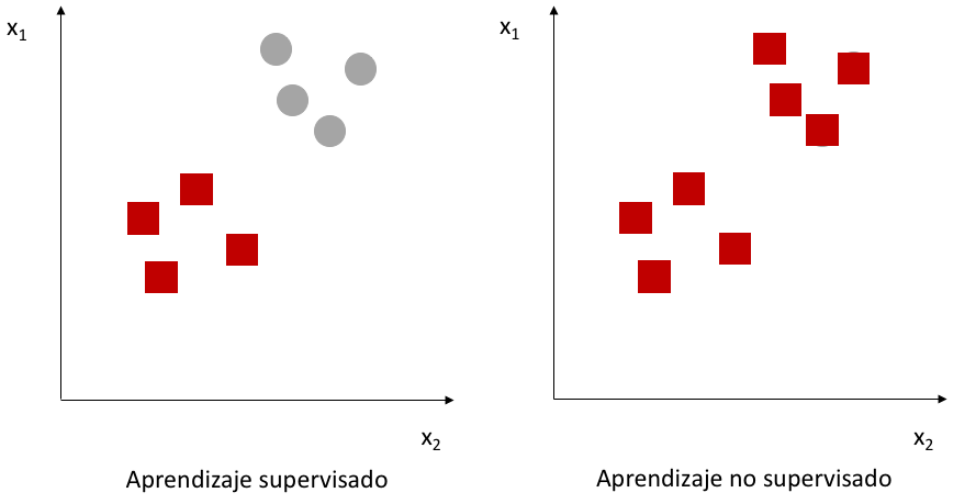


Figura 3. Diferencias entre los conjuntos de ejemplos de entrenamiento del AA supervisado y el no supervisado.

Esta diferencia viene ilustrada en la Figura 3: los datos de entrada del Aprendizaje Automático supervisado vienen ya clasificados en cuadrados rojos y círculos grises, mientras que los datos de entrada del Aprendizaje Automático no supervisado no tienen clase asignada. De ahí que sean todos cuadrados rojos.

2.3.4 OTROS TIPOS DE APRENDIZAJE AUTOMÁTICO

Además de los aprendizajes supervisado y no supervisado, existen otros tipos:

- *El aprendizaje semisupervisado* que combina los dos anteriores, ya que utiliza tanto ejemplos clasificados como no clasificados para su entrenamiento.
- *El aprendizaje por refuerzo*. Éste permite al algoritmo aprender mediante un flujo continuo de información en dos direcciones, del mundo real a la máquina y viceversa, para realizar una serie de ensayos y reforzar las operaciones que son validadas por el mundo.

- *La transconducción* que es similar al aprendizaje supervisado, pero -a diferencia de éste- no pretende definir explícitamente la función de predicción.
- *El aprendizaje multi-tarea* que incluye todos los métodos de aprendizaje que se entrenan previamente para hacer frente a problemas parecidos a los ya estudiados.

2.4 *ELECCIÓN DEL ALGORITMO MÁS ADECUADO*

La gran variedad de algoritmos ya concebidos y de fines para los que se pueden utilizar complica la elección del algoritmo más adaptado a cada problema que se desea resolver.

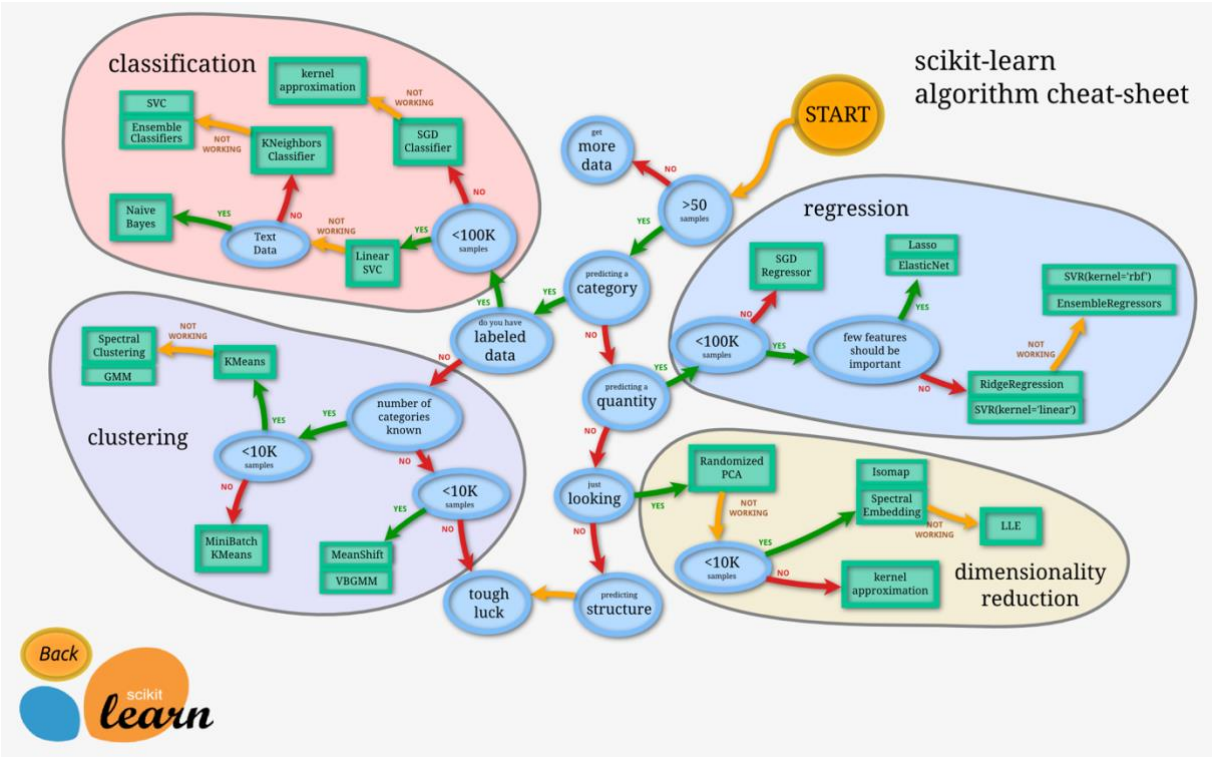


Figura 4. La chuleta o cheat-sheet de algoritmos de scikit-learn.

En primer lugar, se debe determinar qué tipo de problema se desea resolver, es decir, qué tipo de respuesta se espera por parte del algoritmo utilizado. Luego, y una vez se ha analizado la información disponible en la base de datos inicial para así precisar el tipo de aprendizaje que se va a aplicar, se pueden utilizar las famosas *cheat-sheets* de algoritmos, que ayudan a escoger el algoritmo que más se adapta al problema tratado.

Existen distintas *cheat-sheets*, algunas de las más conocidas son la de *scikit-learn*, la biblioteca de Python dedicada a Aprendizaje Automático (ver Figura 4) o la de *Microsoft Azure*, plataforma en la nube ofrecida por Microsoft.

2.5 MODELOS DE PREDICCIÓN

El modelo empleado por un algoritmo de Aprendizaje Automático debe ser lo más preciso posible, es decir, conducir al mínimo error. Sin embargo, hay que tener en cuenta que, normalmente, la disminución de la tasa de error conlleva un aumento de la complejidad del modelo, aumento que perjudica la calidad del algoritmo. Se debe por lo tanto encontrar el punto medio entre los requisitos de complejidad y error de predicción.

2.5.1 ERROR DE PREDICCIÓN DE UN MODELO

El error de predicción de un modelo se puede descomponer de la siguiente manera:

$$Error = Sesgo^2 + Varianza + Error\ irreductible$$

Se recuerdan las definiciones de sesgo y varianza.

El sesgo estadístico es la diferencia entre la esperanza matemática del estimador, es decir el valor esperado, y el valor numérico estimado. Cuanto más simple sea un modelo, mayor será su sesgo. Formalmente, el sesgo se escribe:

$$Sesgo\big(\hat{f}(x)\big) = E\big(\hat{f}(x) - f(x)\big)$$

La varianza es una medida de dispersión de una variable aleatoria. Contrariamente al sesgo, la varianza se ve aumentada con la complejidad del modelo. Formalmente, la varianza se escribe:

$$Varianza\big(\hat{f}(x)\big) = E\big(\hat{f}(x)^2\big) - E\big(\hat{f}(x)\big)^2$$

El error de predicción depende directamente de los valores del sesgo y de la varianza, por lo que ambos valores se deben intentar limitar.

2.5.2 OVERFITTING Y UNDERFITTING

Las principales causas de obtener malos resultados en los problemas de Aprendizaje Automático son el *overfitting* y el *underfitting* de los datos. La resolución de dichos problemas consiste en encajar, verbo que se traduce por *to fit* en inglés, las entradas con las salidas del algoritmo. Las palabras *overfitting y underfitting* parecen poder traducirse en este contexto por "sobreajuste" y "subajuste", respectivamente. Los tres casos de modelos, en cuanto a complejidad, vienen ilustrados por un ejemplo en la Figura 5.

Se recuerda que los algoritmos de Aprendizaje Automático deben poder generalizar un concepto para poder reconocer un objeto desconocido. El algoritmo no podrá generalizar un concepto si no se le proporcionan suficientes ejemplos de entrenamiento, por una falta de conocimientos: es el caso del *underfitting*. El *overfitting*, muy común entre los usuarios de Aprendizaje Automático menos experimentados, se da cuando se suministran demasiados ejemplos al algoritmo. Éste sólo aprende casos particulares, por lo que no será capaz de reconocer nuevos objetos. Encontrar un modelo que no esté en ninguna de estas dos situaciones puede resultar ser una tarea muy difícil.

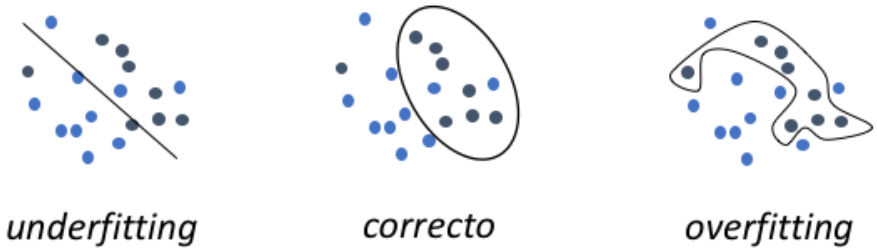


Figura 5. Los tres casos de modelos de predicción en cuanto a complejidad.

Existe por lo tanto una estrecha relación entre el error de predicción y la complejidad de un modelo. Más particularmente, un sesgo elevado provoca un *underfitting* y una varianza elevada provoca un *overfitting*. Un modelo es óptimo cuando su error de predicción es mínimo. Este mínimo, como se puede observar en la Figura 6, se alcanza cuando el modelo no es ni demasiado simple ni demasiado complejo, por lo que se debe encontrar un

compromiso adecuado entre el sesgo y la varianza, y por lo tanto entre el posible *underfitting* y el *overfitting*, mediante el ajuste de la complejidad del modelo.

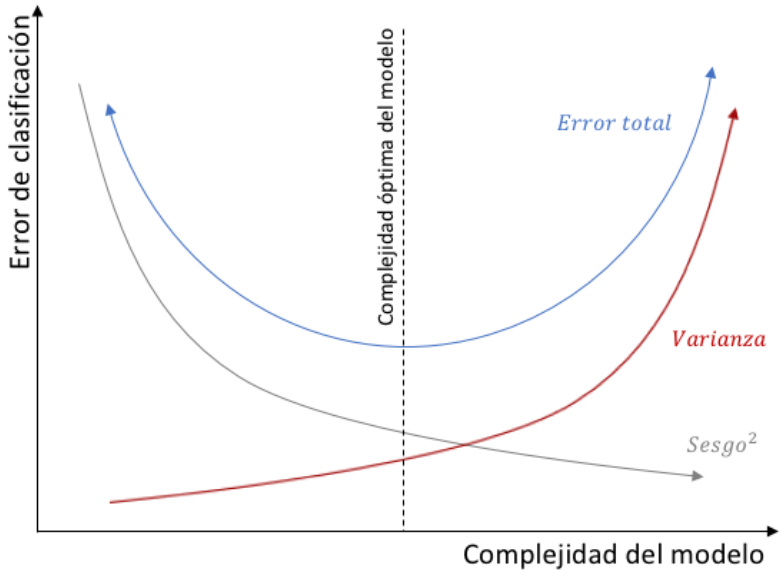


Figura 6. Relación entre la complejidad de un modelo y su error de predicción.

Todo lo expuesto anteriormente se resume en la Figura 7.

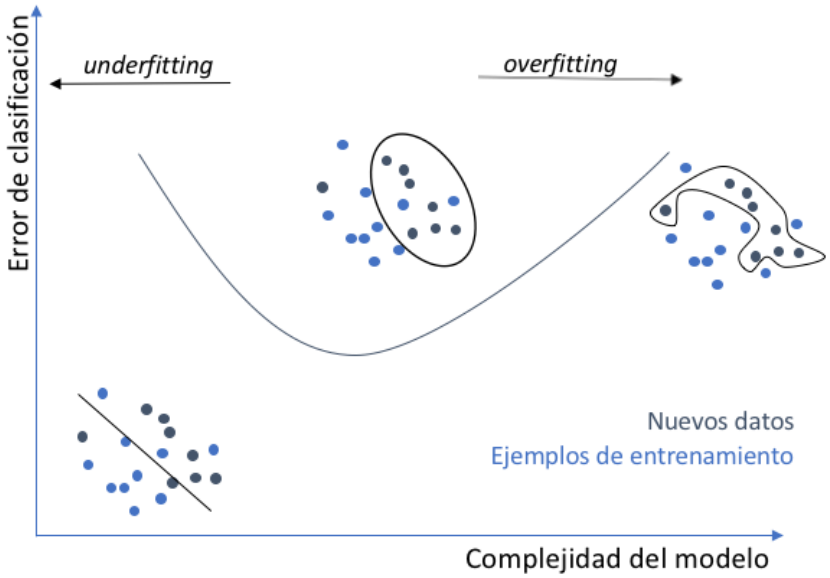


Figura 7. Resumen sobre la precisión de los modelos de clasificación.

Existen diferentes maneras de proceder a la hora de encontrar el equilibrio requerido entre el sesgo y la varianza. Una de ellas consiste en reducir las dimensiones del modelo, es decir, el número de variables de entrada. Esto permite simplificar el modelo, mediante la disminución del valor de la varianza.

2.6 MÉTRICAS DE RENDIMIENTO DE LOS PROBLEMAS DE CLASIFICACIÓN

Para mejorar los resultados de los problemas de clasificación, se han creado diferentes herramientas capaces de medir el rendimiento de un algoritmo. Las métricas que se presentan a continuación responden a distintas cuestiones, que pueden ser de mayor o menor interés según el problema al que se aplican.

Para simplificar las explicaciones, se utilizará el ejemplo siguiente. Un buzón de correo electrónico debe clasificar los correos que recibe y decidir si son "deseados" o si son "spam". Se considera la clase "deseados" como clase positiva y la de "spam" como clase negativa. Se trata de un caso de clasificación binaria, puesto que se consideran únicamente dos clases distintas.

La Tabla 1 contiene lo que se denomina una matriz de confusión. Se trata de una matriz $N \times N$, siendo N el número de clases consideradas, que resume las predicciones, ya sean correctas o incorrectas, de un modelo de clasificación. Presenta, por lo tanto, la correlación entre la etiqueta y la clasificación del objeto. Un eje de la matriz, en este caso el vertical, indica la clase real mientras que el otro, en este caso el horizontal, presenta la predicción de la clase.

Los objetos en verde, es decir los verdaderos positivos (VP) y los verdaderos negativos (VN), corresponden a aquellos que se han asociado a la clase real a la que pertenecen, mientras que los objetos en rojo, los falsos positivos (FP) y los falsos negativos (FN) no se han clasificado correctamente.

		Predicción de la clase	
		Clase = Deseado (S)	Clase = Spam (N)
Clase real	Clase = Deseado (P)	Verdadero positivo (VP)	Falso negativo (FN)
	Clase = Spam (N)	Falso positivo (FP)	Verdadero negativo (VN)

Tabla 1. Matriz de confusión. Ejemplo del correo electrónico.

El objetivo de todo problema de clasificación es, lógicamente, minimizar los errores de predicción y maximizar el número de predicciones correctas. La matriz de confusión puede ayudar a detectar patrones de error, ya que contiene la información necesaria para calcular varias métricas de rendimiento. A continuación, se detallan cuatro de estas métricas de rendimiento.

2.6.1 LA EXACTITUD

En primer lugar está la exactitud, la métrica más intuitiva, puesto que simplemente permite calcular la relación entre el número de predicciones realizadas correctamente y el número de predicciones total. Sin embargo, la exactitud no es capaz de analizar correctamente un conjunto de datos desequilibrados, es decir, donde existe un gran contraste entre el número de etiquetas positivas y negativas. El uso exclusivo de la exactitud no da una visión completa del desempeño del algoritmo. Formalmente, la exactitud se escribe:

$$Exactitud = \frac{VP + VN}{VP + VN + FP + FN}$$

2.6.2 LA PRECISIÓN

En segundo lugar está la precisión, que permite calcular cuántas predicciones positivas se han realizado correctamente. La precisión también puede calcular cuántas predicciones negativas se han llevado a cabo correctamente. La precisión se escribe:

$$Precisión = \frac{VP}{VP + FP}$$

ó

$$Precisión = \frac{VN}{VN + FN}$$

2.6.3 LA EXHAUSTIVIDAD

A continuación está la exhaustividad, que permite calcular cuántos objetos de clase real positiva se han atribuido correctamente a su clase. Al igual que la precisión, la exhaustividad también puede calcular cuántos objetos de clase real negativa se han clasificado correctamente. La exhaustividad se define de la siguiente manera:

$$Exhaustividad = \frac{VP}{VP + FN}$$

ó

$$Exhaustividad = \frac{VN}{VN + FP}$$

La precisión y exhaustividad evalúan mejor el rendimiento de problemas de clasificación cuando se dispone de un conjunto de datos desequilibrados. Ambas métricas deben examinarse para evaluar de manera óptima la efectividad del modelo, pero generalmente la mejora de una conlleva el deterioro de la otra.

2.6.4 EL VALOR-F

Finalmente, está el valor-F, la métrica más completa de las presentadas en este proyecto, además de ser la más valorada entre los expertos de *Machine Learning*, como es por otro lógico. Es la media armónica de los valores de precisión y exhaustividad, considerando así todas las predicciones positivas y negativas realizadas, ya sean correctas o incorrectas. La fórmula general depende de un parámetro β:

$$F_\beta = (1+\beta^2)\frac{Precisión \cdot Exhaustividad}{(\beta^2 \cdot Precisión) + Exhaustividad}$$

- si $\beta = 1$, la Precisión y la Exhaustividad tienen la misma ponderación.
- si $\beta > 1$, la Exhaustividad tiene mayor importancia.
- si $\beta < 1$, la Precisión tiene mayor importancia.

Para $\beta = 1$, el Valor-F se escribe de la siguiente manera:

$$F1 = 2\frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad}$$

El desempeño de un algoritmo de clasificación se puede valorar gracias a las cuatro métricas de rendimiento detalladas en este último apartado.

Tras detallar las características clave del Aprendizaje Automático, en el capítulo siguiente, se estudia cuál es el estado de la cuestión del objeto del presente trabajo.

Capítulo 3. ESTADO DE LA CUESTIÓN

El concepto de carga se puede definir de distintas maneras, por lo que el reconocimiento de cargas domésticas se puede plantear desde varios puntos de vista. En este capítulo se exponen dos planteamientos diferentes: el procesamiento de señales y la desagregación de los consumos.

3.1 EL PROCESAMIENTO DE SEÑALES Y ANÁLISIS ARMÓNICO

El estudio de señales, o bien procesamiento de señales, es una disciplina que nace mucho antes de la aparición de los ordenadores. Realmente, esta ciencia requiere una limitada serie de cálculos, que antiguamente se realizaban a mano y que se utilizaban para predecir el movimiento de cuerpos celestes o el análisis de las mareas.

El procesamiento de señales es un proceso por el que se aplican una serie de operaciones matemáticas a un conjunto de datos extraídos de una señal para obtener una determinada información. Este proceso, que se puede considerar como una simple manipulación de datos, está fuertemente relacionado con muchas disciplinas científicas y técnicas, como son por ejemplo, las matemáticas, la estadística o la optimización. El procesamiento de señales tiene actualmente una gran gama de aplicaciones en ámbitos como la electrónica y la automática, aunque también se utiliza en otras aplicaciones como el reconocimiento de voz, la localización por radar o la sismología.

El procesamiento de señales no sería una disciplina interesante si sólo considerase el aspecto temporal de las señales. Efectivamente, gran parte de la información de las señales se esconde en sus características frecuenciales. El análisis armónico se ha convertido, a lo largo de los siglos XIX y XX, por lo tanto, en una herramienta fundamental de esta ciencia.

La principal herramienta del análisis armónico es la transformada de Fourier, de notación $\mathbb{F}$. Se trata de una aplicación matemática que permite transformar señales del dominio del

tiempo al dominio de la frecuencia, y viceversa. Formalmente, hace corresponder una función del tiempo f y una función de la frecuencia g de la manera siguiente:

$$\mathbb{F}[f(t)] = g(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)\cdot e^{-j\omega t}\partial t$$

$$\mathbb{F}^{-1}[g(\omega)] = f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} g(\omega)\cdot e^{j\omega t}\partial \omega$$

donde f es necesariamente una función L^1, es decir integrable en el sentido de la integral de Lebesgue.

La transformada de Fourier se aplicará en este proyecto únicamente a funciones periódicas. Para estas funciones en particular, la transformada de Fourier se puede asimilar simplemente al cálculo de un conjunto de amplitudes complejas, también conocidas como coeficientes de la serie de Fourier. Las series de Fourier tienen la forma siguiente:

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}\left[a_n \cos\frac{2n\pi}{T}t + b_n sin\frac{2n\pi}{T}t\right]$$

Donde a_n y b_n son los llamados coeficientes de Fourier de la serie de Fourier de la función temporal $f(t)$.

No se entrará en más detalles sobre este punto puesto que no se utilizarán más conceptos teóricos que los que ya se han explicado anteriormente.

El primer planteamiento que se propone consiste en estimar que una carga es en realidad una simple señal eléctrica. La utilización de medidores inteligentes en las casas permite obtener una versión digital de diversas magnitudes eléctricas, como la potencia suministrada. Si se considera que cada carga puede caracterizarse por una señal temporal distinta, el reconocimiento de cargas se reduce en este caso al reconocimiento de señales.

3.2 LA DESAGREGACIÓN DE LOS CONSUMOS

El consumo responsable de electricidad se ha convertido, en los últimos años, en una de las preocupaciones principales de la sociedad actual. Ya sea por razones económicas o ambientales, el seguimiento y control del gasto eléctrico tienen una gran importancia, tanto a nivel individual como a nivel nacional.

En algunos sectores, como puede ser el industrial, el control del gasto energético es bastante riguroso, lo que permite elaborar medidas capaces de reducir al máximo consumos innecesarios. Sin embargo, numerosos estudios demuestran que, a nivel residencial, los usuarios son incapaces de realizar estimaciones correctas de su consumo particular.

Tomando dos hogares que contengan exactamente los mismos aparatos consumidores, no se registrará el mismo gasto energético en ambos puesto que éste depende fuertemente de las costumbres de los usuarios en cada una de las residencias. La utilización de dispositivos inteligentes en el sistema eléctrico podría mejorar las estimaciones y permitir así ahorros energéticos importantes. Para proporcionar información personalizada sobre el consumo de cada hogar, se han desarrollado nuevas herramientas y mecanismos como son los medidores inteligentes.

3.2.1 MEDIDORES INTELIGENTES

El reciente aumento del número de instalaciones residenciales que cuentan con medidores inteligentes, proporcionados por las compañías eléctricas, ha suscitado un interés en la monitorización de las cargas que las componen. Este despliegue tiene varios objetivos entre los cuales destacan la mejora del servicio de suministro eléctrico y la recopilación de informaciones sobre los hábitos de consumo.

Los medidores inteligentes, también llamados contadores inteligentes, son nuevos contadores que permiten obtener un registro más detallado del consumo que el que ofrecen los convencionales. Estos medidores proporcionan lecturas más frecuentes a la compañía eléctrica, además de avisar en caso de manipulación o robo del servicio. Sin embargo, lo

más interesante de estos sistemas es que ofrecen la posibilidad de transmitir toda la información que reciben a un centro donde se procesan todos estos datos, para facilitar el seguimiento, control y otros servicios personalizados del consumo particular.

3.2.2 DESAGREGACIÓN DEL CONSUMO ENERGÉTICO

Los medidores inteligentes, al igual que las facturas de electricidad, proporcionan únicamente medidas sobre el consumo total de energía, ya sea diario, mensual o anual. ¿Pero, qué pasaría si se pudiese proporcionar al cliente un desglose preciso del gasto de cada aparato?

Muchos estudios defienden que el hecho de disponer de información del consumo eléctrico, aparato por aparato, permite a los usuarios gestionar mejor su consumo. Esta gestión, que puede llegar a ser hasta un 15% más eficiente, puede llegar a tener importantes consecuencias medioambientales, puesto que la gran mayoría de la energía consumida en el sector residencial proviene actualmente de combustibles fósiles, como son el carbón y el gas. La falta de conocimiento sobre la energía consumida individualmente por algunos dispositivos conlleva además la aparición de falsas creencias entre los usuarios. Por ejemplo, estos tienden a subestimar la potencia requerida por los sistemas de climatización y a sobrestimar aquella que necesitan la iluminación o ciertos dispositivos como la televisión.

La monitorización de todos y cada uno de los dispositivos utilizados en una residencia es evidentemente imposible, por razones de coste. Se plantea por lo tanto la cuestión siguiente: ¿cómo efectuar el desglose de un consumo eléctrico de manera más barata?

La desagregación del consumo energético, también llamada en inglés *Non-intrusive Load Monitoring* (NILM) o *Non-intrusive Appliance Load Monitoring* (NIALM), se ha convertido en el objeto de numerosos estudios. Este proceso permite deducir qué dispositivos se están utilizando en una residencia, conociendo únicamente la corriente y el voltaje que se le está suministrando. Fue originalmente propuesto por George W. Hart, Ed Kern y Fred Schweppe del MIT a principios de los años 80, pero fue Hart quien continuó a desarrollarlo durante los años 90.

Como se ha mencionado brevemente, la desagregación del consumo se obtiene a partir de la señal del consumo total, también llamado agregado, y mediante el desarrollo de algoritmos, llamados de desagregación. La instalación de contadores eléctricos inteligentes en muchas viviendas ha impulsado el desarrollo de estos estudios.

3.2.3 BENEFICIOS DE LA DESAGREGACIÓN DEL CONSUMO ELÉCTRICO

Los beneficios que puede aportar la utilización del método de desagregación de la energía son numerosos y tocan diversos ámbitos. Algunos de ellos ya se han mencionado anteriormente, pero a continuación se elabora una lista más detallada.

- Reducción de energía en el sector residencial.
  - Automatización de los diagnósticos: detección rápida de fallos y, por lo tanto, limitación de gastos innecesarios.
  - Elaboración de recomendaciones y/o programas específicos para cada residencia según su patrón de funcionamiento.
  - Ahorro eléctrico mediante la sensibilización de los usuarios.
- Progreso y diseño de edificios y dispositivos.
  - Mejoras en la calificación energética de los edificios existentes.
  - Mejoras en el diseño de edificios de nueva construcción.
  - Avances de los aparatos consumidores mediante progresos en I+D.
- Progreso de los modelos económicos.
- Promoción de nuevas costumbres.
  - Avances de estudios de comportamiento.
  - Desarrollo de un marketing específico.

Aunque no se entrará en detalles en este capítulo sobre el desarrollo de los algoritmos de desagregación, se puede decir que éstos funcionan mediante la identificación de modelos estadísticos y de patrones dentro de la señal medida. Se necesitan por lo tanto registros reales de consumo eléctrico. Sin embargo, estas medidas eléctricas no se obtienen fácilmente, por lo que se han creado diferentes bases de datos de libre acceso que contienen registros eléctricos reales de distintas residencias.

Los investigadores de MIT fueron los precursores de esta iniciativa, con la publicación de "The Reference Energy Disaggregation Data Set (REDD)" en 2011. Sin embargo, son muchos los grupos de investigadores que, posteriormente, han creado bases de datos similares en países como Canadá, India, Francia o Reino Unido.

Para una gestión óptima de estas bases de datos, es importante tener en cuenta distintas informaciones sobre el país donde se han realizado estas medidas. En primer lugar, conocer las características del sistema eléctrico existente es indispensable para realizar un análisis correcto. En segundo lugar, hay que tener en cuenta que el conjunto de aparatos consumidores utilizados varía mucho de un país a otro. Por último, como ya se ha comentado, existe una clara relación entre ciertos hábitos y el consumo eléctrico. Se puede por lo tanto suponer que existe un patrón de costumbres o hábitos que pueda contribuir a la caracterización del consumo energético de cada país.

3.2.4 REQUISITOS DE LA BASE DE DATOS

Toda base de datos creada para estudiar la desagregación de la energía debe incluir la potencia consumida por el conjunto de la residencia, que será el dato de entrada para el algoritmo de desagregación. También se requiere una frecuencia de muestreo suficientemente elevada para limitar la tasa de errores, así como un periodo de registro de valores lo más largo posible para disponer de un número elevado de datos con los que trabajar.

Para mejorar la robustez del algoritmo, la incorporación del consumo individual de ciertos aparatos consumidores puede ser muy interesante. Los datos sobre el consumo individual permitirán, una vez se haya desarrollado el algoritmo, validar los resultados obtenidos. Cuantos más aparatos se incluyan, mejores rendimientos del algoritmo se podrán alcanzar.

3.3 UK-DOMESTIC APPLIANCE-LEVEL ELECTRICITY

En 2015 se crea en Reino Unido la primera base de datos de libre acceso y de alta resolución temporal registrada. Esta base de datos, llamada "*UK Domestic Appliance-Level Electricity*", aunque también conocida como UK-DALE, se inspira en la base de datos REDD.

3.3.1 DATOS CONTENIDOS EN UK-DALE

Los datos recogidos están disponibles de tres maneras distintas en UK-DALE, que se detallan a continuación.

En primer lugar, los datos se recogen mediante un contador de potencia a una frecuencia de 16 kHz (cada $6.25 * 10^{-5}$ segundos). La utilización de este conjunto de datos requiere un almacenamiento de 4 TBytes. Viene dividido en ficheros de 200 MBytes de manera que cada uno de ellos contiene una hora completa de medidas. Esta base de datos será difícilmente utilizable, por la cantidad de datos que contiene. En segundo lugar, los datos se recogen mediante un contador de potencia, a una frecuencia de 1 Hz (se realiza una medida cada segundo). Por último, se realizan medidas cada 6 segundos, es decir a una frecuencia de 0.16 Hz.

Se decide utilizar esta última base de datos para facilitar su uso, ya que es la que menos datos contiene.

3.3.2 ESTRUCTURA DE UK-DALE

Se describe a continuación la morfología de esta base de datos, que incluye registros de cinco viviendas inglesas diferentes. Estos cinco conjuntos de medidas vienen recogidos en cinco carpetas llamadas `house_x`, siendo `x` un número entero entre 1 y 5.

Cada una de estas carpetas contiene, como se muestra en la Figura 8:

- una serie de archivos, cada uno llamado `channel_i.dat`, de manera que cada uno corresponde a un contador eléctrico `i`. Dicho contador está asociado a un aparato consumidor.
- un fichero `labels.dat`, también en formato .CSV, que permite asociar cada archivo `channel` al nombre del aparato (ver Figura 9).

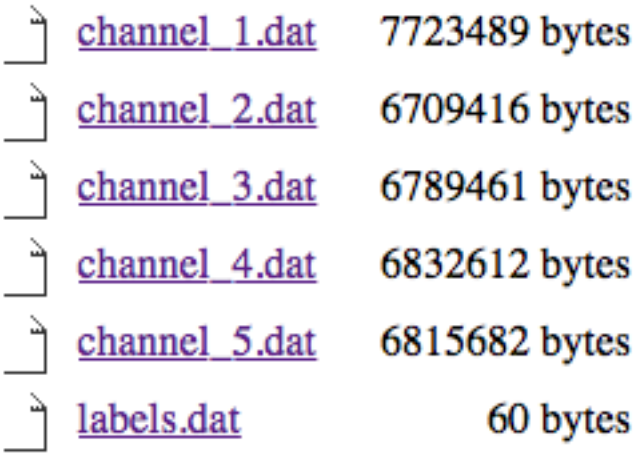


Figura 8. Contenido de la carpeta `house_3`.

```
1 aggregate
2 kettle
3 electric_heater
4 laptop
5 projector
```

Figura 9. Contenido del fichero `labels.dat` de la vivienda número 3.

Cada fichero `channel` está compuesto por dos columnas, como se muestra en la Figura 10.

- La primera columna es una marca de tiempo UNIX, es decir, el número de segundos que han transcurrido desde las 00:00:00 UTC del 1 de enero de 1970.
- La segunda columna contiene la propia medida de la potencia consumida en ese instante.

Para las medidas de potencia, se utilizan dos tipos de medidores: por una parte, los contadores que monitorizan la potencia activa (en watts), y por otra parte, los transformadores de corriente que miden la potencia aparente (en VA).

```
1361997314 5
1361997320 4
1361997326 5
1361997332 5
1361997338 4
1361997344 4
1361997350 4
```

Figura 10. Estructura de un archivo `channel_i`.

Los monitores individuales de los aparatos disponen de un interruptor que permiten encenderlos y apagarlos. Estos cambios de actividad se registran en un archivo `channel_i_button_press.dat` (ver Figura 11). Si el interruptor se acaba de encender, se registra un "1", y si se acaba de apagar, se registra un "0". Los eventos de encendido están perfectamente registrados, ya que la única manera de que se produzca este evento es que el usuario apriete el botón. Sin embargo, los eventos de apagado incluyen falsos positivos. A veces, los derechos de acceso se apagan espontáneamente, evento que no se puede distinguir de un apagado voluntario. Si éstos quedan apagados durante más de 12 segundos, el sistema asume que se han apagado deliberadamente.

```
1409028286 0
1409028292 1
1413885630 0
1413889885 1
1414430685 1
1415730428 1
1418386236 0
1418386246 1
```

Figura 11. Contenido del fichero `channel_i_button_press`.

Capítulo 4. APLICACIÓN AL RECONOCIMIENTO DE CARGAS DOMÉSTICAS

El Aprendizaje Automático es una disciplina muy versátil que se puede utilizar para numerosas aplicaciones. Tras haber descrito las tecnologías de Aprendizaje Automático existentes y haber estudiado la situación actual de la cuestión del reconocimiento de cargas domésticas, llegan las aplicaciones.

En el presente capítulo se describen los diferentes trabajos de aplicación realizados para este proyecto. En primer lugar, se procede a describir el algoritmo a utilizar. Después, se aplica dicho algoritmo al reconocimiento de cargas, según dos enfoques distintos del problema.

4.1 DESARROLLO DEL ALGORITMO

4.1.1 ELECCIÓN DEL ALGORITMO

La determinación del algoritmo ideal no siempre es fácil por lo que, en este caso, se ha utilizado la *cheat-sheet* de *skicit-learn* (ver Figura 4), de la que ya se ha hablado en el apartado 2.4 para escoger el que más se adapta al problema de reconocimiento de cargas.

Se comienza en la casilla START y partiendo de la premisa de que se dispone de más de 50 ejemplos, surge la siguiente pregunta: ¿se desea predecir una categoría? Un reconocimiento es efectivamente una clasificación, por lo que se puede afirmar que el objetivo de este proyecto es la determinación de una categoría. En un intento de simplificar el estudio, se parte también de la premisa de que los datos de entrenamiento ya vienen etiquetados, por lo que se entra directamente en el ámbito de los algoritmos de clasificación por aprendizaje supervisado. Para limitar el tiempo de computación por falta de material adecuado, se decide no utilizar más de 100.000 ejemplos de entrenamiento.

La *cheat-sheet* nos aconseja utilizar el algoritmo de Clasificación de vectores de soporte. Sin embargo, como su desarrollo parece complejo y se desea desarrollar el algoritmo desde cero, se decide pasar al siguiente paso.

La base de datos que se pretende utilizar contiene valores de consumos, es decir, valores numéricos, por lo que se acaba seleccionando el algoritmo que será utilizando en el presente proyecto: el clasificador de los *k* vecinos más próximos.

4.1.2 MÉTODO DE LOS K VECINOS MÁS CERCANOS

El método de los *k* vecinos más cercanos, o *k*-Nearest Neighbours en inglés, abreviado *k*-NN, es un método de clasificación que permite predecir las etiquetas de objetos de clase desconocida. Al ser un método de aprendizaje supervisado, y como ya se ha explicado en el apartado 2.3, el algoritmo necesita una fase de entrenamiento previa para "aprenderse" un conjunto de ejemplos de clasificación ya conocida. Para que el algoritmo funcione correctamente, es importante escoger ejemplos que sean relativamente "cercanos" a aquellos que se desean predecir.

Esta sección incluye una descripción detallada del funcionamiento del algoritmo.

4.1.2.1 Funcionamiento del algoritmo

4.1.2.1.1 Ejemplo

Aunque el concepto que hay detrás del *k*-NN no es especialmente complejo, se utilizará un ejemplo para facilitar su comprensión.

Se considera un conjunto de puntos pertenecientes a un plano, como se ilustra en la Figura 12. Este conjunto de datos se denomina conjunto de ejemplos de entrenamiento, o *training set* en inglés. Cada punto representa un ejemplo de entrenamiento, y como se puede apreciar en la figura, puede pertenecer a dos clases distintas: la clase "roja" o la clase "gris".

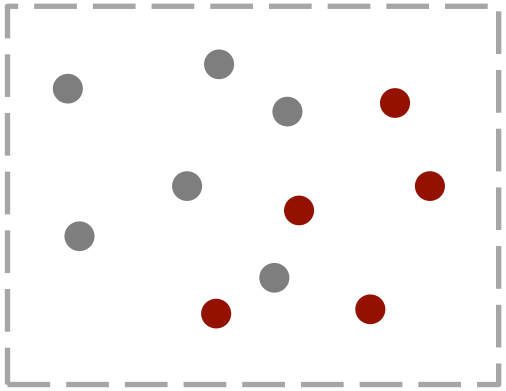


Figura 12. Ejemplo funcionamiento. Conjunto de ejemplos de entrenamiento.

A continuación, se introduce un nuevo punto al plano, denominado objeto de prueba, o de *test* en inglés. La clase a la que pertenece este objeto se desconoce, por lo que se representa en negro en la Figura 13. El algoritmo debe poder determinar si el nuevo objeto es realmente de clase "roja" o de clase "gris".

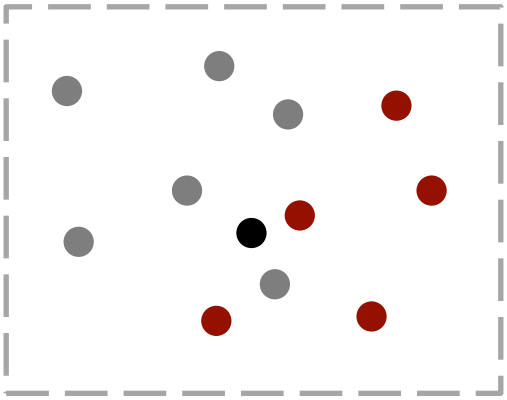


Figura 13. Ejemplo funcionamiento. Introducción de un objeto de clase desconocida.

Para ello, se seleccionan los k vecinos más próximos al punto de clase desconocida. Para este ejemplo, se toma el valor $k = 3$, por lo que se seleccionan los tres puntos indicados en la Figura 14.

Por último, se analizan los puntos seleccionados: predominan los grises (hay dos puntos grises, frente a uno rojo). El algoritmo estima, por lo tanto, que el nuevo objeto debe pertenecer a la categoría "gris".

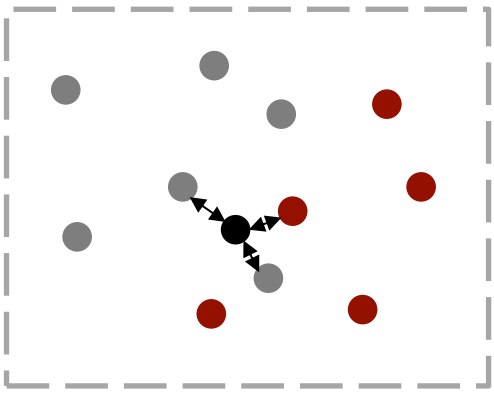


Figura 14. Ejemplo funcionamiento. Cálculo de los k vecinos más cercanos.

4.1.2.1.2 Explicación formal del método

El método de los k vecinos más cercanos guarda en memoria un conjunto de N parejas "*entrada - salida*". A partir de estos ejemplos, es capaz de estimar la salida de cualquier nueva entrada x. Dado que se trata de un algoritmo de clasificación, las salidas son las clases de los objetos.

Este método se aplica en dos fases consecutivas: fase de entrenamiento y posteriormente, fase de clasificación.

La primera fase, la fase de entrenamiento, consiste en "enseñar" al algoritmo el conjunto de ejemplos cuya clasificación se conoce a priori. Estos ejemplos están compuestos por dos partes "*entrada"* y "*salida*". La primera es un vector multidimensional compuesto por p atributos:

$$x_i = (x_{1i},\ x_{2i}, \dots, x_{pi}) \in X$$

donde:

X es el conjunto de ejemplos de entrenamiento

$(x_{1i}, x_{2i}, \dots, x_{pi})$ son los p atributos que caracterizan al objeto ejemplo de entrenamiento

La segunda es una etiqueta, como puede ser el color "rojo", que representa la clase a la que pertenece el ejemplo.

Un ejemplo x se puede por lo tanto escribir de la siguiente manera:

$$(x_1, x_2, \dots, x_p), "rojo"$$

Durante la fase de entrenamiento, el algoritmo almacena el conjunto de ejemplos x_i. Cada uno de ellos se puede representar en el espacio de dimensión p.

En la fase de clasificación, se introduce un nuevo objeto del que se conocen únicamente sus p atributos. El algoritmo calcula la distancia entre el nuevo punto introducido y todos los puntos de los ejemplos de entrenamiento. Se seleccionan los k ejemplos más cercanos, es decir, los de distancia mínima al nuevo punto. Finalmente, se puede asociar al nuevo objeto la clase más común entre los k ejemplos seleccionados.

4.1.2.2 Especificaciones del algoritmo

4.1.2.2.1 Cálculo de las distancias

El cálculo de distancias entre puntos multidimensionales se puede realizar de diferentes maneras.

En este caso se utiliza la distancia euclídea entre dos puntos n-dimensionales. Ésta se calcula de la siguiente manera:

$$d_E(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

En otros casos, este cálculo puede estar ponderado para dar una mayor importancia a los atributos que mejor caractericen las clases o que mejor permitan distinguirlas entre ellas.

4.1.2.2.2 Cálculo del valor del parámetro k

El valor que toma el parámetro k es crítico para el buen funcionamiento del algoritmo. El cálculo de este valor se debe realizar durante la fase de entrenamiento para poder empezar a utilizarlo desde el comienzo de la fase de prueba. Sin embargo, durante la primera etapa, los únicos datos conocidos son los ejemplos de entrenamiento. Para poder calcular el parámetro k, éstos se dividen aleatoriamente en dos subconjuntos: el de entrenamiento y el de prueba.

El método comúnmente utilizado para el cálculo del parámetro k es la validación cruzada, o *cross validation*. Se trata de un método que evalúa los resultados de un análisis estadístico para verificar que éstos son independientes de la partición elegida entre datos de entrenamiento y datos de prueba. Entre los distintos tipos de validación cruzada, la más utilizada es la validación cruzada de K iteraciones, o K—*cross validation*.

En la validación cruzada de K iteraciones, el conjunto de ejemplos de entrenamiento inicial se divide en K subconjuntos de mismo tamaño. En una primera iteración, el primer subconjunto se toma como conjunto de datos de prueba y los $(K-1)$ restantes como conjunto de datos de entrenamiento. Este proceso se repite durante K iteraciones, seleccionando cada vez unos datos de prueba distintos. El resultado final es la media de los resultados de todas las iteraciones.

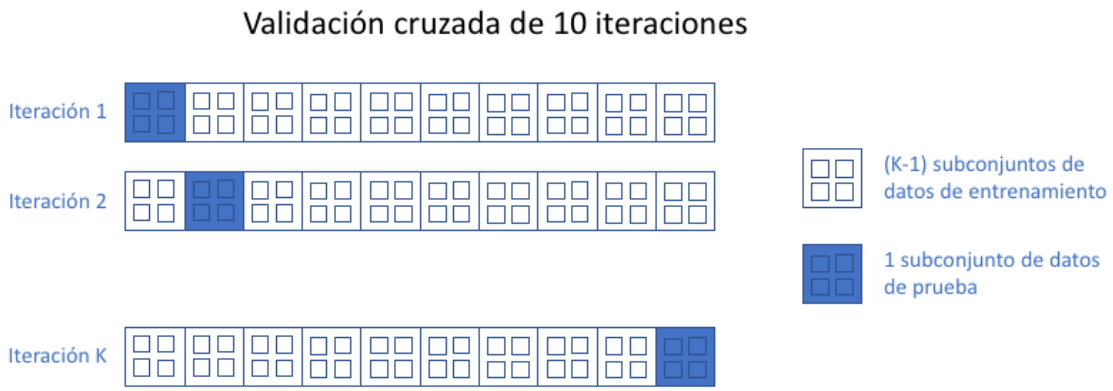


Figura 15. Método de la validación cruzada de 10 iteraciones.

Se trata de un método muy preciso ya que se obtiene a partir del análisis de K combinaciones diferentes de los datos. Sin embargo, esta gran precisión está inconvenientemente acompañada de una gran complejidad computacional: cuanto mayor sea el valor de K o el tamaño del conjunto de datos, mayor precisión, pero también es mayor el número de cálculos que debe realizar el algoritmo.

El ejemplo que se muestra en la Figura 15 presenta una aplicación de validación cruzada con de K iteraciones, donde $K = 10$.

Hay un total de 40 datos de entrenamiento, que se dividen en 10 subconjuntos de 4 datos cada uno. A cada iteración, el subconjunto que se utiliza de prueba es distinto, como viene ilustrado en la Figura 15.

El cálculo del valor óptimo de k es esencial para evitar un subajuste o un sobreajuste del algoritmo. El valor óptimo de k es aquél que corresponde a una tasa de error de predicción mínima.

4.1.3 PRIMER ESTUDIO

4.1.3.1 Base de datos

El conjunto de datos *Flor de Iris*, también conocido como el conjunto de datos *Iris de Fisher*, es un conjunto de datos multivariante que fue utilizado por primera vez como ejemplo de análisis discriminante lineal por el estadístico y biólogo inglés Ronald Fisher, en 1936, para su artículo "*The use of multiple measurements in taxonomic problems*". Contiene informaciones de 50 muestras de cada una de las especies de flor de iris siguientes: *Iris setosa*, *Iris virginica* e *Iris versicolor*. Para cada muestra, se midieron las cuatro características siguientes: longitud y anchura de sus pétalos, así como de sus sépalos (ver Tabla 2).

Flor de Iris será utilizado como base de datos para comprobar el funcionamiento de un primer algoritmo desarrollado en Python. Los datos están recogidos en una tabla donde cada fila es un ejemplo de entrenamiento, es decir, una muestra de flor. Cada fila contiene en

primer lugar los cuatro atributos que caracterizan cada muestra y, en segundo lugar, la etiqueta que determina a qué especie pertenece dicha muestra.

Largo de sépalo	**Ancho de sépalo**	**Largo de pétalo**	**Ancho de pétalo**	**Especie**
4,9	3,0	1,4	0,2	*Iris setosa*
4,7	3,2	1,3	0,2	*Iris setosa*
4,6	3,1	1,5	0,2	*Iris setosa*
5,0	3,6	1,4	0,2	*Iris setosa*
7,0	3,2	4,7	1,4	*Iris versicolor*
6,4	3,2	4,5	1,5	*Iris versicolor*
6,9	3,1	4,9	1,5	*Iris versicolor*

Tabla 2. Estructura de la base de datos Iris de Fisher.

4.1.3.2 Desarrollo del código

En primer lugar, se crea un vector de $(p + 1)$ dimensiones de etiquetas. Éste contiene, por orden, los nombres de los p atributos y una última etiqueta llamada `class`. En este caso, se obtiene el vector `names` siguiente:

```
# define column names
names = ['sepal_length','sepal_width','petal_length',
'petal_width', 'class']
```

A continuación se importan los N ejemplos de entrenamiento contenidos en el fichero gracias al comando `pd.read_csv`.

El comando `drop` permite eliminar una o varias columnas de una matriz. Aquí se utiliza para eliminar la columna `class` y poder crear el vector `X_train`. Éste contiene los N vectores de atributos asociados a cada ejemplo de entrenamiento.

La función `get_dummies`, que se aplica a la columna `class`, simplifica el tratamiento de las etiquetas de clase ya que sirve para reemplazar caracteres por valores numéricos. Se recorre la columna `class` y cada etiqueta distinta que se encuentra, se introduce en otro vector `y_train` como valor. Es decir, en vez de almacenar las etiquetas como "*iris setosa*" y "*iris versicolor*", la guarda como 0 y 1. El vector `y_train` contiene por tanto una serie de longitud *N* de valores numéricos.

```
# loading training data
df = pd.read_csv('data/iris.csv', header=None, names=names)
df.head()

X_train = df.drop(['class'], axis=1).values
y_train = pd.get_dummies(df['class']).values[:,1]
```

A menudo, la representación gráfica de un conjunto de datos facilita su comprensión. Como en este caso los puntos que se desea representar son 4-dimensionales, éstos no se pueden representar en un plano. Por lo tanto, se separan en dos gráficos distintos, de manera que uno contenga la longitud y el ancho de los pétalos y el otro los de los sépalos.

El gráfico superior de la Figura 16 representa el ancho y el largo de los pétalos y el gráfico inferior, los de los sépalos. Las muestras de la misma especie están representadas por puntos de mismo color.

```
# print(X_train.shape)
plt.scatter(X_train[:,0], X_train[:,1], c=y_train)
plt.show()
# plt.scatter(X_train[:,2], X_train[:,3], c=y_train)
```

Las partes del código siguientes se exponen de manera lógica, y no siguiendo la estructura del código.

Una vez terminada la fase de entrenamiento, comienza la de prueba. La variable `X_test` definida en el código `main` permite introducir tantos objetos de prueba como se deseen clasificar. Cada uno debe ser un vector de cuatro valores numéricos que caracterizan las dimensiones de los pétalos y de los sépalos de la flor de prueba que se pretende clasificar. En el código presentado más abajo, sólo se introduce un objeto de prueba.

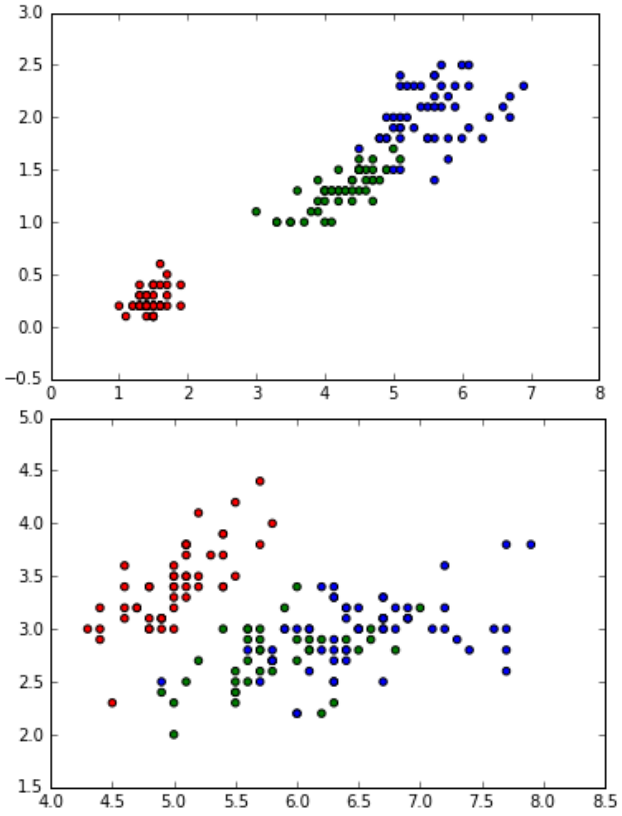


Figura 16. Training set *del conjunto de datos* Flor de Iris.

También en el `main`, se llama a la función `kNearestNeighbor`, que utiliza a su vez otra función llamada `predict`. Se supone que para este caso el valor óptimo de k es 10, aunque luego se describa el código necesario para obtener dicho valor.

```
X_test = np.array([[ 5.5 , 3, 3.5, 1.25]])
kNearestNeighbor(X_train, y_train, X_test, 10)
```

La función `predict`, como su nombre bien indica, permite predecir la clase más común entre los k vecinos más próximos de la muestra de prueba. Calcula, una por una, todas las distancias euclídeas que hay entre el objeto de prueba `x_test` introducido en la función y los N objetos de entrenamiento. Éstas se almacenan en el vector `distances` y se organizan por orden creciente. En el vector `index` se almacenan los objetos correspondientes a las k primeras distancias de `distances`. Es así como se localizan los k vecinos más próximos.

La función `Counter().most_common()` calcula cuál es el valor más común entre los objetos de `index` para finalmente poder estimar a qué clase pertenece nuestro nuevo objeto.

```
def predict(X_train, y_train, x_test, k):
      # create list for distances and targets
      distances = []
      targets = []

      for i in range(len(X_train)):
      # first we compute the euclidean distance
      distance = np.sqrt(np.sum(np.square(x_test - X_train[i, :])))
      # add it to list of distances
      distances.append([distance, i])

      # sort the list
      distances = sorted(distances)

      # make a list of the k neighbors' targets
      for i in range(k):
      index = distances[i][1]
      targets.append(y_train[index])

      # return most common target
      return Counter(targets).most_common(1)[0][0]
```

La función `kNearestNeighbor`, llamada desde el `main`, recibe una lista `X_test` compuesta de objetos de clase desconocida. Mediante un bucle `for`, se encarga de aplicar la función `predict` a cada uno de los objetos `x_test` de dicha lista. Finalmente se obtiene un vector llamado `predictions` que contiene, una a una, todas las clases estimadas por la función `predict` para cada objeto.

```
def kNearestNeighbor(X_train, y_train, X_test, k):
      predictions = []
      # transforming them into lists
      X1 = X_train.tolist()
      y1 = y_train.tolist()
      for i in range(len(X_test)):
      a = predict(X_train, y_train, X_test[i, :], k)
      predictions.append(a)
      X1+=[X_test[i].tolist()]
      y1+=[a]
      y = np.array(y1)
```

Por último, para facilitar la lectura de los resultados, se incluyen los objetos recientemente clasificados en la representación anterior del conjunto de entrenamiento. Lógicamente, los nuevos puntos se trazan del color asociado a su clase.

```
        X = np.array(X1)
        plt.scatter(X[:,0], X[:,1], c=y)
        plt.show()
        plt.scatter(X[:,2], X[:,3], c=y)
```

4.1.3.3 Mejoras del código: validación cruzada

El valor de k utilizado en el párrafo anterior no es el que minimiza los errores para esta base de datos. Para obtener este valor, se debe escribir el código que permita aplicar la validación cruzada al conjunto de datos.

```
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X_train,
y_train, test_size = 0.33, random_state = 42)
Predictions = kNearestNeighbor(X_train, y_train, X_test,
predictions, 1)
Accuracy = accuracy_score(y_test, predictions)
```

El código anterior permite dividir el *training set* en un subconjunto de entrenamiento y un subconjunto de prueba. A continuación se aplica el algoritmo de clasificación k-NN, considerando dichos subconjuntos, y para $k = 1$. Se evalúa la exactitud con la función `accuracy`, que se importa de la biblioteca `sklean.metrics`.

Si se generaliza este proceso para una lista de valores de k, se obtiene el método de validación cruzada. Se utilizan únicamente valores impares de k para evitar situaciones no concluyentes. Por ejemplo, si se toma $k = 2$ y que el algoritmo detecta que los dos vecinos más próximos son uno azul y otro rojo, será imposible decidir cuál de las dos clases es la correctamente estimada.

```
# creating odd list of K for KNN
myList = list(range(1,50))
# subsetting just the odd ones
```

```
neighbors = [x for x in myList if x%2 != 0] # empty list that will hold cv scores
cv_scores = []
# perform 10-fold cross validation
for k in neighbors:
     knn = KNeighborsClassifier(n_neighbors=k)
     scores = cross_val_score(knn, X_train, y_train, cv=10,
scoring='accuracy')
     cv_scores.append(scores.mean())
# changing to misclassification error
MSE = [1 - x for x in cv_scores]
# determining best k
optimal_k = neighbors[MSE.index(min(MSE))
```

Para visualizar mejor la relación que existe entre el valor de k y la tasa de error de predicción, ésta se puede representar en un gráfico que se obtiene mediante el código siguiente.

```
plt.plot(neighbors, MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
```

4.1.4 EJEMPLO DE APLICACIÓN

Se supone que se quiere determinar la especie de una muestra de flor de iris que tiene las siguientes características:

$$Longitud_{pétalo} = 5{,}5$$

$$Ancho_{pétalo} = 3$$

$$Longitud_{sépalo} = 3{,}5$$

$$Ancho_{sépalo} = 1{,}25$$

Se introduce el siguiente vector en el código:

```
X_test = np.array([[ 5.5 , 3, 3.5, 1.25]])
```

En este caso se utilizan sólo dos especies de iris para crear el conjunto de ejemplos de entrenamiento. Las dos especies se representan en colores azul y rojo. Se introduce el vector `X_test` siguiente en el `main` y obtenemos el siguiente resultado: el nuevo objeto clasificado aparece de clase "roja" en la Figura 17.

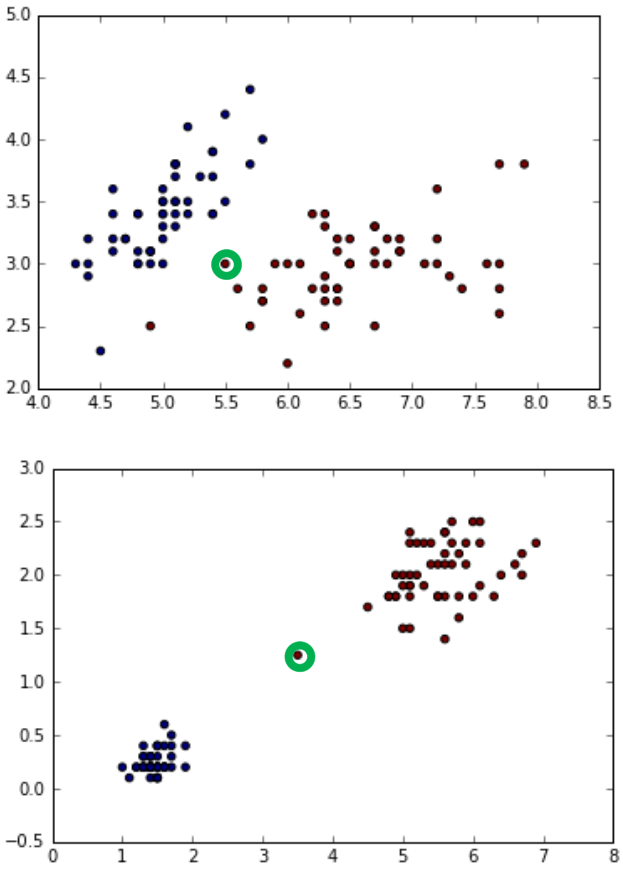


Figura 17. Conjunto de ejemplos de entrenamiento con dos especies y nuevo objeto de clase "roja".

Además, como bien indica el gráfico de la Figura 18, el valor óptimo de k para este conjunto de datos es $k = 7$.

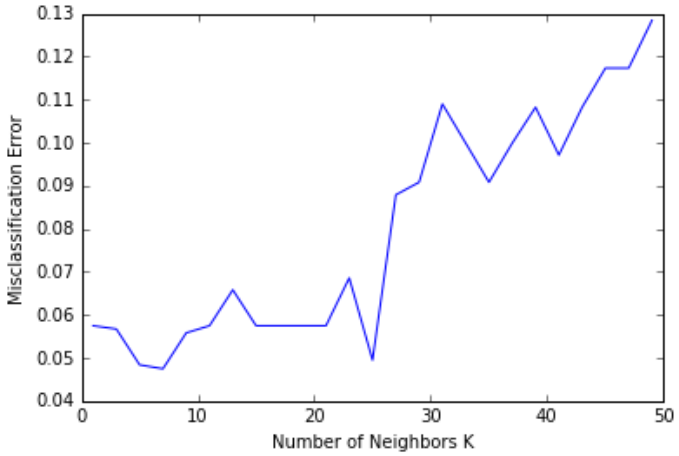


Figura 18. Cálculo del valor óptimo de k en función de la tasa de error de predicción.

El algoritmo desarrollado se utiliza a continuación para cumplir el objetivo de este proyecto, a saber, el reconocimiento de cargas domésticas. Se realizan dos aplicaciones distintas según la manera con la que se definen las cargas. Este problema se enfoca de dos maneras diferentes: en un primer lugar, se hace un reconocimiento de señales y, en segundo lugar, se reconocen las cargas encendidas en un momento determinado.

4.2 APLICACIÓN Nº1: RECONOCIMIENTO DE SEÑALES

Muchas magnitudes eléctricas, como pueden ser la intensidad, el voltaje o la potencia, son realmente señales periódicas, con una cierta amplitud y frecuencia. En esta primera aplicación, se utilizan señales para caracterizar las cargas que se quieren reconocer.

Las bases de datos empleadas para este estudio son muy sencillas de construir. Están formadas de vectores bidimensionales, donde la primera coordenada es una referencia temporal y, la segunda, es la amplitud de la señal estudiada.

4.2.1 VERIFICACIÓN

Para comprobar que se obtienen los resultados esperados, se plantea un primer ejemplo de uso del algoritmo.

Para un mismo periodo de tiempo, se consideran las dos funciones siguientes: la función $sen(\omega t)$, y la función $sen(2\omega t)$, en azul y en rojo, respectivamente, en la Figura 19. El muestreo de ambas funciones nos permite obtener el conjunto de ejemplos de entrenamiento, que contiene objetos de dos funciones y, por lo tanto, de dos clases.

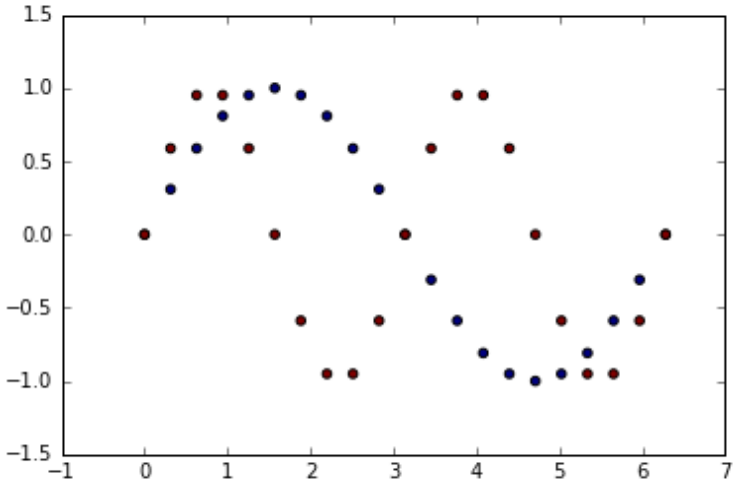


Figura 19. Nuevo conjunto de datos de entrenamiento.

Para aclarar la equivalencia entre las estructuras de los datos de *Flor de Iris* y los que se crean para esta parte, se vuelven a definir las variables utilizadas para la elaboración del código:

- El vector `X_train` contiene las coordenadas (t, x) de cada punto.
- El vector `y_train` es una lista de valores numéricos que contiene la etiqueta de todos los puntos.

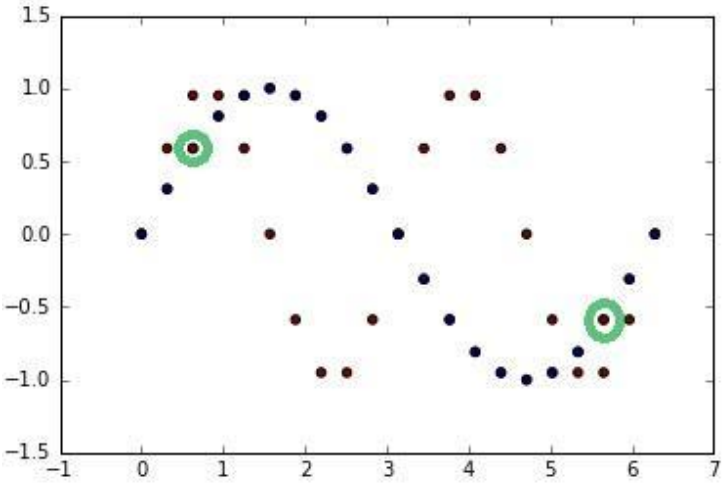


Figura 20. Test con $sen(\omega t)$.

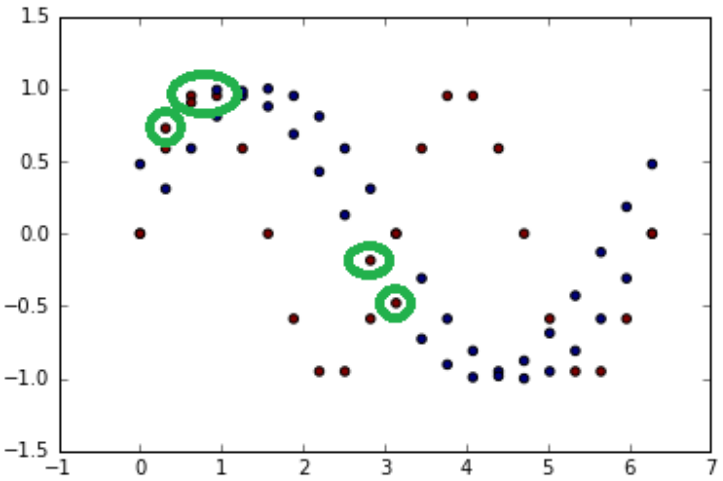


Figura 21. Test *con* $sen(2\omega t)$.

Para comprobar el correcto funcionamiento del algoritmo se utilizan los propios ejemplos como datos de prueba. Es decir, se introducen los puntos de la función $sen(\omega t)$ para que el algoritmo los clasifique. Teóricamente, las estimaciones de éste deberían ser perfectas, ya que se ha "aprendido" exactamente estos puntos durante la fase de entrenamiento. Este mismo proceso también se lleva a cabo con los puntos de la función $sen(2\omega t)$. Las clasificaciones obtenidas se encuentran representadas en la Figura 20 y Figura 21, respectivamente. Los puntos indicados con círculos verdes son aquellos que no se han clasificado correctamente. En la Figura 20, los puntos marcados son realmente de clase azul porque pertenecen a la función $sen(2\omega t)$, aunque se hayan identificado como rojos. Pasa lo mismo con los puntos rodeados en la Figura 21, que deberían ser rojos ya que pertenecen a la función $sen(\omega t)$ y sin embargo, están representados con puntos azules.

La falta de precisión constatada podría limitarse con un aumento de la frecuencia de muestreo. Sin embargo se puede admitir que el algoritmo ha reconocido correctamente ambas funciones, puesto que la mayoría de los puntos que la componen pertenecen a la clase adecuada.

4.2.2 INTRODUCCIÓN DE UN DESFASE

Se quiere que el algoritmo sea capaz de reconocer una función $sen(\omega t)$, incluso si ésta está desfasada. Se introducen los puntos que componen la función $sen(\omega t)$ con un desfase de 0,5 como datos de prueba. Se obtiene la clasificación representada en la Figura 22.

Si se razona igual que en el apartado anterior, se admite que el algoritmo ha reconocido la señal porque la mayoría de sus puntos están correctamente clasificados y representados en azul. Se prevé, sin embargo, que el aumento del desfase perjudica fuertemente los resultados obtenidos.

La modificación de la amplitud de la función tiene las mismas consecuencias sobre la calidad de la clasificación que la introducción de un desfase. Si dicha modificación es demasiado importante, se puede llegar a no reconocer la función correctamente.

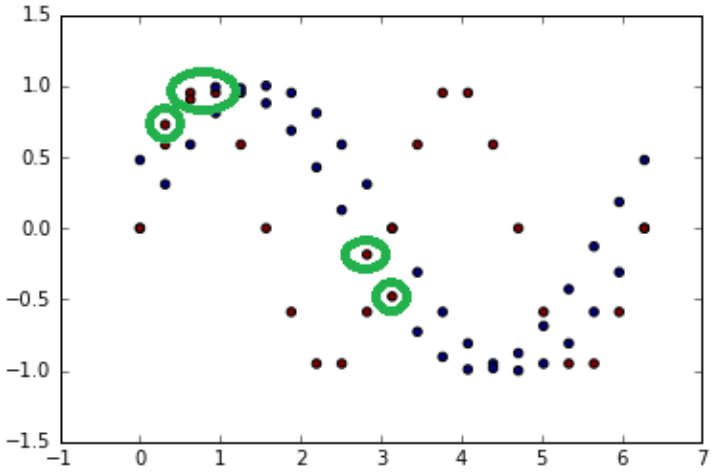


Figura 22. Test *con* $sen(\omega t + 0{,}5)$.

4.2.3 OTRAS SEÑALES MÁS COMPLEJAS

La representación temporal de magnitudes eléctricas será aún más complicada que las señales que se han tratado hasta ahora. Se construye por lo tanto el conjunto de datos de entrenamiento representado en la Figura 23, que está formado por los puntos muestreados de las siguientes funciones:

- $y = sin(x)$
- $y = sin(x) + 3sin(2x)$
- $y = sin(x) + 3sin(2x) + cos(x)$
- $y = sin(x) + 3sin(2x) + cos(3x)$
- $y = sin(x) + sin(4x) + cos(5x)$

La función de Python `train_test_split` divide el conjunto de datos de entrenamiento en dos subconjuntos: uno de entrenamiento y otro de prueba. Para cuantificar rápidamente el rendimiento del algoritmo con respecto a esta base de datos, se clasifican los puntos del subconjunto de prueba. A continuación, se calcula la exactitud como se muestra en el código inferior.

```
X_train, X_test, y_train, y_test = train_test_split(X_train,y_train,
test_size=0.33, random_state=42)

# making our predictions
predictions = []
```

```
predictions = kNearestNeighborsinus(X_train, y_train, X_test, 1)

# transform the list into an array
predictions = np.asarray(predictions)

# evaluating accuracy
print(y_test)
print(predictions)
accuracy = accuracy_score(y_test, predictions)
```

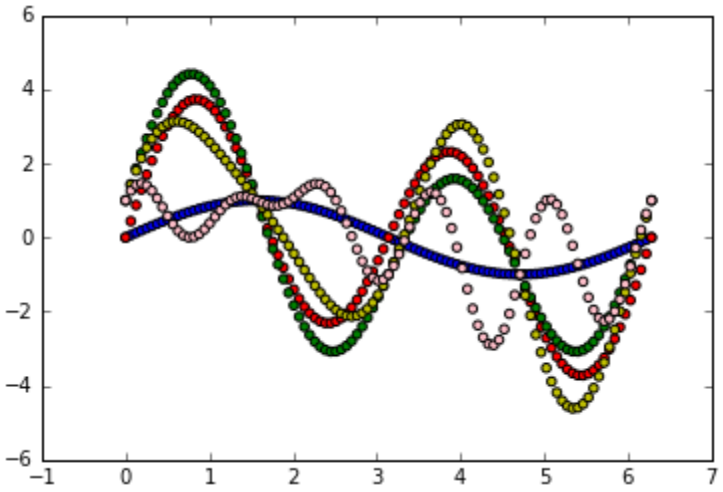


Figura 23. Conjunto de datos de entrenamiento compuesto por cinco señales diferentes.

En la Figura 24 se muestran los puntos del subconjunto de prueba, una vez se han clasificado. Por problemas de manejo del programa, los colores de las clases cambian con respecto a la representación del *training set* en la Figura 23. El azul marino se mantiene, pero el rosa es ahora burdeos, el amarillo es naranja, el verde oscuro es verde claro y el rojo es azul claro. Tras la clasificación se obtiene una exactitud de 0,47305.

Un segundo test se lleva a cabo, utilizando como datos de prueba los puntos de la señal número 2 desfasada de 0,5. Es una señal muy próxima a la original, por lo que se espera obtener una mejor exactitud. Sin embargo, la clasificación es la mostrada en la Figura 25 y la exactitud es esta vez de 0,38614.

Las dos exactitudes obtenidas son mediocres, y evidencian los límites del algoritmo k-NN en cuanto a reconocimiento de señales temporales.

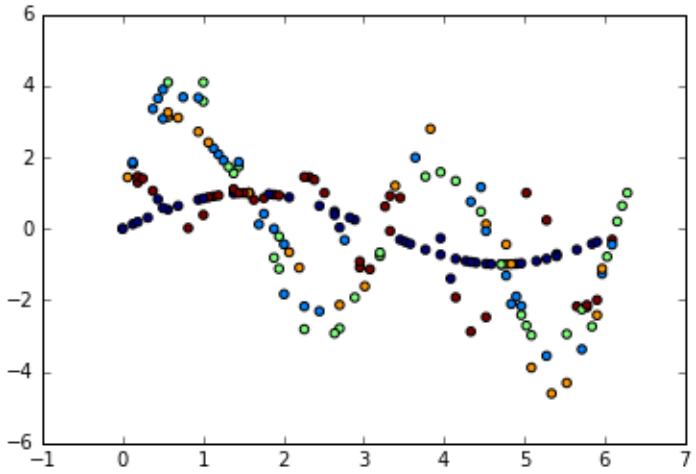


Figura 24. Clasificación de los puntos del test set *elaborado por la función* `train_test_split`.

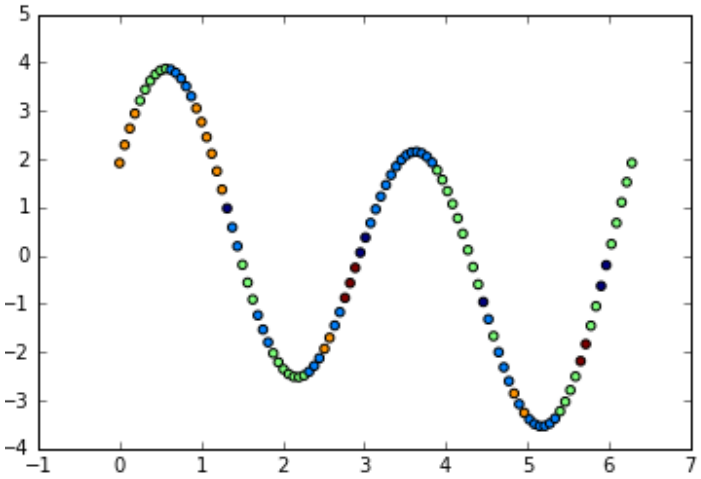


Figura 25. Clasificación de la función número 2 con un desfase de 0,5.

4.2.4 ESTUDIO FRECUENCIAL

Para intentar resolver el problema observado en el apartado anterior se considera un planteamiento alternativo del problema. Hasta ahora sólo se ha considerado el aspecto temporal de las señales. Para poder distinguir ciertas señales o tipos de señales, puede ser interesante analizar sus coeficientes de Fourier.

Se propone utilizar el algoritmo k-NN para distinguir tres funciones: una sinusoidal, una triangular y una rectangular. Sus representaciones de Fourier son bastante peculiares y

diferentes entre sí, por lo que se puede suponer que el reconocimiento de las señales se podrá llevar a cabo de forma correcta.

El conjunto de datos de entrenamiento son de nuevo vectores bidimensionales, cuya primera componente es el coeficiente de Fourier y la segunda es la frecuencia asociada a dicho coeficiente. Estos coeficientes se obtienen gracias a la función `fft` de la biblioteca de Python `spicy.fftpack`, como se muestra en el código siguiente. Dado que la transformada de Fourier será una función simétrica con respecto al eje de las ordenadas, se consideran únicamente las frecuencias positivas. Además, se muestrean las tres transformadas, para reducir el número de puntos y facilitar el manejo de datos. Dado que sólo las funciones seno y rectangular existen en Python, también se describe en el código cómo hacer para crear la función triangular. Esta señal se construye por repetición del patrón elemental en cada final de periodo.

```
# définition des constantes du signal
K = 2*pi      # facteur conversion période/fréquence
A0 = 4        # amplitude fréquence fondamentale
A1 = 8        # amplitude première harmonique
f0 = 2        # fréquence fondamentale (Hz)
f1 = 8        # fréquence première harmonique (Hz)

# définition temporelle du signal
t0 = 0              # début de l'acquisition du signal
t1 = 10             # fin de l'acquisition (s)

# définition des paramètres d'échantillonnage
FreqEch = 1024            # fréquence d'échantillonnage
PerEch = 1./FreqEch       # période d'échantillonnage
N = FreqEch*(t1 - t0)     # nombre de points échantillonnés sur l'intervalle

# définition du temps
t = linspace(t0, t1, N)

# définition du signal
def triangle(length, amplitude):
     section = length // 4
for direction in (1, -1):
       for i in range(section):
              yield i * (amplitude / section) * direction
       for i in range(section):
              yield (amplitude - (i * (amplitude / section))) * direction
length = len(t)
L = np.zeros(length)
```

```
L=list(L)
duree = int(length/((t1-t0)*f0)) reference = list(triangle(duree,A0))

for k in range(duree):
    L[k] = reference[k]
for k in range(duree+1,length):
    L[k] = L[k-duree]
signal0 = (A0)*sin(f0*K*t)
signal1 = A0*sin(f0*K*t) + A1*sin(f1*K*t)

signal_carre = A0*signal.square(2 * np.pi * f0 * t)
signal_triangulaire = np.array(L)
```

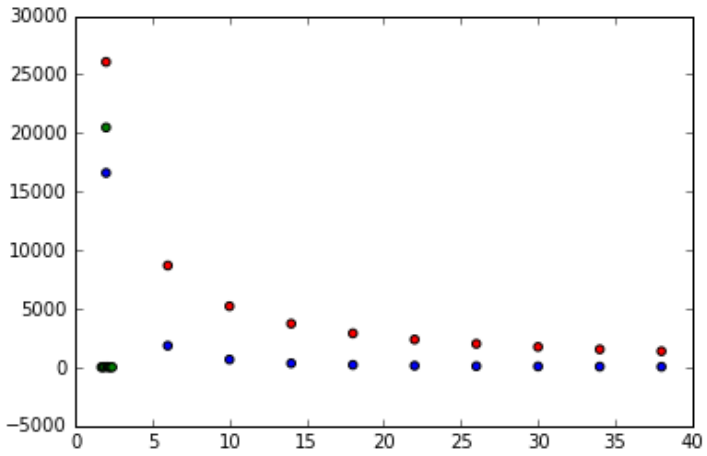


Figura 26. Conjunto de datos de entrenamiento. Coeficientes de Fourier de tres señales: sinusoidal, triangular y cuadrada.

La representación del conjunto de datos se encuentra en la Figura 26. Se reconoce claramente la transformada de Fourier de la señal sinusoidal de frecuencia $f_0 = 2\ Hz$ en verde, que es simplemente un dirac a la frecuencia fundamental f_0.

Se vuelve a dividir el conjunto de datos de entrenamiento en dos subconjuntos de entrenamiento y de prueba, como ya se ha hecho en el apartado 4.2.3. mediante la función `train_test_split`. El resultado obtenido es el que se representa en la Figura 27, y se obtiene una exactitud de 0,82331, mucho mayor que las obtenidas anteriormente.

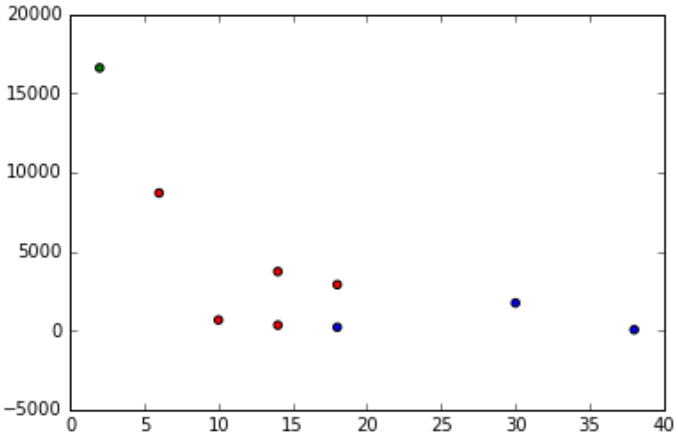


Figura 27. Resultado de la fase de test con *conjunto de prueba aleatorio.*

A continuación se propone introducir ciertas modificaciones de amplitud y fase para verificar el buen funcionamiento del algoritmo aun con estas modificaciones.

Como se puede observar en el conjunto de figuras 28 a 33, las señales seno y cuadrada son fácilmente reconocibles, independientemente del desfase y de la variación de la amplitud. En la Figura 28 y la Figura 29, aunque los puntos no sean fácilmente visibles por estar acumulados unos encima de otros, la mayoría de ellos son verdes, color que corresponde a la función seno. En la Figura 30 y la Figura 31, todos los puntos están representados en rojo, color que corresponde a la señal cuadrada.

Sin embargo, la señal triangular parece más difícil de reconocer. En el caso de la modificación de su amplitud, el algoritmo no sabe distinguir si se trata de una señal triangular o cuadrada, ya que la mitad de los puntos son azules y la otra mitad son rojos, ver Figura 33.

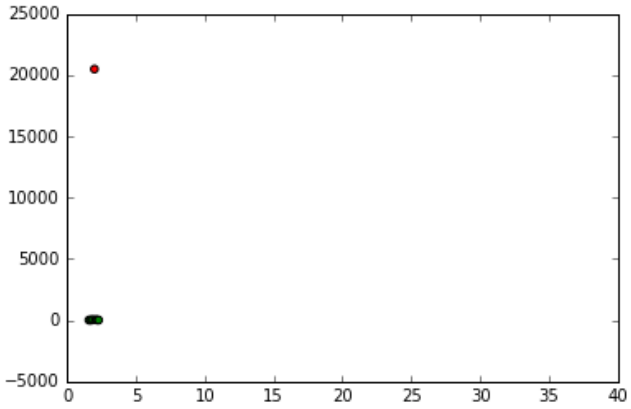


Figura 28. Test *con seno desfasado.*

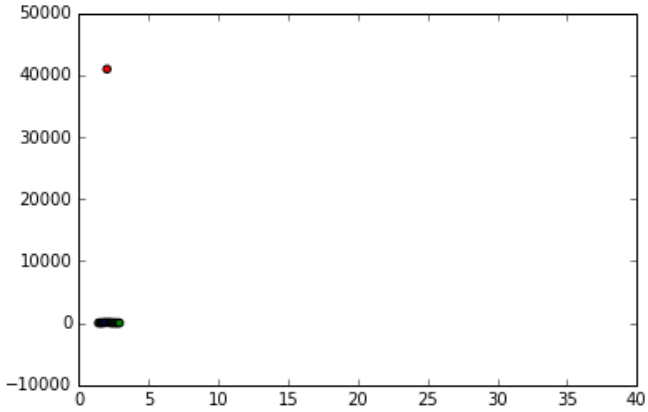


Figura 29. Test *con seno de amplitud doble.*

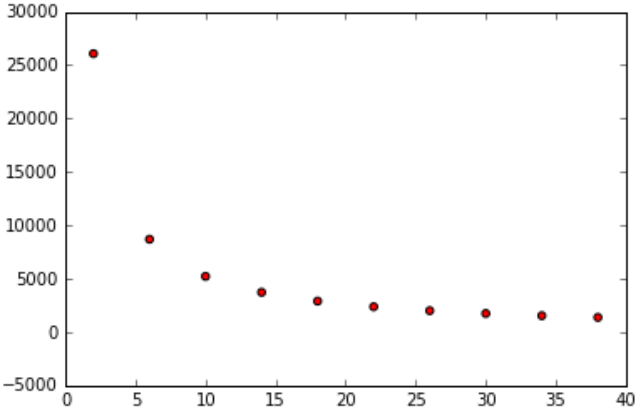


Figura 30. Test *con señal cuadrada desfasada.*

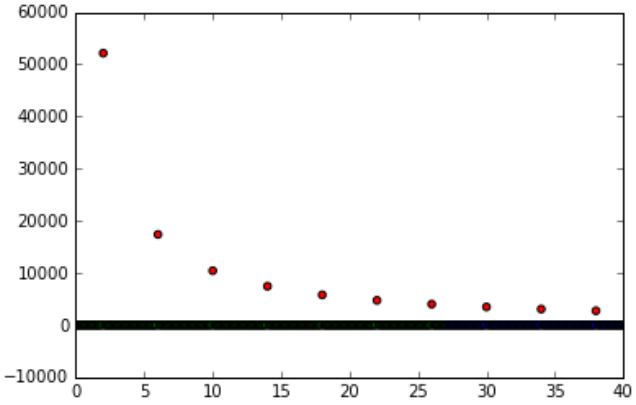


Figura 31. Test *con señal cuadrada de amplitud doble.*

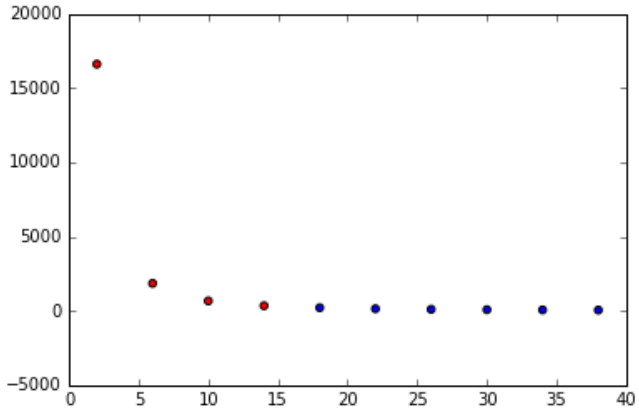


Figura 32. Test *con señal triangular.*

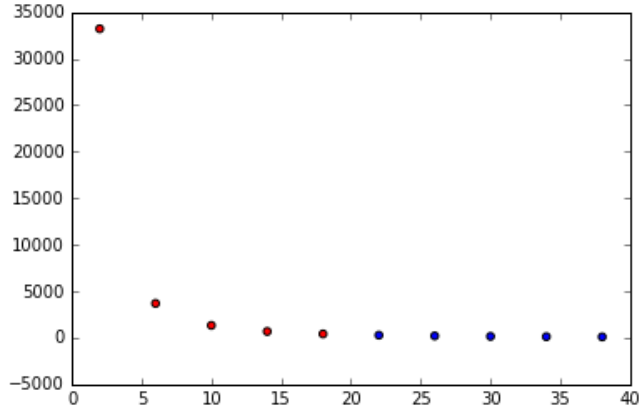


Figura 33. Test *con señal triangular de amplitud doble.*

Además de esta dificultad encontrada con las señales triangulares, hay que saber que, si se modifican las frecuencias de las señales, nuestro algoritmo no sería capaz de reconocerlas con el conjunto de ejemplos de entrenamiento que se le ha proporcionado hasta ahora. Habría por tanto que añadir tantas funciones de entrenamiento como frecuencias se consideren, aumentando notablemente el número de cálculos. Además, la superposición de informaciones complicaría o incluso impediría el reconocimiento de señales.

4.3 APLICACIÓN Nº2: RECONOCIMIENTO DE ESTADO DE APARATOS

Para el segundo planteamiento que se propone en este proyecto, se utiliza el concepto de la desagregación del consumo de energía, método que se explica en el apartado 3.2. El reconocimiento de cargas domésticas es, en realidad, el reconocimiento del estado de la carga. Se pretende utilizar el algoritmo *k*-NN para predecir si un aparato está encendido o no, a partir del consumo total de una vivienda.

4.3.1 CREACIÓN DE UNA BASE DE DATOS

Para juzgar la pertinencia de este segundo enfoque se construye una base de datos ficticios de consumos eléctricos de una vivienda. Los valores de consumo utilizados no son representativos.

La base de datos elaborada contiene un registro eléctrico de una vivienda: se supone que se mide cada hora, durante tres días consecutivos, la cantidad de potencia demandada por la totalidad de las cargas conectadas. Se escogen además dos dispositivos consumidores distintos: el sistema de iluminación y el horno. Se construye la base de datos que tiene la estructura mostrada en la Tabla 3.

Hora del día	*Consumo total del hogar*	*Estado ON/OFF de la iluminación*	*Estado ON/OFF del horno*

Tabla 3. Estructura de la base de datos elaborada.

La estructura de los datos que se considera es la siguiente: la hora del registro y el consumo total del hogar son los dos atributos de cada instancia, mientras que los estados de los aparatos determinan su clase. Esta estructura es la que se utiliza para el resto de aplicaciones que se realizan más adelante en este planteamiento del problema.

Dado que en este caso se dispone de dos aparatos, habrá $2^2 = 4$ clases posibles y por lo tanto en la representación de los datos, que se encuentra en la Figura 34, se utilizan cuatro colores distintos:

- *Rojo*, si la iluminación y el horno están apagados. La mayoría de estos puntos se dan durante las horas de noche.
- *Verde*, si la iluminación está encendida pero el horno está apagado, caso que se da durante la mayor parte de horas del día.
- *Azul*, si la iluminación y el horno están encendidos, situación que se produce principalmente a la hora de cenar.
- El caso en el que el horno está encendido pero la iluminación está apagada no es lógico, por lo que no viene representado.

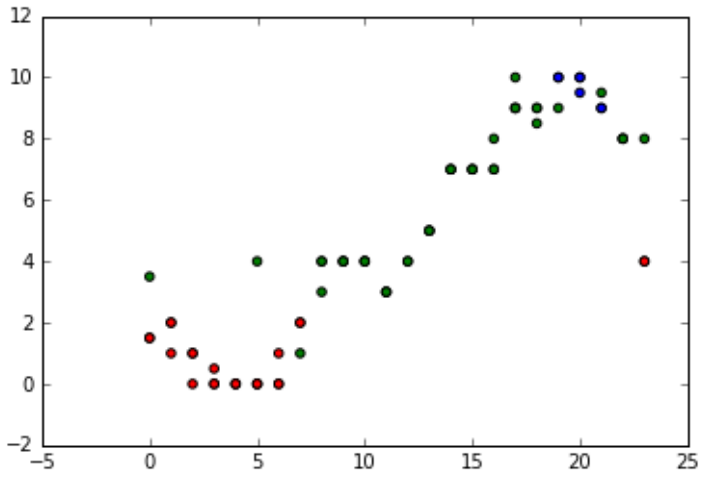


Figura 34. Training set *elaborado. Vivienda con los aparatos conectados.*

Para el *test* se utiliza el punto de coordenadas `(5,5, 1,5)`, es decir un registro de 1,5 unidades de potencia consumida a las 5h30. Se recuerda que el valor de las potencias no es

representativo. La clasificación realizada por el algoritmo, mostrada en la Figura 35, indica que, para esos valores, ambos aparatos deben estar apagados, puesto que el punto viene representado en rojo.

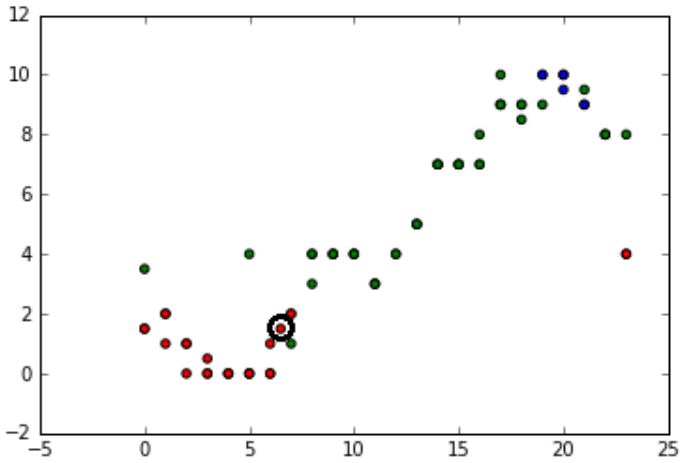


Figura 35. Clasificación del punto `(5,5 1,5)`.

4.3.2 UTILIZACIÓN DE UNA BASE DE DATOS REAL

Este algoritmo sólo es de utilidad si sirve en situaciones reales, por lo que a continuación se escoge un registro eléctrico auténtico, la base de datos UK-DALE. Su estructura ya se ha descrito en el apartado 3.3.2, por lo que se procede directamente a explicar el conjunto de ensayos que se han realizado.

El primer paso es la elección de la vivienda que se va a analizar. La preferencia evidente es aquella que tenga el menor número de datos a tratar, es decir, menos aparatos conectados a la red. Esta elección es importante porque este número varía entre 4 aparatos para la vivienda número 3, y 52 aparatos para la número 5. Se escoge, por lo tanto, trabajar con los registros de la vivienda número 3.

4.3.2.1 Incoherencias en las potencias medidas

Sin embargo, los ficheros de la vivienda número 3 presentan una incoherencia en las unidades: la potencia agregada, que se encuentra en el fichero `channel_1`, no es igual a la

suma de potencias desagregadas del resto de ficheros `channel_x`. Esta incoherencia viene ilustrada en la Figura 36, y se debe al hecho de que no todas las potencias registradas son potencias aparentes. El tipo de potencia medida depende del contador utilizado.

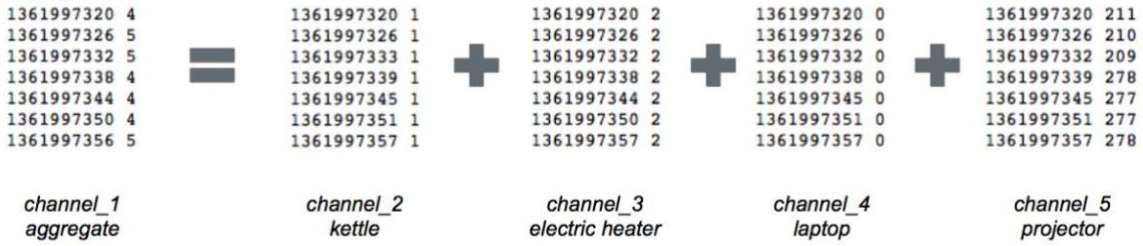


Figura 36. Potencias registradas incoherentes.

La carpeta de la vivienda número 1 incluye un fichero de texto, llamado `README.txt`, mostrado en la Figura 37, que detalla el tipo de potencia que se mide en cada uno de los aparatos. De esta manera, si se escoge trabajar con los registros de la vivienda número 1, se pueden escoger los cuatro dispositivos cuya potencia registrada es la aparente, al igual que la potencia agregada. Estos cuatro aparatos son los siguientes: "`boiler`", "`solar_thermal`", "`kitchen_lights`" y "`lighting_circuit`".

```
SENSORS USED
============

For details of these sensors, see:
https://github.com/JackKelly/rfm_ecomanager_logger/wiki/Data-format#wiki-Two_types_of_sensor


Current Transformer (CT) sensors (recording apparent power in units of VA)
--------------------------------------------------------------------------

"aggregate", "boiler", "solar_thermal", "kitchen_lights" and "lighting_circuit"


Individual Appliance Monitors (recording real power in units of watts)
----------------------------------------------------------------------

All other channels.
```

Figura 37. Extracto del fichero "README.txt" de la vivienda número 1. Las potencias registradas no son todas activas.

4.3.2.2 No disponibilidad de los estados ON/OFF

Sin embargo, al ir a utilizar dichos aparatos, se presenta otro problema. Ninguno de ellos tiene un fichero `channel_x_press_button` asociado. Se recuerda que este documento contiene los cambios de estado ON/OFF del dispositivo `x`.

El algoritmo desarrollado en este proyecto necesita estos estados para poder clasificar dichos aparatos. Esto se debe a que las diferentes clases en las que se clasifican las nuevas instancias dependen del estado de todos los dispositivos considerados. Por ejemplo, en el apartado 4.3.1 hay cuatro clases diferentes, dependiendo del estado de los dos dispositivos considerados, en este caso, la iluminación y el horno.

Esta falta de información dificulta la adaptación de los datos al planteamiento utilizado hasta ahora.

Dado que el único registro del que se conocen los tipos de potencia medida es el de la vivienda número 1, se procede a trabajar con él.

4.3.2.3 Referencias diferentes y falta de datos

Para solucionar el problema de las unidades, se decide implementar en el algoritmo una función que permita calcular la potencia total a mano. Es decir, que, a partir de las medidas de los aparatos individuales, se calcula la suma de todas ellas para así obtener una "potencia agregada", aunque ésta no sea la global de la casa.

Es importante escoger aparatos de los que se conozcan los cambios de estado, como ya se ha explicado anteriormente. Por ello, se eligen los `channels` 14 y 15 de la vivienda número 1 (`lcd_office` y `hifi_office`). El *training set* correspondiente se muestra en la Figura 38.

Sin embargo, otras dos dificultades aparecen en esta porción de datos. En primer lugar, las medidas se efectúan cada seis segundos, pero uno de los registros está adelantado de un segundo con respecto al otro. Esto no se aprecia en la Figura 38 por la gran cantidad de puntos que están representados, pero dificulta mucho la elaboración de un *training set*

interesante. Esto se debe a que no existen puntos a los que se le pueda asociar el estado de ambos aparatos, puesto que la referencia temporal no es la misma.

Se podría desplazar todo un registro de un segundo para superponerlo con el otro y obtener así un conjunto de ejemplos de entrenamiento utilizable. Pero es que, además, falta una gran porción de datos, como bien se muestra en la Figura 38.

Más allá de estos inconvenientes, algunos dispositivos se han incorporado a la vivienda más tarde, por lo que las referencias temporales no siempre coinciden.

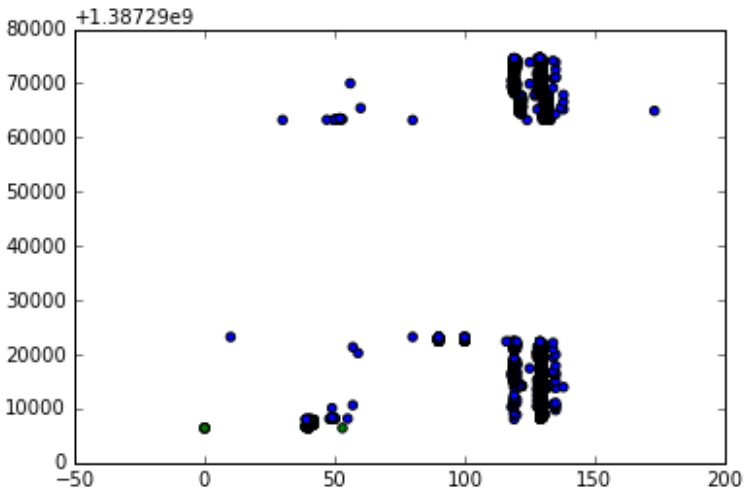


Figura 38. Ejemplos de entrenamiento `lcd_office` *y* `hifi_office`.

Dado que no se dispone de programas suficientemente potentes como para tratar la enorme cantidad de datos disponibles en UK-DALE, además de no gozar de más tiempo para encontrar una solución a todos los problemas expuestos en esta parte, se procede

directamente a exponer las soluciones que se estiman apropiadas para mejorar el rendimiento del algoritmo para esta aplicación.

Capítulo 5. ANÁLISIS DE RESULTADOS Y MEJORAS

El análisis de resultados obtenidos con este proyecto y la presentación de las posibles mejoras que se podrían llevar a cabo se exponen en este capítulo en dos partes, ya que cada planteamiento del problema presenta sus propios límites y, por tanto, sus propias posibilidades de mejora.

Se comienza por una propuesta para el primer planteamiento, a saber, el reconocimiento de señales puesto que, como ya se ha visto, el algoritmo k-NN no es capaz de reconocer formas. A continuación, y para hacer frente a todas las dificultades que se han encontrado en la segunda aplicación (ver apartado 4.3.2), se expone un tratamiento completamente diferente de la base de datos.

5.1 MEJORAS PARA EL RECONOCIMIENTO DE SEÑALES

El algoritmo k-NN se basa en distancias para tomar sus decisiones de clasificación, y no en formas. Por ello su aplicación al reconocimiento de señales, ya sean temporales o frecuenciales, no permite obtener resultados satisfactorios.

La forma del conjunto de datos de entrenamiento obtenido en el estudio frecuencial (ver apartado 4.2.4) invita a utilizar métodos de regresión. Éstos permitirían distinguir la tendencia de la transformada de Fourier de cada función considerada, mediante la caracterización de su envolvente.

5.1.1 LAS METACARACTERÍSTICAS

Además, el algoritmo que se ha desarrollado es muy rudimentario y no proporciona exactitudes suficientemente elevadas en sus predicciones. Esta alta tasa de error se debe, entre otras, a que las informaciones que ofrecen los datos tratados en bruto son muy limitadas. Se propone como mejora la extracción de informaciones suplementarias, estudiando ciertas estructuras o características de los datos.

Para enriquecer las informaciones que sirven como entrada para el algoritmo, se pueden definir determinadas metacaracterísticas, como explica Kaustav Basu en su tesis "*Techniques avancées de classification pour l'identification et la prédiction non intrusive de l'état des charges dans le bâtiment*", en inglés "*Classification techniques for non-intrusive load monitoring and prediction of residential loads*". Estas metacaracterísticas se encargan de especificar los datos de una manera diferente.

Para la determinación de dichas metacaracterísticas se debe:

- definir una secuencia temporal $t = t_1, t_2, \dots, t_n$a la que se asocian las potencias consumidas, y
- particionar el conjunto de datos inicial en $n - w + 1$ subsecuencias $C_k = t_k, \dots, t_{k+w-1}$ de longitud w para crear la *sliding window*. A cada subsecuencia se le asocia el estado del aparato, ON/OFF, en el que se encuentre durante dicha subsecuencia.

La construcción de la *sliding window* se ilustra en la Figura 39. Las metacaracterísticas se definen para cada subsecuencia C_k.

Por ejemplo, si se toma C_k centrada en t_0, un ejemplo de metacaracterística puede ser la distancia -temporal- de t_0 al máximo y mínimo local de potencia consumida dentro de la subsecuncia. La variación de consumos entre subsecuencias, el gradiente, y otros conceptos estadísticos como la varianza y la variación típica son otras metacaracterísticas propuestas por Basu.

La introducción de metacaracterísticas en los datos de entrada simplifica la distinción de clases, puesto que hay más atributos por instancia. Esto conlleva una mayor precisión en la clasificación obtenida a partir del algoritmo k-NN.

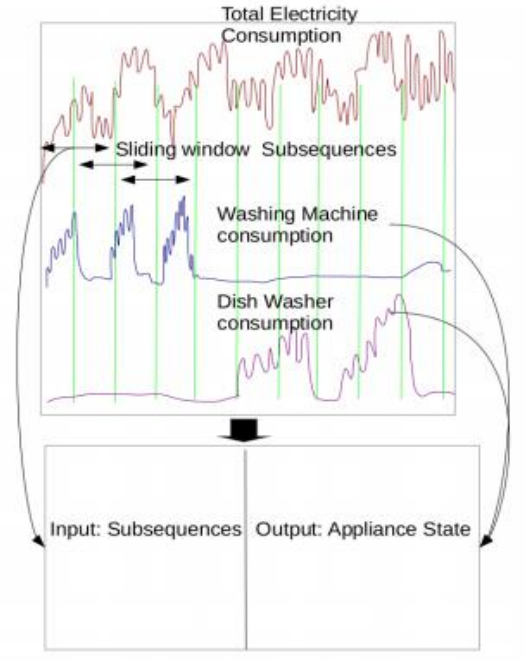


Figura 39. Construcción de la sliding window.

5.2 MEJORAS PARA EL RECONOCIMIENTO DE ESTADOS

En este proyecto, la principal dificultad encontrada para el reconocimiento de estados ha sido la falta de tiempo y de material adecuado para realizar un tratamiento correcto de una base de datos tan extensa y compleja como lo es UK-DALE.

Además, el algoritmo *k*-NN no parece ser el adecuado para la resolución del problema debido a varios factores. La altísima frecuencia de recogida de datos provoca la superposición de los puntos y, por lo tanto, no se obtienen buenos resultados. Asimismo, el tiempo computacional que supone el cálculo de todas y cada una de las distancias entre los diferentes puntos, ralentiza considerablemente el sistema.

5.2.1 DESCOMPOSICIÓN DEL CONSUMO COMO FUNCIÓN CASI PERIÓDICA

El consumo energético de un hogar en función del tiempo se puede considerar como una señal casi periódica. Ya sea tomando una escala de tiempo de un día o de un año, se puede suponer que el consumo energético tiene un patrón bastante claro.

Es por ello por lo que se propone utilizar esta casi-periodicidad para descomponer la señal en una suma de señales sinusoidales. El conjunto de datos de entrenamiento necesario es, por lo tanto, una tabla que contenga las amplitudes de la señal a una frecuencia de muestreo predefinida. En este caso, ya no se utiliza el algoritmo k-NN, sino un algoritmo que permita reconocer señales sinusoidales según los coeficientes de Fourier que las definan, para así calcular la suma de dichas señales. Esta suma es la señal original que se debe reconocer.

Esta es una descripción breve de la fase de entrenamiento y por una extensión del intervalo de tiempo, se puede llevar a cabo la fase de prueba, en la que se predecirían los consumos futuros de una vivienda.

Incluso si este método parece interesante, hay que reconocer que se aleja bastante del objetivo principal de este proyecto, puesto que, si bien permitiría estimar los futuros consumos energéticos, no permite reconocer el estado de las cargas domésticas conectadas a la red.

Capítulo 6. BIBLIOGRAFÍA

Asamblea General de las Naciones Unidas. (1986). Declaración sobre el derecho al desarrollo.

Basu, K. (s.f.). *Techniques avancées de classification pour l'identification et la prédiction non intrusive de l'état des charges dans le bâtiment.*

Comisión de Medio Ambiente y Desarrollo de Naciones Unidas. (1987). Informe de Brundtland.

Comisión de Medio Ambiente y Desarrollo de Naciones Unidas. (1992). Declaración de Río.

Comisión Europea. (2016). Energía Limpia para todos.

Energía y sociedad. (s.f.). *Manual de la Energía*. Obtenido de Energía y sociedad: http://www.energiaysociedad.es/manenergia/1-1-energia-y-sociedad/

Fisher, R. (1936). *The use of multiple measurements in taxonomic problems.*

Instituto para la Diversificación y el Ahorro de la Energía. (s.f.). Obtenido de www.idae.es

Kelly, J., & Knottenbelt, W. (2015). *The UK-DALE dataset, domestic appliance-level electricity demand and whole house demand from five UK homes.*

López Michelone, M. (2015). Obtenido de https://www.unocero.com/ciencia/inteligencia-artificial-y-uno-de-sus-pioneros-a-l-samuel/

Mitchell, T. M. (1997). *Machine Learning.*

NA8. (2017). Obtenido de Aprende Machine Learning: http://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

Real Academia Española. (s.f.).

Red Eléctrica de España. (2014). *Informe del Sistema Eléctrico Español.*

Samuel, A. L. (1959). *Some Studies of Machine Learning Using the Game of Checkers.*

Sancho Caparrini, F. (2017). Obtenido de http://www.cs.us.es/~fsancho/?e=75

scikit-learn. (s.f.). Obtenido de

http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

ANEXO A: CÓDIGO K-NN PARA FLOR DE IRIS

```
# -*- coding: utf-8 -*-
"""
Created on Wed Dec 20 14:36:53 2017

@author: ubedaromero_cla
"""

#https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
# loading libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#from sklearn import preprocessing
#from sklearn import model_selection
#from sklearn.cross_validation import train_test_split
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import neighbors
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_score

#def stratifiedMFolds(y, num_folds):
# kf = model_selection.StratifiedKFold(n_splits=num_folds)
# folds_regr = [(tr, te) for (tr, te) in kf.split(np.zeros(y.size), y)]
# return folds_regr

# define column names
names = ['sepal_length','sepal_width','petal_length', 'petal_width', 'class']

# loading training data
df = pd.read_csv('Desktop\projet_long\projetlong\projetlong\data\iris.csv',
header=None, names=names)
df.head()

X_train = df.drop(['class'], axis=1).values
y_train = pd.get_dummies(df['class']).values[:,1]

#print(X_train.shape)
plt.scatter(X_train[:,0], X_train[:,1], c=y_train)
plt.show()
#plt.scatter(X_train[:,2], X_train[:,3], c=y_train)

def predict(X_train, y_train, x_test, k):
# create list for distances and targets
```

```
distances = []
targets = []
for i in range(len(X_train)):
# first we compute the euclidean distance
distance = np.sqrt(np.sum(np.square(x_test - X_train[i, :])))
# add it to list of distances
distances.append([distance, i])
# sort the list
distances = sorted(distances)
# make a list of the k neighbors' targets
for i in range(k):
index = distances[i][1]
targets.append(y_train[index])
# return most common target
return Counter(targets).most_common(1)[0][0]
def kNearestNeighbor(X_train, y_train, X_test, k):
predictions = []
# transforming them into lists
X1 = X_train.tolist()
y1 = y_train.tolist()
for i in range(len(X_test)):
a = predict(X_train, y_train, X_test[i, :], k)
predictions.append(a)
X1+=[X_test[i].tolist()]
y1+=[a]
y = np.array(y1)
X = np.array(X1)
plt.scatter(X[:,0], X[:,1], c=y)
plt.show()
plt.scatter(X[:,2], X[:,3], c=y)

X_test = np.array([[ 4.8 , 2.5, 3.5, 1]])
precisions = kNearestNeighbor(X_train, y_train, X_test, 5)

# creating odd list of K for KNN
myList = list(range(1,50)

# subsetting just the odd ones
#neighbors = filter(lambda x: x % 2 != 0, myList)
neighbors = [x for x in myList if x%2 != 0]

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for k in neighbors:
knn = KNeighborsClassifier(n_neighbors=k)
scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
cv_scores.append(scores.mean())
# changing to misclassification error
MSE = [1 - x for x in cv_scores]
print(MSE)
```

```
# determining best k

optimal_k = neighbors[MSE.index(min(MSE))]
print(optimal_k)

# plot misclassification error vs k
plt.plot(neighbors, MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
```

ANEXO B: CÓDIGO PARA RECONOCIMIENTO DE FUNCIÓN SENO

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 11 16:49:18 2018

@author: ubedaromeroa_cla
"""

#https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/
# loading libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#from sklearn import preprocessing
#from sklearn import model_selection
#from sklearn.cross_validation import train_test_split
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import neighbors
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_score
from math import pi
from math import sin

n=20
X_train1 = []
y_train1 = []
y_color1 = []
for k in range (0,n+1):
    x = 2*pi*(k)/n
    y = sin(2*pi*(k)/n)
    y_train1 += [0]
    y_color1 += ["b"]
    X_train1 += [[x,y]]

#plt.scatter(X_train[:,0], X_train[:,1])
#plt.show()
```

```
X_train2 = []
y_train2 = []
y_color2 = []
for k in range(0,n+1):
    x = 2*pi*k/n
    y = sin(2*(2*pi*k/n))
    y_color2 += ["r"]
    y_train2 += [1]
    X_train2 += [[x,y]]

X_train = X_train1 + X_train2
y_train = y_train1 + y_train2
y_color = y_color1 + y_color2

X_train = np.array(X_train)
y_train = np.array(y_train)
y_color = np.array(y_color)

plt.scatter(X_train[:,0], X_train[:,1], c = y_train)
plt.show()

#for i in range(0,len(X_train)):
#    plt.plot(X_train[i,0],X_train[i,1], "o", color = y_color[i])

def predict(X_train, y_train, x_test, k):
        # create list for distances and targets
    distances = []
    targets = []

    for i in range(len(X_train)):
            # first we compute the euclidean distance
        distance = np.sqrt(np.sum(np.square(x_test - X_train[i, :])))
            # add it to list of distances
        distances.append([distance, i])

      # sort the list
    distances = sorted(distances)

      # make a list of the k neighbors' targets
    for i in range(k):
        index = distances[i][1]
        targets.append(y_train[index])

      # return most common target
    return Counter(targets).most_common(1)[0][0]


def kNearestNeighborsinus(X_train, y_train, X_test, k):
    predictions = []
    # transforming them into lists
    X1 = X_train.tolist()
    y1 = y_train.tolist()
```

```
#     X1 = []
#     y1 = []
    for i in range(len(X_test)):
        a = predict(X_train, y_train, X_test[i, :], k)
        predictions.append(a)
        X1+=[X_test[i].tolist()]
        y1+=[a]
#     count = 0
#     for j in y1:
#         count+=j
#     count = count/(n+1)
#     if count > 0.5:
#         y1 = y_train.tolist()+[1]*(n+1)
#     else:
#         y1 = y_train.tolist()+[0]*(n+1)
#     print(count)
#     print(y1)
    y = np.array(y1)
    X = np.array(X1)
    print(len(X1))
    plt.scatter(X[:,0], X[:,1], c=y)
    plt.show()
    return(predictions)


#X_test = np.array(X_train1)

n=20
X_test1 = []
for k in range (0,n+1):
    x = 2*pi*(k)/n
    y = sin((2*pi*(k)/n))
    X_test1 += [[x,y]]

X_test1 = np.array(X_test1)
pred = kNearestNeighborsinus(X_train, y_train, X_test1,3)
print(accuracy_score(y_train1, pred))
```

ANEXO C: CÓDIGO PARA RECONOCIMIENTO DE FUNCIONES SENO, CUADRADA Y TRIANGULAR

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 29 10:05:53 2018

@author: chauvin_mez
"""
from numpy import pi, sin, linspace, log10
from scipy.fftpack import fft, fftfreq
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
from scipy.fftpack import fft, fftshift
import pandas as pd
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn import neighbors
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_score
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection

from sklearn.linear_model import LinearRegression
from sklearn.isotonic import IsotonicRegression
from sklearn.utils import check_random_state

def retirerTendanceLineaire(data):
    x = []
    l = len(data)
    #On considere que les donnees sont separees d'un pas
    #de 1 dans le temps
    for i in range(l):
        x += [i]
    #On effectue une regression lineaire
    sum_x,sum_y=0,0
    sum_xx,sum_xy=0,0
    for k in range(l):
        sum_x += x[k]
        sum_y += data[k]
```

```
        sum_xx += x[k]**2
        sum_xy += x[k]*data[k]
    #On calcule le coefficient directeur de la droite
    slope = (l*sum_xy - sum_x*sum_y)/(l*sum_xx - sum_x*sum_x)
    #On soustrait la tendance (coef directeur ) aux donnees
    donnees_sans_tendance = []
    for j in range(l):
        donnees_sans_tendance += [data[j] - slope*j]
    #On retourne un objet contenant les donnees sans la
    #tendance Ainsi que le coefficient de la tendance dont
    #nous aurons besoin a la fin
    donnees = donnees_sans_tendance
    coef = slope
    return(donnees,slope)

#Nous utilisons le jeu de donnees ci-dessous :
#data = [669, 592, 664, 1005, 699, 401, 646, 472, 598, 681, 1126, 1260, 562, 491,
714, 530, 521, 687, 776, 802, 499, 536, 871, 801, 965, 768, 381, 497, 458, 699,
549, 427, 358, 219, 635, 756, 775, 969, 598, 630, 649, 722, 835, 812, 724, 966,
778, 584, 697, 737, 777, 1059, 1218, 848, 713, 884, 879, 1056, 1273, 1848, 780,
1206, 1404, 1444, 1412, 1493, 1576, 1178, 836, 1087, 1101, 1082, 775, 698, 620,
651, 731, 906, 958, 1039, 1105, 620, 576, 707, 888, 1052, 1072, 1357, 768, 986,
816, 889, 973, 983, 1351, 1266, 1053, 1879, 2085, 2419, 1880, 2045, 2212, 1491,
1378, 1524, 1231, 1577, 2459, 1848, 1506, 1589, 1386, 1111, 1180, 1075, 1595,
1309, 2092, 1846, 2321, 2036, 3587, 1637, 1416, 1432, 1110, 1135, 1233, 1439,
894, 628, 967, 1176, 1069, 1193, 1771, 1199, 888, 1155, 1254, 1403, 1502, 1692,
1187, 1110, 1382, 1808, 2039, 1810, 1819, 1408, 803, 1568, 1227, 1270, 1268,
1535, 873, 1006, 1328, 1733, 1352, 1906, 2029, 1734, 1314, 1810, 1540, 1958,
1420, 1530, 1126, 721, 771, 874, 997, 1186, 1415, 973, 1146, 1147, 1079, 3854,
3407, 2257, 1200, 734, 1051, 1030, 1370, 2422, 1531, 1062, 530, 1030, 1061, 1249,
2080, 2251, 1190, 756, 1161, 1053, 1063, 932, 1604, 1130, 744, 930, 948, 1107,
1161, 1194, 1366, 1155, 785, 602, 903, 1142, 1410, 1256, 742, 985, 1037, 1067,
1196, 1412, 1127, 779, 911, 989, 946, 888, 1349, 1124, 761, 994, 1068, 971, 1157,
1558, 1223, 782, 2790, 1835, 1444, 1098, 1399, 1255, 950, 1110, 1345, 1224, 1092,
1446, 1210, 1122, 1259, 1181, 1035, 1325, 1481, 1278, 769, 911, 876, 877, 950,
1383, 980, 705, 888, 877, 638, 1065, 1142, 1090, 1316, 1270, 1048, 1256, 1009,
1175, 1176, 870, 856, 860]
data = [0,np.sqrt(3)/2,np.sin(2*pi/3),np.sin(pi),np.sin(4*pi/3),np.sin(5*pi/3),0]
noTendanceObj = retirerTendanceLineaire(data)
donnees_sans_tendance = noTendanceObj[0]
coef_tendance = noTendanceObj[1]

def transformeeFourier(data):
    tab = fft(data)
    return(tab)

domaine_freq = fft(donnees_sans_tendance)

def fftfreq1(n):
    val = 1/n
    results = []
    N = int((n-1)/2)+1
    p1 = []
```

```
    for i in range(N):
        results += [i*val]
    for i in range(N,n):
        results += [(-int(n/2) - (N-i))*val]
    return(results)

def trier_indice_frequences(freqs):
    tmpTab = []
    for i in range(len(freqs)):
        tmpTab += [[freqs[i],i]]
    tmpTab = np.array(tmpTab)
    indices_tries = np.lexsort((tmpTab[:,1],tmpTab[:,0],))
    return(indices_tries)

f = fftfreq1(len(data));
#on trie les indices des fréquences selon la valeur absolue des éléments de f
indices_tries = trier_indice_frequences(f);

def predire(horizon,nb_harmonique,domaine_freq,f,indices_tries,coef_tendance):
    length = len(domaine_freq)
    indice = 0
    amplitude = 0
    signal_restaure = [0]*(length + horizon)
    for i in range((1+nb_harmonique*2)):
        indice = indices_tries[i]
        domaine_freq_reel = np.real(domaine_freq)
        domaine_freq_imag = np.imag(domaine_freq)
        amplitude =
np.sqrt(domaine_freq_reel[indice]**2+domaine_freq_imag[indice]**2)/len(domaine_fr
eq)
        phase = np.arctan(domaine_freq_imag[indice]/domaine_freq_reel[indice])
        print(phase)
        print(amplitude)
        facteur = 2*pi*f[indice]
        for j in range(length+horizon):
            signal_restaure[j] += amplitude*np.cos(facteur*j+phase)

    for i in range(len(signal_restaure)):
        signal_restaure[i] += coef_tendance*i
    return(signal_restaure)

horizon = 2;
nb_harmonique = 2;
prediction = predire(horizon, nb_harmonique, domaine_freq, f, indices_tries,
coef_tendance);
```