Universidad Pablo de Olavide

Doctorado en Ingeniería Informática

TÉCNICAS DE PREDICCIÓN ESCALABLES PARA BIG DATA TEMPORALES

D. Antonio Galicia de Castro

Tesis doctoral dirigida por

Drª. Alicia Troncoso Lora y Dr. Francisco Martínez Álvarez

TÉCNICAS DE PREDICCIÓN ESCALABLES PARA BIG DATA TEMPORALES

Autor
D. Antonio Galicia de Castro

Directores
Dr^{a}. Alicia Troncoso Lora

Dr. Francisco Martínez Álvarez

Centro De Estudios De Postgrado
Universidad Pablo de Olavide

Sevilla, enero de 2019

Esta Tesis Doctoral ha sido parcialmente financiada por el Ministerio de Economía, Industria y Competitividad mediante el proyecto “Big Time-Aware Data: Análisis de datos masivos indexados en el tiempo clasificación y predicción” (TIN2014-55894-C2-2-R) y por el grupo de investigación TIC-254: Data Science & Big Data de la Universidad Pablo de Olavide.

"Big Data, Big Knowledge"
– Data Science & Big Data Research Lab.

Agradecimientos

En especial a Alicia Troncoso y Francisco Martínez, mis directores de Tesis, quienes han vuelto a acompañarme en otra etapa de mi vida. Ha sido un placer compartir tantos momentos con ellos, tantas risas y tantos debates en innumerables temáticas, que ciertamente me han hecho crecer como profesional y como persona. Sigue sorprendiéndome la vocación que tienen, la dedicación y el esfuerzo constante que realizan, y siempre acompañados por una gran sonrisa. Ellos son grandes profesionales, grandes docentes y grandes inspiradores.

Otras personas que me han acompañado en estos últimos años son David, Ricardo, Rubén, Antonio y Pepe. Gracias por hacer más llevaderas tantas horas de trabajo y esfuerzo, por ser como sois, por vuestra amistad sincera y honesta.

También me gustaría agradecer a mi familia, mis tíos, mi hermana y en especial mis padres, quienes han puesto tanto empeño en mi formación y me han labrado como persona. Ellos son la parte más importante de mi vida.

A mi familia.

Resumen

En esta Tesis se presenta una metodología para pronosticar series temporales de gran longitud basada en el framework de computación distribuida Apache Spark y su librería MLlib para Machine Learning.

La predicción de los h valores futuros se realiza dividiendo el problema de predicción en h subproblemas de predicción, uno para cada valor del horizonte. Esto nos permite resolver en paralelo todos los subproblemas, asegurando la escalabilidad de la metodología.

Además, se propone un ensemble que nos permite predecir h valores futuros, mediante la combinación de los resultados de k modelos generados en base a distintos algoritmos. De forma concreta, se han utilizado las implementaciones de los algoritmos Decision Tree, Gradient-Boosted Trees y Random Forest que ofrece la librería MLlib de Spark. Se consideran dos estrategias, un modelo de ensemble estático y un modelo dinámico que actualiza los pesos para mejorar el modelo de predicción. Los pesos del ensemble se calculan con el método de mínimos cuadrados ponderados, y las predicciones para cada modelo que forma el ensemble se obtienen de forma distribuida.

El comportamiento de los modelos se evalúa con dos casos de uso: el consumo eléctrico en España, en el que se genera un modelo para predecir las siguientes 4 horas de la serie temporal, partiendo de un histórico de 10 años de registros con una frecuencia de 10 minutos; y datos de producción de energía solar fotovoltaica de Australia, recogidos por la Universidad de Queensland durante dos años, con una frecuencia de 30 minutos entre las mediciones.

Los resultados han mostrado que tanto los ensemble dinámicos como los estáticos se comportaron bien, mejorando los resultados de cualquiera de los algoritmos que componen el ensemble. El ensemble dinámico fue el modelo más preciso cometiendo un error relativo medio del 2 % en la predicción de la demanda de energía eléctrica de España, resultado muy prometedor para esta serie temporal.

Los resultados obtenidos para la predicción de producción de energía solar fotovoltaica se han comparado, además, con redes neurales artificiales, el algoritmo PSF el cual está basado en secuencia de patrones y con Deep Learning, obteniendo las mejores predicciones en esta serie temporal.

Índice general

Parte I

Trabajo de Tesis Doctoral

1

Introducción

1.1. Motivación de la investigación

Es conocido que en los últimos años la democratización de la tecnología ha significado que el acceso a internet y a las telecomunicaciones ya no son exclusivas de las personas más jóvenes. Los servicios están cada vez más digitalizados y diseñados para todos los sectores de la población, difuminando a su paso la brecha generacional. Esto quiere decir que para las organizaciones, la mayor parte de sus usuarios y clientes hacen uso de la tecnología, y la satisfacción de los usuarios depende de la mejora de la tecnología que sustenta los servicios y procesos de negocio. El crecimiento de una sociedad digital está vinculada a una integración vertical de la tecnología y los sistemas de información, generando cada vez más datos que deben ser gestionados en aras de obtener mejores servicios y, de paso, obtener conocimiento de los datos.

Por tanto, considerándonos en la era de la información, estamos también en la era de la Ciencia de Datos, campo interdisciplinario cuyo objetivo es obtener un mejor entendimiento de los datos. Y es que la forma de adquirir conocimiento ha cambiado a lo largo de la historia. En los últimos siglos la ciencia pasó de la experimentación y descripción de fenómenos naturales (primer paradigma) a ser teórica, basándose en modelos y generalizaciones. En las últimas décadas llegó la era computacional con la simulación de fenómenos complejos. Las capacidades actuales de la Ciencia de Datos –impulsada por la ley de Moore en términos de capacidad de cálculo, y la ley de Kryder en cuanto a capacidad de almacenamiento– nos ha abierto las puertas del cuarto paradigma [1], la ciencia intensiva en datos, en la cual podemos obtener conocimiento –o descubrimiento científico– utilizando todos los datos disponibles de un problema, también conocido como Big Data.

Por otro lado, un componente esencial en la naturaleza de los datos es que normalmente la información se encuentra indexada en el tiempo, lo que se conoce como series temporales, dando lugar ahora al término Big Data Time Series según la literatura. Estos tipos de datos tienen sentido si su análisis se realiza con respecto a su evolución en el tiempo. Por ejemplo, datos como la demanda eléctrica o la contaminación de carbono en el aire [2] pueden ser analizados con diversos objetivos: para predecir su evolución, para predecir valores anómalos, para obtener patrones que nos permitan comparar su evolución con otros datos, para establecer relaciones de unas variables con respecto a otras, etc.

En Big Data es necesario utilizar, de forma inherente, herramientas de computación distribuida. Los principales frameworks existentes para el procesamiento de datos masivos han sido desarrollados gracias a compañías tecnológicas punteras como son Google y Yahoo!. Google desarrolló la tecnología MapReduce [3] que, para el procesamiento, divide los datos de entrada en bloques y después integra la información de salida de cada bloque en una única solución, permitiendo que estas divisiones puedan distribuirse entre diferentes máquinas. Después,

Yahoo! desarrolló Hadoop [4], una implementación de código abierto basado en el paradigma MapReduce, actualmente integrado en la Fundación Apache. Las limitaciones de MapReduce a la hora de implementar algoritmos que necesitan iterar sobre los datos, ha requerido la creación de nuevas herramientas, como Spark [5], una herramienta de propósito general para procesamiento distribuido desarrollado por la Universidad de Berkeley en California.

El despliegue de Spark sobre el sistema de ficheros distribuido de Hadoop (Hadoop Distributed File System, HDFS), permite realizar el procesamiento de datos en memoria, consiguiendo con ello mucha mayor velocidad de procesamiento que con Hadoop. Spark proporciona operadores de alto nivel y soporta varios lenguajes (Java, Python, R) además de su lenguaje nativo llamado Scala. Además, ofrece diferentes módulos especializados, como la librería de aprendizaje automático MLlib [6].

En esta Tesis Doctoral, se propone un ensemble de algoritmos para predecir series temporales de gran longitud con un horizonte de predicción de más de un paso, siendo necesario utilizar un entorno de computación distribuida como Apache Spark.

1.2. Objetivos de la Tesis

Se plantea el problema de predecir una serie temporal de gran dimensión con un horizonte temporal determinado. Para resolver este problema en un contexto de Big Data se ha seleccionado el motor de computación distribuida Apache Spark, que tiene un módulo específico para Machine Learning. Sin embargo, actualmente la librería presenta algunas desventajas que se detallan a continuación.

Las técnicas de regresión disponibles en esta librería no soportan la regresión multi-output, es decir, la predicción de más de un paso. Por otro lado, los algoritmos de regresión no están diseñados para trabajar con conjuntos de datos donde el orden temporal sea un factor importante, ya que ninguna operación de alto nivel conserva el orden cronológico, aspecto crucial en una serie temporal. De este modo, uno de los objetivos de este trabajo es introducir una metodología para la predicción de series temporales, siendo el orden temporal una de las principales características de estos conjuntos de datos, y además que nos permita la predicción de un horizonte temporal formado por h valores. Para ello, se parte de la base de las implementaciones existentes en la librería de Machine Learning MLlib.

El trabajo de investigación ha sido desarrollado teniendo en cuenta los siguientes objetivos:

- Estudio teórico-práctico de los métodos de predicción de la librería MLlib.
- Formulación matemática del problema de predicción multipaso a resolver.
- Propuesta de una metodología basada en regresión de un único paso que permita resolver el problema.
- Validar la metodología analizando la precisión de las predicciones obtenidas en casos de estudio con datos reales.
- Estudio de la escalabilidad de la metodología propuesta.

1.3. Principales contribuciones

La formulación matemática del problema de predicción multipaso que se resuelve en este trabajo de investigación ha sido publicado por primera vez en [7]. En esta publicación se realiza una primera aproximación al problema de predicción multipaso con una propuesta de metodología basada en regresión de un único paso. Una descripción más detallada fue publicada en [8], en la que se valida la metodología analizando la precisión de las predicciones, para un caso de estudio basado en datos reales. Además, como parte de la validación, se realizó un estudio de la escalabilidad de la metodología propuesta. En ambas publicaciones se utilizaron diferentes algoritmos, como regresión lineal, un árbol de regresión simple y dos algoritmos que utilizan múltiples árboles, tales como Gradient-Boosted Trees y Random Forest. Estos algoritmos se compararon con el algoritmo de Deep Learning desarrollado en [9].

- [7] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «Scalable Forecasting Techniques Applied to Big Electricity Time Series». *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cadiz, Spain, June 14-16, 2017, Proceedings, Part II*. Springer International Publishing, 2017, pp. 165–175. DOI: 10.1007/978-3-319-59147-6_15. Conference Ranking: CORE-B

- [8] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «A novel Spark-based multi-step forecasting algorithm for big data time series». *Information Sciences* (2018). DOI: 10.1016/j.ins.2018.06.010. IF: 4,305 (12/148) Computer Science - Information Systems Q1

- [9] J.F. Torres, A. Galicia, A. Troncoso, and F. Martínez-Álvarez. «A scalable approach based on deep learning for big data time series forecasting». *Integrated Computer-Aided Engineering* 25 (2018), pp. 1–14. DOI: 10.3233/ICA-180580. IF: 3,667 (21/132) Computer Science - Artificial Intelligence Q1

El siguiente paso ha sido utilizar tres de los algoritmos antes mencionados para desarrollar un nuevo algoritmo ensemble para la predicción. El enfoque de ensemble asigna diferentes pesos a cada método utilizando un método de mínimos cuadrados ponderados, que optimiza la contribución de cada pronóstico individual en la predicción combinada para un horizonte temporal dado. Por lo tanto, nuestro ensemble no es un ensemble clásico de tipo boosting sino un ensemble de votación ponderada, ya que combinamos tres modelos base usando un sistema de voto ponderado donde los pesos se calculan resolviendo un problema de mínimos cuadrados cuya función objetivo es el error en un conjunto de entrenamiento. Este ensemble ha sido publicado en [10], donde la metodología se valida utilizando un ensemble estático y otro dinámico. La experimentación del ensemble se ha llevado a cabo con dos casos de uso, uno sobre demanda de energía eléctrica en España, y una segunda experimentación con datos de producción de energía solar fotovoltaica en Australia. Los resultados muestran el rendimiento de ambos ensemble, superando los modelos base que combinaban, y particularmente mostrando el potencial de los ensemble dinámicos para la predicción de Big Data Time Series.

- [10] A. Galicia, R. Talavera-Llames, A. Troncoso, I. Koprinska, and F. Martínez-Álvarez. «Multi-step forecasting for big data time series based on ensemble learning». *Knowledge-Based Systems* (2018). DOI: 10.1016/j.knosys.2018.10.009 IF: 4,396 (14/132) Computer Science - Artificial Intelligence Q1

1.4. Estructura de la Memoria

En esta sección se describe la organización de esta memoria, que se compone de tres bloques temáticos:

- Parte I. Trabajo de Tesis Doctoral
- Parte II. Big Data Time Series
- Parte III. Publicaciones

La primera parte contempla diversos apartados generales como la introducción, donde se ha justificado la unidad temática y la motivación de la Tesis. También se han concretado los objetivos que se pretenden alcanzar y el marco en el que se encuadra el conjunto de publicaciones donde se abordan cada uno de los objetivos.

A continuación, en la segunda parte del documento, se analiza en primer lugar el marco teórico de esta Tesis Doctoral, enmarcada en la Ciencia de Datos. Aquí se describen los métodos de predicción para series temporales más utilizados en la literatura, diferenciando la predicción clásica de series temporales de corta y mediana longitud, y la predicción series temporales de gran longitud. También se consideran los principales entornos de desarrollo que existen para soluciones Big Data.

Para continuar, se aborda el problema de predicción objeto de la investigación llevada a cabo. Para ello, se establece la formulación matemática del problema que se pretende resolver y se describe el método propuesto para su resolución. Se detalla además, cómo se representan los datos para poder predecir una serie temporal usando algoritmos de regresión.

Más adelante, se resumen los resultados obtenidos de la metodología propuesta para la predicción de Big Data Time Series con horizontes multipaso, utilizando para ello dos casos de uso: demanda de energía eléctrica y producción de energía solar fotovoltaica.

Para terminar este bloque, se recapitulan las principales reflexiones sobre el trabajo desarrollado en esta Tesis, incluyendo las futuras líneas de investigación que se llevarán a cabo.

Finalmente se detallan las publicaciones, las cuales se encuentran anexadas en el formato original de su publicación.

En la última parte del documento de Tesis, se pueden encontrar las referencias bibliográficas.

Parte II

Big Data Time Series

2

Contexto de la investigación

La Ciencia de Datos es un campo multidisciplinar que abarca, principalmente, el dominio de las matemáticas, la estadística y las ciencias de la computación. Además de la propia experiencia en el sector o negocio, es el componente principal dentro del proceso de descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases, KDD) [11]. Este proceso incluye operaciones de selección, preprocesamiento, transformación de los datos, aprendizaje y evaluación de los resultados, conceptos englobados dentro de la minería de datos. El proceso KDD culmina con la interpretación de los modelos, y por tanto, la obtención de conocimiento.

La Ciencia de Datos considera principalmente tres paradigmas del aprendizaje, aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. En el primero, se consideran las entradas y las salidas del evento sobre el que se desea aprender. El aprendizaje culmina con la interpretación del modelo generado a partir de los datos,

los cuales están etiquetados en base a la experiencia y el conocimiento previo. Sin embargo, en el aprendizaje no supervisado, no se dispone de dicha experiencia previa, y por tanto, sólo se contempla un conjunto de atributos de entrada. Un estudio de dichas técnicas pueden consultarse en [12, 13]. En el aprendizaje por refuerzo, inspirado en la psicología conductista, se introduce la noción de recompensa como vía para la obtención del conocimiento. En la Ciencia de Datos también se tratan otros tipos de problemas, como la selección de atributos o la imputación de valores autentes.

En esencia, toda representación de la realidad se realiza mediante la generación de un modelo, para lo cual se necesitan datos, y gracias a Big Data, se están desarollando herramientas de almacenamiento y análisis de ingentes cantidades de información.

2.1. Técnicas de Machine Learning aplicadas a Big Data

La mayoría de las técnicas clásicas de la Ciencia de Datos no pueden aplicarse a problemas de Big Data puesto que no están implementadas para procesar una cantidad masiva de datos, sino para trabajar con una sólo máquina y, además, teniendo todos sus datos almacenados en memoria. Por otro lado, la complejidad de los modelos también ha aumentado, lo que requiere la adaptación de estas técnicas de minería de datos a la computación distribuida, con el objeto de resolver tareas típicas de Machine Learning, como clustering, clasificación y regresión. A continuación, se introduce una breve descripción de los avances de las principales técnicas de Machine Learning que se están aplicando a problemas de datos masivos [14, 15].

En los últimos años cada vez se está prestando más atención al clustering para Big Data [16, 17]. Un estudio detallado de estas técnicas puede consultarse en [18]. En particular, muchas aproximaciones han sido recientemente propuestas para aplicar clustering a series temporales de gran longitud. En concreto, en [19] los autores proponen un nuevo algoritmo de clustering basado en un clustering previo aplicado a una muestra de los datos de entrada. En [20] se hace un procesamiento de datos basado en MapReduce para obtener clusters y en [21] se propone un método distribuido para la inicialización del algoritmo k-Means.

En cuanto a tareas de clasificación, Kotsiantis realiza un análisis detallado en [22]. En [23] las máquinas de vectores soporte (Support Vector Machine, SVM) se han modificado para adaptarse a la computación de alto rendimiento dando lugar a SVMs paralelas. Para la clasificación de series temporales en procesos industriales, se ha propuesto una Deep Feedforward Neural Network (DFNN) con una primera etapa destinada a la selección de los atributos más relevantes [24]. Este autoencoder permite eliminar la necesidad de conocimiento de expertos en la determinación de características útiles. En lo que respecta al método de vecinos cercanos (k Nearest Neighbors, kNN), en [25] se propone una implementación en paralelo de vecinos cercanos. Además, existen técnicas basadas en métodos de reducción de instancias en un paradigma MapReduce que proponen reducir el coste computacional y el requerimiento de almacenamiento para los algoritmos de clasificación basados en vecinos cercanos [26]. Para grandes volúmenes de datos, en [27] los autores desarrollaron una solución MapReduce iterativa para el algoritmo de vecinos cercanos utilizando el motor de computación distribuida Apache Spark, obteniendo un tiempo de ejecución 10 veces mejor que con Hadoop.

En el campo de la regresión, todavía hay mucho por investigar, teniendo en cuenta que se han publicado muy pocos trabajos. Las técnicas de ensemble de árboles son las más recurrentes en la literatura

debido, en parte, a su fácil adaptación a un entorno de computación distribuida. En [28] se han construido árboles de regresión utilizando aprendizaje distribuido con la tecnología MapReduce en un cluster de máquinas.

Random Forest ha sido aplicado en algunos problemas concretos, mostrando un buen rendimiento para conjuntos de datos de gran longitud [29]. Haciendo uso de tecnologías Big Data en la nube, se aplicaron algunos algoritmos de regresión a la predicción de terremotos en California [30]. En 2015, Hassani realizó un análisis de predicción con el foco en grandes conjuntos de datos [31]. En [32] se propone un sistema difuso escalable para la regresión, ya que el rendimiento de las reglas difusas depende del tamaño del problema.

Centrándonos en las técnicas de ensemble, éstas se desarrollaron para mejorar la precisión de un sistema automatizado de toma de decisiones, con el objetivo de reducir la varianza. Desde entonces, los ensemble han sido ampliamente utilizados en problemas de Machine Learning de clasificación, selección de atributos, aprendizaje incremental, predicción, etcétera. En particular, Polikar [33] ofreció una visión general de los ensemble, sus propiedades y cómo pueden ser aplicados a diferentes tareas. En [34] se han propuesto diversos métodos de ensemble y han sido utilizados con éxito en aplicaciones prácticas.

Por otro lado, Hadoop y su librería de aprendizaje automático Mahout han sido usados en [35] para tareas de clasificación utilizando Random Forest. También se han propuesto metaclasificadores combinados para la detección de malware en [36]. En [37] se propuso un algoritmo de ensemble para la clasificación multimedia, donde se utilizó un árbol de decisión para combinar las predicciones de los miembros individuales del ensemble. En cuanto al análisis en streaming, se puede encontrar un estudio detallado del uso de técnicas de ensemble en [38].

2.2. Métodos de predicción de series temporales

Una serie temporal es una secuencia de observaciones tomadas secuencialmente en el tiempo. La predicción de series temporales de corta y mediana longitud ha sido ampliamente estudiada en la literatura. Los métodos basados en ARMA (Autoregressive Moving Average Model) y GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) son un estándar en la literatura, utilizados para crear modelos estadísticos de series temporales. Ambas técnicas utilizan modelos autorregresivos, ARMA impone una estructura específica de la media, y GARCH utiliza la varianza. De este modo, los métodos para predicción de series temporales pueden clasificarse en métodos clásicos basados en Box y Jenkins [39], tales como ARMA o GARCH; y técnicas de minería de datos [40], como redes neuronales artificiales (Artificial Neural Networks, ANN), máquinas de vectores soporte o técnicas de vecinos cercanos.

A continuación se hará un breve recorrido por los principales trabajos publicados que han sido aplicados a los casos de estudio presentado en esta Tesis, series temporales en el ámbito de la energía. Una revisión completa puede ser consultada en [41], con aplicaciones a la demanda y al precio de los mercados eléctricos.

En el ámbito de la demanda eléctrica, en [42] se presenta un modelo estacional autorregresivo integrado de medias móviles (Autoregressive Integrated Moving Average, ARIMA) para predecir el máximo de la demanda mensual para los siguientes dieciocho meses, usando datos de la ciudad de Maharashtra (India) desde 1980 hasta 1999. Los resultados obtenidos por el modelo ARIMA son buenos, debido a que este mercado no presenta grandes variaciones en su tendencia a lo largo de las estaciones. No obstante, para mercados eléctricos que presentan mayor volatilidad, uno de los métodos que mejores resultados proporciona

es el modelo GARCH. Por ejemplo, en [43] se estudia con detalle la volatilidad de las series, aplicando ARIMA y GARCH a la demanda eléctrica desde 1993 a 2014 de un operador de electricidad americano, para predecir el año siguiente. Concluyen que la varianza en un modelo ARIMA permanece constante por lo que las series no estacionarias deben ser transformadas.

En el estudio llevado a cabo en [44] se analiza por primera vez la opción de aplicar SVM para predecir la demanda de energía en Taiwan, modelando 50 años de demanda desde 1945 a 1994. Utilizando los siguientes 9 años, se compararon con la aplicación de una red neuronal, de una regresión lineal y ARIMA, concluyendo que SVM es una alternativa válida para este tipo de problemas. Por otro lado, Fan et al. [45] propusieron un modelo de aprendizaje híbrido basado en clasificadores bayesianos y SVM. Primero, se usaron técnicas de clustering bayesianas para dividir el conjunto de datos en 24 subconjuntos, y entonces se aplicó una SVM a cada subconjunto para obtener las predicciones de la demanda horaria. Obtuvieron un error promedio de 1.39 % en la predicción de la demanda horaria de los dos meses siguientes, utilizando un modelo generado con los años completos desde 2001 hasta 2003. Los autores en [46] aplicaron SVM a 11 años de demanda eléctrica mensual de una provincia china para predecir los 5 años siguientes, obteniendo un error MAPE de 2.89 %, que mejoraba los resultados obtenidos con una ANN.

En [47] también se toma ARIMA como referencia para comparar el rendimiento de tres ANN, una entrenada con un algoritmo backpropagation estándar, otra ANN entrenada con el algoritmo genético CGA (Cellular Genetic Algorithm) y otra red neuronal difusa. Los resultados mostraron que, para predecir la demanda de energía en Victoria (Australia), la red neuronal difusa obtenía mejores resultados que los restantes métodos. Se utilizaron diez meses de registros de demanda tomados cada media hora. En [48], los autores proponen un algoritmo de optimización de enjambre de partículas

(Particle Swarm Optimization, PSO) para entrenar la red neuronal, y la comparan con un entrenamiento clásico de la red neuronal basado en backpropagation. PSO consigue obtener el modelo más preciso con un MAPE de 2.52 %, mejorando también a un modelo ARMA utilizado como referencia. Otra aplicación de la técnica PSO puede verse en [49], donde la comparan con un algoritmo de colonia de abejas (Artificial Bee Colony, ABC) para la demanda eléctrica en Turquía. También se utiliza ANN en [50], comparándose con técnicas de media móvil y de suavizado exponencial para el consumo eléctrico de la Universidad Covenant, en Nigeria. El mejor modelo fue obtenido con ANN, resultando un MAPE de 8.25 %. También se utilizó ANN en [51], donde se propone una nueva aproximación basada en combinar la optimización de colonia de hormigas con un algoritmo genético, mejorando los resultados de ANN y de un algoritmo difuso (ANFIS). En este caso, analizaron las demandas anuales de electricidad en Irán (también estudiaron gas y productos petrolíferos), considerando además 20 atributos socieconómicos, los cuales fueron filtrados mediante selección de atributos. El histórico utilizado está comprendido entre 1971 y 2000, y preciden los 7 años siguientes, es decir, hasta 2007. Una revisión más detallada del uso de redes neuronales artificiales aplicadas a la predicción de la demanda eléctrica puede verse en [52].

Una revisión más completa de técnicas aplicadas a la demanda eléctrica puede verse en [53, 54], donde se han analizado numerosas técnicas como métodos Naïve, ARMA, suavizado exponencial, Holt-Winters estacionales, modelos de filtrado Kalman, modelos basados en técnicas de expansión espectral, además de técnicas recientes que se basan en métodos de inteligencia artificial, como redes neuronales profundas, lógica difusa, sistemas expertos o máquinas de vectores soporte.

Una variable muy relacionada con la demanda eléctrica y que ha sido ampliamente estudiada es el precio de la energía, debido al impacto

económico, especialmente desde la desregulación del mercado, donde se pide a los participantes en el mercado eléctrico que expresen sus ofertas en términos de volumen de energía y precio de la misma. Por tanto, un operador puede ajustar su propio programa de producción y precios en función de sus propios costes de producción y la previsión del pool en cada hora. Debido a la importancia del sector eléctrico, se siguen investigando nuevas técnicas y métodos que permitan obtener mejores modelos, y aún en los últimos años siguen produciéndose avances.

Lo mercados de Ontario, Omel, Austria, y varios mercados nórdicos y australianos son utilizados en [55], donde las series de precios entre 2004 y 2009 son analizadas mediante técnicas de clustering. El mayor mercado energético de Europa en términos de consumo está en Alemania, comercializado en la Bolsa Europea de la Energía (EEX), en Leipzig. En [56] se analiza dicho mercado para proponer una nueva técnica de modelado, basada en una persectiva de los propios datos de precios, llamada Functional Factor Model (FFM).

Los autores en [57] usaron el método GARCH para predecir precios de la electricidad en dos regiones de Nueva York. Los resultados que obtuvieron fueron comparados con diferentes técnicas como regresión dinámica, modelos de función de transferencia y modelos de suavizado exponencial. En este trabajo se muestra que tener en cuenta los valores en los que la demanda es muy alta y la varianza de la serie temporal mejora la predicción, alcanzando errores menores que 2.5 %. En [58] también se propuso un modelo GARCH, para periodos de alta volatibilidad del mercado eléctrico español y californiano, con errores de predicción alrededor de 9 %.

Por otro lado, Taylor et al. [59] comparó seis modelos de series temporales univariable para predecir la demanda de electricidad de los mercados de Río de Janeiro, Inglaterra y Gales. Los métodos usados fueron un modelo ARIMA, un suavizado exponencial, una ANN y una regresión lineal. La comparación arrojó como mejores métodos

los modelos de regresión y de suavizado exponencial, que obtuvieron muy buenos resultados para la demanda en Inglaterra y Gales.

Una metodología, llamada Weighted Nearest Neighbor (WNN), basada en vecinos cercanos para la predicción del precio de la electricidad en el mercado eléctrico español fue propuesta en [60]. Se trata de una técnica de pronóstico basada en la combinación ponderada de lo acentecido anteriormente. Considerando los últimos valores pasados como patrón de referencia, se buscan los k vecinos cercanos donde el comportamiento fue parecido. Los valores siguientes a los vecinos deben ser igualmente parecidos al horizonte que queremos predecir, y por tanto, son combinados para obtener la predicción.

También en [61], se propone una discretización de kNN llamada Pattern Sequence-Based Forcasting (PSF), cuya implementación en R puede ser encontrada en [62]. Esta técnica transforma la búsqueda de vecinos cercanos en la búsqueda de secuencias discretas iguales. Aplicado al precio del mercado español, fue comparado con ANN, ARIMA y WNN. En el mercado eléctrico de Nueva York se compara el rendimiento predictivo con ARIMA; y en el mercado australiano se compara con un algoritmo para la descomposición de una señal en ondas (Discrete Wavelet Transform, DWT), una red neuronal y SVM. En todos estos mercados, el algoritmo propuesto obtuvo mejores resultados que las técnicas de referencia.

En [63] se propone PSF-NN, una combinación de PSF y ANN bajo un esquema de predicción iterada, mejorando las predicciones de la demanda eléctrica australiana obtenidas por PSF. En este trabajo se utilizaron datos horarios de 2009 y 2010 para el entrenamiento de un modelo para predecir el año siguiente, resultando en un MAPE de 3.37 %.

Una extensión de vecinos cercanos (EWNN) ha sido propuesta en [64] como mejora a WNN, en la que se usa un esquema de

predicción iterada junto a un módulo de selección de atributos. Realiza una comparación con una versión iterativa de vecinos cercanos [65] (Iterative Nearest Neighbor, INN), PSF con WNN muestra mejoras considerables para la demanda eléctrica australiana y española, obteniendo un MAPE de 3.14 % y 5.55 %, respectivamente. Y en el caso de la demanda portuguesa obtiene un MAPE de 13.55 %, sólo mejorado por INN con un MAPE de 11.70 %.

Otro campo que está mostrando relevancia es la generación eléctrica a partir de fuentes de energía renovable, principalmente debido a la dificultad de generar modelos suficientemente robustos, ya que están influenciados por variables meteorológicas. Fortuna en su libro [66] analiza diferentes modelos no lineales de aprendizaje automático para modelar un problema de radiación solar y de velocidad del viento. Una revisión de la literatura sobre la predicción de energía eólica puede consultarse en [67].

Ya en 2007 se empezaron a modelar datos meteorológicos de 16 días para precedir la generación de energía fotovoltaica de las siguientes 24 horas [68]. Consideraron 3 algoritmos de redes neuronales, Feed-Forward Neural Network (FFNN), Radial Basis Function Neural Network (RBFNN), y Recurrent Neural Network (RNN), con los que obtuvieron errores relativos entre 14.7 % y 18.9 %. También se han propuesto nuevas técnicas especializadas en el ámbito. Por ejemplo, en [69] se propone una técnica llamada HIstorical SImilar MIning (HISIMI), que utiliza un algoritmo genético con datos meteorológicos para su optimización. En [70] proponen un ensemble de ANN y un algoritmo SVM para generar las predicciones de un sistema fotovoltaico en Australia, con horizontes de 5 a 60 minutos. Además de un modelo univariante, comparan sus resultados con otro multivariante, donde se consideran datos meteorológicos. Considerando un conjunto de datos de 2 años, sólo obtienen errores relativos cercanos a 9 % para predicciones horarias, llegando en torno al 4 % cuando se predicen los siguientes 5 minutos. Tras incorporar datos meteorológicos al

problema, el modelo multivariante reduce sensiblemente el error cometido en las predicciones de la primera media hora, pero después el error incluso se incrementa debido a la falta de correlación entre las variables (en [71] hablan de la necesidad de una correlación superior a 80 %). Comello en [72] trata cómo ha evolucionado la posición de la energía solar fotovoltaica en el mercado estadounidense, y cómo es probable que evolucione en un futuro prometedor.

Sin embargo, ninguno de estos métodos de predicción se puede aplicar a series temporales de larga longitud debido a su alto coste computacional. Pfenninger [73] trata series temporales de larga longitud, pero con el objetivo de reducir la tasa de muestreo con diferentes métodos. Las investigaciones sobre regresión aplicada a Big Data están aumentando, sin embargo, hay un número reducido de trabajos publicados para realizar predicciones de series temporales big data.

En particular, en [74] se combina lógica difusa y ANN para predecir la cotización bursátil del State Bank of India (SBI). Con aplicación concreta al ámbito de la energía, en [75, 76] se utiliza un algoritmo basado en kNN para predecir series temporales mediante la ponderación de k ventanas similares al horizonte de predicción. También se ha propuesto utilizar Deep Learning para predecir series temporales de larga longitud en [77, 9].

2.3. Frameworks para Big Data

Hasta hace unos años, para realizar análisis de datos bastaba con utilizar desde herramientas domésticas como hojas de cálculo, hasta herramientas especializadas como SPSS o Weka. Con el objetivo de abordar problemas de mayor complejidad y procesar una mayor cantidad de datos, el uso de computadores cada vez más potentes

se hizo necesario, cobrando cada vez más relevancia la computación de alto rendimiento (High performance Computing, HPC), a pesar del elevado coste de adquisición de este tipo de máquinas. La madurez de la tecnología de clusters ha avanzado hasta hacer relevante la computación distribuida, que nos permite resolver problemas de computación masiva utilizando un gran número de ordenadores organizados en clusters, que resultan más rentables que los HPC y sobre todo, ofrecen una escalabilidad mucho mayor.

Por todo ello se han desarrollado nuevas herramientas de procesamiento y análisis de datos que nos ofrece un entorno trabajo más cómodo. Muchas de estas herramientas han sido desarrolladas por la comunidad de código abierto, lo que ha derivado en una mayor visibilidad. A continuación se detallan algunos de los principales frameworks de procesamiento de Big Data que más se utilizan.

- Apache Hadoop: Es el primer framework para computación distribuida que apareció de la mano de Yahoo! (después de que Google publicase el concepto del paradigma MapReduce), y sigue siendo una herramienta generalizada en la industria. Es tan frecuente que casi se ha convertido en sinónimo de Big Data. Alrededor de Hadoop hay todo un ecosistema de herramientas y tecnologías, incluyendo MapReduce, YARN, Pig, Hive, Flume y HDFS. Además, tiene una librería que ofrece algoritmos distribuidosde Machine Learning llamada Mahout.

- Apache Storm: Es un sistema de computación distribuida en tiempo real. Es escalable y tolerante a fallos. Mientras que Hadoop realiza procesamiento por lotes, Storm realiza el procesamiento en tiempo real.

- Apache Spark: Utiliza un motor de procesamiento de datos en memoria, lo que significa que en ciertas situaciones puede realizar tareas hasta cien veces más rápido que Hadoop,

manteniendo los datos en memoria. Incluso si los datos no pueden estar completamente contenidos en la memoria, tiende a ser 10 veces más rápido que su homólogo de MapReduce. También tiene disponible un conjunto de herramientas, incluyendo Spark SQL para procesamiento estructurado de datos, GraphX para el procesamiento de gráficos y Spark Streaming para el procesamiento en tiempo real.

- Apache Flink: Es una plataforma de procesamiento distribuido para análisis y aplicaciones en tiempo real, optimizado para procesos cíclicos o iterativos. Además, Flink tiene un modo de compatibilidad muy fuerte que hace posible integrarlo con Storm o MapReduce.

Tabla 2.1: Versiones de los principales frameworks de procesamiento.

Año	**Hadoop**	**Spark**	**Storm**	**Flink**
2007	0.1			
2008				
2009	0.2			
2010				
2011	1	0.2		
2012	1.1	0.7	0.8	
2013	2.2	0.8	0.9	
2014	2.4	1.2		0.7
2015	2.6	1.5		
2016	2.7	1.6	1	1.1
2017	2.9	2.2	1.1	
2018	3.1	2.4	1.2	1.4

Estos frameworks no son los únicos, sino una pequeña muestra de las herramientas de código abierto disponibles que permite una visión general de lo que se puede lograr con las herramientas seleccionadas. Además, estas herramientas tienen en común que son las principales alternativas para aprendizaje automático. Flink es un proyecto muy

prometedor, que aún está en fase de desarrollo temprano. Por otro lado, Hadoop ya ha alcanzado la madurez con su tercera *major version*. Hasta ahora, Spark se ha considerado el estándar de facto, debido a las librerías específicas que tiene y al gran apoyo que ha tenido de la comunidad, demostrado tras lanzar su segunda *major version*. En la Tabla 2.1 pude verse la evolución completa de las *minor version* a lo largo de los años.

2.3.1. Spark y la librería MLlib

Apache Spark es un potente motor de procesamiento distribuido que se ha situado como alternativa al motor MapReduce de Hadoop. Originalmente fue desarrollado en 2009 en AMPLab de la Universidad de Berkeley, y desde 2010 se encuentra como proyecto Open Source. En los años siguientes se ha visto una rápida adopción. Actualmente es utilizado en una amplia gama de industrias por más de 1000 organizaciones como Alibaba, Amazon, Cisco, eBay, Elsevier, Groupon, Hitachi, IBM, Microsoft, NASA, Nokia, Oracle, Samsung, Tripadvisor, Verizon, Visa, Yahoo!... Se ha convertido rápidamente en una de las mayores comunidades de código abierto para procesar grandes volúmenes de datos, con más de 200 colaboradores de más de 50 organizaciones.

En el ámbito de la investigación, Spark se considera una herramienta estándar para la computación intensiva de datos, habiéndose realizado numerosos estudios de su rendimiento [78, 79]. Por otro lado, Spark ha sido utilizado en una amplia gama de problemas relacionados con datos climáticos [80], el sistema de agua [81], los terremotos [82], la equiparación de entidades para la integración de información y la depuración de datos [83] o datos energéticos en edificios [84].

Para realizar el procesamiento de datos en memoria, Spark dispone de un ecosistema propio, y además, es compatible con el ecosistema de Hadoop, ya que se puede utilizar con su sistema de almacenamiento HDFS y con su gestor de recursos YARN. Este procesamiento de datos en memoria de Spark es más rápido que MapReduce de Hadoop por el hecho de que no tiene que escribir en disco los resultados obtenidos en las etapas intermedias. Esto permite a Spark que las aplicaciones puedan ejecutarse hasta 100 veces más rápido en la memoria, y 10 veces más rápido cuando incluso necesita acceder a disco. Spark destaca por la facilidad de uso, permitiendo escribir aplicaciones en Scala, Java, Python, o R; con un conjunto integrado de más de 80 operadores de alto nivel, como pueden ser filtrar, agrupar, ordenar, etc. Además, cuenta con diferentes módulos que ofrecen soporte a consultas SQL, flujos de datos, algoritmos de grafos y análisis complejos como aprendizaje automático, pudiendo combinar todas estas capacidades en un único flujo de trabajo.

A continuación se describirán brevemente los módulos principales de Spark.

Spark Core

Spark Core contiene la funcionalidad básica de Spark, incluyendo componentes para la programación de tareas, gestión de memoria, recuperación de fallos e interacción con sistemas de almacenamiento. Está diseñado para escalar eficientemente hasta utilizar miles de nodos. Spark funciona con una variedad de administradores de clústeres, incluyendo Hadoop YARN, Apache Mesos y un simple administrador propio que viene integrado.

Además, define los conjuntos de datos distribuidos resilientes (RDD), que son la principal abstracción de programación de Spark. Un RDD es una colección particionada de datos de solo lectura, y por ello contienen información sobre cómo calcularse a sí mismos, incluyendo las dependencias con otros RDD, creando un gráfico acíclico dirigido

(DAG) de cálculos, que permite realizar operaciones sobre grandes cantidades de datos de una manera rápida. Gracias a los DAG, Spark es capaz de generar un plan de ejecución optimizado que ordene las operaciones del modo más favorable (por ejemplo, filtrar antes de unir).

Spark SQL

Spark SQL es el módulo para trabajar con datos estructurados. Permite la consulta de datos a través de SQL y soporta diferentes fuentes de datos, incluyendo tablas Hive, Parquet y JSON. Además, permite a los desarrolladores entremezclar consultas SQL con operaciones de RDD, todo dentro de una sola aplicación, combinando así SQL con análisis complejos. Esta estrecha integración hace que Spark SQL sea diferente a cualquier otra herramienta de almacenamiento de datos de código abierto.

Spark Streaming

Spark Streaming es el componente que permite procesar flujos de datos, proporcionando una API para la manipulación de flujos de datos que se asemeja mucho a la API de RDD. Del mismo modo, Spark Streaming fue diseñado para proporcionar tolerancia a fallos, rendimiento y escalabilidad.

GraphX

GraphX es una librería para manipular grafos y realizar cálculos paralelos. Extiende la API de RDD, permitiéndo crear un grafo dirigido con propiedades en cada vértice y borde. También proporciona algoritmos algunos comunes, como el algoritmo PageRank y un contador de triángulos.

MLlib

MLlib es la librería de aprendizaje automático que proporciona algoritmos escalables en un cluster, diseñados para tareas de clasificación, regresión, agrupamiento y filtrado colaborativo (utilizado normalmente en sistemas de recomendación). También

tiene herramientas de caracterización, como extracción de atributos, transformación, reducción dimensional y selección de características. También tiene disponible utilidades de álgebra lineal, estadísticas, manejo de datos, persistencia y carga de los algoritmos y los modelos generados, construcción y evaluación de pipelines, etc.

Esta librería está en constante desarrollo y aún existen muchos algoritmos que no están soportados por la misma. Este es el principal motivo del crecimiento actual tanto de la librería como de la comunidad de desarrolladores que contribuyen a su desarrollo. Este trabajo presenta una metodología para realizar una predicción multi-output, la cual no está soportada directamente por la librería, aunque sí podemos aprovechar muchos recursos que ofrece tanto Spark como la librería MLlib. En este trabajo, los métodos de regresión usados han sido seleccionados con el objetivo de cubrir diferentes paradigmas, como modelos lineales, modelos basados en árboles y técnicas de ensemble. Los modelos basados en árboles se han propuesto principalmente porque los modelos obtenidos son fáciles de interpretar. Por otro lado, las técnicas de ensemble normalmente mejoran los resultados obtenidos por un único árbol, además de obtener muy buenos resultados en numerosas aplicaciones reales.

Librerías de terceros. Spark-TS

Es una librería que fue desarrollada inicialmente por el equipo de Data Science de Cloudera que permite el análisis de conjuntos de datos con millones de series temporales. La principal funcionalidad es la manipulación de series temporales, ofreciendo de una forma optimizada operaciones de alineación, filtros por fecha y hora, imputación de valores ausentes y conversión entre diferentes esquemas de datos temporales.

2.4. Fundamentos teóricos

Este trabajo se enmarca dentro del aprendizaje supervisado, cuya principal característica es que los ejemplos que forman parte del entrenamiento están etiquetados. Dentro del aprendizaje supervisado, se enmarca dentro de la regresión, donde las etiquetas de los ejemplos consisten en un valor numérico conocido como predicción. La generación del modelo de predicción se aborda con métodos lineales, entre los que se encuentran los métodos de regresión lineal, y con métodos no lineales basados en árboles de decisión, los cuales usan un aprendizaje inductivo.

La regresión lineal más clásica está basada en el método de mínimos cuadrados, pudiéndose usar diferentes funciones de pérdida sin regularización o con regularización, como la regresión Lasso, la regresión Ridge o la regresión elástica. En cuanto a los árboles de decisión, se comparan métodos que generan un único árbol o técnicas ensemble que generan muchos árboles, como son los métodos Gradient-Boosted Trees y Random Forest.

A continuación se describen brevemente los métodos que se han utilizado, los cuales se diferencian en el modelo que generan, que puede ser lineal o no lineal. Una forma sencilla de ver si un modelo es lineal o no, es examinando las derivadas de la función con respecto a cada uno de los parámetros que la definen. Si las derivadas no dependen de ninguno de los parámetros, se dice que el modelo es lineal en los parámetros o simplemente lineal.

2.4.1. Modelos lineales

Existen diversos modelos que, no siendo directamente modelos lineales, admiten una transformación que los permite transformar en

modelos lineales, como ocurre con el modelo exponencial, el potencial o el logarítmico. El caso más simple de regresión lineal se modela como la ecuación de una recta:

$$y = \alpha + \beta x + \varepsilon \tag{2.1}$$

donde α es la ordenada en el origen, β es la pendiente de la recta y ε es el error o promedio de los residuos.

La regresión lineal múltiple (término utilizado por primera vez por Pearson en [85]) se basa en obtener una relación lineal entre un conjunto de variables independientes $x_1, x_2, ..., x_d$ con una variable dependiente y, es decir:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_d x_d + \varepsilon \tag{2.2}$$

Por tanto, para obtener un modelo de predicción a partir de los datos de entrenamiento, se necesita calcular la ecuación lineal que más se ajusta a los datos. La ecuación lineal se puede expresar de forma vectorizada como sigue:

$$y = w^T x \tag{2.3}$$

donde w es un vector de dimensión d conocido como vector de pesos, d es el número de atributos del conjunto de datos, x es el vector formado por los valores de los atributos e y es la variable que se quiere predecir.

De este modo, es necesario calcular los pesos w a partir del conjunto de entrenamiento. Para ello, se usa un método estándar en Machine Learning que consiste en calcular los pesos resolviendo un problema de optimización convexa. Es decir, la tarea de encontrar el mínimo de una función convexa f en función del vector w. Formalmente, se puede escribir como el siguiente problema de optimización:

$$\min_{w} f(w) := \lambda \mathbb{R}(w) + \frac{1}{n} \sum_{i=1}^{n} L(w; x_i; y_i) \tag{2.4}$$

La función objetivo f tiene dos partes: la parte correspondiente a la regularización, que controla el sobreajuste del modelo, y la parte correspondiente a la función de pérdida, que mide el error del modelo cuando se predice y_i a partir de los datos de entrenamiento x_i. Esta función de coste es una función convexa en w, donde el vector x_i contiene los valores de las instancias que forman parte del conjunto de entrenamiento para $1 \leq i \leq n$, donde n es el número de instancias, e y_i tiene los correspondientes valores reales que se quieren predecir. El parámetro para ajustar la regularización $\lambda \geq 0$ define el equilibrio entre los dos objetivos a minimizar: el coste (error de entrenamiento) y la complejidad del modelo para evitar el sobreajuste, lo que impediría obtener buenos resultados cuando se usa un conjunto de test.

Para los métodos de regresión, la librería MLlib de Apache Spark permite utilizar la función de coste definida por el error cuadrático medio $L := \frac{1}{2}\left(w^T x - y\right)^2$, donde la predicción obtenida a partir de la instancia x viene dada por $w^T x$ y el valor real es y.

El método más utilizado para resolver problemas de optimización del tipo $\min_{w} f(w)$ es el Descenso de Gradiente Estocástico (Stochastic Gradient Descent, SGD). Se utilizan para encontrar los mínimos de una función, iterando paso a paso en la dirección descendente de la derivada de la función en cada punto. El parámetro γ es el tamaño del paso que dado un valor s inicial, va decreciendo según la raíz cuadrada de la variable contador de las iteraciones t, y es un valor muy crítico para la convergencia del método.

$$\gamma := \frac{s}{\sqrt{t}} \tag{2.5}$$

2.4.2. Modelos no lineales

Para obtener modelos no lineales se pueden utilizar algoritmos basados en árboles de decisión (Morgan, 1963). Los árboles de decisión (Decision Tree, DT) son muy comunes en el aprendizaje automático, tanto para clasificación como para regresión, ya que los modelos generados son fáciles de interpretar, son capaces de modelar relaciones no lineales entre atributos y soportan atributos discretos.

Los árboles de decisión se obtienen a través de un algoritmo voraz que realiza una partición binaria recursiva del espacio de atributos, donde se asigna la misma etiqueta para cada hoja del árbol. En cada iteración, cada atributo se elige de forma que maximice, en el caso de la librería MLlib, la ganancia de información o la varianza.

Para problemas de regresión, la librería MLlib de Spark utiliza la varianza como medida de impureza, dada por:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu)^2 \tag{2.6}$$

donde y_i es el valor real para una instancia x_i, n es el número de instancias y μ es la media de los valores y_i.

La construcción recursiva del árbol se detiene cuando se alcanza la profundidad máxima, cuando los nodos hijos no alcanzan un número mínimo de instancias o cuando no se encuentra un atributo que aumente la ganancia de información en el caso de clasificación, o que disminuya la varianza en el caso de regresión.

2.4.2.1. Ensemble de árboles de decisión

Los ensemble de modelos de predicción son uno de los métodos más exitosos utilizados en aplicaciones prácticas. Un método ensemble es un algoritmo de aprendizaje que crea un modelo compuesto de un conjunto de otros modelos base, y gracias a ello, la combinación tiene un rendimiento predictivo que mejora el que podría obtenerse con cualquiera de los modelos individuales que lo componen.

La librería MLlib de Spark soporta dos ensemble importantes, Gradient-Boosted Trees y Random Forest. Al igual que Decision Tree, pueden ser utilizados para clasificación binaria, clasificación multiclase y para regresión, permitiendo en cualquier caso, considerar atributos discretos y continuos. Estos ensemble tienen en común que utilizan árboles de decisión como modelo base, pero los procesos de entrenamiento son diferentes. Y es por ello que se propone combinar Decision Tree, Gradient-Boosted Trees y Random Forest en un único ensemble para resolver problemas de predicción de series temporales en un entorno Big Data, debido a los buenos resultados obtenidos individualmente por cada uno de estos algoritmos.

Gradient-Boosted Trees [86] (en adelante, GBT), es un método basado en una secuencia de árboles de decisión, entrenados iterativamente para minimizar la función de error. En cada iteración, el algoritmo utiliza el conjunto de árboles actual para predecir la etiqueta de cada instancia de entrenamiento, y después compara la predicción obtenida con la etiqueta real. El conjunto de datos es reetiquetado para poner más énfasis en las instancias de entrenamiento con peores predicciones. De este modo, en la siguiente iteración, el árbol de decisión ayudará a corregir los errores cometidos en la predicción con el uso del ensemble de árboles de la iteración anterior.

Random Forest [87] (en adelante, RF), es también uno de los modelos de aprendizaje automático más utilizados, tanto para problemas de clasificación como de regresión. A diferencia de GBT, este tipo de ensemble entrena un conjunto de árboles de decisión de forma paralela. El algoritmo introduce aleatoriedad al proceso de entrenamiento para que cada árbol entrenado sea diferente, seleccionando diferentes subconjuntos de instancias y subconjuntos de atributos.

Para predecir una nueva instancia, RF tiene en cuenta las predicciones de cada árbol del conjunto, procediendo de forma diferente según se trate de un problema de clasificación o regresión. En el caso de la clasificación se basa en un sistema de voto mayoritario, donde la predicción de cada árbol cuenta como un voto para la clase que corresponda, y por tanto, la etiqueta seleccionada como predicción será la clase que más votos reciba. Si se trata de un problema de regresión, donde cada árbol de decisión predice un valor real, la etiqueta será la media de las predicciones obtenidas con cada árbol.

En el ensemble propuesto, formado por DT, GBT y RF, en lugar de combinar las predicciones usando el promedio, que sería la combinación más simple y directa, se propone usar un promedio ponderado calculando diferentes pesos para cada algoritmo basado en su rendimiento previo. Para calcular los pesos, se minimiza el error de predicción en un conjunto de validación.

Sin embargo, se pueden utilizar diferentes estrategias adaptativas para actualizar los pesos después de un intervalo de tiempo dado, y así crear ensemble dinámicos.

La Figura 2.1 muestra un diagrama general del modelo de ensemble estático propuesto, para predecir una serie temporal de larga longitud. Nótese que los pesos están definidos por una matriz debido a que el problema de predicción tiene un horizonte de h periodos.

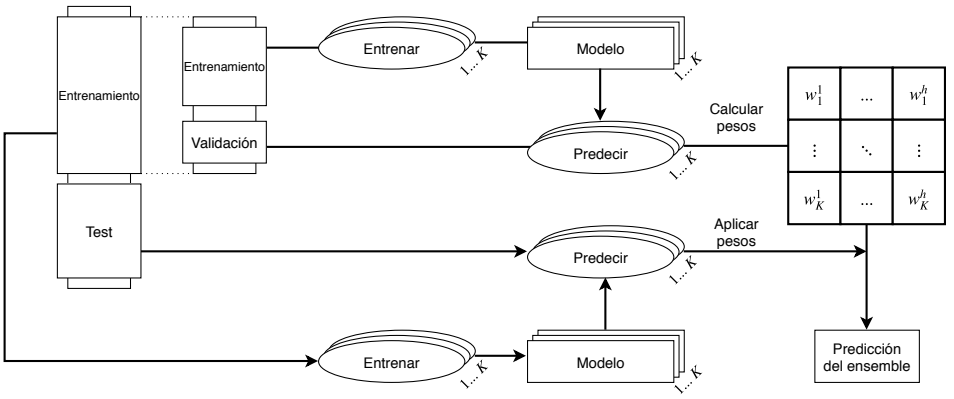


Figura 2.1: Diagrama de un ensemble estático.

Supongamos que el conjunto de test está compuesto de M instancias, es decir, un total de Mxh mediciones a predecir. En el modelo de ensemble dinámico, se considera R como el período de actualización para los pesos. A continuación, el conjunto de test TS se puede dividir en subconjuntos de la siguiente manera:

$$TS = \bigcup_{t=1}^{R} TS^t \tag{2.7}$$

donde el subconjunto TS^t está compuesto por $[M/R]$ instancias y $[\cdot]$ denota la parte entera.

Además, el conjunto de entrenamiento se actualiza desplazando $(t-1)[M/R]$ instancias hacia adelante según la ecuación siguiente:

$$TRS \leftarrow TRS^t \bigcup \left(\bigcup_{l=1}^{t-1} TS^l \right) \quad t = 1, ..., R \tag{2.8}$$

La Figura 2.2 representa gráficamente cómo funciona el ensemble dinámico propiesto en esta Tesis, donde se puede observar cómo los pesos se calculan de forma periódica con distintos conjuntos de entrenamiento.

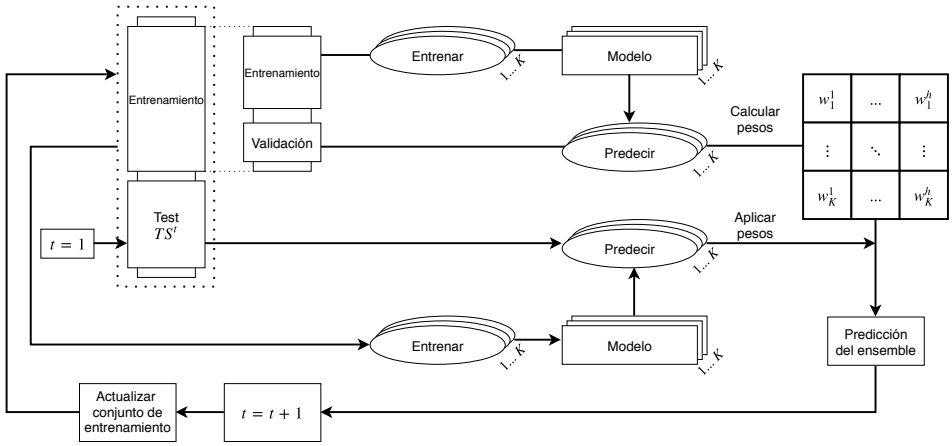


Figura 2.2: Diagrama de un ensemble dinámico.

3

Discusión de los resultados

3.1. Demanda de energía eléctrica

En este apartado se analiza la metodología, mediante su aplicación a la demanda de energía eléctrica. Se describe la serie temporal utilizada y los experimentos llevados a cabo, en los que se han utilizado distintos algoritmos (detallados en la Sección 2.4). Se comparan los resultados obtenidos por cada uno de ellos, para así poder comprobar la viabilidad de la metodología para datos masivos, en términos de precisión y escalabilidad.

En la web[1] de Red Eléctrica de España, además de publicar numerosos informes y datos estadísticos, se puede consultar la demanda peninsular de energía en tiempo real. Dada esta serie de temporal, se pretende obtener la predicción de los 24 valores siguientes, que corresponde a un horizonte de 4 horas.

[1]Portal de Red Eléctrica de España: `http://www.ree.es`

Para la aplicación de la metodología se han usado recursos de alta computación en la nube, concretamente un clúster de 5 computadores: 1 maestro y 4 esclavos. Cada nodo tiene 60 GB de RAM y 8 cores lógicos (o cores virtuales) de un Intel Xeon E5-2658 v3 @ 2.20 GHz, que tiene 30 MB de caché. Las versiones de software utilizadas han sido Apache Spark 2.0.1 y Hadoop 2.6 sobre Ubuntu 16.04 LTS.

3.1.1. El conjunto de datos

A partir de consultas a la web de Red Eléctrica de España, se ha generado un fichero CSV con los datos de la demanda de energía eléctrica en España, con cerca de medio millón de registros medidos en intervalos de 10 minutos. Como primer valor de cada fila tenemos un formato Timestamp con fecha y hora, y como segundo valor tenemos el consumo producido en MW. A continuación se muestra un fragmento del fichero de datos:

2007-01-01 00:00,28667
2007-01-01 00:10,28204
2007-01-01 00:20,27781
...
2016-06-22 03:00,24326

Este conjunto de datos se divide en un conjunto de entrenamiento, correspondiendo al 60 %, para generar el modelo de predicción para cada algoritmo, y un conjunto de test con el 40 % restante. El conjunto de entrenamiento tiene 298608 mediciones, cuyo intervalo temporal comienza el 1 de enero de 2007 a las 00:00 y termina el 8 de septiembre de 2012 a las 10:30. El conjunto de test está formado por 199080 mediciones, que corresponden a los valores comprendidos desde el 8 de septiembre de 2012 a las 10:40 hasta el 22 de junio de 2016 a las 03:00.

3.1.2. Análisis de sensibilidad

Se han seleccionado los siguientes algoritmos de la librería MLlib, los cuales serán utilizados en la metodología propuesta para la solución de los subproblemas, comparando sus resultados:

- Linear Regression (LR)
- Decision Tree (DT)
- Gradient-Boosted Trees (GBT)
- Random Forest (RF)

Cada uno de los algoritmos de regresión usados, necesita unos parámetros específicos que dependen directamente de las características de los datos que queremos predecir. En el método de regresión lineal, el descenso del gradiente estocástico requiere un número adecuado de iteraciones y de una tasa de aprendizaje γ. En cuanto a los métodos basados en árboles de regresión, el número de árboles y la profundidad máxima de los mismos son parámetros a definir. Por consiguiente, es necesario realizar un análisis de la precisión de los modelos de predicción obtenidos a partir de diferentes valores para los parámetros con el objeto de seleccionar los valores óptimos de los mismos para la generación del modelo. Para ello, utilizaremos el error relativo medio (Mean Relative Error, MRE) como medida de evaluación para comparar la precisión de las predicciones obtenidas. Su formulación es la siguiente:

$$MRE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\widehat{y}_i - y_i|}{y_i} \tag{3.1}$$

donde y_i y $\widehat{y}_i$ representan los valores reales y predichos de la serie temporal, respectivamente. Este error es conocido también en la literatura como error porcentual absoluto medio (Mean Absolute Percentage Error, MAPE).

Por otro lado, el número de valores pasados que hay que usar para predecir los h valores siguientes es otro parámetro w a determinar. La experimentación llevada a cabo ha consistido en un total de 672 ejecuciones, habiéndose obtenido un total de 64512 submodelos de predicción para la serie temporal de consumo eléctrico. Esta experimentación se ha basado en los criterios descritos a continuación:

- El tamaño de la ventana w formada por valores pasados se ha establecido en 24, 48, 72, 96, 120, 144 y 168, correspondiéndose con un histórico de 4, 8, 12, 16, 20, 24 y 28 horas, respectivamente. Con este número de valores pasados se pretende predecir los 24 valores siguientes.
- En LR, el descenso del gradiente estocástico requiere un número adecuado de iteraciones, que ha sido establecido en 25, 50, 75 y 100, y de un paso γ de 1E-10, 5E-10 y 1E-9.
- El número de árboles y la profundidad máxima de los mismos son parámetros de entrada en GBT y RF. Para ambas técnicas de ensemble, se ha establecido una profundidad de 4 y de 8. Para GBT se han establecido 5 árboles y para RF se han realizado experimentos con 25, 50, 75 y 100 árboles.

Estas experimentaciones han permitido optimizar los parámetros de los algoritmos para un mejor ajuste de los modelos. En cuanto al tamaño de la ventana con w atributos, los resultados indican que para todos los métodos basados en árboles, a mayor número de atributos, la precisión del modelo aumenta.

No obstante, estas mejoras no son lineales, y conforme se aumenta el número de atributos, la mejora obtenida es menor. Sin embargo, en el modelo LR se produce una mejora drástica al utilizar una ventana de 144 valores. No es casualidad, pues este tamaño representa un día completo de mediciones, lo que refleja una fuerte estacionalidad

de la serie temporal de demanda eléctrica en ciclos diarios. Por ese motivo, se fija $w = 144$ como valor del parámetro para las siguientes experimentaciones.

En cuanto a LR, la configuración óptima se obtuvo con un paso γ de 1E-10 y 100 iteraciones, resultando un MRE de 7.3397 %. En el caso de los métodos basados en árboles, la profundidad de los árboles es un factor crítico, reduciendo el error cometido en las precidicciones cuando se usan árboles de mayor profundidad (con un mayor coste computacional), por lo que utilizamos árboles de profundidad 8. También apreciamos que, el aumento del número de árboles no mejora la calidad del modelo de forma lineal. Por ejemplo, GBT de 5 árboles consiguió un MRE de 2.7520 %, mientras que el error de RF con 25 árboles fue 2.2338 %, con 75 árboles llegó a conseguir un MRE de 2.1863 %. Por otro lado, el tiempo de entrenamiento y generación de los modelos aumenta de forma casi lineal según aumenta el número de árboles.

3.1.3. Análisis de escalabilidad

Este análisis se ha realizado usando la configuración de los algoritmos que han dado al menor error y un número de 144 atributos. Para ello, se ha comparado el tiempo que tarda cada uno de los algoritmos, en generar los modelos a partir del conjunto de instancias de entrenamiento cuando la longitud de la serie temporal se ha incrementado multiplicando hasta por 32 veces su longitud. Además, se ha analizado la influencia de múltiples hilos de ejecución en la generación de los modelos de predicción.

En particular, se consideraron la serie temporal con su longitud original y las series temporales obtenidas multiplicando la longitud de la serie original por 2, 4, 8, 16 y 32, es decir, desde casi medio millón

de mediciones y 21 MiB de tamaño hasta 16 millones de mediciones y un tamaño de 670 MiB. En este análisis se ha observado cómo el tiempo de entrenamiento se incrementa de forma lineal conforme se incrementa la longitud de la serie temporal, lo cual demuestra el comportamiento escalable de la metodología propuesta basada en los métodos de regresión de MLlib.

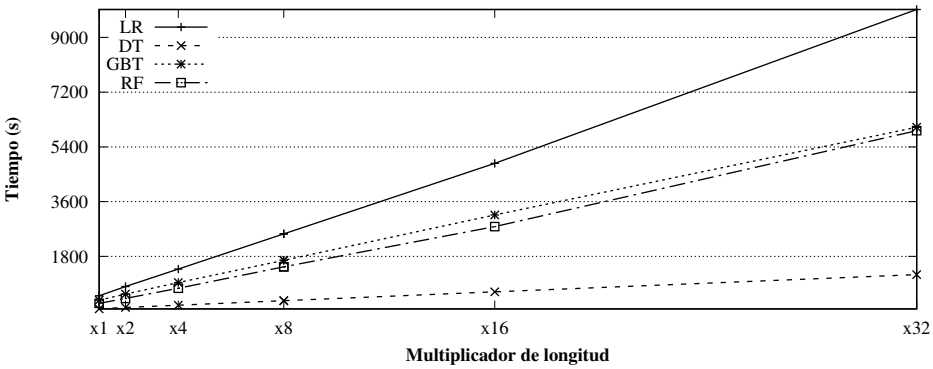


Figura 3.1: Escalabilidad de los algoritmos.

Para evaluar la influencia de los recursos de computación disponibles, se ha variado el número de hilos utilizados en el entrenamiento de los modelos, desde 1 hilo hasta 8 hilos. En primer lugar, apreciamos que los *workers* del framework de computación Spark no obtienen una mejora de rendimiento significativa más allá de los 4 hilos de procesamiento. No obstante, de forma general, los algoritmos analizados entrenan sus modelos en menos tiempo ante una mayor disponibilidad de recursos de computación, hecho observado desde la serie con longitud original hasta la serie temporal multiplicada su longitud 32 veces.

3.1.4. Ensemble estáticos y dinámicos

Hasta ahora hemos analizado el comportamiento de la regresión lineal, un único árbol de regresión, y dos algoritmos de ensemble de árboles, Gradient-Boosted Trees y Random Forest. Ahora vamos a combinar estos tres algoritmos basados en árboles de regresión, para ver su rendimiento al combinar dichos modelos.

Para la combinación de estos tres algoritmos se han promediado las predicciones de cada uno de ellos de forma ponderada. Para calcular los coeficientes de la predicción ponderada, se minimiza el error de predicción en un conjunto de validación. Para ello, se aplica un método de mínimos cuadrados para minimizar el error cuadrático cometido por los algoritmos que componen el ensemble.

Hasta ahora hemos utilizado como entrenamiento el 60 % inicial de la serie temporal, que dividiremos nuevamente en dos partes de la misma relación 60 %-40 % para obtener el conjunto de validación. De este modo, utilizaremos el 36 % inicial de la serie temporal como conjunto de entrenamiento, con el que generaremos el modelo para obtener las predicciones. La siguiente parte del conjunto de datos, que es un 24 %, será el conjunto de validación que será utilizado para obtener los valores de ponderación del ensemble. Con el 40 % final, el conjunto de test, haremos las predicciones para verificar la precisión de los algoritmos y la metodología propuesta.

Nótese que ahora el modelo se genera con sólo el 36 % inicial de los datos, correspondiendo con algo menos de 3 años y medio, mientras que en la experimentación anterior utilizábamos el 60 %, lo que suponía casi 6 años de histórico. Debido a que para el estudio del nuevo ensemble se utiliza un histórico mucho menor, es necesario obtener nuevamente los resultados para los algoritmos que componen el ensemble de forma independiente, ya que la precisión de las predicciones se verá afectada.

Además, en la experimentación realizada hasta ahora, se ha utilizado todo el conjunto de entrenamiento, el 60 % inicial de los datos, para generar un único modelo que se ha mantenido a lo largo de todas las predicciones del conjunto de test. Sin embargo, se puede generar un modelo actualizado conforme vamos avanzando en el conjunto de test. La primera aproximación se conoce como modelo estático, y cuando el modelo se actualiza a lo largo del conjunto de test, se denomina modelo dinámico. En este caso, se ha creado un nuevo modelo mensualmente, que es utilizado para predecir todos los valores del mes siguiente. En el análisis de resultados se han considerado los peores y mejores días pronosticados para los diferentes métodos de predicción. Para estudiar el MRE diario agrupamos las predicciones de cada algoritmo en grupos de 144 valores, ya que las mediciones se realizan cada 10 minutos y predecimos usando 144 valores pasados, es decir, 24 horas.

Analizando los resultados obtenidos por ambos ensemble, se ha comprobado el buen rendimiento del modelo dinámico, el cual obtuvo una mejora apreciable en comparación con el modelo estático. Para el promedio de los días predichos, actualizando el modelo mensualmente se redujo el error del conjunto de test, pasando de un MRE de 2.33 % a 2 %, lo que representa una mejora de casi un 13 % en términos relativos. Observando el peor y mejor día predicho, mientras que el ensemble estático obtuvo un error de 9.32 % y 0.82 % respectivamente, el modelo dinámico mejoró reduciendo los errores MRE hasta 8.6 % y 0.72 %, resultando en una mejora relativa cercana a 8 % y a 13 %, respectivamente, para cada uno de esos días.

3.1.4.1. Distribución del error en el horizonte de predicción

Dado que la metodología propuesta consiste en calcular las predicciones para cada valor del horizonte de predicción (en este caso, $h = 24$) con un modelo diferente, se ha realizado un análisis de la precisión de cada modelo.

Hemos visto cómo el MRE de los modelos aumenta mientras más lejos en el horizonte se encuentra el valor a predecir. En general, los MRE más bajos y más altos se obtienen cuando se pronostican el primer y el último valor del horizonte de predicción, respectivamente. En la Figura 3.2 podemos ver gráficamente cómo el ensemble dinámico mejora al estático, y la ventaja es mayor conforme el horizonte de predicción es más distante.

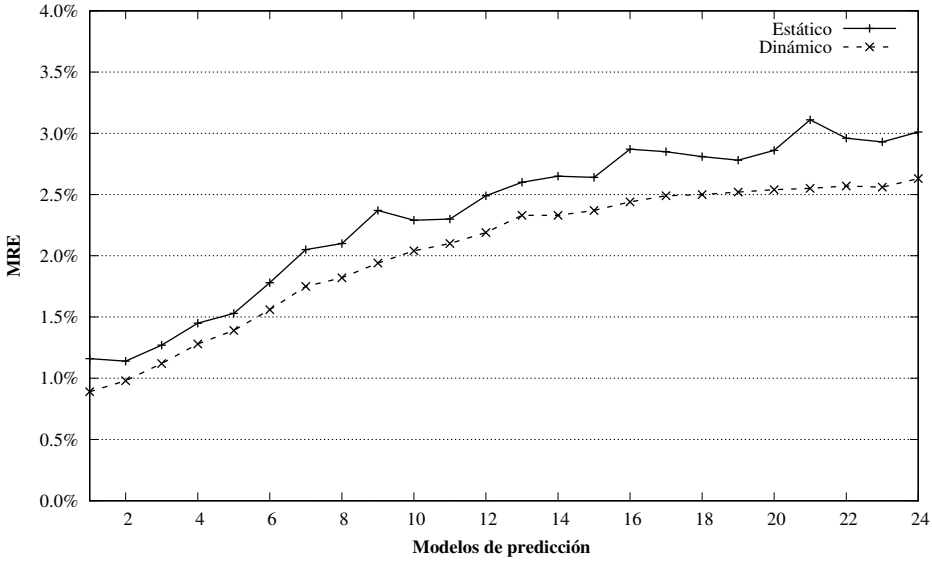


Figura 3.2: Evolución del MRE de los ensemble para cada modelo.

3.1.4.2. Análisis diario

En la Figura 3.3 se muestra el histograma del MRE diario para los ensemble dinámico y estático. El histograma representa la frecuencia del MRE diario en diferentes intervalos de error. Podemos ver que el ensemble dinámico aumenta considerablemente el número de días con errores en el intervalo entre 0.5 % y 1.5 %, donde la precisión es mayor. El impacto de esta mejora también es notable en el intervalo entre

1.5 % y 3 %, donde el número de días con estos errores es mayor en el caso del modelo estático, lo que refleja que en un gran número de días, el MRE diario al utilizar el modelo dinámico se ha reducido entre 1 y 1.5 puntos.

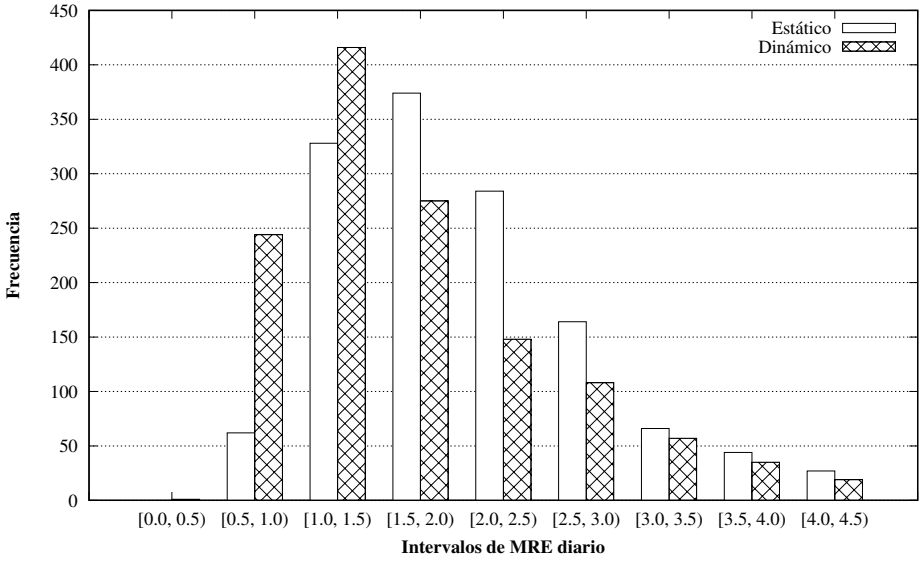


Figura 3.3: Histograma comparativo de los errores diarios.

Una característica común de ambos ensemble es que el porcentaje acumulado hasta un MRE de 5 % agrupa al 98 % de los días predichos en el conjunto de test. El 2 % restante de los días han superado dicho umbral, y hacen aumentar considerablemente el error promedio de los días. Analizando dichos días se ha podido comprobar que corresponden a días festivos o puentes, principalmente localizados en Navidad, Año Nuevo y Semana Santa, como queda resumido en la Tabla 3.1.

Tabla 3.1: Días con peores predicciones.

Ensemble estático			**Ensemble dinámico**		
MRE	Día	Tipo	MRE	Día	Tipo
9.32	24/12/13	Nochebuena	8.60	24/12/13	Nochebuena
7.71	24/12/12	Nochebuena	7.18	19/04/14	Semana Santa
7.66	19/04/14	Semana Santa	7.16	30/03/13	Semana Santa
7.40	30/03/13	Semana Santa	7.01	29/03/13	Semana Santa
7.31	24/12/15	Nochebuena	6.98	24/12/12	Nochebuena
7.15	31/12/12	Nochevieja	6.88	31/12/12	Nochevieja
7.14	24/12/14	Nochebuena	6.71	24/12/15	Nochebuena
6.96	31/12/15	Nochevieja	6.33	31/03/13	Semana Santa
6.56	31/03/13	Semana Santa	6.00	30/04/14	Día del Trabajador
6.41	30/04/13	Día del Trabajador	5.83	30/04/15	Día del Trabajador
6.29	17/04/14	Semana Santa	5.72	01/04/13	Semana Santa
6.29	31/12/13	Nochevieja	5.58	31/12/15	Nochevieja
6.00	30/04/15	Día del Trabajodor	5.54	31/12/13	Nochevieja
5.97	29/03/13	Semana Santa	5.54	07/12/14	Inmaculada Concepción
5.95	21/04/14	Semana Santa	5.48	26/12/12	Siguiente a Navidad

También es interesante estudiar los peores y mejores días pronosticados de los ensemble y de los diferentes métodos que los componen. El modelo entrenado con el algoritmo DT tiene una precisión en promedio de 96.9 %. De forma concreta, el día con peor precisión se ve reducida hasta e 89.8 %, mientras que el día con mayor precisión fue de 98.8 %. Con el algoritmo GBT, la precisión media resultó en un 97 %, 89.8 % para el día peor predicho y 98.8 % para el mejor. Estos resultados son mejorados por RF, que en promedio ha resultado en una precisión de 97.5 %, 91.2 % para el día peor predicho y 99.2 % de precisión para el día mejor predicho. Combinando el aprendizaje de estos algoritmos de forma ponderada, obtenemos un ensemble que consigue obtener una mejora global en el promedio de los días, con una precisión de las predicciones de 97.7 %.

El ensemble estático mejora el promedio del MRE diario alrededor de un 25 % en comparación con DT, un 21 % en comparación con GBT y es un 6 % más preciso que RF. En el caso del ensemble dinámico, la precisión es un 28 % más precisa que DT, un 23 % en comparación con

GBT y un 8 % mejor en comparación con RF. Además, se ha observado que ambos ensemble son capaces de reducir la varianza del error de las predicciones respecto a los modelos individuales que lo componen.

Por tanto, dentro de cada grupo (estático y dinámico), el ensemble supera los modelos de predicción individuales que combina, donde el modelo de predicción individual más preciso es RF, seguido de GBT y DT. El árbol de decisión DT es un único predictor, por lo que se espera que sea superado por los ensemble de árboles como GBT y RF. El buen desempeño de RF demuestra la ventaja de utilizar dos estrategias para generar diversos miembros del ensemble RF: bagging y selección aleatoria de características cuando se selecciona el mejor atributo de corte.

3.2. Producción de energía solar

La energía solar es una fuente de energía renovable muy prometedora, que todavía está infrautilizada. Sin embargo, en los últimos años se ha producido un aumento considerable de la utilización de placas fotovoltaicas en todo el mundo. La producción de energía solar depende en gran medida de las condiciones meteorológicas, como la radiación solar, la cobertura nubosa, las precipitaciones y la temperatura. Esta dependencia crea incertidumbre cuando es importante garantizar un suministro fiable de electricidad, lo que dificulta la integración de la energía solar en los mercados eléctricos. Por lo tanto, la capacidad de predecir la energía solar generada es una tarea crítica para los actores del sector energético.

En esta sección, se presenta y discute la aplicación de la metodología propuesta para la predicción de energía solar fotovoltaica, a partir de los datos recogidos por la Universidad de Queensland, en Australia.

3.2.1. El conjunto de datos

La serie temporal utilizada[2] abarca un periodo de dos años, desde el 1 de enero de 2015 hasta el 31 de diciembre de 2016. Es una serie temporal con intervalos de 30 minutos, donde cada día tiene 20 mediciones correspondientes al día solar, establecido desde las 07:00 hasta las 17:00.

Cuando se utiliza la metodología propuesta con un horizonte de predicción de 10 horas (h se establece en 20 valores), el conjunto de datos consta de 730 instancias y 20 atributos. Estos atributos corresponden a una ventana w de 20 valores anteriores (las 10 horas anteriores). Este conjunto de datos se divide en un conjunto de entrenamiento y un conjunto de test que consiste en un 70 % y un 30 % de los datos, respectivamente. El conjunto de entrenamiento se divide de nuevo en un conjunto de entrenamiento (70 % utilizado para generar el modelo de predicción para cada algoritmo) y un conjunto de validación (el 30 % restante utilizado para obtener los pesos del ensemble).

En el caso del ensemble dinámico, el modelo de predicción se actualiza cada dos semanas. Es decir, los pesos del ensemble se actualizan cada 280 valores pronosticados. De este modo, el conjunto de entrenamiento se desplaza 280 mediciones hacia adelante, manteniendo por tanto, el mismo tamaño del conjunto de entrenamiento y los mismos tiempos de ejecución.

3.2.2. Parámetros de los algoritmos

De acuerdo con el trabajo de referencia con el que se han comparado los resultados [88], el entorno experimental se resume a continuación:

[2]UQ Solar: `https://solar-energy.uq.edu.au/`

- El tamaño de la ventana w está formado por 20 valores pasados, correspondientes a 10 horas. Dado este número de valores pasados, el objetivo es predecir los próximos 20 valores correspondientes a las 10 horas futuras.
- El número de árboles y la profundidad máxima de los árboles son parámetros de entrada en GBT y RF. Específicamente, se ha utilizado una profundidad de 8 para ambos algoritmos, 5 árboles para GBT y 100 árboles para RF.
- La técnica de ensemble combina DT, GBT y RF.
- En el modelo dinámico, los pesos se actualizan cada dos semanas.

El error absoluto medio (Mean Absolute Error, MAE) y el error cuadrático medio (Root Mean Squared Error, RMSE) se han utilizado como medidas de evaluación para comparar la precisión de las predicciones obtenidas por los diferentes métodos de predicción. MAE y RMSE se definen como sigue:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y}_i - y_i| \tag{3.2}$$

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\widehat{y}_i - y_i)^2} \tag{3.3}$$

donde y_i y $\widehat{y}_i$ representan los valores reales y predichos de la serie temporal, respectivamente, y n es el número de muestras que hay que predecir.

3.2.3. Análisis de resultados

En esta sección, presentamos y discutimos la precisión de los modelos de predicción ensemble estáticos y dinámicos, analizando

los errores diarios junto con los peores y mejores días y los errores relativos promedio. Además se establece una comparativa con los errores cometidos por los algoritmos ANN, PSF y DL.

Recordemos que en el caso del modelo estático, construimos un modelo de predicción utilizando el 70 % de los datos como conjunto de entrenamiento. Para el modelo dinámico, el conjunto de entrenamiento siempre mantiene el mismo tamaño, pero el modelo se actualiza cada dos semanas, es decir, cada vez que se predicen 280 valores del conjunto de test. De este modo, el conjunto de entrenamiento se desplaza hacia adelante 280 mediciones y los pesos del modelo se calculan de nuevo a partir del nuevo conjunto de entrenamiento. De esta forma, se obtiene un nuevo modelo de predicción actualizado para predecir los 280 valores siguientes.

Al analizar el RMSE a partir de las predicciones del conjunto de test, el modelo ANN obtuvo el mayor error en promedio, con un RMSE de 145.16. Los modelos PSF y DL obtuvieron un rendimiento similar, un RMSE de 147.52 y 148.98, respectivamente. Ambos modelos ensemble mostraron un rendimiento superior a los otros métodos de predicción. Las predicciones del modelo estático obtuvieron un RMSE de 130.98, y el mejor modelo fue el generado por el ensemble dinámico, con un RMSE de 129.46.

3.2.3.1. Distribución del error en el horizonte de predicción

Debido a que la metodología propuesta utiliza diferentes modelos de predicción para cada valor del horizonte a predecir, esto provoca que desaparezcan las posibles relaciones entre valores consecutivos del horizonte. Por ello, se ha estudiado el comportamiento de los 20 modelos de predicción, analizando cómo de precisos son en las 10 horas predichas.

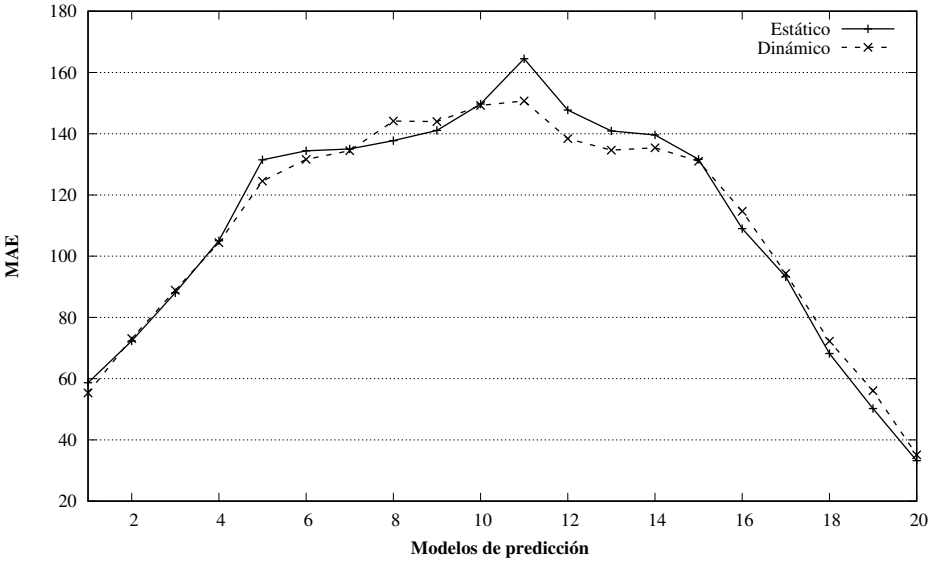


Figura 3.4: Comparación de los errores MAE.

Como se puede ver en la Figura 3.4, el MAE para ambos ensemble muestra que inicialmente, durante las primeras 1–3 horas del horizonte de predicción, la precisión es alta. Para las horas 4–7, la predicción es más difícil y el error aumenta. Para las horas siguientes del horizonte de predicción, ambos ensemble vuelven a disminuir el error. Estos resultados son consistentes tanto para MAE como para RMSE, donde el comportamiento de ambos ensemble son similares. Los intervalos donde la precisión es más alta están caracterizados por tener una concentración mayor de los datos, dado por una desviación estándar leve. Sin embargo, como puede verse en la Figura 3.5, en las horas centrales del día, donde la desviación de las mediciones es mayor, es donde se concentran las predicciones con mayor error.

Aunque el modelo dinámico mejora la predicción, presenta dificultades para predecir datos influenciados por atributos meteorológicos. Un conjunto de datos con un número mayor de registros ayudaría notablemente a obtener mejores modelos. Además,

sería apropiado contar con datos meteorológicos para modelar de forma más precisa el problema, lo que implicaría resolver el problema con algoritmos multivariantes.

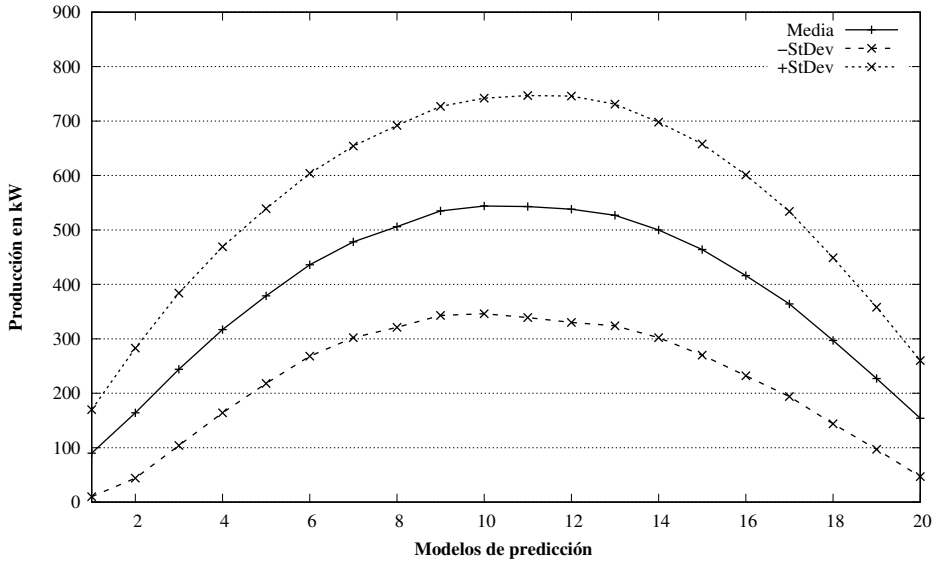


Figura 3.5: Valores medios y desviaciones de los datos de entrenamiento.

3.2.3.2. Análisis diario

Para estudiar el MAE diario agrupamos las predicciones de cada algoritmo en grupos de 20 valores. De forma general, en el promedio de los días, el método de ensemble dinámico aumenta levemente la precisión con respecto al modelo estático. Analizando la distribución de los errores (representados en la Figura 3.6), hemos apreciado que tanto utilizando el modelo estático como el dinámico, la mayoría de los días presentan errores MAE que están entre 25 y 150, siendo poco frecuente la aparición de errores más grandes. En el caso del ensemble dinámico, se aprecia que el número de días con MAE mayor que 125 es similar al ensemble estático. Es en el intervalo con un MAE mayor que

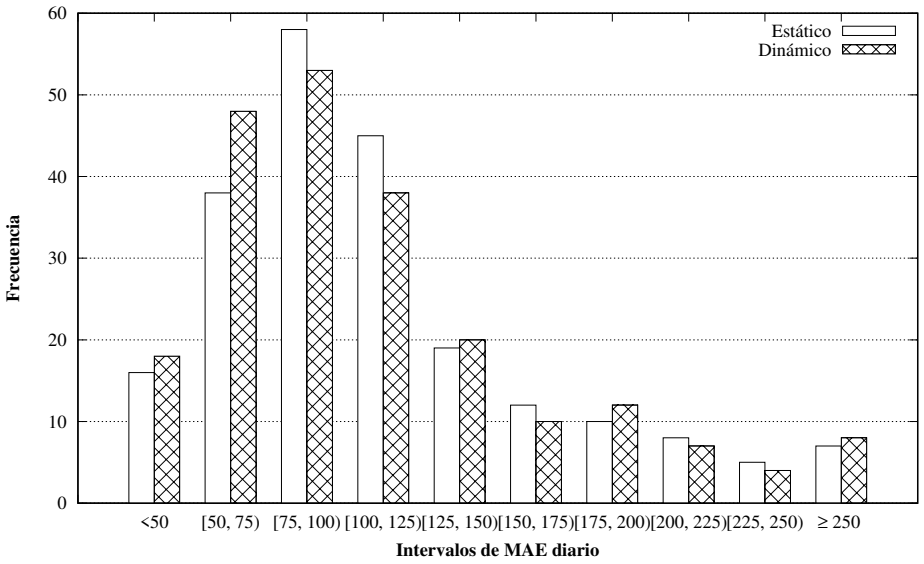


Figura 3.6: Histograma comparativo de los errores diarios.

75 y menor que 125 donde hay una disminución del número de días, y por tanto, donde el ensemble dinámico obtiene mejores predicciones que el ensemble estático. Estos días aparecen en el intervalo con un MAE mayor que 50 y menor que 75, donde el ensemble dinámico ha aumentado el número de días con este error con respecto al ensemble estático.

Deteniendo la mirada en los mejores y peores días predichos, observamos que el modelo dinámico no consigue una predicción más precisa que el modelo estático. En el caso del día con peor predicción, el MAE ha sido 26 puntos mayor que el MAE del modelo estático, aunque en el día mejor predicho el error tan solo ha sido 0.79 mayor. Estos datos, junto con la leve mejora de poco más de 1 % que consigue el modelo dinámico con respecto al estático, demuestra la necesidad de disponer de un histórico más amplio, especialmente cuando se trata con datos afectados por la meteorología.

Por otro lado, se han analizado los peores y los mejores días predichos. En el primer caso, los modelos ensemble han mostrado un rendimiento inferior a los modelos ANN, PSF y DL. ANN obtuvo un MAE cercano a 192, PSF obtuvo un MAE de 253 y DL resultó en un MAE de 206. Sin embargo, el peor día del modelo estático ha obtenido un MAE de 304 y el modelo dinámico un MAE de 330. Esta pérdida de rendimiento del modelo dinámico respecto al modelo estático es debida al deslizamiento del conjunto de entrenamiento cada 280 mediciones.

En el análisis de los días con mejor predicción, mientras que el ensemble estático obtuvo un MAE de 26, el MAE del ensemble dinámico aumentó levemente hasta 27. En el caso del mejor día predicho por los modelos PSF y DL el error de ambos incrementó hasta un MAE de 32, mientras que en el caso de ANN alcanzó un MAE de 59. Como vemos, en el caso del mejor día predicho, los ensemble redujeron el error en un 55 % respecto a los modelos comparados.

4

Conclusiones

En este trabajo se ha propuesto una formulación formal para obtener la predicción de series temporales con una alta frecuencia de muestreo y un horizonte de predicción de varios pasos, usando el framework Apache Spark. Para la implementación de los algoritmos de predicción hemos utilizado la librería MLlib, para asegurar la escalabilidad de los algoritmos utilizados y así asegurar su idoneidad para grandes conjuntos de datos. Se ha seleccionado un conjunto de modelos de regresión, lineales y no lineales, como la regresión lineal, los árboles de decisión y dos técnicas de ensemble de árboles. Aunque en nuestro caso de estudio hemos optado por combinar modelos de regresión basados en árboles (DT, GBT y RF), la metodología propuesta puede utilizarse para combinar otro tipo de modelos de predicción. Se han mostrado dos ejemplos de aplicación de la metodología propuesta, con los que se ha demostrado la viabilidad de la metodología tras obtener unos errores notablemente bajos. Igualmente se han hecho experimentos que muestran el grado de escalabilidad de cada uno de los métodos, concluyendo la viabilidad

de la metodología para la predicción de series temporales de gran dimensión. Además, la metodología ha sido validada utilizando dos series temporales de diferente naturaleza, una caracterizada por la alta frecuencia de sus registros y otra caracterizada por verse afectada por factores climatológicos que rompen la estacionalidad de la serie.

En la Sección 3.1 se ha realizado una evaluación exhaustiva utilizando datos de consumo eléctrico español durante 10 años que consistió en medio millón de registros medidos en intervalos de 10 minutos. Propusimos un método ensemble que calcula los pesos para cada miembro del ensemble usando un método de mínimos cuadrados, asignando pesos más altos a los miembros del ensemble que mejores errores obtuvieron en el pasado. Se han establecido dos estrategias para actualizar los pesos, dando como resultado un ensemble dinámico y estático. Nuestros resultados mostraron que ambos métodos de ensemble obtuvieron buenos resultados, superando a los miembros individuales del ensemble que combinaron. El ensemble dinámico fue el mejor método, el cual superó considerablemente al ensemble estático, obteniendo una precisión en las predicciones de 98 % (alcanzando hasta 99.3 % en situaciones favorables). Este es un resultado muy competitivo, que muestra la viabilidad de la metodología propuesta para la predicción de big data time series. Los resultados detallados pueden encontrarse en [7, 8].

En la Sección 3.2, un segundo caso de uso relativo a la generación de electricidad en Australia ha sido analizado. Al tratarse de generación mediante placas fotovoltaicas, los factores meteorológicos tienen una gran influencia en la producción, lo cual dificulta obtener predicciones precisas. Ha quedado patente la necesidad de un histórico de datos mucho mayor debido a los factores climatológicos. Un conjunto de datos más grande con más instancias de entrenamiento podría ayudar al ensemble dinámico, puesto que al generar un nuevo modelo con los datos más actuales, se pierde parte del conocimiento anterior. Además, cuando el día para predecir es muy atípico, incluso los

ensemble han tenido dificultades para obtener predicciones precisas. Igualmente, una selección más acertada de los modelos individuales que componen el ensemble pueden mejorar las capacidades de predicción, aunque ha quedado patente el potencial de la metodología en general y de los métodos ensemble en concreto. No obstante, la metodología utilizada ha mostrado un rendimiento aceptable en ambos ensemble. Los días mejor pronosticados por los ensemble ven reducido el error a 55 %, mejorando a los otros algoritmos con los que se comparan. Los resultados detallados pueden encontrarse en [7, 10].

Como futuros trabajos se plantea calcular los parámetros óptimos de los métodos de forma automática, optimizando el error con un conjunto de validación. Por otro lado, se analizará cómo afecta el número de particiones en el que se distribuye el conjunto de datos a la escalabilidad de los algoritmos. Además, sería muy interesante estudiar la estacionalidad de la serie temporal y su influencia en el modelo de predicción que se genera en el entrenamiento. Por último, se considera necesario verificar el comportamiento de los métodos con otros conjuntos de datos de mayor tamaño y de diferente naturaleza.

Planeamos estudiar la adición de otros tipos de modelos de predicción al ensemble, que sean adecuados para datos de gran dimensionalidad, con el fin de aumentar la diversidad entre los miembros del ensemble. También investigaremos otras estrategias de aprendizaje para determinar los pesos de una manera dinámica. Por último, también evaluaremos el rendimiento del ensemble dinámico usando otros conjuntos de datos de diferente naturaleza.

Parte III

Publicaciones

5

Informe sobre las publicaciones

A continuación se presentan el conjunto de aportaciones científicas que se han publicado durante los años en el que se ha desarrollado esta Tesis. Estas publicaciones demuestran el interés de la comunidad científica en los avances y el impacto sobre la misma. Han sido sometidas a revisión por pares, por parte de investigadores expertos; y discutidos en foros de impacto. Por cada una de las aportaciones, se incluye además, una breve historia del medio publicador, así como sus principales métricas, como el índice de impacto en el JRC, entre otros.

- [7] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «Scalable Forecasting Techniques Applied to Big Electricity Time Series». *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cadiz, Spain, June 14-16, 2017, Proceedings, Part II*. Springer International Publishing, 2017, pp. 165–175. DOI: 10.1007/978-3-319-59147-6_15. Conference Ranking: CORE-B

- [8] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «A novel Spark-based multi-step forecasting algorithm for big data time series». *Information Sciences* (2018). DOI: 10.1016/j.ins.2018.06.010. IF: 4,305 (12/148) Computer Science - Information Systems Q1

- [9] J.F. Torres, A. Galicia, A. Troncoso, and F. Martínez-Álvarez. «A scalable approach based on deep learning for big data time series forecasting». *Integrated Computer-Aided Engineering* 25 (2018), pp. 1–14. DOI: 10.3233/ICA-180580. IF: 3,667 (21/132) Computer Science - Artificial Intelligence Q1

- [10] A. Galicia, R. Talavera-Llames, A. Troncoso, I. Koprinska, and F. Martínez-Álvarez. «Multi-step forecasting for big data time series based on ensemble learning». *Knowledge-Based Systems* (2018). DOI: 10.1016/j.knosys.2018.10.009 IF: 4,396 (14/132) Computer Science - Artificial Intelligence Q1

5.1. Int. Work-Conference on Artificial Neural Networks

Desde sus comienzos en 1991, el International Work-Conference on Artificial Neural Networks –o IWANN– se ha celebrado bienalmente en las principales ciudades de España. Es un encuentro bienal que busca proporcionar un foro de discusión para científicos, ingenieros, educadores y estudiantes de postgrado sobre las últimas ideas y avances en los fundamentos, teoría, modelos y aplicaciones de sistemas inspirados en la naturaleza (redes neuronales, lógica difusa y sistemas evolutivos), así como en áreas emergentes relacionadas con los temas anteriores. IWANN está incluido en el "CORE Conference Ranking" de "Computing Research and Education Association" desde 2008, posicionándose como "CORE B". Las actas de IWANN están indexadas por INSPEC, CiteSeer, CORE y DBLP; y son publicadas por Springer en "Lecture Notes in Computer Science" (LNCS). Además, han publicado versiones ampliadas de artículos seleccionados de las ediciones más recientes en números especiales de Neurocomputing.

- [7] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «Scalable Forecasting Techniques Applied to Big Electricity Time Series». *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cadiz, Spain, June 14-16, 2017, Proceedings, Part II*. Springer International Publishing, 2017, pp. 165–175. DOI: 10.1007/978-3-319-59147-6_15. Conference Ranking: CORE-B

Scalable Forecasting Techniques Applied to Big Electricity Time Series

Antonio Galicia, José F. Torres, Francisco Martínez-Álvarez, and Alicia Troncoso( )

Division of Computer Science, Universidad Pablo de Olavide, 41013 Seville, Spain
{agalde,jftormal}@alu.upo.es, {fmaralv,ali}@upo.es

Abstract. This paper presents different scalable methods to predict time series of very long length such as time series with a high sampling frequency. The Apache Spark framework for distributed computing is proposed in order to achieve the scalability of the methods. Namely, the existing MLlib machine learning library from Spark has been used. Since MLlib does not support multivariate regression, the forecasting problem has been split into h forecasting subproblems, where h is the number of future values to predict. Then, representative forecasting methods of different nature have been chosen such as models based on trees, two ensembles techniques (gradient-boosted trees and random forests), and a linear regression as a reference method. Finally, the methodology has been tested on a real-world dataset from the Spanish electricity load data with a ten-minute frequency.

Keywords: Big data · Scalable · Electricity time series · Forecasting

1 Introduction

It is known that advances in technology have meant that the amount of data being generated and stored is increasing to the point that 90% of the data in the world have been generated in the last years. The need to process this huge amount of data has become essential for the evolution of the data mining tools giving rise to the term big data. On the other hand, an essential component in the nature of the big data is that they are commonly indexed over time, called here big time series, and its prediction in future time periods can be extremely important in diverse areas such as energy, traffic, pollution and so forth.

Nowadays, the main existing frameworks for processing big time series have been developed by over the top tech companies like Google or Yahoo. Google developed the MapReduce technology [5], which divides input data for processing in *blocks* and then integrates the output information of each block in a single solution. Later, Yahoo developed Hadoop technology [22], an open code implementation of the MapReduce paradigm, currently integrated with the Apache foundation. The limitations of MapReduce in the implementation of algorithms, which iterate

I. Rojas et al. (Eds.): IWANN 2017, Part II, LNCS 10306, pp. 165–175, 2017.
DOI: 10.1007/978-3-319-59147-6_15

over the data, have required the creation of new tools, such as Spark [9], developed by the University of Berkeley and also today in the Apache Foundation. Spark installed on a Hadoop distributed file system (HDFS) allows in-memory parallel data processing, achieving a much higher processing speed than Hadoop. Apache Spark is also an open source software project that allows the multi-pass computations, provides high-level operators, uses diverse languages (Java, Python, R) in addition to its own language called Scala, and finally, offers the machine learning library MLlib [8].

In this work, a collection of scalable algorithms are proposed in order to forecast big data time series. In particular, representative prediction methods of different nature have been chosen such as models based on trees, linear regression and two ensembles techniques (gradient-boosted trees and random forests). The algorithms have been developed in the framework Apache Spark under the Scala programming language by using the library MLlib. All the methods have been tested on a real-world big time series related to energy consumption.

The rest of the paper is structured as follows. Section 2 reviews of the existing literature related to the machine learning algorithms for big data. In Sect. 3 the proposed methodology to forecast big data time series is introduced. Section 4 presents the experimental results corresponding to the prediction of the energy consumption. Finally, Sect. 5 closes the paper giving some final conclusions.

2 Related Work

The prediction of future events has always fascinated humankind. Not in vain, many of these efforts can be seen in everyday activities, such as weather forecasting, the prediction of exchange rate fluctuations or of pollution.

The methods for time series forecasting can be roughly classified as follows: classical Box and Jenkins-based methods such as ARMA, ARIMA, ARCH or GARCH [1] and data mining techniques (the reader is referred to [12] for a taxonomy of these techniques applied to energy time series forecasting). However, the majority of the data mining techniques cannot be applied when big data have to be processed due to the high computational cost. Therefore, big data mining techniques [21,24] are being developed for distributed computing in order to solve typical tasks as clustering, classification or regression. A brief description of the main advances is made below.

Increased attention has been paid to big data clustering in recent years [11,15]. A survey on this topic can be found in [7]. Specifically, several approaches have been recently proposed to apply clustering to big data time series. Namely, in [6] the authors propose a new clustering algorithm based on a previous clustering of a sample of the input data. The dynamic time warping was tested to measure the similarity between big time series in [16]. In [23] a data processing based on MapReduce was used to obtain clusters. A distributed method for the initialization of the k-means is proposed in [3].

Regarding classification tasks, several MapReduce-based approaches in big data scenarios have been recently provided. A MapReduce-based framework

focused on several instance reduction methods is proposed in [20] to reduce the computational cost and storage requirements of the k Nearest Neighbors (kNN) classification algorithm. Also, several parallel implementations of the kNN algorithm based on Spark have been proposed in the literature [17,19]. Support vector machines (SVM) were recently adapted to the field of high performance computing giving rise to parallel SVMs [4].

In the regression field, there is still much research to be conducted, especially considering that very few works have been published. For instance, the ensemble techniques based on trees have been the most studied topic in the literature due to its easy adaptation to a distributed computing framework. Random forests have been applied to some particular problems showing a good performance for high-dimensional data [10]. On the other hand, regression trees have been built by parallel learning based on MapReduce on computer clusters in [14]. However, these methods based on a distributed computing have not used for big time series forecasting in to the best of authors' knowledge, and therefore, this work aims at filling this gap.

3 Methodology

This section describes the methodology proposed in order to forecast big data time series by using the MLlib library.

Given a time series recorded in the past up to the time t, $[x_1,...,x_t]$, the problem consists in predicting the h next values for the time series from a historical windows composed of w-values (h is known as the prediction horizon). This can be formulated as:

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = f(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \tag{1}$$

where f is the model to be found by the forecasting method in the training phase.

Nevertheless, the existing regression techniques in MLlib do not support the multivariate regression, that is, the multi-step forecasting. Therefore, the first stage splits the problem into h forecasting subproblems as follows:

$$\begin{aligned} x_{t+1} &= f_1(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \\ x_{t+2} &= f_2(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \\ &\ldots \\ x_{t+h} &= f_h(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \end{aligned} \tag{2}$$

The existing possible relations between the h consecutive values $x_{t+1}, ..., x_{t+h}$ are missed with this formulation. However, if the prediction of previous values is used to predict the next values a greater error is obtained, as the errors are accumulated in the last time stamps of the prediction horizon. Additionally, to obtain h models $f_1, ..., f_h$ to predict h values has a greater computational cost than the building of a just model f to predict all the values.

The next stage consists in solving each forecasting subproblem in the Spark distributed computing framework by using the regression methods of the MLlib library. The main variable in Apache Spark is the Resilient Distributed Dataset (RDD), which is an immutable and partitioned collection of elements that can be operated in a distributed way. Thus, every RDD created is split in blocks of the same size approximately across the nodes that integrate the cluster, as it is shown in Fig. 1.

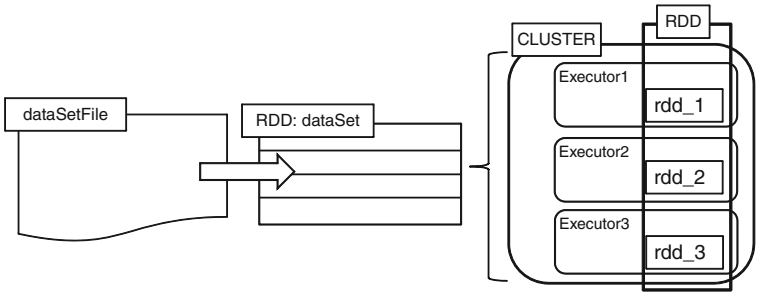


Fig. 1. A RDD variable in a spark cluster.

Once the dataset has been distributed, the MLlib algorithms firstly obtain a model from each worker node, and later, aggregate the predictions obtained for each model in a stage called reducer. It is important to highlight that RDD variables do not preserve the order, and therefore, all instances have to be indexed to deal with time series by using MLlib. An illustration of the methodology is presented in Fig. 2.

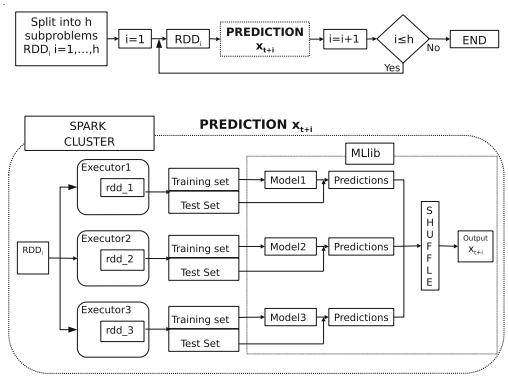


Fig. 2. Illustration of the proposed methodology.

Regression methods from MLlib have been selected according to cover different paradigms such as linear models, models based on trees and, finally, techniques ensembles.

The models based on trees have been mainly proposed because interpretable results are always desirable for the end-user. Furthermore, the ensemble techniques usually improve the results obtained by a single regressor in addition to obtain very good results for many real applications. Finally, a linear model has been selected as a state-of-the-art reference method. A brief description of the methods used for each paradigm is made below.

Within the models based on trees, a greedy algorithm [18] that performs a recursive binary partitioning of the feature space in order to build a decision tree has been used. The tree predicts the same value for all instances that reach the same leaf node. The root nodes are selected from a set of possible splits, but no from all attributes, by maximizing the information gain. In this approach, the possible split candidates are a quantile over the block of the data, which is being processed by a certain worker machine in the cluster. Moreover, once the splits are ordered, a maximum number of bins is allowed.

Two ensemble of decision trees have been considered: random forests [2] and the gradient-boosted trees (GBTs) [13]. Both algorithms learn ensembles of trees, but the training processes are very different. GBTs train one tree at a time, being the longer training than random forests, which can train multiple trees in parallel. Random forests improves the performance when the number of trees increases, however, GBTs can present overfitting if the number of trees grows too large.

Random forests is an ensemble of decision trees trained separately in the same way as detailed above for individual decision trees. The trees generated are different because of different training sets from a bootstrap subsampling and different random subsets of features to split on at each tree node are used. To make a prediction on a new instance, a random forest makes the average of the predictions from its set of decision trees.

GBTs iteratively train a sequence of decision trees. On each iteration, the algorithm uses the current ensemble to predict the label of each training instance and then compares the prediction with the true label by computing the mean square error. The training instances with poor predictions are re-labeled, and therefore, in the next iteration, the decision tree will help correct for previous mistakes.

Finally, a linear regression has been selected as linear model. The well-known stochastic gradient descent method has been used to minimize the mean square error for the training set in order to obtain the model.

4 Results

This section presents the results obtained from the application of the proposed methodology to electricity consumption big data time series to predict the 24 next values, that is, the forecast horizon set to $h = 24$ (4 h). Hence, Sect. 4.1

describes the used dataset. The experimental setup carried out is detailed in Sect. 4.2. Finally, the results are discussed in Sect. 4.3.

4.1 Datasets Description

The time series used is related to the electrical energy consumption, which ranges from January 1st 2007 at 00:00 am to June 21st 2016 at 23:40 am. The consumption is measured every ten minutes during this period. This makes a time series with a total length of 497832 measurements, which have been split into 298608 samples for the training set corresponding to the period from January 1st, 2007 at 00:00 am to September 8th 2012 at 10:30 am and 199080 samples for the test set corresponding to the period from September 8th 2012 at 10:40 am to June 21st 2016 at 11:40 pm.

4.2 Design of Experiments

The experimental setting of the algorithms is as follows:

1. The number of past values used to predict the 24 next values has been set to 144 (window $w = 144$), which represents all the values for a whole day.
2. In the linear regression, the stochastic gradient descent method requires an adequate number of iterations and rate of learning in order to guarantee the convergence of the optimization technique. In this work, values of $1.0E-10$ for the rate and 100 for the iterations have shown to be suitable.
3. The number of trees and the maximum depth are the main inputs for random forests and GBTs. Different depth levels have been tested for both ensembles, namely, four and eight. A number of five trees has been set for GBTs and values of 50, 75, 100, 125 and 150 trees for random forests.

The experimentation has been launched on a cluster, which is composed of three nodes: the master and two slaves nodes. Each node has two Intel Xeon E7-5820K processors at 3.3 GHz, 15 MB cache, 6 cores per processor and 16 GB of main memory working under Linux Ubuntu. The cluster works with Apache Spark 2.0.2 and Hadoop 2.6.

Finally, the well-known mean relative error (MRE) measure has been selected to assess the accuracy of the predictions. Its formula is:

$$MRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|\widehat{x}_i - x_i|}{x_i} \tag{3}$$

where $\widehat{x}_i$ stands for the predicted values and x_i for the actual consumption values.

4.3 Electricity Consumption Big Data Time Series Forecasting

Table 1 summarizes the MRE obtained by all methods based on trees when predicting the test set. A study of how the number of trees has an influence on the error is made for the random forests ensemble. In addition, the depth of the trees used for all methods has been analyzed. It can be seen that a greater accuracy is provided when the depth of the trees increases due to trees more specific are obtained. By contrast, it seems that the number of trees to be used by the random forest has not a high impact over the error, and therefore, fifty trees was a sufficient number to obtain a good performance of the method.

Table 1. MRE for different depth levels and number of trees.

	Decision tree	Random forests					GBTs
Number of trees	1	50	75	100	125	150	5
Depth 4	5.1516	4.2823	4.2583	4.2415	4.2415	4.2427	4.3402
Depth 8	2.8783	2.2005	2.1853	2.1842	2.1810	2.1773	2.7190

Table 2 shows the MRE for the methods based on trees when a depth of 8 and a number of 50 trees for random forests has been used. Additionally, it shows the MRE obtained by means of a linear regression as baseline method to establish a benchmarking. All non linear methods based on trees achieved better errors than the linear regression, namely a difference of 5% approximately. Although the best results are obtained by the random forests ensemble technique, it can be concluded that the decision tree is the more adequate method in terms of accuracy and CPU time to predict big data time series.

Table 2. MRE for the test set and CPU time for training.

	MRE (%)	Time (seconds)
Linear regression	7.3395	553
Decision tree	2.8783	81
Random forests	2.2005	277
GBTs	2.7190	417

Figures 3 and 4 present the predicted values along with the actual values for the random forest algorithm for the two days from the test set leading to the largest and smallest errors, respectively. The worst prediction corresponds to an error of 9.12% associated to the period from December 24th 2013 at 10:50 am to December 25th 2013 at 10:40 am and the error of the best prediction is 0.67% corresponding to the day from September 20th 2012 at 10:40 am to September 21st 2012 at 10:30 am. It can be noted that the worst day is a special day, namely, Christmas Eve.

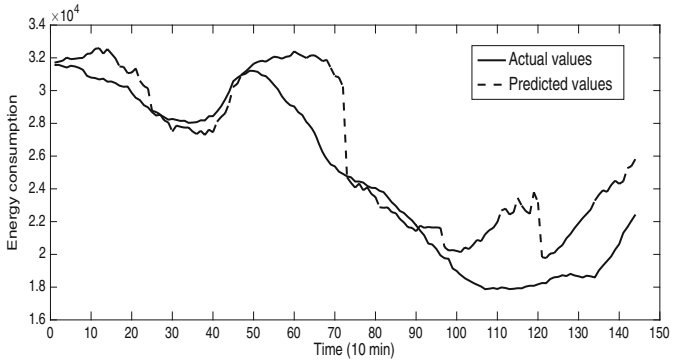


Fig. 3. The day corresponding to the worst prediction when using random forests.

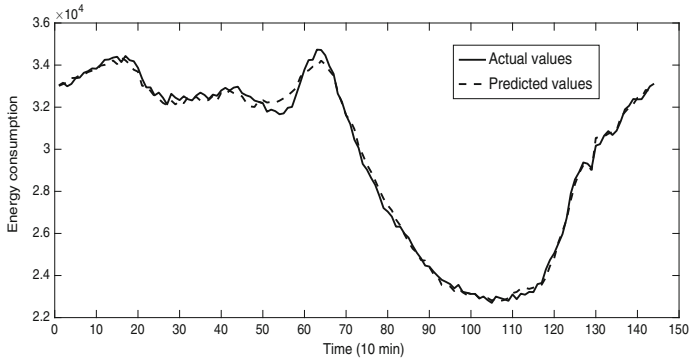


Fig. 4. The day corresponding to the best prediction when using random forests.

Finally, the training time versus the length of the time series for all algorithms proposed here are shown in the Fig. 5. The execution time has been obtained with time series of two, four, eight, sixteen and thirty and two times the length of the original time series. It is necessary to highlight the building of the dataset from the time series for each subproblem is not included in the training time as that is not made in a distributed way, but in an iterative way. From this figure, it can be observed that the most scalable method is the decision tree.

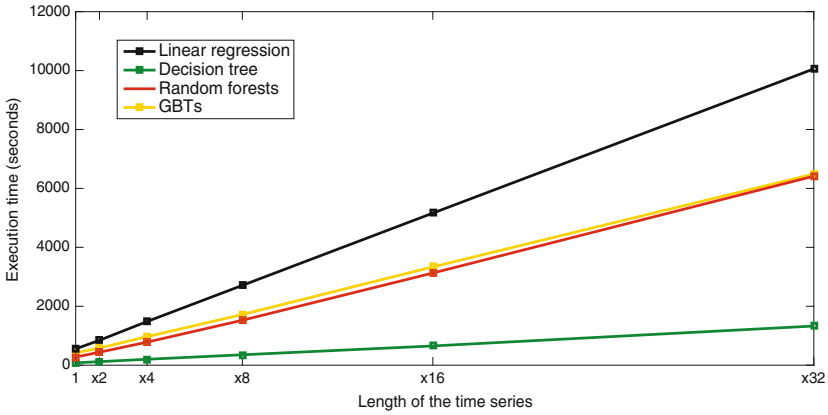


Fig. 5. Runtime and scalability for all algorithms.

5 Conclusions

In this work, a new formulation has been proposed for multi-step forecasting problems in order to be able to use the MLlib library from Apache Spark framework. The use of this library guarantees that the methods applied to predict the energy consumption for the next twenty four values are scalable, and therefore, they can be used for big data time series. A pool of linear and non linear methods have been selected, e.g., methods based on trees, ensemble techniques based on trees and a linear regression. Results for the Spanish electricity demand time series have been reported, showing the good performance of the methods proposed here and the grade of scalability for each of them.

Future work is directed towards solving the forecasting subproblems in a distributed way by using technology based on multithreads.

Acknowledgments. The authors would like to thank the Spanish Ministry of Economy and Competitiveness and Junta de Andalucía for the support under projects TIN2014-55894-C2-R and P12-TIC-1728, respectively.

References

1. Box, G., Jenkins, G.: Time Series Analysis: Forecasting and Control. Wiley, New York (2008)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Capó, M., Pérez, A., Lozano, J.A.: A Recursive k-means initialization algorithm for massive data. In: Proceedings of the Spanish Association for Artificial Intelligence, pp. 929–938 (2015)

4. Cavallaro, G., Riedel, M., Richerzhagen, M., Benediktsson, J.A.: On understanding big data impacts in remotely sensed image classification using support vector machine methods. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. **8**, 4634–4646 (2015)
5. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
6. Ding, R., Wang, Q., Dan, Y., Fu, Q., Zhang, H., Zhang, D.: Yading: fast clustering of large-scale time series data. Proc. VLDB Endow. **8**(5), 473–484 (2015)
7. Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Zomaya, A.Y., Khalil, I., Sebti, F., Bouras, A.: A survey of clustering algorithms for big data: taxonomy & empirical analysis. IEEE Trans. Emerg. Top. Comput. **5**, 267–279 (2014)
8. Machine Learning Library (MLlib) for Spark. On-line (2016). http://spark.apache.org/docs/latest/mllib-guide.html
9. Hamstra, M., Karau, H., Zaharia, M., Knwinski, A., Wendell, P., Spark, L.: Lightning-Fast Big Analytics. O' Really Media, USA (2015)
10. Li, L., Bagheri, S., Goote, H., Hassan, A., Hazard, G., Risk adjustment of patient expenditures: a big data analytics approach. In: Proceedings of the IEEE International Conference on Big Data, pp. 12–14 (2013)
11. Luna-Romera, J.M., Martínez-Ballesteros, M., García-Gutiérrez, J., Riquelme-Santos, J.C.: An approach to Silhouette and Dunn clustering indices applied to big data in spark. In: Luaces, O., Gámez, J.A., Barrenechea, E., Troncoso, A., Galar, M., Quintián, H., Corchado, E. (eds.) CAEPIA 2016. LNCS, vol. 9868, pp. 160–169. Springer, Cham (2016). doi:10.1007/978-3-319-44636-3_15
12. Martínez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., Riquelme, J.C.: A survey on data mining techniques applied to electricity-related time series forecasting. Energies **8**(11), 13162–13193 (2015)
13. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: Proceedings of the Neural Information Processing Systems Conference, NIPS, pp. 512–518 (1999)
14. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: PLANET: massively parallel learning of tree ensembles with mapreduce. In: Proceedings of the Very Large Databases, pp. 1426–1437 (2009)
15. Perez-Chacon, R., Talavera-Llames, R.L., Martinez-Alvarez, F., Troncoso, A.: Finding electric energy consumption patterns in big time series data. In: Omatu, S. (ed.) Proceedings of the International Conference on Distributed Computing and Artificial Intelligence. Advances in Intelligent Systems and Computing, vol. 474. Springer, Cham (1991)
16. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Addressing big data time series: mining trillions of time series subsequences under dynamic time warping. ACM Trans. Knowl. Discov. Data **7**(3), 267–279 (2014)
17. Reyes-Ortiz, J.L., Oneto, L., Anguita, D.: Big data analytics in the cloud: spark on Hadoop vs MPI/OpenMP on Beowulf. Procedia Comput. Sci. **53**, 121–130 (2015)
18. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers - a survey. IEEE Trans. Syst. Man Cybern. Part C **35**(4), 476–487 (2005)
19. Talavera-Llames, R.L., Pérez-Chacón, R., Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F.: A nearest neighbours-based algorithm for big time series data forecasting. In: Martínez-Álvarez, F., Troncoso, A., Quintián, H., Corchado, E. (eds.) HAIS 2016. LNCS, vol. 9648, pp. 174–185. Springer, Cham (2016). doi:10.1007/978-3-319-32034-2_15

20. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: MRPR: a mapreduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
21. Tsai, C.-W., Lai, C.-F., Chao, H.-C., Vasilakos, A.: Big data analytics: a survey. J. Big Data **2**(1), 21 (2015)
22. White, T.: Hadoop, The Definitive Guide. O' Really Media, USA (2012)
23. Zhao, W., Ma, H., He, Q.: Parallel k-means clustering based on mapreduce. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5391, pp. 674–679. Springer, Heidelberg (2009). doi:10.1007/978-3-540-95885-7_24
24. Zhou, L., Pan, S., Wang, J., Vasilakos, A.V.: Machine learning on big data: opportunities and challenges. Neurocomputing **237**, 350–361 (2017)

5.2. Information Sciences

La revista Information Sciences fue creada en 1986, y está diseñada para investigadores, desarrolladores y otros interesados en las ciencias de la información. Abarca campos como la ingeniería, matemáticas, estadística, física, informática, neurobiología, biología celular y molecular, bioquímica, etcétera. Es una revista internacional del grupo Elsevier, la mayor editorial de libros de medicina y literatura científica del mundo, fundada en 1880.

Information Sciences	
ISSN / eISSN	0020-0255 / 1872-6291
Editorial	Elsevier Science Inc., USA
Categoría	Computer Science - Information Systems
Idioma	Inglés
Frecuencia	36 números al año
InCites Journal Citation Reports de 2017	
Nuevas publicaciones	586
Citas totales	25329
Factor de impacto (F.I.)	4,305
F.I. sin autocitas	3,689
F.I. últimos 5 años	4,378

- [8] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «A novel Spark-based multi-step forecasting algorithm for big data time series». *Information Sciences* (2018). DOI: 10.1016/j.ins.2018.06.010. IF: 4,305 (12/148) Computer Science - Information Systems Q1

Information Sciences 467 (2018) 800–818

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A novel spark-based multi-step forecasting algorithm for big data time series

A. Galicia, J.F. Torres, F. Martínez-Álvarez*, A. Troncoso

Division of Computer Science, Universidad Pablo de Olavide, Seville ES-41013, Spain

ARTICLE INFO

Article history:
Received 10 August 2017
Revised 30 May 2018
Accepted 3 June 2018
Available online 15 June 2018

Keywords:
Big data
Scalable
Electricity time series
Forecasting

ABSTRACT

This paper presents different scalable methods for predicting big time series, namely time series with a high frequency measurement. Methods are also developed to deal with arbitrary prediction horizons. The Apache Spark framework is proposed for distributed computing in order to achieve the scalability of the methods. Prediction methods have been developed using Spark's MLlib library for machine learning. Since the library does not support multivariate regression, the prediction problem is formulated as h prediction subproblems, where h is the number of future values to predict, that is, the prediction horizon. Furthermore, different kinds of representative methods have been chosen, such as decision trees, two tree-based ensemble techniques (Gradient-Boosted and Random Forest) and a linear regression method as a reference method for comparisons. Finally, the methodology has been tested in a real time series of electrical demand in Spain, with a time interval of ten minutes between measurements.

1. Introduction

It is well known that advances in technology have led, in recent years, to the increasing amount of data generated and stored, to the extent that 90% of the data that exist in the world has been generated during the last two years. The need to process this huge amount of information has made it essential in recent years to develop and evolve tools that have been included under the heading of Data Mining. This evolution has given rise to the term Big Data. An essential component in the nature of the data is that information is normally indexed over time, a process that is known in the literature as time series. This case is very common in the field of Big Data, giving rise to the term Big Data Time Series. For example, two of Big Data's main sources are open data repositories, which are proposed by management for transparency policies, such as smart cities, where multiple sensors provide information on consumption, traffic, pollution, etc. These two types of data make sense if their analysis is performed with respect to their evolution over time: data that measure electrical demand or pollution can be analysed for various purposes: to predict their evolution; to predict anomalous values; to obtain patterns that allow us to compare their evolution with other data; to establish relations between certain variables with respect to others, and so forth.

Nowadays, the main existing frameworks for the massive data processing have been developed thanks to leading technology companies such as Google and Yahoo!. MapReduce technology was developed by Google [6], which for processing purposes divides the input data into blocks and then integrates the output information of each block into a single solution.

* Corresponding author.
E-mail addresses: agalde@alu.upo.es (A. Galicia), jftormal@alu.upo.es (J.F. Torres), fmaralv@upo.es (F. Martínez-Álvarez), ali@upo.es (A. Troncoso).

https://doi.org/10.1016/j.ins.2018.06.010
0020-0255/

Later, Yahoo! developed Hadoop [37], an open-source implementation based on the MapReduce paradigm, now part of the Apache Foundation. The limitations of MapReduce when implementing algorithms that need to iterate over data have required the creation of new tools, such as Spark [15], developed by the University of Berkeley in California, also within the Apache Foundation.

Spark's deployment on the Hadoop Distributed File System (HDFS) allows the parallelization of data processing in-memory, achieving much faster processing speeds than with Hadoop. Apache Spark is also an open source project that allows iterative calculations, provides high-level operators and supports several languages (Java, Python, R) in addition to its native language called Scala. Furthermore, it offers different specialised modules, such as the MLlib machine learning library [19].

The main goal of this study is to predict a large time series with a specific (but arbitrary) time horizon in the context of Big Data. To solve this problem in a Big Data context, the MLlib library has been selected. However, the MLlib library currently has certain disadvantages which are detailed below. Although some approaches for Big Data can be found in the literature, e.g. Spark TS [33]. Insufficient support is provided for these approaches as they are not officially included in the Apache Spark project.

On the one hand, the regression techniques available in MLlib do not support multivariate regression, i.e. prediction of more than one step. On the other hand, the MLlib regression methods are not designed to work with datasets where the temporal order is an important factor since no high-level operation of the Scala language retains the chronological order, a crucial aspect in a time series.

Hence, one of the main objectives of this work is to introduce a methodology, which allows MLlib to be used for the prediction of time series, where the temporal order is the main characteristic of these datasets, and also allows the prediction of a time horizon formed by h values.

In conclusion, a set of scalable algorithms are studied and adapted for very large time series forecasting. In particular, different kinds of representative methods, such as linear regression, decision trees and two tree ensembles techniques such as Gradient-Boosted and Random Forest have been chosen. The algorithms have been developed with the MLlib library of the Apache Spark framework, using Scala as the programming language. All the methods have been tested with a real time series, related to the consumption of electric energy in Spain. Reported results discuss the suitable number of cores, linearity of algorithms and speed up, among other relevant issues.

To achieve the goal set for this paper, Section 2 reviews the literature related to time series forecasting techniques and machine learning for big data. Theoretical background is also included in Section 3, where the proposed methodology and supported algorithms are detailed. Later, in Section 4.4 results are shown and discussed. Finally, Section 5 summarises the main conclusions.

2. Related work

This section discusses the most relevant related works. Due to the nature of the proposed approach, two sections have been created. First, Section 2.1 reviews works in the context of time series forecasting. Second, Section 2.2 specifically reviews works within the fields of Big Data and Machine Learning.

2.1. Time series forecasting

The prediction of time series for short and medium term has been extensively studied in the literature. The methods for predicting time series can be classified into classical methods based on Box and Jenkins [2], such as ARIMA or GARCH; and data mining techniques [38], such as neural networks (ANN), Support Vector Machine (SVM) or near-neighbor techniques(kNN).

The following will be a brief tour of the main published works, which have been applied to the study case presented here a temporal series in the field of energy. A complete and more detailed review can be found in [22].

In [12], a variation of the ARIMA model, namely a seasonal ARIMA model, is presented to predict the maximum monthly demand in the city of Maharashtra in India. They used the data from April 1980 to June 1999 and obtained the prediction of the following eighteen months. The results obtained are good because this market does not show great variations in its tendency throughout the seasons. However, for electric markets with greater volatility, one of the methods that provide the best results is the GARCH model. The authors in [13] used the GARCH method to predict electricity prices in two regions of New York. The results obtained were compared using different techniques such as dynamic regression, transfer function models and exponential smoothing models. This work shows that taking into account the values in which the demand is very high and the variance of the time series improves the prediction since they reached errors smaller than 2,5%. García et al. [11] also proposed a GARCH model. This work focuses on the prediction of electricity prices in periods of high volatility for the Spanish and Californian electricity market. Equally striking is the technique proposed by Malo and Kanto in [21], which considered multivariable GARCH models for electric markets in Nordic countries.

The performance of a standard ANN, a fuzzy ANN, and ARIMA models when predicting energy demand in Victoria (Australia) is compared in [1]. The results showed that the fuzzy neural network improves the results of the remaining methods. Taylor [32] compared six univariate time series models to predict electricity demand in the markets of Rio de Janeiro, England and Wales. The methods used were an ARIMA model, an exponential smoothing, an ANN and a linear regression. The

comparison showed the best methods to be the exponential smoothing and regression models, which obtained very good results for the demand in England and Wales. In [8], the authors presented the results obtained from an ANN applied to the prediction of energy demand in Jordan. The ANN was trained with an optimisation algorithm based on particle swarm simulation and compared to an ANN with a classic training based on back propagation.

In the study carried out in [25], the feasibility of applying SVM to predict energy demand in Taiwan was analysed. The results, were compared with those obtained from an ANN and a linear regression. Likewise, the authors in [14] reached an optimal prediction globally by applying SVM in the Chinese electricity market. Fan et al. [10] proposed a hybrid learning model based on Bayesian classifiers and SVM. First, Bayesian clustering techniques were used to divide the dataset into twenty-four subsets, and then a SVM was applied to each subset to obtain hourly demand predictions.

A methodology based on kNN was proposed in [35] for the prediction of electricity prices in the Spanish electricity market. An extension of kNN was proposed in [28] in which an iterated prediction scheme was used and an attribute selection module was incorporated. A kNN (Pattern Sequence-Based Forecasting (PSF) discretisation is proposed in [23]. PSF transforms the search of nearby neighbours in the search for equal discrete sequences. A combination of PSF and ANN under an iterated prediction scheme was proposed in [16].

2.2. Machine learning for big data

Currently, data mining techniques [36,40] are being developed for distributed computing in order to solve typical machine learning tasks, such as clustering, classification or regression for big data. The following is a brief description of the main developments obtained over the last few years.

Increasing attention has been paid in recent years to clustering for big data [18,27]. A detailed study of clustering techniques for big data can be found in [9]. In particular, many approaches have recently been proposed to apply clustering to large time series. Specifically, in [7] the authors propose a new clustering algorithm based on a previous clustering applied to a sample of the input data. In [39] the authors use a MapReduce-based data processing to obtain clusters and in [4] a distributed method is proposed for the initialisation of the k-means algorithm.

As for classification tasks, there are techniques based on methods of reduction of instances in a MapReduce paradigm [34] that propose to reduce the computational cost and storage requirement for kNN-based classification algorithms. In addition, several parallel implementations of the kNN algorithm are proposed in [29,31]. In [5], the support vector machines (SVMs) have been modified to accommodate high performance computing resulting in parallel SVMs. For large-datasets, in [20] the authors developed an iterative MapReduce solution for the k-Nearest Neighbors algorithm based on Apache Spark, obtaining a runtime 10-times better than using Hadoop.

In the field of regression, there is still much to investigate, bearing in mind that very few papers have been published. Tree ensemble techniques are the most recurrent topic in the literature due, in part, to their easy adaptation to a distributed computing environment. Random Forest has been applied to some specific problems, showing good performance for large datasets [17]. On the other hand, regression trees have been constructed using parallel learning with MapReduce technology in a machine cluster [26]. However, a large study of the literature reveals that these methods have not been applied to the prediction of large time series, and therefore, this work seeks to fill this gap in the literature.

Following a thorough review of these previously published works, it can be concluded that the prediction of time series has been extensively studied, but there is still much to investigate, bearing in mind that very few papers have been published using distributed computing system to compute large time series. These facts justify the need for research in the topic described in this paper.

3. Methodology

3.1. Theoretical background

This work is framed within supervised learning, the main characteristic of which is that the examples that are part of the training are labelled. To be precise, it entails a regression approach, where the labels of the examples consist of a numerical value known as the prediction. The generation of the prediction model is carried out with linear methods, specifically regression methods, and with non-linear methods based on decision trees, which use inductive learning.

The classical regression is based on the method of least squares, being able to use different functions of loss such as Lasso regression, Ridge regression and elastic regression, depending on whether regularisation is considered or not. As for decision trees, methods that generate a single tree or ensemble techniques that generate many trees, such as the Gradient-Boosted (GBT) and Random Forest methods, are compared.

3.2. Description of the methodology

This section describes the methodology proposed in order to forecast big data time series by using the MLlib library.

Given a time series recorded in the past up to the time t, $[x_1, ..., x_t]$, the problem consists of predicting the next h values (h is known as the prediction horizon) for the time series from a historical window composed of w-values. This is represented in the Fig. 1.

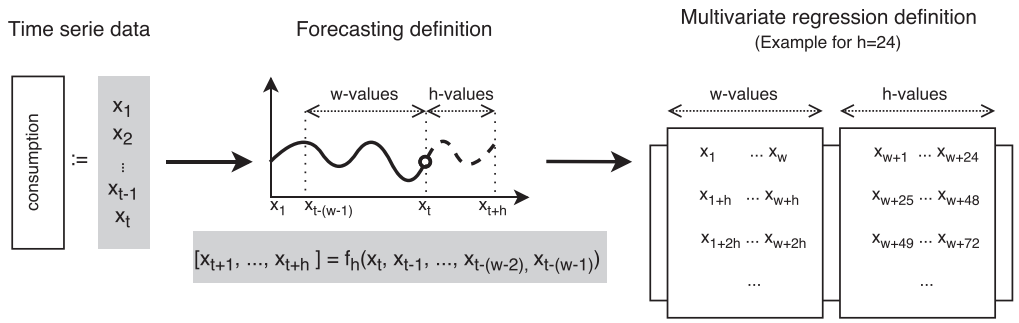


Fig. 1. Illustration of the multivariate problem.

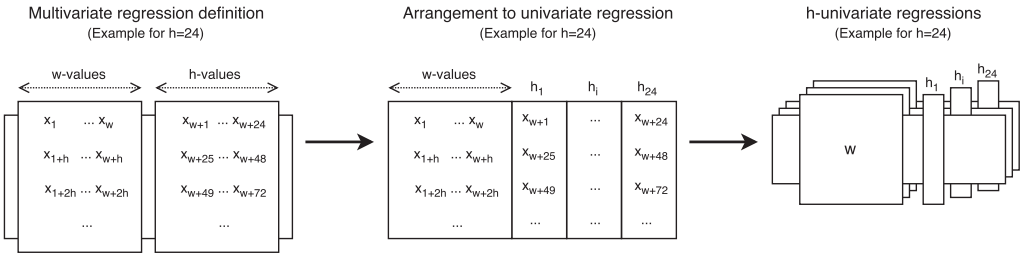


Fig. 2. Proposed methodology for multivariate to univariate adaptation.

This forecasting problem can be formulated as below, where f is the model to be found by the forecasting method in the training phase.

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = f\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \tag{1}$$

Nevertheless, the existing regression techniques in MLlib do not support the multivariate regression, that is, the multi-step forecasting. Therefore, the first stage splits the problem into h forecasting sub-problems as follows, also represented in Fig. 2:

$$\begin{aligned} x_{t+1} &= f_1\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \\ x_{t+2} &= f_2\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \\ &\vdots = \vdots \\ x_{t+h} &= f_h\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \end{aligned} \tag{2}$$

The existing possible relations between the h consecutive values $x_{t+1}, \ldots, x_{t+h}$ are missed with this formulation. However, if the prediction of previous values is used to predict the next values a greater error is obtained, as the errors are accumulated in the last time stamps of the prediction horizon.

Additionally, obtaining h models $f_1, \ldots, f_h$ to predict h values carries a greater computational cost than the building of a just model f to predict all the values.

The next stage entails solving each forecasting sub-problem in the Spark distributed computing framework by using the regression methods of the MLlib library. The main variable in Apache Spark is the Resilient Distributed Dataset (RDD), which is an immutable and partitioned collection of elements that can be operated in a distributed way. Thus, every RDD created is split into blocks of the same size approximately across the nodes that integrate the machine cluster, as it is shown in Fig. 3.

Once the dataset has been distributed, the MLlib algorithms firstly obtain a model from each worker node, and later, aggregate the predictions obtained for each model in a stage called reducer. It is important to highlight that RDD variables do not preserve the order, and therefore, all instances have to be indexed to deal with time series by using MLlib. An illustration of the methodology is presented in Fig. 4. The split strategy is represented in Fig. 4(a), where each sub-problem is executed in parallel. In Fig. 4(b) each problem is solved in a distributed way using the Spark cluster.

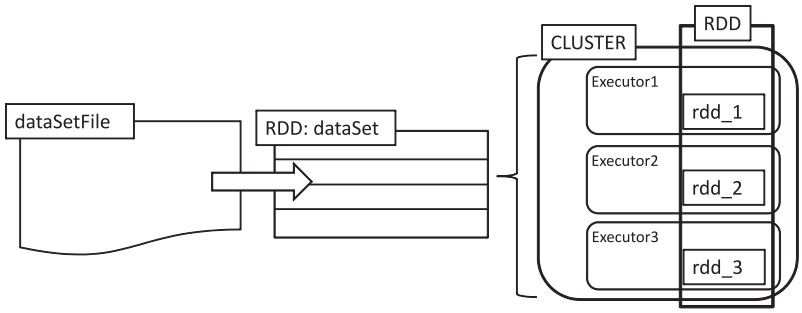
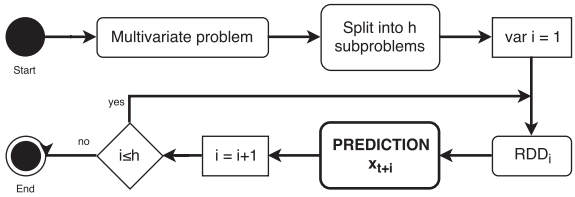
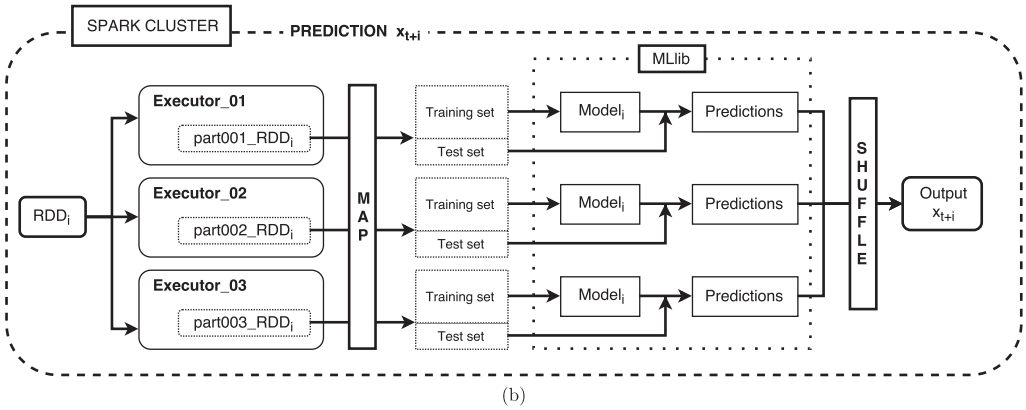


Fig. 3. A RDD variable in a Spark cluster.



(a)



(b)

Fig. 4. Methodology.

Furthermore, Fig. 5 represents how the proposed methodology generates h-models from the training set. These models and the test set are used to predict some values, and the predicted values are compared with the real value of the dataset.

Regression methods from MLlib have been selected in order to cover different paradigms such as linear models, models based on trees and, finally, ensemble techniques.

In Fig. 6, h univariate regression problems are solved. Using the instances (composed of w-features and the label h) from each training set, a representative model is generated by MLlib. With each h-model, w-features from the test set (TS_h) are used to predict the corresponding label h. The differences between the actual label and the predicted are measured by certain quality metric.

This methodology has been tested with four different methods. The models based on trees have been mainly proposed because interpretable results are always desirable for the end-user. Furthermore, the ensemble techniques usually improve the results obtained by a single regressor and also obtain very good results for many real applications. Finally, a linear

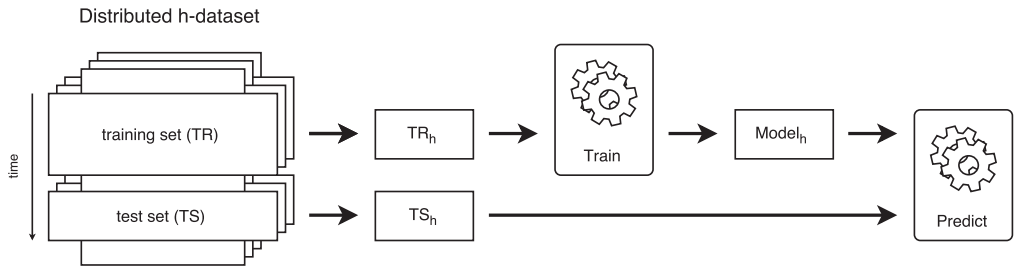


Fig. 5. *h*-model training and generation to predict the test set.

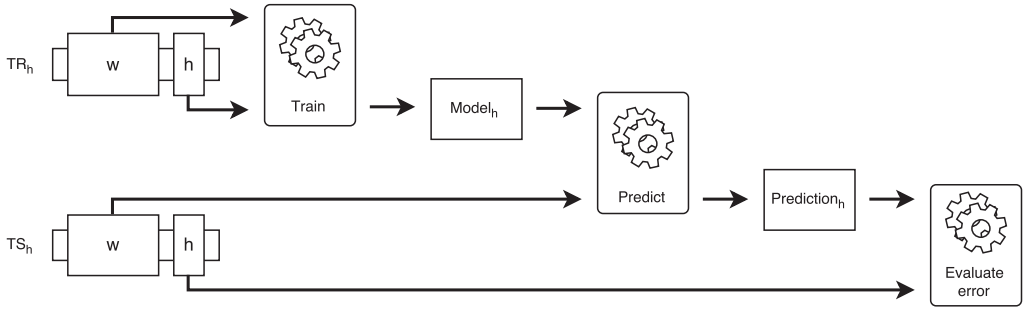


Fig. 6. Using the test set to evaluate the model.

model has been selected as a state-of-the-art reference method. A brief description of the methods used for each paradigm is provided below.

Within the models based on trees, a greedy algorithm [30] that performs a recursive binary partitioning of the feature space in order to build a decision tree has been used. The tree predicts the same value for all instances that reach the same leaf node. The root nodes are selected from a set of possible splits, but not from all attributes, by maximising the information gain. In this approach, the possible split candidates are a quantile over the data block, which is being processed by a certain worker machine in the cluster. Moreover, once the splits are ordered, a maximum number of bins is allowed.

Two ensembles of trees have been considered: Random Forest [3] and the Gradient-Boosted Trees (GBT) [24]. Both algorithms learn ensembles of trees, but the training processes are very different. GBTs train one tree at a time, providing the longer training than Random Forest, which can train multiple trees in parallel. Random Forest improves the performance when the number of trees increases. However, GBTs can present overfitting when a large number of trees is used.

Random Forest is an ensemble of decision trees trained separately in the same way as detailed above for individual decision trees. The trees generated are different because of different training sets from a bootstrap subsampling and different random subsets of features to split on at each tree node are used. To make a prediction on a new instance, a Random Forest makes the average of the predictions from its set of decision trees.

GBTs iteratively train a sequence of decision trees. On each iteration, the algorithm uses the current ensemble to predict the label of each training instance and then compares the prediction with the true label by computing the mean square error. The training instances with poor predictions are re-labelled, and therefore, in the next iteration, the decision tree will help correct for previous mistakes.

Finally, a linear regression has been selected as the reference model. The well-known stochastic gradient descent method has been used to minimise the mean square error for the training set in order to obtain the model.

4. Results

This section sets out the results obtained from the application of the proposed methodology to the prediction of big data time series for electrical consumption are shown. The methodology has been applied to a set of linear and nonlinear regression methods.

Section 4.1 sets an adequate window of historical data used to determinate the prediction in Section 4.2 for the electricity consumption dataset described in Section 4.3. With an adequate size for the window w selected, an analysis of the results from the methods is given in Section 4.4, which indicates the viability of the methodology, analysing in Section 4.5 the influence of the amount of computational resources and how the methodology responds to different time series lengths.

4.1. Design of experiments

The experimentation carried out consists of a total of 168 executions, obtaining a total of 4032 prediction models for the time series of electrical consumption in the Spanish electricity market. This experimentation was based on the criteria described below:

1. The size of the window w made up of past values has been set to 24, 48, 72, 96, 120, 144 and 168, corresponding to a history of 4, 8, 12, 16, 20, 24 and 28 hours, respectively. With this number of past values, The intention is to predict the following 24 values.
2. In linear regression, the stochastic gradient descent requires an appropriate number of iterations, which has been set to 25, 50, 75 and 100, and a step size γ (also known as the learning rate) to 1E-10, 5E-10 and 1E-9.
3. The number of trees and the maximum depth of trees are input parameters in GBT and Random Forest. For both ensemble techniques, a depth of 4 and 8 has been tested. For GBT, 5 trees have been established and for Random Forest experiments with 25, 50, 75 and 100 trees have been performed.

In all methods, the mean relative error (MRE) has been used as an evaluation measure to compare the accuracy of the predictions obtained by the different prediction methods, which are formulated as follows:

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\widehat{y}_i - y_i|}{y_i}, \tag{3}$$

where y_i and $\widehat{y}_i$ represent real and predicted values of the time series, respectively.

The experimentation has been launched on High-Performance Computing Resources on the Open Telekom Cloud Platform using five machines: the master and four slave nodes. Each node has 60 GB of main memory and 8 logical cores from an Intel Xeon E5-2658 v3 @ 2,20 GHz processor that has 30 MB L3 cache.

4.2. Sensitivity analysis

This section provides a sensitivity analysis of the window of past attributes, known as w-features. Each of the proposed methods requires different parameters, affecting to the convergence.

Table 1 shows the results obtained by applying a linear regression (LR, hereinafter) using the stochastic gradient (known as LinearRegressionWithSGD in MLlib) as the optimisation method. SGD requires two parameters: *stepSize*, referring to the learning rate 1E-10, 5E-10 and 1E-9; and *numIterations*, which is the number of iterations set at 25, 50, 75 and 100. In this way, 84 prediction models have been obtained. The SGD parameters clearly affect the convergence of the optimisation problem. Optimal configuration was obtained with a window of 144 values, a step of 1E-10 and 100 iterations, obtaining an MRE of 7,3397%. When *numIterations* and *stepSize* mean that the method is not converged, the MRE is represented by NC (not converged).

Table 2 shows the results obtained by applying a regression tree using the method known in MLlib as DecisionTreeRegression (DT). This method entails specifying the maximum depth of the tree, *maxDepth*, which has been set to 4 and 8. In this way, 14 prediction models have been obtained. The optimum configuration was obtained with a window of 168 values and a depth of 8, obtaining a MRE of 2,8958%. Smaller errors are obtained with deeper trees.

Table 3 shows the results obtained by applying the ensemble GBT technique, known in MLlib as GradientBoostingRegression, to the prediction of the test set. In addition to the number of trees to train, which has been set at 5, this method involves specifying *maxDepth*, also established at 4 and 8. Fourteen models have been obtained, the optimal model being the one that uses a window of 168 passed values and trees of depth 8. The error obtained for this model was 2,7431%. Likewise, deeper trees are closer than those of lower depth.

Finally, the ensemble Random Forest technique, known as RandomForestRegression in MLlib, has been applied to obtain the prediction of the test set. Table 4 shows the MRE obtained depending on the parameters of the method. These parameters are the number of trees to train, considering in this experiment 25, 50, 75 and 100 trees; and also 4 and 8 have been set as the maximum depth of the tree. Finally, 56 models have been obtained, with the smallest error (2,0831%) achieved for a window of 168 past values and 100 trees of depth 8.

For each method, Table 5 shows the minimum MRE obtained in the prediction of the test set for each value of the window, independently of the rest of the parameters.

Table 5 and Fig. 7 shows the evolution of MRE when increasing the window size increases for all proposed methods, selecting the lowest MRE for each window size. For all tree-based methods, an improvement in the MRE can be seen when the size of w grows. However, a significant improvement is not achieved when the window is increased from 144 values to 168, and is barely appreciable for DT and GBT. Nevertheless, MRE is increasing even in the case of linear regression using a window with 168 previous values.

Table 1
MRE for LR.

w	stepSize	numIterations	MRE (%)	w	stepSize	numIterations	MRE (%)
24	1,00E−11	25	16,3889	96	5,00E−11	75	15,2191
24	1,00E−11	50	14,9937	96	5,00E−11	100	15,2191
24	1,00E−11	75	14,9937	96	1,00E−10	25	NC
24	1,00E−11	100	14,9937	96	1,00E−10	50	13,5324
24	5,00E−11	25	12,8400	96	1,00E−10	75	13,5324
24	5,00E−11	50	12,8400	96	1,00E−10	100	13,5324
24	5,00E−11	75	12,8400	120	1,00E−11	25	14,4325
24	5,00E−11	100	12,8400	120	1,00E−11	50	14,4325
24	1,00E−10	25	12,7129	120	1,00E−11	75	14,4325
24	1,00E−10	50	12,7129	120	1,00E−11	100	14,4325
24	1,00E−10	75	12,7129	120	5,00E−11	25	13,0596
24	1,00E−10	100	12,7129	120	5,00E−11	50	13,0596
48	1,00E−11	25	14,9596	120	5,00E−11	75	13,0596
48	1,00E−11	50	14,9596	120	5,00E−11	100	13,0596
48	1,00E−11	75	14,9596	120	1,00E−10	25	NC
48	1,00E−11	100	14,9596	120	1,00E−10	50	NC
48	5,00E−11	25	14,6481	120	1,00E−10	75	10,4554
48	5,00E−11	50	14,6481	120	1,00E−10	100	10,4554
48	5,00E−11	75	14,6481	144	1,00E−11	25	12,5119
48	5,00E−11	100	14,6481	144	1,00E−11	50	12,5119
48	1,00E−10	25	13,9949	144	1,00E−11	75	12,5119
48	1,00E−10	50	13,9949	144	1,00E−11	100	12,5119
48	1,00E−10	75	13,9949	144	5,00E−11	25	10,4821
48	1,00E−10	100	13,9949	144	5,00E−11	50	10,3061
72	1,00E−11	25	15,8229	144	5,00E−11	75	10,3061
72	1,00E−11	50	15,8229	144	5,00E−11	100	10,3061
72	1,00E−11	75	15,8229	144	1,00E−10	25	NC
72	1,00E−11	100	15,8229	144	1,00E−10	50	NC
72	5,00E−11	25	15,1816	144	1,00E−10	75	NC
72	5,00E−11	50	15,1816	144	1,00E−10	100	**7,33970**
72	5,00E−11	75	15,1816	168	1,00E−11	25	12,3389
72	5,00E−11	100	15,1816	168	1,00E−11	50	12,3389
72	1,00E−10	25	14,1608	168	1,00E−11	75	12,3389
72	1,00E−10	50	14,0328	168	1,00E−11	100	12,3389
72	1,00E−10	75	14,0328	168	5,00E−11	25	NC
72	1,00E−10	100	14,0328	168	5,00E−11	50	10,0876
96	1,00E−11	25	16,0632	168	5,00E−11	75	10,0876
96	1,00E−11	50	16,0632	168	5,00E−11	100	10,0876
96	1,00E−11	75	16,0632	168	1,00E−10	25	NC
96	1,00E−11	100	16,0632	168	1,00E−10	50	NC
96	5,00E−11	25	15,2191	168	1,00E−10	75	NC
96	5,00E−11	50	15,2191	168	1,00E−10	100	NC

Table 2
MRE for DT.

w	maxDepth	MRE (%)
24	4	6,6991
24	8	4,7625
48	4	6,4666
48	8	4,0322
72	4	5,9180
72	8	3,4386
96	4	5,8596
96	8	3,3032
120	4	5,3441
120	8	3,1801
144	4	5,1291
144	8	2,9271
168	4	5,0214
168	8	**2,8958**

Table 3
MRE for GBT.

w	maxDepth	MRE (%)
24	4	6,1276
24	8	4,4633
48	4	5,8249
48	8	3,7019
72	4	5,1246
72	8	3,2383
96	4	4,9933
96	8	3,1334
120	4	4,5709
120	8	3,0165
144	4	4,2949
144	8	2,7520
168	4	4,2567
168	8	**2,7431**

Table 4
MRE for RF.

w	stepSize	numIterations	MRE (%)	w	stepSize	numIterations	MRE (%)
24	25	4	6,5787	96	75	4	5,3174
24	25	8	4,5122	96	75	8	2,7045
24	50	4	6,5566	96	100	4	5,3106
24	50	8	4,4915	96	100	8	2,7098
24	75	4	6,5599	120	25	4	4,6510
24	75	8	4,5021	120	25	8	2,4728
24	100	4	6,5615	120	50	4	4,6274
24	100	8	4,4846	120	50	8	2,4344
48	25	4	6,1533	120	75	4	4,6177
48	25	8	3,6477	120	75	8	2,4229
48	50	4	6,1435	120	100	4	4,6081
48	50	8	3,6185	120	100	8	2,4160
48	75	4	6,1277	144	25	4	4,2856
48	75	8	3,5969	144	25	8	2,2338
48	100	4	6,1333	144	50	4	4,2354
48	100	8	3,6006	144	50	8	2,1898
72	25	4	5,5598	144	75	4	4,2533
72	25	8	2,9286	144	75	8	2,1863
72	50	4	5,4919	144	100	4	4,2387
72	50	8	2,8984	144	100	8	2,1867
72	75	4	5,5253	168	25	4	4,0934
72	75	8	2,8912	168	25	8	2,1281
72	100	4	5,4969	168	50	4	4,0520
72	100	8	2,8893	168	50	8	2,0964
96	25	4	5,3290	168	75	4	4,0527
96	25	8	2,7466	168	75	8	2,0855
96	50	4	5,3299	168	100	4	4,0510
96	50	8	2,7245	168	100	8	**2,0831**

Table 5
Minimum MRE (%) for all methods.

w	LR	DT	GBT	RF
24	10,8781	4,7625	4,4633	4,4846
48	13,9949	4,0322	3,7019	3,5969
72	14,0328	3,4386	3,2383	2,8912
96	13,5324	3,3032	3,1334	2,7045
120	10,4554	3,1801	3,0165	2,4160
144	**7,3397**	2,9271	2,7520	2,1863
168	10,0876	**2,8958**	**2,7431**	**2,0831**

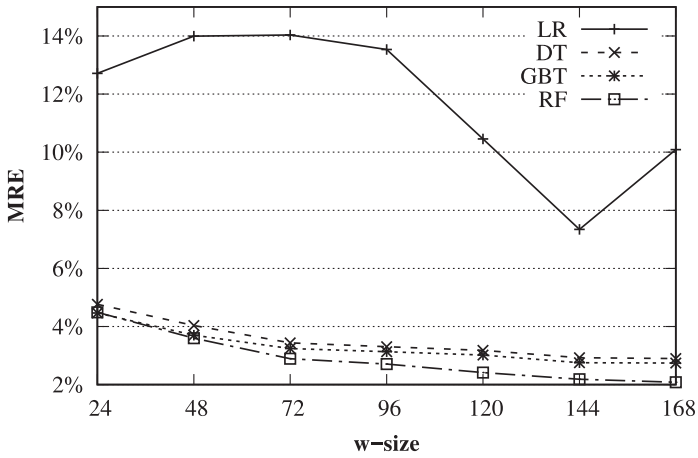


Fig. 7. MRE evolution as the window size increases.

Table 6
MRE for different depth levels and number of trees.

	DT	GBT	RF			
Number of trees	1	5	25	50	75	100
Depth 4	**5,1291**	**4,2949**	4,2856	**4,2354**	4,2533	4,2387
Depth 8	**2,9271**	**2,7520**	2,2338	2,1898	**2,1863**	2,1867

For this reason, $w = 144$ is the selected value for the analysis of the results shown in the following sections. This value is not accidental since it represents the values corresponding to the 24 hours of knowledge window before the day to be predicted, thus demonstrating the strong stationarity of the time series for electric demand in daily periods.

4.3. Dataset description

The time series used is related to the total electrical energy consumption in Spain, which ranges from January 1st 2007 at midnight to June 21st 2016 at 11:40 pm. In short, it is a time series of 9 and a half years which has a high sampling frequency - 10 min intervals - giving a total of 497832 measurements.

With a prediction horizon of 4 hours (h is set to 24 values), the dataset consists of 20742 instances and 144 attributes, corresponding to 5,70 MiB of storage size. These 144 attributes correspond to a window w of 144 past values (24 h). This dataset is divided into a training set, corresponding to 60%, to generate the prediction model for each method, and a test set corresponding to 40%. The training set has 298752 measurements, whose time interval begins on January 1st, 2007 at midnight and ends on September 8th, 2012 at 10:30 am. Therefore, the test set consists of 199080 measurements, which correspond to the values included from September 8th, 2012 at 10:40 am to June 21st, 2016 at 11:40 pm.

4.4. Analysis of results

After obtaining the optimum window to generate the models for each of the methods, Table 6 summarises the MRE (in percentage) obtained when the test set is predicted for each of the tree-based methods. The depth of the trees clearly influences the error and the number of trees in the case of Random Forest.

The same information summarised in Table 6 is shown graphically in Fig. 8.

Tree depth is a critical factor, reducing the error made in the predicted values when using deeper trees. However, by increasing depth, more computation time is needed to obtain the prediction model. Furthermore, in the Random Forest technique, although the optimum error is obtained with 75 trees, there are no significant differences when using a smaller or larger number of trees.

Table 7 summarises the generation times of the prediction model, i.e. the training times (in seconds), for each of the methods, using trees of depth 8 and 75 trees in the case of Random Forest. All non-linear tree-based methods have achieved

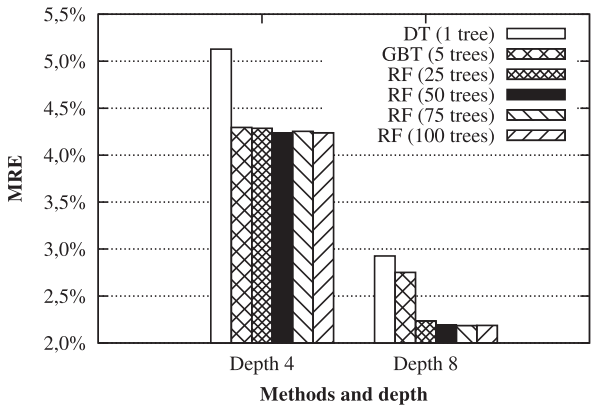


Fig. 8. MRE for different depth levels and number of trees.

Table 7
Execution time for training and MRE for test set.

	MRE (%)	Time (s)
LR	7,3397	503
DT	2,9271	**72**
GBT	2,7520	358
RF	**2,1863**	253

Table 8
Errors of worst and best predicted days at test set.

	LR	DT	GBT	RF
Worst	14,0004	10,1348	9,7966	**9,1872**
Mean	7,3397	2,9274	2,7520	**2,1863**
Best	3,3762	1,1877	1,0656	**0,6745**

errors less than linear regression, with a 5% difference approximately. Although the Random Forest ensemble technique has obtained the best result, it is possible to conclude that the decision tree could be considered the most appropriate method, especially considering the time required to generate the model with long time series.

So far the average relative error obtained in the prediction of the test set has been analysed. However, it is interesting to study maximum and minimum errors of methods analysed.

The time series for electrical demand has measurements every 10 min. In order to study of daily errors, the predictions obtained must be grouped into groups of 144 values (24 h). Hence, Table 8 presents the error of the best and worst predicted day for each method.

Fig. 9 shows the average relative error of the predictions made on the test set for each of the algorithms, as well as the errors corresponding to the days with the best and the worst prediction.

Due to the large difference between the worst predicted day and the average of every predicted day in the test set, the assumed MRE after predicting each day is shown in Fig. 10. The figure shows the MRE of the test set, which consists of 199,080 measurements, corresponding to the values included from September 8th, 2012 at 10:40 am to June 21st, 2016 at 11:40 pm.

The best daily predictions for each of the methods are shown graphically in Fig. 11. Fig. 11(a) shows the day with the best prediction obtained with the Linear Regression. The MRE is 3,37% and corresponds to measurements from Tuesday June 17th, 2014 at 10:50 am until Wednesday June 18th, 2014 at 10:40 am. Fig. 11(b) shows the day with the best prediction obtained with DecisionTreeRegression, which has resulted in an MRE of 1,1877%, corresponding to the 24 hours from Wednesday January 21st, 2015 at 10:50 am to Thursday January 22nd, 2015 at 10:40 am. Fig. 11(c) shows the day with the best prediction obtained with the GBT ensemble technique, corresponding to an MRE of 1,0656%, between Wednesday July

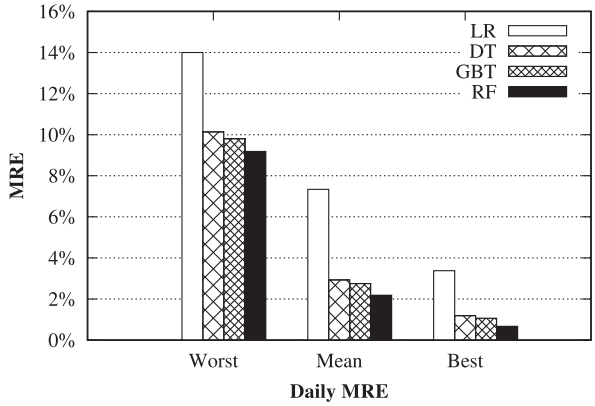
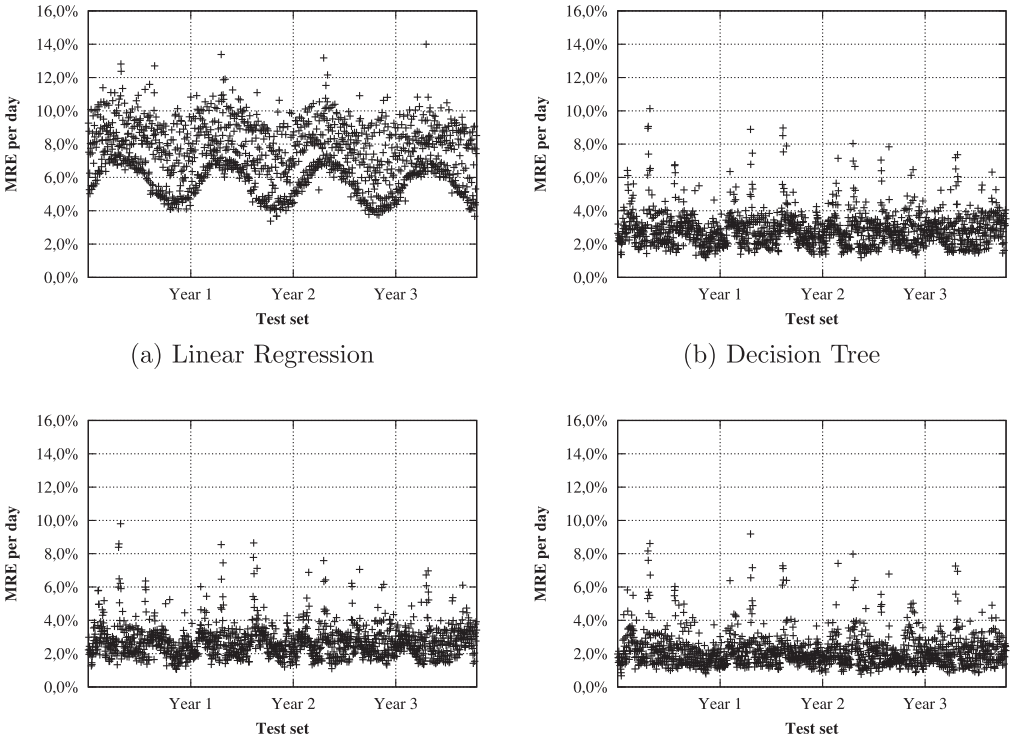


Fig. 9. Errors of worst and best predicted days at test set.



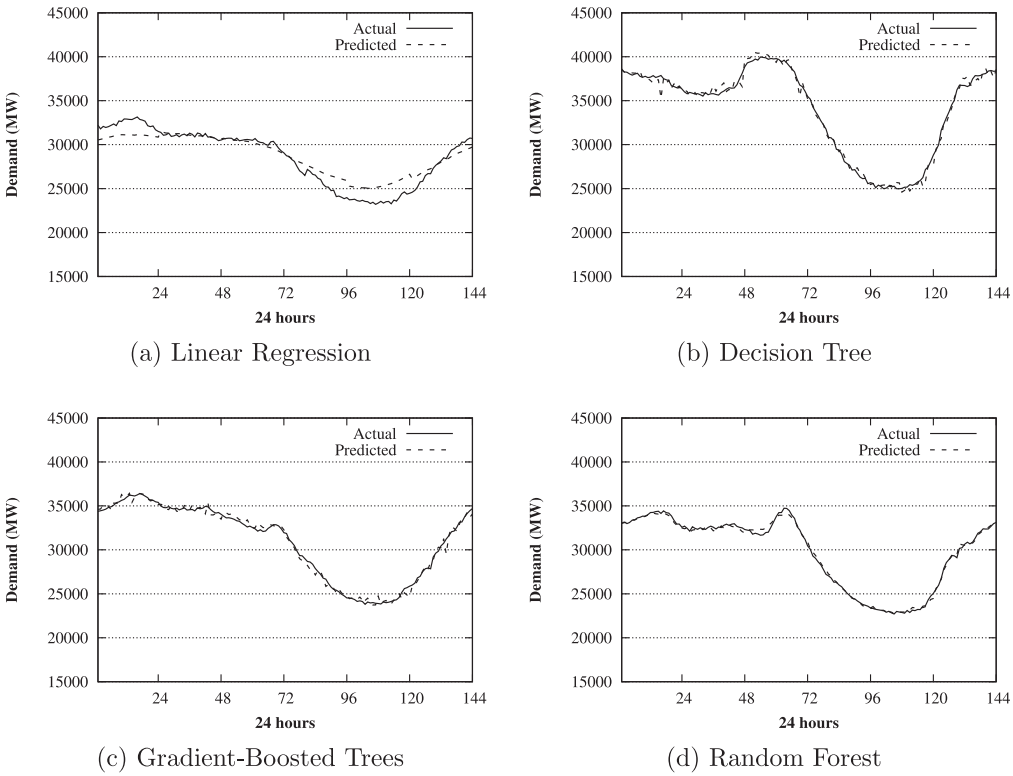
(a) Linear Regression

(b) Decision Tree

(c) Gradient-Boosted Trees

(d) Random Forest

Fig. 10. Daily MRE at the test set.



(a) Linear Regression (b) Decision Tree

(c) Gradient-Boosted Trees (d) Random Forest

Fig. 11. Day for the best prediction.

17th, 2013 at 10:50 am and Thursday July 18th, 2013 at 10:40 am. Fig. 11(d) shows the best predicted day obtained with Random Forest, corresponding to an MRE of 0,6745%, between Wednesday September 19th, 2012 at 10:50 am and Thursday September 20th, 2012 at 10:40 am. The lowest daily error in the test set corresponds to Random Forest.

The relative error assumed for each best predicted day is shown in Fig. 12. The highest daily error was obtained using Linear Regression and the lowest daily error in the test set corresponds to Random Forest.

In addition, the worst daily predictions for each of the methods are shown graphically in Fig. 13.

Fig. 13(a) shows the day with the worst prediction obtained using the linear regression, resulting in an MRE of 14,0004%, corresponding to the measurements from Wednesday December 23rd, 2015 at 10:50 am hours until Thursday December 24th, 2015 at 10:40 am. In this particular case, it corresponds to a special day within the month of December. Fig. 13(b) shows the worst prediction obtained with the DecisionTreeRegression method of MLlib. The error obtained is 10,1348% corresponding to the interval from Sunday December 30th 2012 at 10:50 am until Monday December 31st, 2012 at 10:40 am. Similarly to linear regression, it is a special day within the period of Christmas. Fig. 13(c) shows the day with the worst prediction obtained with the GBT ensemble technique, which has resulted in an MRE of 9,7966%, corresponding to the 24 h included from Sunday December 30th, 2012 at 10:50 am until Monday December 31st, 2012 at 10:40 am. Fig. 13(d) shows the day with the worst prediction obtained using Random Forest, which has resulted in an MRE of 9,1872%, between Monday December 23rd, 2013 at 10:50 am and Tuesday December 24th 2013 at 10:40 am.

In addition, it is important to observe the worst predictions since they contribute to the average increase in errors.

Table 9 shows a summary of the days in which the largest daily error is obtained for each of the algorithms analysed. In all cases, they correspond to very special days during the holiday season.

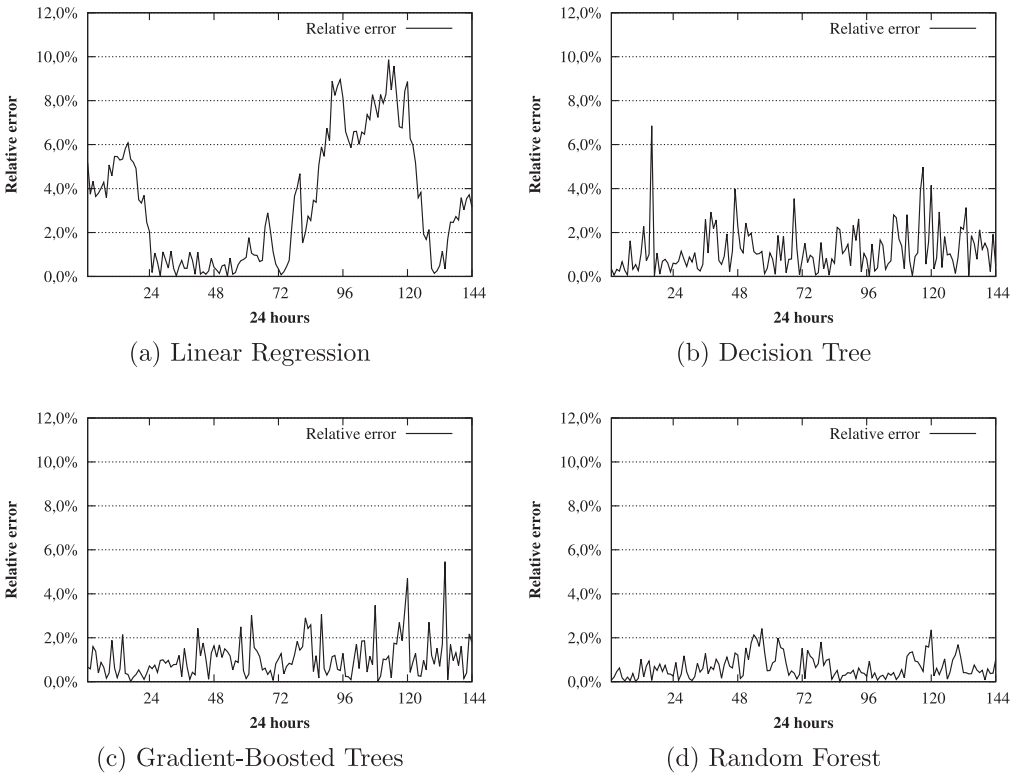


Fig. 12. Relative error corresponding to each best predicted day.

Table 9
Days with the worst predictions.

	From	To	MRE (%)
LR	X. 2015-12-23 10:50	J. 2015-12-24 10:40	14,0004
DT	D. 2012-12-30 10:50	L. 2012-12-31 10:40	10,1348
GBT	D. 2012-12-30 10:50	L. 2012-12-31 10:40	9,7966
RF	L. 2013-12-23 10:50	M. 2013-12-24 10:40	9,1872

4.5. Scalability analysis

Having studied the precision of the models generated by the different algorithms, this next section analyses the scalability of the proposed methodology. On the one hand, the influence of multiple threads in the generation of models is considered. On the other hand, the length of the time series is increased, multiplying its length by up to 32 times. These tests are performed with the configuration of the algorithms that have given rise to lowest errors, considering the number of attributes $w = 144$ and prediction horizon $h = 24$.

4.5.1. Computing resources remarks

To verify how scalable the various methods are according to available computing resources, the four algorithms are analysed when the number of computing threads varies from 1 to 8 and when the length of the time series is the original length and when the length is multiplied by 2, 4, 8, 16 and 32 (x1, x2, x4, x8, x16, x32, respectively). Only one slave has been used to obtain these results. Table 11 shows a summary of the sizes of the time series.

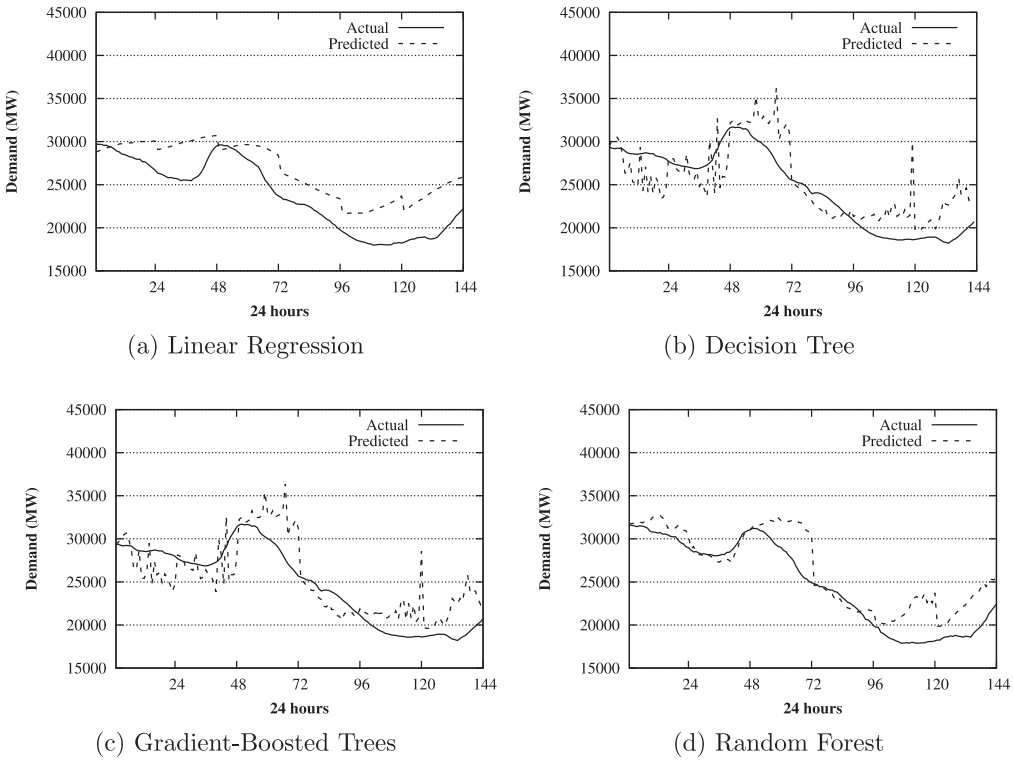


Fig. 13. Day for the worst prediction.

The time series with initial length –x1– has 497832 measurements, corresponding to 20742 records in the dataset and with a size of 5,70 MiB. As shown in Table 11, multiplying the length of the time series (twice –x2– until thirty two times –x32–) the length grows up to 15930624 measurements, corresponding to 663744 instances in the dataset and with a size of 18,230 MiB.

The results obtained for all methods using different time series length for time scalability analysis are shown in Table 10, where results are expressed in seconds. The algorithms analysed train their models in less time as availability of computing resources is increased. In addition, there is a dependence observed, related with the length of the time series. The algorithms are more sensitive to the increment in the number of threads; that is, the greater the scalability of the algorithms, the longer the length of the time series. However, the decrease in computing time differs very little when increasing from 4 to 8 threads for all algorithms.

In Fig. 14, the behaviour of each algorithm is represented, as the size of the time series and the number of processing threads increases. It also shows the reduction in runtime required to generate the model, when the Spark worker increases the number of processing threads. However, the decrease in computing time differs very little for all algorithms when increasing from 4 to 8 threads. Regardless of the algorithm used, this time reduction becomes more noticeable for longer time series, since with the original dataset x1, the time is reduced. This behaviour shows a clear dependence on the size of the time series, since Spark is designed to process sets of data of the order of gigabytes, and therefore, the greater the scalability of the algorithms the greater the length of the time series.

4.5.2. Data size remarks

Runtime has been obtained for the time series x2, x4, x8, x16 and x32, whose sizes are summarised in Table 11, respect a length multiplier, using one master and four slaves.

Table 10
Time scalability for all methods using different time series length.

Multiplier	Threads	LR	DT	GBT	RF
x1	1	722	165	765	815
	2	574	114	523	462
	3	522	98	443	381
	4	519	93	417	351
	5	487	88	387	317
	6	513	86	378	307
	7	518	86	379	302
	8	521	85	376	298
x2	1	1412	253	1195	1328
	2	1024	169	785	737
	3	964	147	679	632
	4	936	140	647	575
	5	933	138	634	555
	6	882	131	600	531
	7	877	130	593	521
	8	875	130	585	521
x4	1	2844	433	2063	2315
	2	1771	264	1242	1268
	3	1553	223	1044	1082
	4	1527	211	996	989
	5	1558	211	1002	942
	6	1465	201	939	905
	7	1461	199	929	890
	8	1462	199	924	895
x8	1	5584	799	3798	4303
	2	3523	495	2351	2376
	3	2742	378	1785	1978
	4	2705	356	1693	1794
	5	2655	346	1647	1714
	6	2633	340	1617	1643
	7	2653	339	1615	1618
	8	2621	336	1601	1620
x16	1	10552	1457	6970	11082
	2	5703	809	3853	5915
	3	5037	667	3196	4798
	4	4990	640	3122	4305
	5	4985	633	3024	3977
	6	4987	634	2997	3731
	7	5058	631	2960	3585
	8	5054	639	3007	3268
x32	1	21062	2891	6970	21495
	2	11563	1589	3853	11445
	3	10444	1391	3196	9387
	4	9870	1271	3122	8470
	5	9850	1238	3024	7899
	6	9862	1241	2997	7446
	7	10376	1275	2960	7017
	8	9791	1222	3007	6772

Table 12 shows the training time with respect to the different lengths of the time series for all proposed algorithms. This information is shown graphically in Fig. 15(a) and Fig. 15(b). The training time increases linearly as the length of the time series increases exponentially, which indicates the good behaviour of all methods with regard to scalability.

A scalability factor can be expressed as:

$$Factor_i = \frac{t_i}{t_{i/2}}, \tag{4}$$

where t_i is the training time for the time series of length x_i with $i = 2, 4, 8, 16$ and 32.

Fig. 16 shows the scalability factor of each method when the length of the time series increases by multiplying by 2, 4, 8, 16 and 32. The scalability factor is usually less than 2, which implies that scalability is even better than linear scalability.

5. Conclusions

In this work, a formal formulation is proposed to obtain multi-pass predictions using the MLlib library of the Apache Spark framework. The use of this framework guarantees that the applied methods to predict the energy consumption of the

Table 11
Size of the time series and dataset.

	Length of series	Number of instances	Size (MiB)
x1	497832	20742	5,70
x2	995664	41484	11,39
x4	1991328	82968	22,79
x8	3982656	165936	45,58
x16	7965312	331872	91,15
x32	15930624	663744	18230

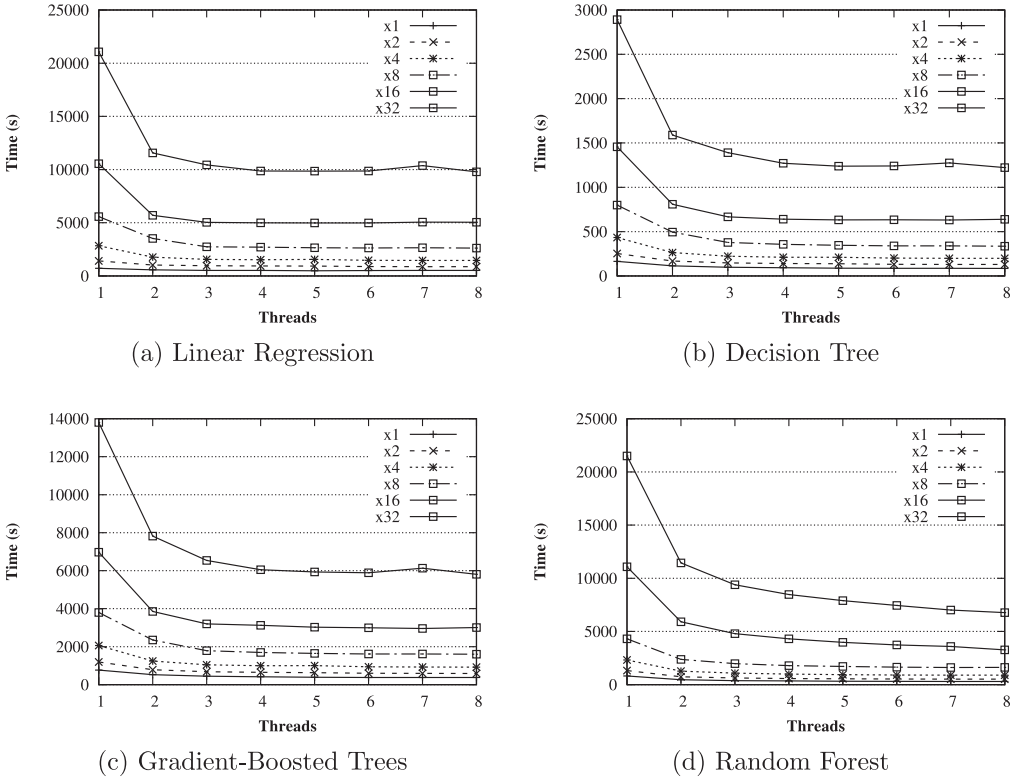


Fig. 14. Scalability of training time.

Table 12
Execution time scalability.

	x1	x2	x4	x8	x16	x32
LR	503	807	1381	2541	4859	9920
DT	72	119	196	342	632	1201
GBT	358	559	939	1671	3161	6046
RF	253	414	749	1456	2779	5935

following 24 values are scalable, and that, consequently, they can be used for long time series. A set of regression models, linear and nonlinear, such as linear regression, decision trees and two tree ensembling techniques, has been selected. The results of the prediction of electricity in the Spanish electricity market are giving with errors of approximately 2%. Likewise, experiments have been carried out showing the degree of scalability of each of the methods, concluding the viability of the methodology for the prediction of large time series.

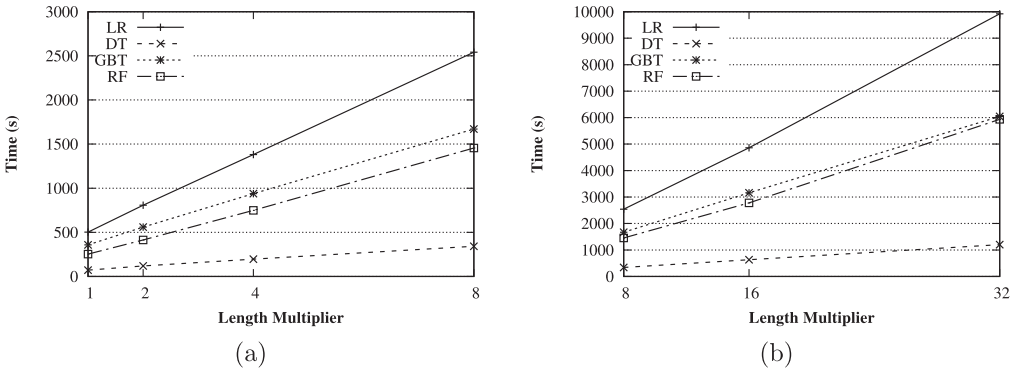


Fig. 15. Runtime and scalability for all algorithms.

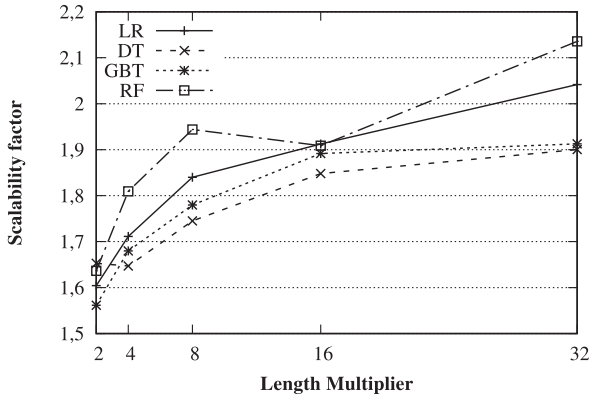


Fig. 16. Scalability factor behaviour.

One proposal for future research is to optimise the error with a validation set. Further studies should also analyse how the number of partitions into which the dataset is distributed affects the scalability of the algorithms. In addition, it would be very interesting to study the periodicity of the time series and its influence on the prediction model generated in the training. Finally, the behaviour of the methods must be verified with other datasets of larger sizes and different natures.

Acknowledgment

The authors would like to thank the Spanish Ministry of Economy and Competitiveness and Junta de Andalucía for the support under projects TIN2014-55894-C2-R and P12-TIC-1728, respectively. Additionally, the authors want to express their gratitude to the T-Systems Iberia company since all experiments were carried out on its Open Telekom Cloud Platform based on the Open-Stack open source.

References

[1] A. Abraham, B. Nath, A neuro-Fuzzy approach for forecasting electricity demand in victoria, Appl. Soft Comput. J. 1 (2) (2001) 127–138.
[2] G. Box, G. Jenkins, Time Series Analysis: Forecasting and Control, John Wiley and Sons, 2008.
[3] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[4] M. Capó, A. Pérez, J.A. Lozano, A recursive K-means initialization algorithm for massive data, in: Proceedings of the Spanish Association for Artificial Intelligence, 2015, pp. 929–938.
[5] G. Cavallaro, M. Riedel, M. Richerzhagen, J.A. Benediktsson, A. Plaza, On understanding big data impacts in remotely sensed image classification using support vector machine methods, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (2015) 4634–4646.
[6] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[7] R. Ding, Q. Wang, Y. Dan, Q. Fu, H. Zhang, D. Zhang, Yading: fast clustering of large-scale time series data, in: Proceedings of the VLDB Endowment, 8, 2015, pp. 473–484.

[8] M. El-Telbany, F. El-Karmi, Short-term forecasting of jordanian electricity demand using particle swarm optimization, Electr. Power Syst. Res. 78 (2008) 425–433.
[9] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, A.Y. Zomaya, I. Khalil, F. Sebti, A. Bouras, A survey of clustering algorithms for big data: taxonomy & empirical analysis, IEEE Trans. Emerg. Top Comput. 5 (2014) 267–279.
[10] S. Fan, C. Mao, J. Zhang, L. Chen, Forecasting electricity demand by hybrid machine learning model, Lect. Notes Comput. Sci. 4233 (2006) 952–963.
[11] R.C. García, J. Contreras, M. van Akkeren, J.B.C. García, A GARCH forecasting model to predict day-ahead electricity prices, IEEE Trans. Power Syst. 20 (2) (2005) 867–874.
[12] S. Ghosh, A. Das, Short-run electricity demand forecasts in maharashtra, Appl. Econ. 34 (8) (2002) 1055–1059.
[13] H.S. Guirguis, F.A. Felder, Further advances in forecasting day-ahead electricity prices using time series models, KIEE Int. Trans. PE 4-A (3) (2004) 159–166.
[14] Y. Guo, D. Niu, Y. Chen, Support-vector machine model in electricity load forecasting, in: Proceedings of the International Conference on Machine Learning and Cybernetics, 2006, pp. 2892–2896.
[15] M. Hamstra, H. Karau, M. Zaharia, A. Knwinski, P. Wendell, Learning Spark: Lightning-Fast Big Analysis, O' Really Media, 2015.
[16] I. Koprinska, M. Rana, A. Troncoso, F. Martínez-Álvarez, Combining pattern sequence similarity with neural networks for forecasting electricity demand time series, in: Proceedings of the International Joint Conference on Neural Networks, 2013, pp. 940–947.
[17] L. Li, S. Bagheri, H. Goote, A. Hassan, G. Hazard, Risk adjustement of patient expenditures: a big data analytics approach, in: Proceedings of the IEEE International Conference on Big Data, 2013, pp. 12–14.
[18] J.M. Luna-Romera, M. Martínez-Ballesteros, J. García-Gutierrez, J.C. Riquelme, An Approach to Silhouette and Dunn Clustering Indices Applied to Big Data in Spark, in: Proceedings of the Conference of the Spanish Association for Artificial Intelligence, 2016, pp. 160–169.
[19] Machine Learning Library (MLlib) for Apache Spark, On-line, http://spark.apache.org/docs/latest/mllib-guide.html (2016).
[20] J. Maillo, S. Ramírez, I. Triguero, F. Herrera, KNN-IS: an iterative spark-based design of the k-Nearest neighbors classifier for big data, Knowl. Based Syst. 117 (2017) 3–15.
[21] P. Malo, A. Kanto, Evaluating multivariate GARCH models in the nordic electricity markets, Commun. Stat Simul. Comput. 35 (1) (2006) 117–148.
[22] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, J.C. Riquelme, A survey on data mining techniques applied to electricity-related time series forecasting, Energies 8 (11) (2015) 13162–13193.
[23] F. Martínez-Álvarez, A. Troncoso, J.C. Riquelme, J.S. Aguilar, Energy time series forecasting based on pattern sequence similarity, IEEE Trans. Knowl. Data Eng. 23 (2011) 1230–1243.
[24] L. Mason, J. Baxter, P. Bartlett, M. Frean, Boosting algorithms as gradient descent, in: Proceedings of the Neural Information Processing Systems Conference, NIPS, 1999, pp. 512–518.
[25] P.F. Pai, W.C. Hong, Support vector machines with simulated annealing algorithms in electricity load forecasting, Energy Convers. Manag. 46 (17) (2005) 2669–2688.
[26] B. Panda, J.S. Herbach, S. Basu, R.J. Bayardo, PLANET: massively parallel learning of tree ensembles with MapReduce, in: Proceedings of the International Conference in Very Large Data Bases, 2009, pp. 1426–1437.
[27] R. Pérez-Chacón, R. Talavera-Llames, F. Martínez-Álvarez, A. Troncoso, Finding electric energy consumption patterns in big time series data, in: Proceedings of the International Conference on Distributed Computing and Artificial Intelligence, 2016, pp. 231–238.
[28] M. Rana, I. Koprinska, A. Troncoso, V.G. Agelidis, Extended Weighted Nearest Neighbor for Electricity Load Forecasting, Springer International Publishing, pp. 299–307.
[29] J.L. Reyes-Ortiz, L. Oneto, D. Anguita, Big data analytics in the cloud: spark on hadoop vs MPI/OpenMP on beowulf, Procedia Comput. Sci. 53 (2015) 121–130.
[30] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers - a survey, IEEE Trans. Syst. Man Cybern. Part C 35 (4) (2005) 476–487.
[31] R. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, A nearest neighbours-based algorithm for big time series data forecasting, in: Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, 2016, pp. 174–185.
[32] J. Taylor, Density forecasting for the efficient balancing of the generation and consumption of electricity, Int. J. Forecast 22 (2006) 707–724.
[33] Time Series for Spark (The spark-ts Package), On-line, https://github.com/sryza/spark-timeseries (2017).
[34] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: a mapreduce solution for prototype reduction in big data classification, Neurocomputing 150 (2015) 331–345.
[35] A. Troncoso, J.C. Riquelme, J.M. Riquelme, J.L. Martínez, A. Gómez, Electricity market price forecasting based on weighted nearest neighbours techniques, IEEE Trans. Power Syst. 22 (3) (2007) 1294–1301.
[36] C.-W. Tsai, C.-F. Lai, H.-C. Chao, A. Vasilakos, Big data analytics: a survey 2 (1) (2015) 21.
[37] T. White, Hadoop, The Definitive Guide, O' Really Media, 2012.
[38] I.H. Witten, E. Frank, M.A. Hall, C.J. Pal, Data mining: Practical Machine Learning Tools and Techniques, fourth ed., Morgan Kaufmann, Burlington, MA, 2016.
[39] W. Zhao, H. Ma, Q. He, Parallel k-means clustering based on mapreduce, Lect. Notes Comput. Sci. 5391 (2009) 674–679.
[40] L. Zhou, S. Pan, J. Wang, A.V. Vasilakos, Machine learning on big data: opportunities and challenges, Neurocomputing (2017). In press.

5.3. Integrated Computer - Aided Engineering

Integrated Computer-Aided Engineering (ICAE) fue fundada en 1993. El enfoque del ICAE es la integración de tecnologías informáticas y de la información de vanguardia y emergentes para la solución innovadora de problemas de ingeniería. La revista fomenta la investigación interdisciplinaria, como se señala en el número inaugural de la revista, "Basado en la premisa de que el pensamiento interdisciplinario y la colaboración sinérgica de las disciplinas pueden resolver problemas complejos, abrir nuevas fronteras y conducir a verdaderas innovaciones y avances, la piedra angular de la competitividad industrial y el avance de la sociedad".

Integrated Computer-Aided Engineering	
ISSN / eISSN	1069-2509 / 1875-8835
Editorial	IOS Press, Netherlands
Categoría	Computer Science - Artificial Intelligence
Idioma	Inglés
Frecuencia	4 números al año

InCites Journal Citation Reports de 2017	
Nuevas publicaciones	26
Citas totales	587
Factor de impacto (F.I.)	3,667
F.I. sin autocitas	2,647
F.I. últimos 5 años	2,689

- [9] J.F. Torres, A. Galicia, A. Troncoso, and F. Martínez-Álvarez. «A scalable approach based on deep learning for big data time series forecasting». *Integrated Computer-Aided Engineering* 25 (2018), pp. 1–14. DOI: 10.3233/ICA-180580. IF: 3,667 (21/132) Computer Science - Artificial Intelligence Q1

A novel scalable approach based on deep learning for big data time series forecasting

J. F. Torres [1], A. Galicia, A. Troncoso, F. Martínez-Álvarez
Division of Computer Science, Universidad Pablo de Olavide, ES-41013 Seville, Spain.
{jftormal, agalde}@alu.upo.es, {ali, fmaralv}@upo.es

Abstract. This paper presents a novel method based on deep learning to deal with big data times series forecasting. The deep feed forward neural network provided by the H2O big data analysis framework has been used along with the Apache Spark platform for distributed computing. Since H2O does not allow the conduction of multi-step regression, a general-purpose methodology that can be used for arbitrary prediction horizons is here proposed. The solution consists in splitting the problem into h forecasting subproblems, being h the number of samples to be simultaneously predicted. Thus, the best prediction model for each subproblem can be obtained, making easier its parallelization and adaptation to the big data context. Moreover, a grid search is carried out to obtain the optimal hyperparameters of the deep learning. Results from a real-world dataset composed of electricity consumption in Spain, with a ten minute frequency sampling rate, from 2007 to 2016 are reported. In particular, the accuracy and runtimes versus computing resources and size of the dataset are analyzed. Finally, the performance and the scalability of the proposed method is compared to other recently published techniques, showing to be a suitable method to process big data time series.

Keywords. Deep learning, time series forecasting, big data.

1. Introduction

Increasing attention is being paid to the issue of time series forecasting nowadays [26], mainly due to its interdisciplinary nature. Almost all scientific disciplines consist of data sampled over time, which makes their forecasting a task of utmost significance and complexity. Participants in electricity markets (both demand and prices) are particularly interested in making accurate predictions [18], since their obtention is critical for many areas in order to increase benefits, such as planning, inventory management, or even in evaluating capacity needs.

When addressing big data problems, computational issues are usually encountered. Therefore, efficient algorithms must be developed to extract knowledge from massive data in a timely manner. Additionally, many artificial intelligence techniques have been

[1]Corresponding Author: J. F. Torres, Pablo de Olavide University of Seville, Ctra. Utrera, Km.1, 41013, Sevilla, Spain; Voice: +34 605 03 57 59; E-mail: jftormal@alu.upo.es.

inspired by the functioning of neural systems [7] and are currently reporting outstanding results in this research field [9,13].

For all the aforementioned, a novel algorithm to forecast big data time series, based on deep learning architectures [11,27] is introduced in this work. Deep learning is an emerging branch of machine learning that extends artificial neural networks. One of the main drawbacks that classical artificial neural networks exhibit is that, with many layers, its training typically becomes too complex [17]. In this sense, deep learning consists of a set of learning algorithms to train artificial neural networks with a large number of hidden layers. Deep learning models are also sensitive to initialization and much attention must be paid at this stage [28].

The algorithm has been developed for arbitrary prediction horizons, being suitable for the short, mid, and long-term forecasting. To achieve this goal, the proposed approach creates as many independent forecasting problems as many samples are desired to be simultaneously forecasted. Later, each subproblem is individually addressed by computing different time slots within the historical data. Deep learning models have been incrusted in the process and are responsible for making prediction. It is worth noting that the deep learning implementation used is that of the well-known H2O library [3], which is open source and has been conceived for distributed environments.

One of the most relevant features of this method lies in its inherent suitability to be launched in parallel environments, which turns this tool ready to be applied to big data. Moreover, Apache Spark has been used to load data in memory, significantly thus speeding up the whole process and thus decreasing the computation time.

The performance of the approach has been assessed in real-world datasets. Electricity consumption in Spain has been used as case study, and data from 2007 to 2016 in the usual 70%-30% training-test sets structure have been analyzed. Satisfactory results are reported in terms of both accuracy and processing time, outperforming those obtained by a linear regression, a decision tree and two ensemble techniques based on trees as Gradient-Boosted Trees, and Random Forest. A scalability analysis has also been conducted in order to show that the proposed method is fully suitable to big data.

The remainder of the paper is structured as follows. Relevant related works are reviewed and discussed in Section 2. The proposed methodology is introduced in Section 3. Results are reported and discussed in Section 4. A comparative analysis to other well established forecasting strategies is shown in Section 5. Finally, the conclusions drawn are summarized in Section 6.

2. Related work

This section reviews relevant works in the context of big data, time series forecasting and deep learning. It also pays attention to works particularly devoted to forecast electricity demand.

Large datasets needs high performance hardware to be processed. Distributed computing can be used to leverage the existing hardware. In this sense, Castillo et al. [4] introduced a novel approach, in which a SVM model was distributed. The authors emphasize that threads shared some data with each other during the training phase to enhance the learning process.

The scalability of association rules techniques combined with evolutionary computation has also been used addressed. The authors in [19] claimed to have developed a

method particularly suitable to be applied to large datasets. Reported results are quite satisfactory and its use is encouraged for future works.

Recently, some studies have appeared discussing about the performance associated with deep learning and the context of forecasting. In 2013, the temperature forecasting issue was analyzed in [25]. The authors paid particular attention to the hyperparameters of deep learning architectures and provided some clues on how to systematically adjust them.

Event driven stock market was also forecasted by means of a novel approach in 2015 [8]. Firstly, a deep convolutional neural network was used and, secondly, both short and long-term stock price fluctuations were modeled. Results were assessed on S&P 500 stock historical data, showing remarkable performance.

Dalto et al. [6] thoroughly reviewed the selection of variables in order to decrease computational time. As a result of their work, they were able to develop a deep learning based forecasting approach with better accuracy than that of compared standard artificial neural networks.

An interesting deep learning architecture, this time particularly designed for air quality prediction, was presented in [16]. Specially remarkable were the spatio-temporal correlations analyzed by means of a stacked autoencoder model for feature extraction that the authors used. The experimentation carried out and the comparisons made were useful to show how promising the approach is.

Later in 2016, another feature data based method was introduced in [1]. The application field was transportation forecast under data-driven perspective. Namely, a deep learning model to forecast bus ridership at the stop and stop-to-stop levels was there adopted.

Deep learning methods have also been used in the field of health. A remarkable approach can be found in [23], in which the authors introduced a new deep learning approach based on voting schemes, with application to accurate early diagnose of Alzheimer cases.

Finally, some works relating to electricity demand forecasting are also discussed in this section. In 2014, a hybrid method was presented a method combined in order to forecast time series [15]. In particular, the authors combined Hinton and Salakhutdinov's networks with gradient descend and back propagation, as well as integrating some other preprocessing techniques.

Hu et al. [14] proposed a novel neural network GM based model to forecast electricity consumption. Turkish Ministry of Energy and Natural Resources and the AsiaŰ-Pacific Economic Cooperation energy database data were used with the purpose of evaluating the quality of the approach

Marvuglia et al. [20] described a recurrent-neural-network-based model to forecast a time series with one hour as prediction horizon to evaluate the influence of the air-conditioning equipments.

Talavera et al. [29] proposed a forecasting algorithm, under the Apache Spark platform [30]. Data from the Spanish market were used to test the approach. Experimentation was conducted towards the successful application to big data time series. Preliminary reported results are of particular interest.

Also with data from the Spanish market, Pérez et al. extracted demand profiles by means of scalable k-means algorithm [24]. The authors claimed the usefulness of using this information as input into a subsequent stage in the forecasting process. Big data time

series were also used and profiles showed remarkable differences between working days and festivities and among seasons.

Large variations in consumption were analyzed in the work introduced in [12]. The authors deeply studied the influence that data size and temporal granularity may exhibit in such a context. The performance of the approach was assessed with data from Canada by means of different configurations of artificial neural networks and support vector regression, reporting promising results.

Finally, Mocanu et al. [22] proposed two new stochastic models based on artificial neural network to predict time series.

After reviewing all these works, it can be easily concluded that approach similar to the one proposed in this work has been developed so far. Although significant effort have been put into deep learning parameters' adjustment or some approaches have been designed to deal with big data time series, no one has been specifically developed with such a parallel implementation and in a big data environment as the one here introduced, to the authors' knowledge.

3. Methodology

The theoretical background in which this work is included is introduced in Section 3.1. Later, Section 3.2 introduces the proposed methodology itself.

3.1. Theoretical background

The research is included in the field of supervised learning, i.e. instances composing the dataset are already labeled. Specifically, it is a regression task where a numeric value, called class, is intended to be forecasted. However, temporal order must be kept since data are sampled over time. To infer a model, from a part of the labeled data well-known as training set, is required to make a prediction. This model can be obtained by means of many techniques, such as linear regression, regression trees, nearest neighbors, neural networks or support vector machines. Deep learning is here proposed to forecast in a big data environment.

Many network architectures for deep learning are available depending on the characteristics of the target problem. Each architecture is designed to be applied to a particular problem, and therefore, each one works in a different way. Some of these architectures can be recurrent networks, convolutional networks, Hopfield networks, Kohonen networks or feed forward networks. A deep feed forward architecture is applied to forecast long time series in this work.

Feed forward neural networks are the most common network architectures for solving forecasting problems. The main characteristic of this type of network is that each neuron is a basic element of processing. This network is defined by the weights, which represent the interactions between each pair of neurons. Both weights and network topology are computed in the training phase.

H2O is a open source platform designed to compute machine learning techniques into a single node or a cluster of machines in a distributed way, being scalable for big data projects. H2O can also be integrated with Apache Spark to store data in memory instead of in hard disk. This framework includes a deep feed forward neuronal network,

which has been used to forecast big data time series. The executions of this algorithm can be parameterized by a high number of parameters (known as hyperparameters) that will depend on the characteristics of the problem to be solved.

The most important parameters used in this study are described below:

- *Hidden.* All possible numbers of hidden layers and numbers of neurons per layer are provided through this parameter.
- *L1.* This parameter deals with the regularization to avoid overfitting, thus improving the generalization.
- *Epsilon* and *Rho.* These parameters are related to the learning rate and they are used to avoid to achieve a local optima. Default values are 1E-8 and 0.99, respectively.
- *Activation.* The activation function is used to model the type of relationship between inputs and outputs of the network. It has been set to the hyperbolic tangent.
- *Distribution.* This parameter represents the loss function to be minimized.
- *Stop metric.* It is the metric to be used for early stopping. The mean square error (MSE) was selected.
- *Stopping tolerance.* This parameter stops the training of the deep network if an improvement of the established value is not achieved. Its default value is 1E-3.
- *Stopping round.* If a moving average composed of the MSE of $stopping\_round$ models does not improve according to a given tolerance, then the deep learning algorithm stops. Its value by default is 5.

H2O allows the creation of a grid that generates all possible combinations according to the selected hyperparameters. Thus, it is possible to test several values of these parameters and generate a model for each combination. These models are sorted in ascending order according to the error, that is, from the best model to the worst model. A full description on how H2O works, in addition to all the parameters allowed by the deep learning algorithm, can be found in [5].

3.2. Description of the methodology

This section describes the methodology proposed to forecast time series using the deep learning approach from H2O framework, under R programming language. The main goal of this study is to predict h next values (hereinafter called prediction horizon) of a time series, expressed as $[x_1,...,x_t]$, from w previous values (hereinafter called historical data window). This process is also called multi-step regression, since more than one value has to be forecasted. A multi-step regression problem is illustrated in Figure 1.

Formally, this problem can be formulated as it is presented in Eq. (1), where the goal is to find the model f, after deep learning method application:

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = f(x_t, x_{t-1}, \ldots, x_{t-(w-1)}) \tag{1}$$

Unfortunately, the deep learning algorithm included in the H20 framework does not support multi-step forecasting. Therefore, a new methodology has to be developed to achieve this goal. A possible way consists in splitting the main problem into h forecasting subproblems, as showed in Figure 2.

This new methodology can be formulated by using h models, one for each forecasting subproblem, as shown in Eq. (2):

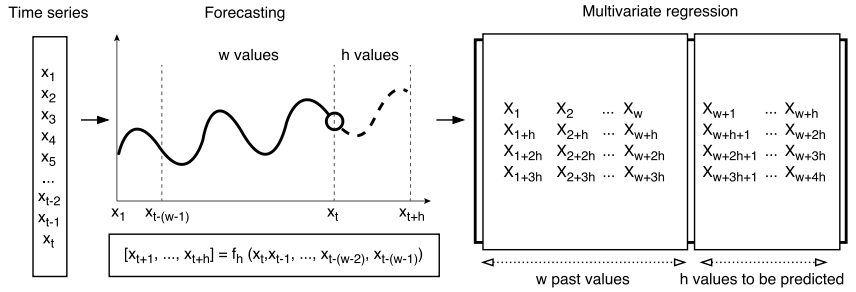


Figure 1. Multivariable forecasting problem

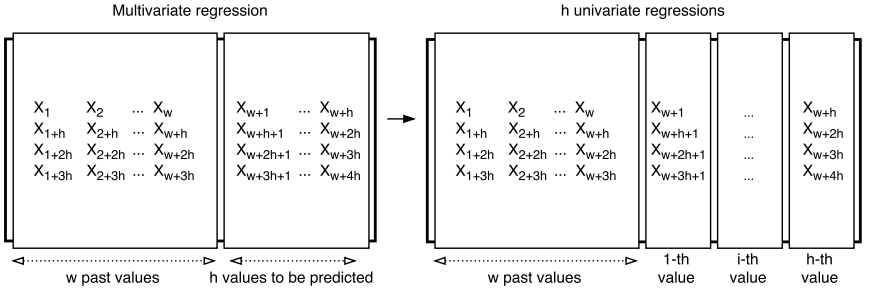


Figure 2. Transformation from multivariate to univariate problem

$$x_{t+1} = f_1(x_t, x_{t-1}, ..., x_{t-(w-1)}) \tag{2}$$

$$x_{t+2} = f_2(x_t, x_{t-1}, ..., x_{t-(w-1)}) \tag{3}$$

$$... \tag{4}$$

$$x_{t+h} = f_h(x_t, x_{t-1}, ..., x_{t-(w-1)}) \tag{5}$$

On the one hand, the relations between consecutive values of the time series are missed in this methodology, as the future value is not predicted using the w previous consecutive values. However, if the predictions of previous values were used to forecast, a greater error would be obtained, giving rise to a wrong prediction.

On the other hand, the obtention of h independent models entails a higher computational cost than building just one model to predict all h values. The deep learning method used in this work has an extra computational cost due to multiple models are computed, by combining different parameters in a grid search. However, since these models are independent, they can be easily parallelized.

A general scheme of the proposed methodology is illustrated in Figure 3.

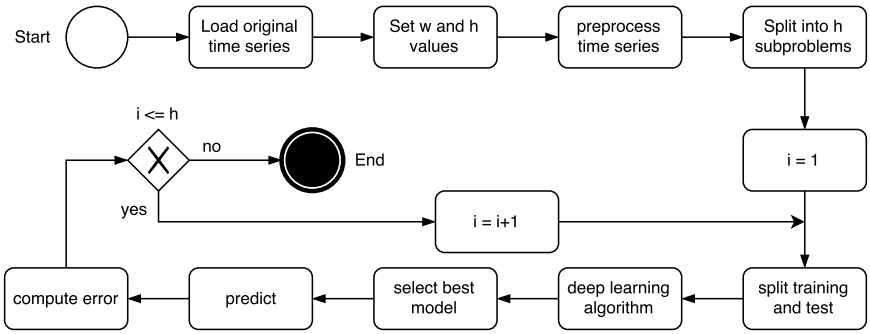


Figure 3. Scheme of the proposed methodology.

4. Results

This section presents the results obtained after applying the previously mentioned methodology to forecast the time series to be described in Section 4.1. Section 4.2 describes the experimental setup designed in order to obtain the optimal hyperparameters. After that, an analysis of the results has been presented in Section 4.3. Finally, Section 4.4 shows the scalability of the proposed deep learning method, providing the computational time of the algorithm for time series of different length and for different computing resources.

4.1. Dataset description

The time series considered in this study is related to the electricity consumption in Spain from January 2007 to June 2016. It is a time series of 9 years and 6 months with a high sampling frequency (10 minutes), resulting in 497832 measures in total into a 33 MB file.

This time series needs to be preprocessed to build a dataset of $w+h$ attributes, being w the number of past values used to forecast the h next values as it is shown in Figure 4. It can be noted that the number of instances of the final dataset can vary depending on w and h values. It is important to highlight that the $w+h$ value could not be multiple of the time series length, being removed in this case the incomplete instances.

The dataset was split into 70% for the training set and 30% for the test set, and in turn, a 30% from the training set has also been selected for the validation set in order to obtain the optimal parameters. The training set covers the period from January 1, 2007 at 00:00 to August 20, 2013 at 02:40. Therefore, the test set comprises the period from August 20, 2013 at 02:50 to June 21, 2016 at 23:40.

4.2. Design of experiments

The experimentation carried out is composed of two phases. First, the optimal parameters of the deep neural network will be calculated. Second, a scalability analysis will be performed using the optimal parameters found in the previous stage.

The different settings applied to make the experiments are as follows:

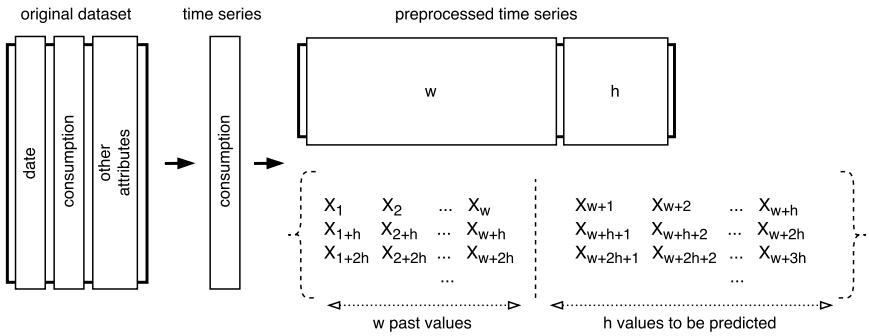


Figure 4. Preprocessing of the original dataset.

1. The *w* number of historical data has been set to 24, 48, 72, 96, 120, 144 and 168, corresponding to 4, 8, 12, 16, 20, 24 and 28 hours, respectively. After training and calculating the validation error for each value of *w*, the value providing the smallest error is selected for the rest of experiments. A value of 168 was finally obtained.
2. The *h* prediction horizon is set to 24, which represents a block of 4 hours to be predicted.
3. The number of hidden layers for applying the deep learning algorithm has been set from 1 to 5 layers and a number of neurons per layer varying from 10 to 100 by steps of 10.
4. The *lambda* regularization parameter is set to 0, 0.1, 0.01, 0.001 and 0.0001 values.
5. Gaussian and Poisson distribution functions have been tested.
6. The remaining deep learning parameters are set by default.

Once the neural network has been trained, the optimal parameters are chosen to analyze the scalability of the proposed deep learning. Information related to the scalability study is detailed below:

1. The size of the time series is increased, multiplying its length by up to 2, 4, 8, 16, 32 and 64 times.
2. The number of local threads is set to 2, 4, 6, 8, 10 and 12 to verify how scalable is the deep learning method according to computing resources.
3. The deep learning method is executed on a cluster of 2 machines, using a total of 24 threads, to check its scalability on distributed computing resources.
4. The scalability of the deep learning is compared to other scalable methods recently published in the literature [10].

The Root Mean Squared Error (RMSE) and the mean absolute error (MAE) have been computed to evaluate the accuracy of the models in the training. On the other hand, the mean relative error (MRE) in percentage has been used to calculate the accuracy of the best deep learning model in the test set. The formulation of these errors is shown below:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - a_i)^2} \tag{6}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i| \tag{7}$$

$$MRE = 100 \cdot \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - a_i|}{a_i} \tag{8}$$

where n, p and a mean the number of samples, predicted values and actual values, respectively.

The hardware used in order to obtain the results reported here has been an Intel Core i7-5820K at 3.3 GHz with 15 MB of cache, 12 cores and 16 GB of RAM memory, working under an Ubuntu 16.04 operating system. The H2O framework was used to apply deep learning by using R language. This framework has available a feed-forward architecture and allows to configure a cluster to launch distributed executions.

4.3. Analysis of results

This section discusses the results obtained by the deep learning algorithm with different hyperparameters described in Section 3.1 for the different configuration settings detailed in Section 4.2.

Table 1 shows the optimal number of neurons for each subproblem and the MRE obtained when varying the number of past values to be used to predict. The number of hidden layers in the net was set to 3, and the number of neurons per layer was varying from 10 to 100 by steps of 10. It can be concluded that 168 is the best window size.

Table 1. MRE depending on the historical data window.

w	neurons per layer and subproblem	MRE
24	[20 50 90 100 30 100 70 100 90 20 70 50 60 100 80 70 60 70 70 100 80 100 60 90]	3.7648
48	[50 60 100 40 80 20 90 30 90 90 100 70 100 100 70 80 50 40 20 80 100 100 100 70]	2.8904
72	[30 50 70 80 100 100 60 40 40 60 40 60 90 70 40 80 50 20 50 20 80 60 70 80]	2.7259
96	[100 80 40 70 60 90 40 60 40 70 20 30 70 100 60 100 60 70 50 40 90 80 50 60]	2.5588
120	[30 30 90 70 20 70 70 80 30 80 80 70 60 70 60 80 80 40 40 30 70 90 100 100]	2.4180
144	[50 80 50 70 60 80 30 80 50 70 60 40 100 40 90 90 90 40 70 40 80 70 90 90]	1.8722
168	[30 80 90 60 60 100 40 80 30 80 50 100 40 80 90 40 70 70 70 60 90 70 100 100]	1.8439

Table 2 summarizes the errors for the validation set when varying the lambda regularization parameter value and the distribution function. These errors are computed by averaging the errors obtained for each subproblem for the validation set. It can be observed that the best values were obtained when the regularization was not considered and for Gaussian distribution function, giving rise to a mean of 587.4677 for RMSE and 440.6434 for MAE. Therefore, the lambda parameter is set to 0 and the distribution function to Gaussian from now on.

Table 3 shows the optimal number of hidden layers and neurons for each subproblem along with the RMSE and MAE for the validation set. These values were internally calculated for each subproblem using a grid search available in H2O in order to compute

Table 2. Errors for different lambda regularization parameters and distribution functions.

Lambda	Distribution	RMSE	MAE
0	**Gaussian**	**587.4677**	**440.6434**
0.1	Gaussian	1526.148	1118.548
0.01	Gaussian	1177.051	812.4854
0.001	Gaussian	857.4803	620.0702
0.0001	Gaussian	636.4495	474.6989
0	Poisson	633.8448	478.203
0.1	Poisson	662.4093	498.6579
0.01	Poisson	637.8108	481.5656
0.001	Poisson	632.1003	477.292
0.0001	Poisson	630.3271	477.2203

the optimal hyperparameters. It can be seen that both RSME and MAE increase as the final of the prediction horizon draw nearer. The reason for this is caused by the existing gap between the last sample in the historical data and the next sample to be predicted.

From the optimal configuration of all parameters previously analyzed, the final value of MRE obtained when predicting the test set is **1.6769%**.

Table 3. Optimal number of neurons and hidden layers for each subproblem.

Subproblem	Hidden layers	Neurons per layer	RMSE	MAE
1	2	80	280.9748	223.3659
2	2	100	334.5473	255.9905
3	5	60	361.0928	279.0836
4	3	60	374.1559	283.35
5	3	80	431.9821	338.0297
6	2	60	457.2543	357.964
7	3	70	488.2656	364.8531
8	5	80	546.8644	415.1822
9	2	100	540.2944	410.5037
10	4	60	557.4836	415.8288
11	3	70	564.0067	424.5466
12	3	100	594.0841	441.9526
13	4	40	595.4264	457.06
14	5	70	648.6574	497.005
15	2	70	644.035	495.1685
16	2	70	667.3852	500.1515
17	4	50	674.7404	508.7588
18	4	80	669.1147	496.9713
19	4	90	698.5957	528.3096
20	4	50	708.2841	520.3575
21	4	90	778.9108	583.7202
22	2	80	799.798	569.1762
23	4	90	825.2674	591.3633
24	5	100	858.0038	616.7493

Figure 5 and Figure 6 present the evolution of actual and forecasted demand corresponding to the best and worst day, respectively, of the test set in terms of prediction accuracy. Note that a day is represented by 144 measures. These days correspond to August 5, 2014 at 02:50 as the best predicted day, and December 26, 2015 at 02:50 as the worst predicted day. It is noteworthy that the worst day is an unusual day, namely, the next day to the Christmas Day.

On the other hand, Figure 7 shows the predicted and actual daily consumption corresponding to the months of April and May in the year 2016. It can be appreciated that the deep learning provides an underestimation at peak times.

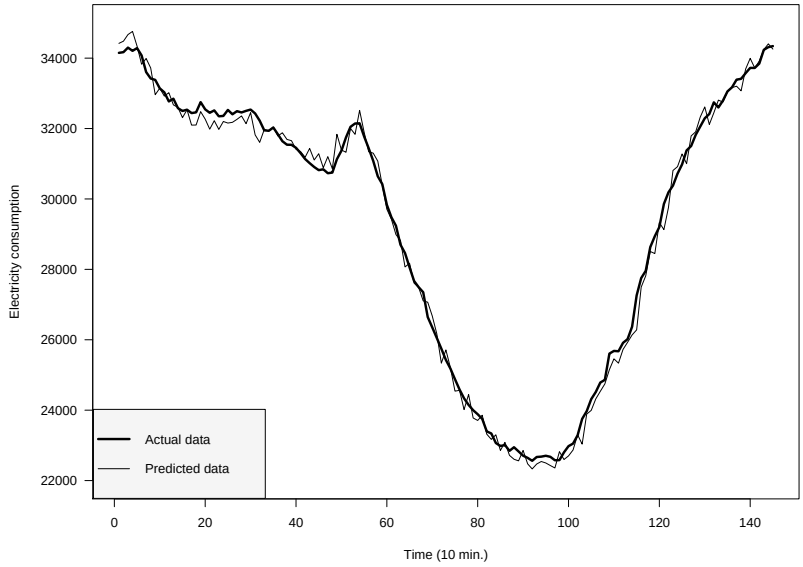


Figure 5. Best daily forecast in the test set.

4.4. Scalability

This section presents a study of scalability of the deep neural network proposed to predict very long time series. For that purpose, the deep learning algorithm has been executed for different lengths of the time series and number of execution threads.

Table 4 shows the computing times of the deep neural network for its training phase when varying the number of threads in a single machine from 2 to 12 by steps of 2, and the length of the series increases depending on a multiplicative factor. Thus, x2 stands for a factor of 2, and so forth. In particular, runtimes have been obtained with time series of two, four, eight, sixteen, thirty and two, and sixty and four times the length of the original time series. Figure 8 graphically summarizes the results collected from Table 4.

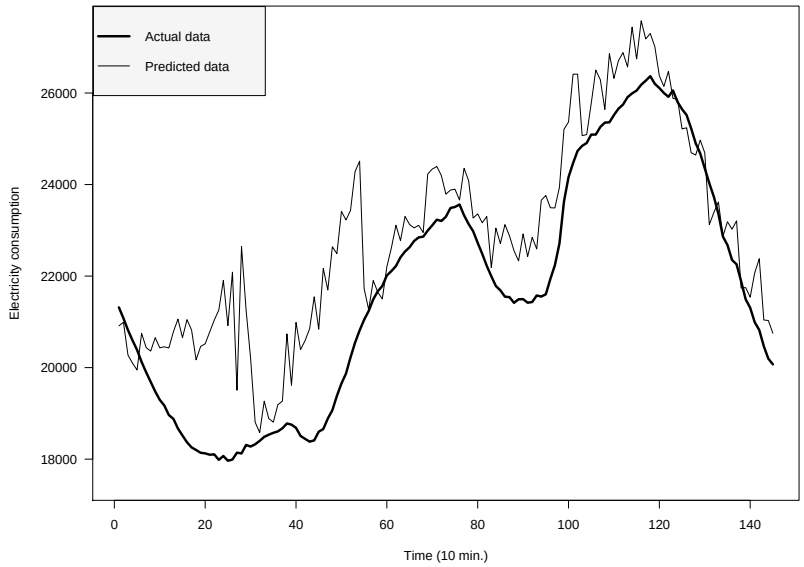


Figure 6. Worst daily forecast in the test set.

It is noticeable that the deep learning model here proposed for big data time series is scalable as the runtimes increase in a linear way when increasing the size of the dataset. Moreover, it can be seen that the optimal resources for the different sizes of the time series used in this experiment are 6 threads as similar runtimes are provided when using a larger number of threads.

Figures 9a and 9b present how the runtime in the training phase decreases as the number of threads in a single machine increases. This phenomenon happens independently of the dataset size, but some important issues can be concluded. For instance, the number of threads for a short time series (for instance x1) is not too relevant as the training computing time by using 6, 8, 10 or 12 threads does not show a great improvement. However, the reduction of runtimes is much more remarkable with very long time series (for instance x64) as it can be seen in Figure 9b.

5. Comparative analysis

The proposed deep learning based methodology has been compared to the methods reported in [10], namely, a linear regression (LR), a decision tree (DT) and two ensemble techniques based on trees as Gradient-Boosted Trees (GBT) and Random Forest (RF). Tree-based methods are very common in machine learning, both for classification and for regression, as they are easy to interpret, support continuous and discrete attributes,

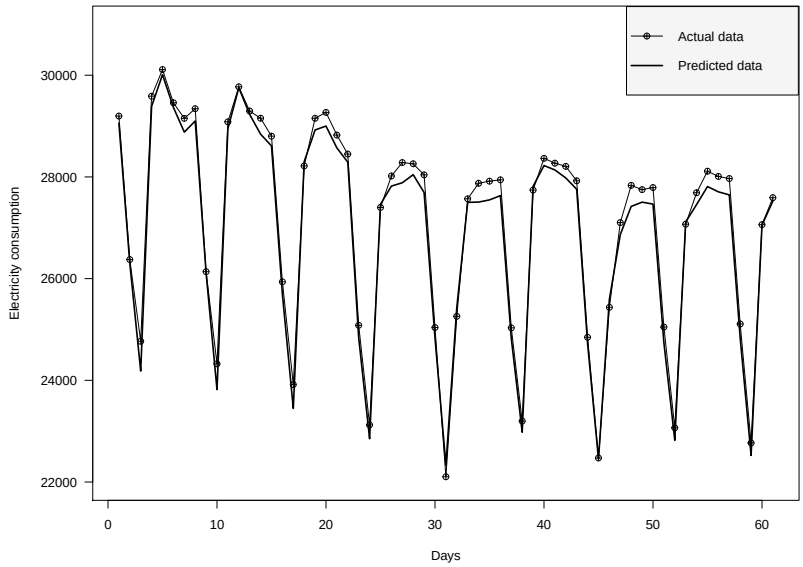


Figure 7. Daily average of the time series in April and May 2016.

do not require attribute scaling and are able to model nonlinear relationships between attributes. A brief description of these methods used for the comparison is made below.

LR minimizes the mean square error of the training set by using the well-known stochastic gradient descent method and is usually selected as a reference model.

DT is obtained through a recursive binary partition of the feature space. At each iteration, the attribute chosen to divide the tree is the one that maximizes the information gain. The recursive construction of the tree stops when there are not enough attributes in the child nodes or the maximum depth is reached.

Ensembles methods are learning algorithms that create a set of basic models to compose the final model. GBT and RF offer very good results for many real applications, showing a high performance in regression tasks and improving the results obtained by a single regression. Both training processes to generate the model are different for each algorithm. In particular, GBT [21] is a set of decision trees trained iteratively. Thus, in each iteration, the algorithm uses the ensemble of trees of the previous iteration to correct the mistakes made in the prediction, thereby improving the accuracy in the following ensemble of trees. On the other hand, RF [2] generates a set of decision trees in parallel. Combining them, the probability of obtaining an overfitted model is reduced. Also, a different training set is used in each tree in order to introduce randomness. In addition, the nodes of each decision tree consider different subsets of attributes. To predict a new instance, RF makes an estimation with the average of the predictions obtained with each tree.

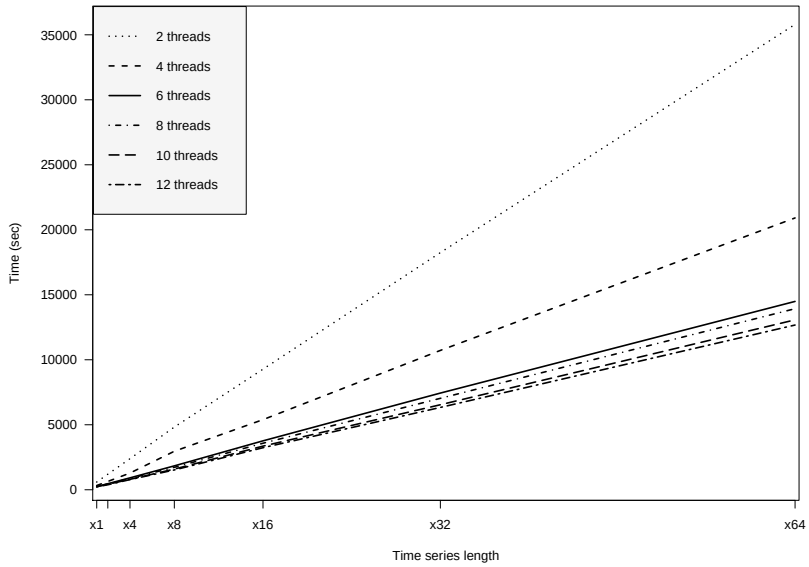


Figure 8. Computing times versus length of the time series.

The results obtained of the application of these methods to the time series described in the Section 4.1 were compared in [10], using an Apache Spark cluster with one master and two slaves with Intel Core i7-5820K @ 3.30GHz processors and 16GB of memory for each machine. A comparison between the accuracy and runtimes (in seconds) for the deep feed-forward neural network method proposed here by using the cluster described above and the results from [10] is shown in Table 5, where methods are ordered by prediction error for the test set. The deep learning achieves a MRE of 1.6769% for the test set, meaning an improvement of 0.52% compared to RF –the method with the best accuracy from [10]–, 1.20% compared to only one decision tree, and a 5.66% in comparison with the linear regression.

Table 6 and Figure 10 show a comparison of the training execution times –expressed in seconds– for different time series lengths in order to compare the scalability of the deep learning, LR, DT, GBT and RF. As can be seen in Table 6, the behavior of all methods is the same, keeping a linear scalability factor according to the time series length. Figure 10 represents graphically how training times increase according to the length of the time series. Both tree-emsemble methods improve execution times regarding the linear regression, but definitely deep learning and DT are at a different level, being DT the most scalable method of the comparison, followed closely by the deep learning method.

Table 4. Computing times for different time series lengths and number of threads.

Multiplier	File size	Threads	Training time(sec)
x1	23.9 Mb	2	595
		4	327
		6	244
		8	237
		10	232
		12	229
x2	47.8 Mb	2	1195
		4	639
		6	464
		8	449
		10	420
		12	384
x4	95.5 Mb	2	2389
		4	1284
		6	915
		8	872
		10	802
		12	782
x8	191.1 Mb	2	4823
		4	2961
		6	1837
		8	1725
		10	1590
		12	1524
x16	382.2 Mb	2	9276
		4	5394
		6	3763
		8	3579
		10	3356
		12	3235
x32	764.4 Mb	2	18244
		4	10719
		6	7438
		8	7034
		10	6540
		12	6333
x64	1.5 Gb	2	35802
		4	20911
		6	14489
		8	13929
		10	13071
		12	12673

6. Conclusions

A deep feed forward neural network applied to time series forecasting has been proposed in this work to deal with big data. The Apache Spark distributed computing platform has

Table 5. Comparison of accuracy and runtimes.

	MRE (%)	Time (s)
Deep Learning	**1.6769**	153
Linear Regression	7.3395	553
Decision Tree	2.8783	**81**
Gradient-Boosted Trees	2.7190	417
Random Forest	2.2005	277

Table 6. Runtimes (expressed in seconds) for different time series lengths.

	x1	x2	x4	x8	x16	x32	x64
Deep Learning	153	218	361	649	1209	2346	4601
Linear Regression	553	846	1483	2710	5162	10057	19871
Decision Tree	81	120	201	353	653	1329	2644
Gradient-Boosted Trees	277	440	783	1525	3128	6416	12518
Random Forest	417	581	968	1720	3336	6490	13141

been used to execute the algorithm in a cluster of machines. The H2O framework has been used for big data analysis, providing the deep learning method here proposed. Reported results have shown that the deep learning configuration setting is important to obtain a good accuracy. A preliminary study of several parameters has been made, obtaining a mean relative error less than a 2%. The scalability of the method has been assessed depending on the time series length and the number of execution threads, showing a linear scalability and a high performance for distributed computing. Finally, the methodology has been compared to other recently published techniques in terms of accuracy and scalability. The deep learning one turned out to be one of the most adequate methods to process big data time series along with decision trees, in terms of scalability, and the best method in terms of accuracy.

Acknowledgments

The authors would like to thank the Spanish Ministry of Economy and Competitiveness and Junta de Andalucía for the support under projects TIN2014- 55894-C2-R and P12-TIC-1728, respectively.

References

[1] J. Baek and K. Sohn. Deep-learning architectures to forecast bus ridership at the stop and stop-to-stop levels for dense and crowded bus networks. *Applied Artificial Intelligence*, 30(9):861–885, 2016.

[2] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] A. Candel, E. LeDell, V. Parmar, and A. Arora. *Deep learning with H2O*. H2O.ai, Inc., 2017.

[4] E. Castillo, D. Peteiro-Barral, B. Guijarro Berdiñas, and O. Fontenla-Romero. Distributed one-class support vector machine. *International Journal of Neural Systems*, 25(07):1550029, 2015.

[5] D. Cook. *Practical Machine Learning with H2O: Powerful, Scalable Techniques for Deep Learning and AI*. O'Reilly Media, 2016.

[6] M. Dalto, J. Matusko, and M. Vasak. Deep neural networks for ultra-short-term wind forecasting. In *Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, pages 1657–1663, 2015.

[7] A. Deleforge, F. Forbes, and R. Horaud. Acoustic Space Learning for Sound-Source Separation and Localization on Binaural Manifolds. *International Journal of Neural Systems*, 25(01):1440003, 2015.

[8] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2327–2334, 2015.

[9] F. Donnarumma, R. Prevete, F. Chersi, and G. Pezzulo. A programmer-interpreter neural network architecture for prefrontal cognitive control. *International Journal of Neural Systems*, 25(6):1–16, 2015.

[10] A. Galicia, J. F. Torres, F. Martínez-Álvarez, and A. Troncoso. Scalable forecasting techniques applied to big electricity time series. In *Proceedings of the 14th International Work-Conference on Artificial Neural Networks (IWANN)*, pages 165–175, 2017.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[12] K. Grolinger, A. L'Heureux, M. A.M. Capretz, and L. Seewald. Energy forecasting for event venues: Big data and prediction accuracy. *Energy and Buildings*, 112:222–233, 2016.

[13] T. Hirschauer, H. Adeli, and T. Buford. Computer-aided diagnosis of parkinson's disease using an enhanced probabilistic neural network. *Journal of Medical Systems*, 39(179):1–12, 2015.

[14] Y-C. Hu. Electricity consumption prediction using a neural-network-based grey forecasting approach. *Journal of the Operational Research Society*, 68(10):1259–1264, 2016.

[15] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi. Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing*, 137:47–56, 2014.

[16] X. Li, L. Peng, Y. Hu, J. Shao, and T. Chi. Deep learning architecture for air quality predictions. *Environmental Science and Pollution Research International*, 23:22408–22417, 2016.

[17] D.J. Livingstone, D.T. Manallack, and I.V. Tetko. Data modelling with neural networks: Advantages and limitations. *Journal of Computer-Aided Molecular Design*, 11:135–142, 1997.

[18] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, and J. C. Riquelme. A survey on data mining techniques applied to energy time series forecasting. *Energies*, 8:1–32, 2015.

[19] M. Martínez-Ballesteros, J. Bacardit, A. Troncoso, and J. C. Riquelme. Enhancing the scalability of a genetic algorithm to discover quantitative association rules in large-scale datasets. *Integrated Computer-Aided Engineering*, 22(01):21–39, 2015.

[20] A. Marvuglia and A. Messineo. Using recurrent artificial neural networks to forecast household electricity consumption. *Energy Procedia*, 14:45 – 55, 2012.

[21] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent. In *Proceedings of the Neural Information Processing Systems Conference (NIPS)*, pages 512–518, 1999.

[22] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling. Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6:91 – 99, 2016.

[23] A. Ortiz, J. Munilla, J. M. Górriz, and J. Ramírez. Ensembles of deep learning architectures for the early diagnosis of the alzheimer's disease. *International Journal of Neural Systems*, 26(07):1650025, 2016.

[24] R. Pérez-Chacón, R. L. Talavera-Llames, A. Troncoso, and F. Martínez-Álvarez. Finding electric energy consumption patterns in big time series data. In *Proceedings of the International Conference on Distributed Computing and Artificial Intelligence (DCAI)*, pages 231–238, 2016.

[25] P. Romeu, F. Zamora-Martínez, P. Botella-Rocamora, and J. Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In *Proceedings of the 23rd International Conference on Artificial Neural Networks (ICANN)*, pages 451–458, 2013.

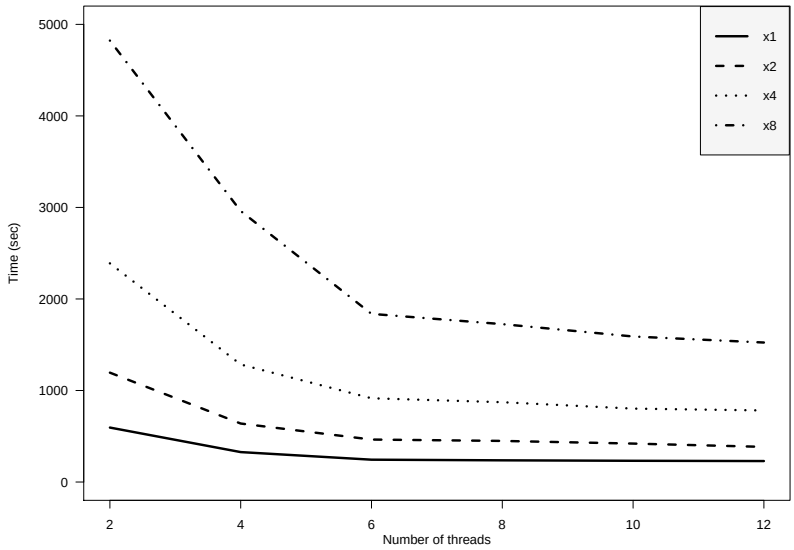
[26] J.L. Rosselló, M.L. Alomar, A. Morro, A. Oliver, and V. Canals. High-density liquid-state machine circuitry for time-series forecasting. *International Journal of Neural Systems*, 26(5):1–12, 2016.

[27] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

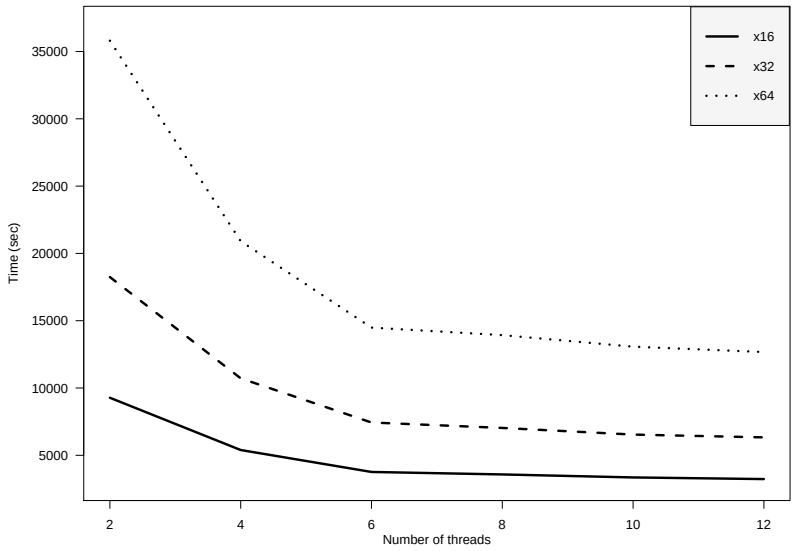
[28] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1139–1147, 2013.

[29] R. L. Talavera-Llames, R. Pérez-Chacón, M Martínez-Ballesteros, A. Troncoso, and F. Martínez-Álvarez. A nearest neighbours-based algorithm for big time series data forecasting. *Lecture Notes in Artificial Intelligence*, 9648:174–185, 2016.

[30] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with-Working Sets. In *Proceedings of the International Conference on Hot Topics in Cloud Computing (ICWS)*, pages 1–10, 2010.



(a)



(b)

Figure 9. Computing times depending on the number of threads.

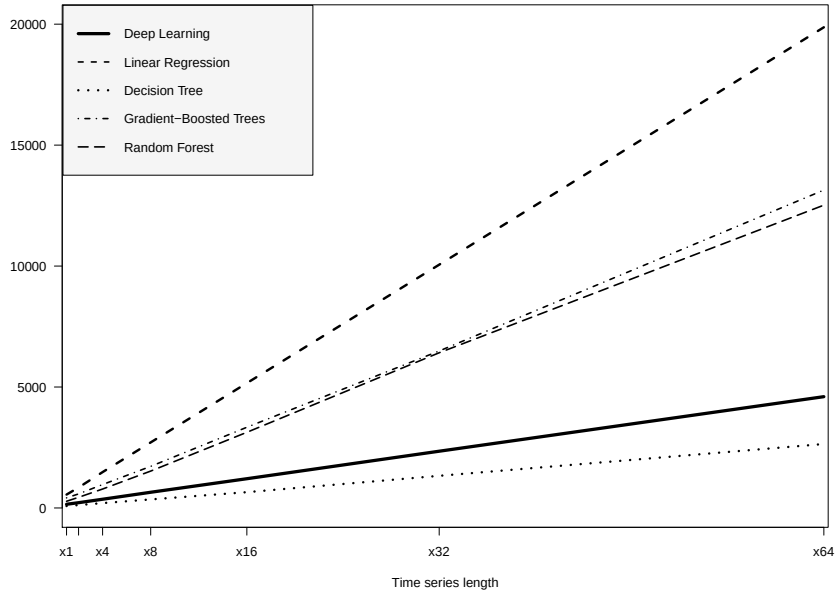


Figure 10. Scalability of the deep learning and all methods used for comparison.

5.4. Knowledge-Based Systems

Es una revista internacional creada en 1987 que pertenece al grupo Elsevier, la mayor editorial de libros de medicina y literatura científica del mundo, fundada en 1880. Knowledge-Based Systems se centra en los sistemas que utilizan técnicas basadas en el conocimiento para apoyar la toma de decisiones y el aprendizaje, lo que incluye la adquisición y representación del conocimiento y arquitecturas de sistemas Hace hincapié en la importancia práctica de dichos sistemas, su arquitectura y uso informático. Aborda la implementación de dichos sistemas en cuanto al proceso de diseño, modelos y métodos, etcétera.

Knowledge-Based Systems	
ISSN / eISSN	0950-7051 / 1872-7409
Editorial	Elsevier Science B.V., Netherlands
Categoría	Computer Science - Artificial Intelligence
Idioma	Inglés
Frecuencia	8 números al año
InCites Journal Citation Reports de 2017	
Nuevas publicaciones	369
Citas totales	9172
Factor de impacto (F.I.)	4,396
F.I. sin autocitas	3,640
F.I. últimos 5 años	4,514

- [10] A. Galicia, R. Talavera-Llames, A. Troncoso, I. Koprinska, and F. Martínez-Álvarez. «Multi-step forecasting for big data time series based on ensemble learning». *Knowledge-Based Systems* (2018). DOI: 10.1016/j.knosys.2018.10.009 IF: 4,396 (14/132) Computer Science - Artificial Intelligence Q1

Knowledge-Based Systems 163 (2019) 830–841

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multi-step forecasting for big data time series based on ensemble learning

A. Galicia [a], R. Talavera-Llames [a], A. Troncoso [a,*], I. Koprinska [b], F. Martínez-Álvarez [a]

[a] Data Science & Big Data Lab, Universidad Pablo de Olavide, ES-41013 Seville, Spain
[b] School of Information Technologies, University of Sydney, Sydney, NSW, Australia

ARTICLE INFO

Article history:
Received 15 March 2018
Received in revised form 4 October 2018
Accepted 5 October 2018
Available online 12 October 2018

Keywords:
Big data
Ensemble
Electricity time series
Forecasting

ABSTRACT

This paper presents ensemble models for forecasting big data time series. An ensemble composed of three methods (decision tree, gradient boosted trees and random forest) is proposed due to the good results these methods have achieved in previous big data applications. The weights of the ensemble are computed by a weighted least square method. Two strategies related to the weight update are considered, leading to a static or dynamic ensemble model. The predictions for each ensemble member are obtained by dividing the forecasting problem into h forecasting sub-problems, one for each value of the prediction horizon. These sub-problems have been solved using machine learning algorithms from the big data engine Apache Spark, ensuring the scalability of our methodology. The performance of the proposed ensemble models is evaluated on Spanish electricity consumption data for 10 years measured with a 10-minute frequency. The results showed that both the dynamic and static ensembles performed well, outperforming the individual ensemble members they combine. The dynamic ensemble was the most accurate model achieving a MRE of 2%, which is a very promising result for the prediction of big time series. Proposed ensembles are also evaluated using solar power from Australia for two years measured with 30-min frequency. The results are successfully compared with Artificial Neural Network, Pattern Sequence-based Forecasting and Deep Learning, improving their results.

1. Introduction

Advances in technology have led to an increasing generation and storage of massive data in recent years [1,2]. These data need to be efficiently processed in order to extract useful and valuable knowledge. Thus, the development of new tools for dealing with big data has become a critical issue. An essential component of the nature of big data is that information is usually captured over time at different points, resulting in big data time series [3]. This information can be analysed for various purposes: to predict the future values, to establish relations among variables, to detect anomalous values, or to discover patterns.

The main existing frameworks for the massive data processing have been developed thanks to leading technology companies. For example, the MapReduce technology was developed by Google [4]; it divides the input data into small blocks, processes them and then aggregates the output into a single solution. Based on this paradigm, Yahoo! developed an open-source implementation called Hadoop [5] and later Spark was developed by the University of Berkeley in California [6].

Spark maximizes parallelization of data processing in-memory, achieving much faster processing speed than Hadoop. Spark also has specific modules for mining big data, such as the Apache Spark's Machine Learning Library (MLlib) [7]. Although its native language is Scala, it also supports Python, R and Java. Spark is seen as a standard framework for data-intensive computing, and is being used in a wide range of problems such as climate data [8], water system [9], earthquakes [10], entity matching for information integration and data cleansing [11] or energy data in buildings [12]. In addition, studies of the performance of the Spark system are shown in [13] and [14].

This study is framed in the time series forecasting context, with arbitrary time horizon and in a big data environment. The previous work in [15] assessed the performance of MLlib for the forecasting of big data time series. A set of scalable algorithms was studied and adapted for very large time series forecasting. In particular, representative methods of different nature, such as linear regression, decision trees, gradient boosted trees and random forest were analysed. The results reported were promising and, for this reason, we now explore the suitability of combining some of these algorithms into ensembles to forecast big data time series.

* Corresponding author.
E-mail addresses: agalde@alu.upo.es (A. Galicia), rltallla@upo.es (R. Talavera-Llames), atrolor@upo.es (A. Troncoso), irena.koprinska@sydney.edu.au (I. Koprinska), fmaralv@upo.es (F. Martínez-Álvarez).

https://doi.org/10.1016/j.knosys.2018.10.009
0950-7051/

In particular, three of the aforementioned methods (decision trees, gradient boosted trees and random forest) have been used to develop a novel ensemble forecasting algorithm. Linear regression has been discarded because its performance, although acceptable in general terms, was too poor when compared to the other methods. The ensemble approach assigns different weights to every method by using a weighted square least method, which optimizes the contribution of each individual forecast in the combined forecast for a given forecasting time. Therefore, our ensemble is not a typical boosting ensemble, but a weighed voting ensemble as we combine three base models by using a weighed majority vote, where the weights are calculated based on the previous performance of these models using a least squares method. We propose two different strategies for training the ensemble models: static, in which the training set remains the same for each target sample to be forecasted, and dynamic, in which the training set slides forwards and changes for every target sample to be forecasted. The performance of the proposed ensemble algorithm has been evaluated using electricity consumption data from Spain. The ensemble outperforms all three methods it combines when they are used individually. In particular, the dynamic ensemble was the best performing model achieving mean relative errors of about 2%, which is a very competitive results [16].

An important feature of the proposed approach is its ability to deal with prediction horizons of arbitrary length. In this sense, if the prediction horizon is composed of h samples, h independent models are generated and simultaneously processed.

Although we used the MLlib implementations of the three base prediction methods, there were some limitations that we had to overcome. On one hand, the regression techniques available in MLlib do not support more than one step ahead prediction. On the other hand, the RDD (Resilient Distributed Dataset) data structure used by Spark is not designed to guarantee the order of data, which is a critical aspect in time series processing . The handling of these two key limitations is another contribution of this work.

In summary, the main contributions of this work are:

(1) We propose a weighted voting ensemble that uses a least squares method to calculate the weights of the base models.

(2) We develop two versions of this ensemble, static and dynamic.

(3) We show how this ensemble can be evaluated on big data for multi-step ahead forecasting by decomposing the task into multiple prediction problems, which can be solved by the Apache Spark big data engine.

(4) We conduct a comprehensive evaluation using Spanish electricity data for 10 years, measured at 10-min intervals, demonstrating that both ensemble members performed, outperforming the base models they combine, and particularly showing the potential of dynamic ensembles for big data forecasting.

(5) Solar photovoltaic dataset from Australia has been used to compare proposed ensembles with algorithms of different nature, such us deep learning, pattern sequence-based forecasting and artificial neural networks.

Section 2 reviews the literature related to time series forecasting techniques, machine learning for big data and ensemble learning. A theoretical background is included in Section 3, where the multi-step methodology is proposed. Section 4 presents and discusses the results of Spanish electricity data. Section 5 discusses and compares the results of Australian solar data . Finally, Section 6 summarizes the main conclusions.

2. Related work

This section discusses the most relevant related works, focusing on big data. Although short and medium term time series forecasting have been extensively studied in the literature, there are very few works on time series forecasting for big data.

In general, the methods for predicting time series can be classified into classical methods based on Box and Jenkins [17], such as ARIMA and GARCH; and data mining methods, such as Support Vector Machine (SVM), k Nearest Neighbour techniques (kNN) and Artificial Neural Networks (ANN). For a taxonomy of these techniques applied to energy time series forecasting, the reader is referred to [16].

However, due to the high computational cost, the majority of the data mining techniques cannot be applied when big data have to be processed. Therefore, big data mining techniques [18,19] are being developed for distributed computing in order to solve typical tasks as classification, clustering or regression. A brief description of the main advances in this area is given below.

Several MapReduce-based approaches for big data scenarios have been recently provided for classification tasks. The SVM algorithm was recently adapted to the field of high performance computing giving rise to parallel SVMs [20]. Based on Spark, several parallel implementations of the kNN algorithm have been proposed in [21–23]. Also, a MapReduce-based framework focused on instance reduction methods was proposed in [24] to reduce the computational cost and storage requirements of kNN. Deep learning has been also used for industry process planning in [25].

In recent years, increased attention has been paid to big data clustering. A survey on this topic can be found in [26,27]. In the particular field of big data time series, K-means has been successfully applied in [28]. Likewise, the challenging task of determining the optimal number of partitions has been addressed in [29], where scalable approaches to determine the quality of the generated partitions using clustering were proposed.

Very few works have been published on using big data for regression tasks, hence there is much room for improvement. A survey on big data forecasting is presented in [30]. Some regression algorithms based on cloud and big data technologies have been used for earthquake prediction in California [31]. An approach based on kNN to forecast big data time series was introduced in 2016 in [22] and approaches based on deep learning have also been published in 2017 [32]. A scalable fuzzy system for regression is proposed in [33], as the performance of the fuzzy rules depends on the size of the problem.

A variety of ensemble methods have been proposed and successfully used in practical applications [34]. The field of ensembles was developed to improve the accuracy of an automated decision-making system, with the aim of reducing variance. Since then, ensembles have been widely used in different machine learning problems, for prediction and classification, feature selection, missing feature, incremental learning, confidence estimation, error correction, among others.

Polikar [35] provided an overview of ensembles, their properties and how they can be applied to different tasks. Ensemble learning is being used for streaming analysis, a survey can be found in [36]. Ensemble techniques based on threes are the most recurrent topic in the literature for big data. This is mainly due to the easy adaptation of these algorithms for distributed computing. Random Forest has been applied to some specific problems, showing good performance for large datasets [37]. Analogously, regression trees have been constructed using parallel learning with MapReduce technology in a cluster [38].

Hadoop and its machine learning library Mahout have been selected for classification tasks using Random Forest in [39]. Meta classifiers combined automatically in an iterative way have been proposed for the detection of malware in [40]. A classifier ensemble algorithm for multimedia classification was proposed in [41], where a decision tree was used to combine the predictions of the individual ensemble members. However, an extensive analysis of the literature reveals that these methods have not been applied to the prediction of big data time series, and therefore, the work here introduced attempts at filling this gap.

Recently, to alleviate some of the issues associated with big data, Do and Poulet [42] developed a parallel ensemble learning algorithm of random local SVM that was able to perform much better than the standard SVM algorithm.

As a large number of resources is necessary to generate the ensemble, in [43] a new ensemble method that minimizes the usage of resources was proposed. Learning a decision multi-tree instead of a decision tree allows to share the common parts of the components of the ensemble and build a shared ensemble.

After a thorough review of the previously published works, it can be concluded that forecasting of big data time series is an emerging topic that should be further investigated. In particular, very few papers have been published using distributed and highly scalable computing systems. Additionally, not many ensemble methods for big data have been proposed and none of them made use of the Spark benefits. In this paper we aim to address these limitations by proposing and investigating the performance of ensemble method for big data forecasting, that makes use of the Spark engine.

In the next section we describe our methodology for forecasting big data time series using static and dynamic ensemble models.

3. Methodology

In this section, we firstly introduce the three regression methods that we use to generate prediction models for big data time series. Then, we present the proposed ensemble model which combines the individual tree-based regression models to further improve the prediction accuracy. For all regression methods we use their MLlib library implementation, to ensure the scalability of the ensemble model, and therefore, its ability to deal with big data time series.

3.1. Decision tree

Decision Trees (DTs) are a very popular and successful machine learning method, for both classification and regression tasks. Some of the advantages they offer are the following: they are able to model nonlinear relations between the attributes and the target variable, do not require attribute scaling, and the resulting tree (a set of if-then rules) is easy to interpret and use for decision making by the end-user.

A DT is built through a recursive binary partition of the feature space. A node in the tree corresponds to a test for the value of an attribute and a leaf node corresponds to a regression function. When building the tree, at each step the attribute with the highest information gain is selected. A test for its value is used to split the tree, creating a branch for the possible outcomes. The test divides the instances into several subsets, based on the attribute value. The process is repeated recursively for each subset until the stopping condition is satisfied, in which case a leaf node is created. The tree growing stops when there is no split candidate with sufficiently high information gain or when a pre-specified maximum tree depth is reached.

To predict the value for a new instance, we start at the root of the tree and follow the path corresponding to the values of the instance until a leaf node is reached and the prediction is obtained.

MLlib supports DTs for both classification and regression, and can be used with both continuous and discrete attributes.

3.2. Gradient boosted trees

Gradient Boosted Trees (GBT) is an ensemble of DTs. An ensemble method combines the predictions of a set of base models. DT-based ensembles such as GBT have showed high performance in regression and classification tasks [44,45]. GBT creates the tree ensemble iteratively. Thus, the errors made by the first tree are taken into account when adding the second tree and so on. The final prediction is the mean of the predictions of the individual trees.

3.3. Random forests

Random Forest (RF) is also an ensemble of DTs. In contrast to GBT, where the trees are built iteratively, RF trains multiple trees in parallel. Each tree uses a different training set generated by creating a bootstrap sample from the training data. In addition, when selecting the best attribute at each node, only a subset of all attributes available at the node will be considered, instead of all features as in DTs. Thus, RF uses both random instance and feature selection. To make a prediction for a new instance, RF takes the average of the predictions obtained from each tree.

In summary, both GBT and RF use decision trees as a base model, but the training process is different and each of the two methods has its advantages and disadvantages. MLlib supports both GBT and RF.

3.4. Proposed ensemble model

Ensembles of prediction models are one of the most successful methods used in practical applications [34]. An ensemble usually improves the results of the single base models it combines. In this paper we focus on ensembles for time series forecasting, and especially for big data. We propose to combine DT, GBT and RF in an ensemble, due to the good results obtained by each of these algorithms when applied to big data time series forecasting [15].

Instead of combining the predictions by taking the average of the individual predictions, which is the most simple and straightforward combination, we propose to use a weighted average combination, where different weights for each algorithm are computed based on its previous performance.

To calculate the coefficients for the weighted prediction, we minimize the forecasting error on a validation set. Let K be the number of algorithms that form the ensemble model. Let us suppose that the validation set is composed of N instances and h is the prediction horizon. Then, a weighted least squares method is applied to minimize the squared error between the predictions of the K algorithms and the actual values for the N instances of the validation set. Thus, the weights that minimize the difference between the predicted and actual values are obtained by solving the following equation for each jth value of the prediction horizon:

$$\widehat{P}^j \alpha^j = b^j \tag{1}$$

where $\widehat{P}^j$ is a matrix with N rows and K columns containing the prediction for the jth value of the prediction horizon for the validation set for each algorithm, α^j is a vector of K elements corresponding to the weights for the jth value of the prediction horizon for each algorithm, and b^j is a vector composed of the N actual values for the jth value of the prediction horizon for the validation set. The predictions $\widehat{P}^j$ for each algorithm are computed following the methodology described in the Section 3.5.

Once the weights have been obtained by the weighted square least method, we use them to make predictions for the test set. Let us suppose the test set is composed of M instances. Then, the Mxh predictions represented by the matrix $\widehat{P}$ are computed by a linear combination of the predictions for the K algorithms, where the coefficients of the linear function are given by the weights:

$$\widehat{P} = \left[\widehat{Q}^1\alpha^1, \ldots, \widehat{Q}^h\alpha^h\right] \tag{2}$$

where $\widehat{Q}^j$ is a matrix with M rows and K columns containing the prediction for the jth value of the prediction horizon for the test set for each algorithm and α^j are the weights obtained by Eq. (1). Thus, $\widehat{Q}^j\alpha^j$ is a vector of M elements containing the predictions obtained by the ensemble model for the jth value of the prediction horizon for the test set.

From Eq. (2) it also follows that:

$$\widehat{p}_{i,j} = \sum_{l=1}^{K} \alpha_l^j \widehat{p}_{i,l}^j \tag{3}$$

where $\widehat{p}_{i,j}$ is the (i, j)-th element of the matrix $\widehat{P}$, $\widehat{p}_{i,l}^j$ is the (i, l)-th element of the matrix $\widehat{Q}^j$ and α_l^j is the lth element of the vector α^j.

Both Eqs. (2) and (3) represent a static ensemble combination, i.e. the same weights are used to predict all Mxh values of the test set. However, different adaptive strategies can be used to update the weights after a given time interval, and thus to create dynamic ensembles.

Fig. 1 shows a general diagram of the static ensemble model described above to predict a big data time series. Note that the weights are defined by a matrix due to a multi-step prediction problem that we address in this paper.

In the dynamic ensemble model, let R be the updating period for the weights. Then the test set TS can be divided into subsets as follows:

$$TS = \bigcup_{t=1}^{R} TS^t \tag{4}$$

where the subset TS^t is composed of [M/R] instances and $[\cdot]$ denotes the whole number part of the division. Then, the weights are found by solving the following equation:

$$\widehat{P}^j(t)\alpha^j(t) = b^j(t) \quad t = 1, \ldots, R \tag{5}$$

where $P^j(t)$ and $b^j(t)$ are the predictions and actual values for the jth value of the prediction horizon respectively, using the validation set from the training set updated as follows:

$$TRS \leftarrow TRS^t \bigcup \left(\bigcup_{l=1}^{t-1} TS^l \right) \quad t = 1, \ldots, R \tag{6}$$

where TRS^t is the training set by removing the oldest $(t-1)[M/R]$ instances. Note that $(t-1)[M/R]$ is the size of the $\bigcup_{l=1}^{t-1} TS^l$, and therefore, the updating of the training set consists of sliding the training set $(t-1)[M/R]$ instances forward while keeping the same training set size.

Once the weights have been obtained by the weighted square least method from Eq. (5), the predictions for each subset TS^t are computed as:

$$\widehat{P}(t) = \left[\widehat{Q}^1(t)\alpha^1(t), \ldots, \widehat{Q}^h(t)\alpha^h(t)\right] \quad t = 1, \ldots, R \tag{7}$$

where $\widehat{Q}^j(t)$ contains the prediction for the jth value of the prediction horizon for the TS^t test subset for each algorithm and $\alpha^j(t)$ are the weights obtained by Eq. (5).

Fig. 2 presents graphically how the dynamic ensemble model works when the weights are periodically updated, in our case, every 3 months.

3.5. Multi-step forecasting

This section summarizes the forecasting methodology proposed in our previous work [15] for h-steps ahead prediction for big data time series using the MLlib library of Apache Spark.

Problem formulation. Given a time series with previous values up to time t, $[x_1, \ldots, x_t]$, the task is to predict the h next values of the time series, from a window of w past values, as shown in Fig. 3.

This forecasting problem can be formulated as below, where f is the model to be learnt by the forecasting method in the training phase:

$$[x_{t+1}, x_{t+2}, \ldots, x_{t+h}] = f\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \tag{8}$$

However, the existing regression techniques in MLlib do not support this multi-step forecasting. Therefore, we split the problem into h forecasting sub-problems as follows:

$$\begin{array}{rcl} x_{t+1} & = & f_1\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \\ x_{t+2} & = & f_2\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \\ \vdots & = & \vdots \\ x_{t+h} & = & f_h\left(x_t, x_{t-1}, \ldots, x_{t-(w-1)}\right) \end{array} \tag{9}$$

Hence, in each sub-problem we use the same input data with size w but predict a different target value from the forecasting window h. We note that using this formulation, the existing possible relations between the h consecutive values $x_{t+1}, \ldots, x_{t+h}$ are not taken into consideration. An alternative approach would be to use a rolling forecasting where the predictions of the previous values are used as actual values when predicting the next value, e.g. after predicting x_{t+1}, it will be included in the w values used to predict x_{t+2}. However, we found that the rolling forecasting method was less accurate due to the accumulation of the error along the prediction horizon. Hence, we chose the first approach.

We build h different training sets. Each training instance is composed of the w features. The target value for each of the h problems corresponds to a different value of the prediction horizon, as shown in Fig. 3. Thus, we learn h prediction models.

To predict a new instance from the test data, the prediction of the ith value of the prediction horizon is obtained by using the ith model with the corresponding w features from the test set as an input.

This methodology was tested in [15] using four different regression methods from MLlib (linear regression, DT, GBT and RF) demonstrating the suitability of these methods for big data time series forecasting.

In the next section we evaluate the performance of the proposed ensemble models on Spanish electricity consumption data for 10 years measured with a 10-minute frequency.

4. Results for electricity consumption data

In this section, we present and discuss the application of our proposed method for prediction of big electricity consumption time series data. We firstly describe the electricity consumption dataset in Section 4.1. The experimental setting is presented in Section 4.2 and the sensitivity analysis used to select an adequate historical window size is provided in Section 4.3. We present and analyse the results obtained by the different static and dynamic ensemble methods in Section 4.4.

4.1. Dataset description

The time series used in this work is related to the total electrical energy consumption in Spain, from January 1st 2007 at midnight to June 21st 2016 at 11:40 pm. In short, it is a time series of nine and a half years with a high sampling frequency, namely 10 min intervals, including 49,832 measurements in total.

When using the proposed methodology with a prediction horizon of 4 h (h is set to 24 values), the dataset consists of 20,742 instances and 144 attributes, corresponding to 5.70 MiB of storage size. These 144 attributes correspond to a window w of 144 past values (24 h).

For the static ensemble, this dataset is divided into a training set and a test set consisting of 60% and 40% of the data, respectively. The training set has 298,752 measurements; it includes data from January 1st, 2007 at midnight to September 8th, 2012 at 10:30 am. The test set contains the remaining data, namely 199,080 measurements from September 8th, 2012 at 10:40 am to June 21st, 2016 at 11:40 pm.

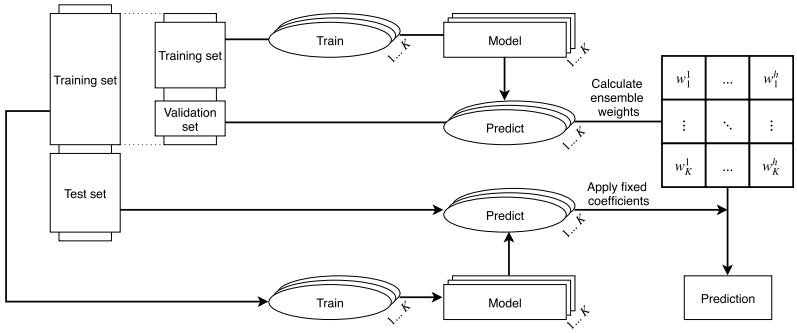


Fig. 1. Workflow of the proposed static ensemble.

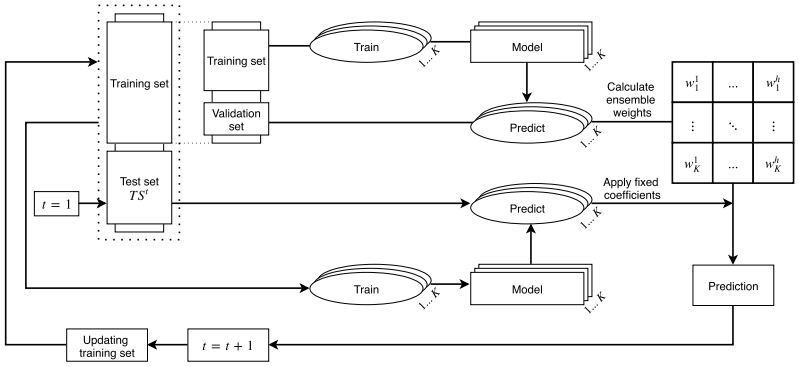


Fig. 2. Workflow of the proposed dynamic ensemble.

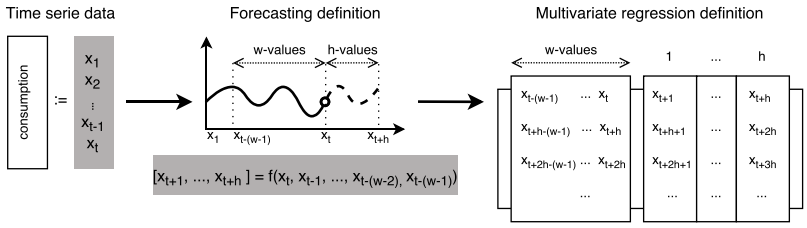


Fig. 3. Illustration of the multivariate problem.

The training set is divided again into a sub-training set –60% used to generate the prediction model for each algorithm , – and a validation set — the remaining 40% used to obtain the weights of the ensemble method .

In the case of dynamic ensemble, the weights of the ensemble are updated every 3 months (every 13,104 predicted values) sliding the training set 13,104 measurements forward while keeping the same training set size. In the same way, the prediction model is updated every 3 months.

4.2. Design of experiments

The experimentation carried out consists of a total of 248 executions, obtaining a total of 5952 prediction models for the time series of electrical consumption in the Spanish electricity market. The experimental setting is summarized below:

1. The size of the window w formed by past values has been set to 24, 48, 72, 96, 120, 144 and 168, corresponding to 4, 8, 12, 16, 20, 24 and 28 h, respectively. Given this number of past values, the goal is to predict the next 24 values.
2. The number of trees and the maximum depth of trees are input parameters in GBT and RF. Both parameters were tested in [15] and the optimal configuration obtained is used in this work. Specifically, a depth of 8 has been used for both algorithms, 5 trees for GBT and 100 trees for RF.
3. The ensemble technique combines DT, GBT and RF. When a dynamic ensemble is applied, the weights are updated every

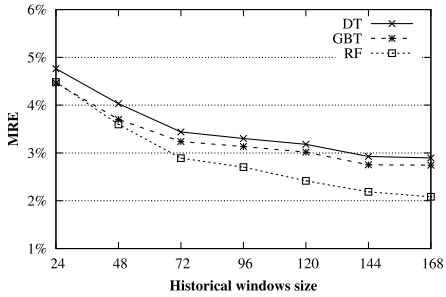


Fig. 4. MRE evolution for different historical window sizes.

3 months. Thus, the dynamic ensemble uses a total of 2304 prediction models.

The mean relative error (MRE) has been used as an evaluation measure to compare the accuracy of the predictions obtained by the different prediction methods. The MRE is defined as follows:

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|p_i - a_i|}{a_i} \quad (10)$$

where a_i and p_i represent actual and predicted values of the time series, respectively, and n is the number of samples to be predicted.

The experimentation was conducted using high-performance computing resources on the Open Telekom Cloud Platform with five machines, one master and four slave nodes. Each node has 60 GB of main memory and 8 logical cores from an Intel Xeon E5-2658 v3 @ 2.20 GHz processor that has 30 MB L3 cache. The cluster works with Apache Spark 2.1.2 and Hadoop 2.6.

4.3. Sensitivity analysis

This section presents a sensitivity analysis regarding the size of the historical window for the DT, GBT and RF algorithms, which are included in the ensemble model. The analysis consists of a total of 152 executions, obtaining 3648 prediction models in total.

Fig. 4 shows the evolution of MRE on the validation set when increasing the window size for the three proposed methods. For all methods, we can see an improvement in MRE as the size of the window w increases. For the DT algorithm, the optimal configuration was obtained with a window of 168 values, resulting in MRE of 2.90%. For GBT, the optimal model used a window of 168 past values obtaining MRE of 2.74%. Finally, for RF, the smallest error of 2.08% was obtained with a window of 168 past values. However, we can see that increasing the window size from 144 to 168 does not lead to a significant improvement for DT and GBT.

Based on the analysis above, $w = 144$ has been selected for the results shown in the following sections. We note that this window size is not accidental – it represents the values corresponding to the past 24 h –demonstrating the strong stationarity of the time series of electric demand during the day.

4.4. Analysis of results

In this section, we present and discuss the accuracy of the ensemble prediction models – the daily errors along with the worst and best days and the average relative errors of the static and dynamic ensemble models on the test set –comparing them to the errors of the single DT, GBT and RF models.

Table 1
MRE (%) distribution.

Interval	Static	(Agg)	Dynamic	(Agg)
[0,0.5)	0.00	(0)	0.00	(0)
[0.5,1)	0.87	(1)	5.50	(5)
[1,1.5)	13.82	(15)	30.82	(36)
[1.5,2)	28.58	(43)	22.79	(59)
[2,2.5)	23.52	(67)	17.80	(77)
[2.5,3)	16.86	(84)	9.41	(86)
[3,3.5)	6.73	(90)	5.72	(92)
[3.5,4)	3.62	(94)	2.46	(95)
[4,4.5)	2.60	(97)	2.75	(97)
[4.5,5)	1.45	(98)	0.51	(98)
[5,9.5)	1.95	(100)	2.24	(100)

Recall that in the case of DT, GBT, RF and the static ensemble model, we build one prediction model by using 60% of the data as training set. For the dynamic ensemble model, the training set always retains the same size, but the model is updated every 3 months, i.e., every 13,104 values. Thus, the training set is slid forward 13,104 measurements and the weights of the ensemble model are calculated again from the new validation set. Then, a new updated model is obtained to predict the 13,104 next values.

4.4.1. Overall performance

Fig. 5 presents the mean relative error for the static and dynamic ensemble and each individual model they combine, for every hour of the 24 h forecasting horizon. Table 3 also shows the aggregated mean values for each prediction algorithm for the whole test set. From Table 3 we can see that the most accurate prediction model is the dynamic ensemble –it outperforms the static ensemble and all other prediction models . This shows the benefits of dynamically adapting to the changes in the time series when building prediction models.

Within each group (dynamic and static), the ensemble outperforms the individual prediction models it combines. The most accurate individual prediction model is RF, followed by GBT and DT. DTs are single classifiers, so it is expected that they will be outperformed by ensembles of trees such as GBT and RF. RF is performing very well showing the advantage of using two strategies for generating diverse ensemble members — bagging and random feature selection when selecting the best attribute .

Fig. 5 shows that initially (during the first 1–2 h of the forecasting horizon) all methods perform similarly. For hours 3–7, RF and the ensemble show similar performance and start outperforming the other methods, and after that the ensemble method outperforms RF. For the following hours of the forecasting horizon, the ranking of the algorithms is consistent (ensemble, RF, GBT and DT).

From Fig. 5 we can also see that MRE increases as the forecasting horizon increases which is as expected. Thus, the lowest and highest MRE are obtained when the first and last value of the prediction horizon are forecasted, respectively.

4.4.2. Daily performance

To study the daily MRE we group the predictions of each algorithm into groups of 144 values as the measurements are taken every 10 min and we predict 24 h ahead. Fig. 6 shows the histogram of the daily MRE of the test set for the dynamic and static ensemble. The histogram represents the frequency of the daily MRE in different intervals when predicting all days in the test set. We can see that the dynamic ensemble considerably reduces the error in the intervals between 0.5% and 1.0% and 1.0% and 1.5%, where the accuracy is the highest. The impact of this improvement is also noticeable for daily MRE between 1.5% and 3%, due to the low number of days with these average prediction errors for the dynamic ensemble model.

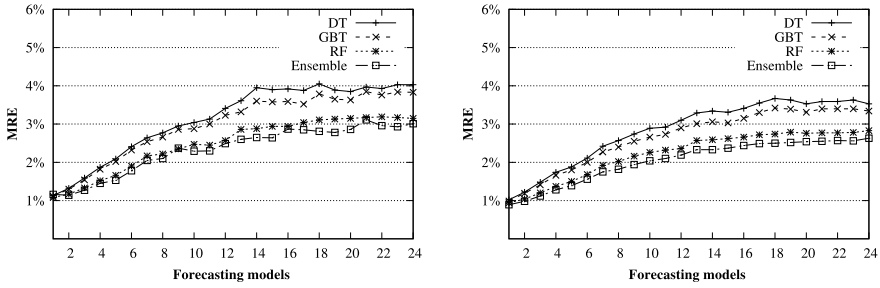


Fig. 5. MRE for each model, for each time point of the prediction horizon.

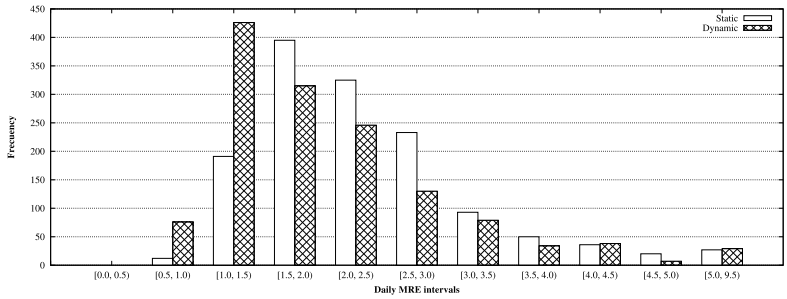


Fig. 6. Histogram of daily errors for static and dynamic ensemble models.

The percentage along with the accumulated number of days for each interval in the histogram is shown in Table 1. As it can be seen, the proposed dynamic ensemble model is stable as the 98% of days have a MRE less than 5% and all days exceeding this threshold correspond to holidays or long weekends as Table 2 shows.

4.4.3. Worst and best days

It is also interesting to study the worst and best predicted days for the different forecasting methods. Table 3 presents the MRE for the best and worst predicted day for DT, GBT, RF and also the static and dynamic ensembles. As it can be observed, the static ensemble model improves the MRE about 25% compared to DT, 21% compared to GBT and 6% compared to RF. In the case of the dynamic ensemble, it achieves a MRE improvement of 28% compared to DT, 23% comparing to GBT and 8% compared to RF. The dynamic ensemble is clearly the best performing model, outperforming the static ensemble with 13%. In addition, it can be noticed that the three ensemble models have similar prediction error variance, which is lower than the variance of the individual methods.

Table 4 compares the static and dynamic ensembles with an ANN that supports the multi-output regression. An ANN configuration with 84 neurons in the hidden layer (the mean of input and output neurons) and a hyperbolic tangent activation function has been selected. It can be seen that the accuracy of ANN is lower than both ensemble models. Compared to the dynamic ensemble, the MRE of ANN for the best predicted day – 2.0199% – is much bigger than the error of the ensemble — 0.7189% . Similarly, the error of the worst predicted day increases from 8.6016% for the ensemble to 17.0503% for ANN.

Fig. 7 shows a graphical representation of the MRE for the test set, and also the MRE corresponding to the days with the best and

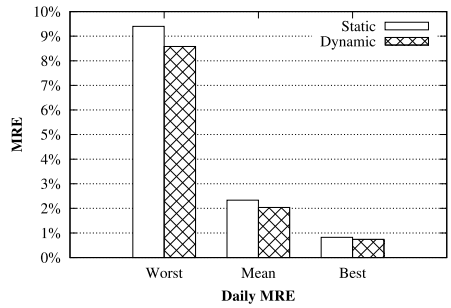


Fig. 7. Comparison of daily MRE of static and dynamic ensemble models.

the worst prediction, for each ensemble model. We can see again that the dynamic ensemble outperforms the static ensemble in all three cases.

Fig. 8 shows how the weight of each individual model in the ensemble is distributed over time. It can be seen that the weights of all three models remain stable for the prediction horizon considered. Moreover, the contribution of the single models ranges from 20% to 40% on average, showing that there is no single dominating model.

Fig. 9 provides information about the hourly predictive performance of the static and dynamic ensemble for the best day. Specifically, Fig. 9(a) shows the actual and predicted electricity demand for the day with the best prediction (MRE of 0.82%), obtained by

Table 2
Worst days for static and dynamic ensemble models.

Static ensemble			Dynamic ensemble		
MRE	Day	Type of Day	MRE	Day	Type of Day
9.32	24/12/13	Christmas Eve	8.60	24/12/13	Christmas Eve
7.71	24/12/12	Christmas Eve	7.18	19/04/14	Easter
7.66	19/04/14	Easter	7.16	30/03/13	Easter
7.40	30/03/13	Easter	7.01	29/03/13	Easter
7.31	24/12/15	Christmas Eve	6.98	24/12/12	Christmas Eve
7.15	31/12/12	New Year's Eve	6.88	31/12/12	New Year's Eve
7.14	24/12/14	Christmas Eve	6.71	24/12/15	Christmas Eve
6.96	31/12/15	New Year's Eve	6.33	31/03/13	Easter
6.56	31/03/13	Easter	6.00	30/04/14	Labour Day
6.41	30/04/13	Labour Day	5.83	30/04/15	Labour Day
6.29	17/04/14	Easter	5.72	01/04/13	Easter
6.29	31/12/13	New Year's Eve	5.58	31/12/15	New Year's Eve
6.00	30/04/15	Labour Day	5.54	31/12/13	New Year's Eve
5.97	29/03/13	Easter	5.54	07/12/14	Immaculate Conception
5.95	21/04/14	Easter	5.48	26/12/12	Christmas Day

Table 3
MRE (mean and variance) on the test set, for the worst and best predicted days.

Model	Algorithm	Worst (%)	Mean (%)	Variance (%)	Best (%)
Static	DT	10.2102	3.1400	0.014	1.2401
	GBT	10.1950	2.9680	0.012	1.2074
	RF	8.8475	2.4838	0.011	0.7621
	Ensemble	**9.3207**	**2.3320**	**0.009**	**0.8230**
Dynamic	DT	9.5022	2.8395	0.015	1.1500
	GBT	9.1633	2.6569	0.014	1.0782
	RF	8.5162	2.2243	0.012	0.6530
	Ensemble	**8.6016**	**2.0362**	**0.010**	**0.7189**

Table 4
Static and dynamic ensembles comparison with ANN.

Method	Worst (%)	Mean (%)	Variance (%)	Best (%)
ANN	17.0503	4.0342	0.136	2.0199
Static ensemble	9.3207	2.3320	**0.009**	0.8230
Dynamic ensemble	**8.6016**	**2.0362**	0.010	**0.7189**

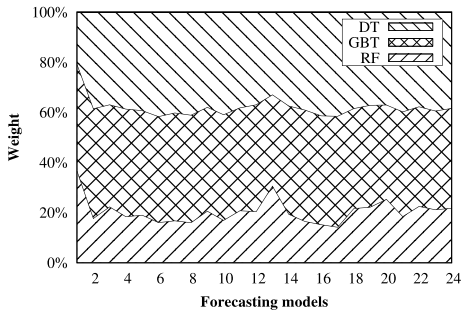


Fig. 8. Static ensemble — weight change during the prediction horizon for the individual models .

the static ensemble model. This day corresponds to the 24 h from Tuesday, August 4th, 2015 at 10:00 pm until Wednesday, August 5th, 2015 at 10:50 pm. Fig. 9(b) shows the same information for the day with the best prediction (MRE of 0.74%), obtained by the dynamic ensemble model. This day, corresponds to the 24 h from Wednesday July 30th, 2014 at 11:00 pm until Thursday July 31st, 2014 at 10:50 pm.

It is also important to analyse the days with worst predictions since they contribute to increasing the average errors. Fig. 10(a) shows the day with the worst prediction obtained with the static ensemble model, resulting in a MRE of 9.32%. This day corresponds to the 24 h from Tuesday December 24th, 2013 at 11:00 pm to Wednesday December 25th at 10:50 pm. Fig. 10(b) shows the day with the worst prediction obtained with the dynamic ensemble model, resulting in MRE of 8.60%, also for the 24 h from Tuesday December 24th, 2013 to Wednesday December 25th. This coincidence is reasonable because December 24th and 25th are special days (Christmas holidays) characterized by more random electricity consumption; they are more different than the previous days, and hence, are more difficult to predict.

In summary, our results showed that both the static and dynamic ensembles performed well and were more accurate than the individual prediction models they combined. The dynamic ensemble was considerably more accurate than the static ensemble, for all predicted days from the test set, achieving an improvement in MRE of about 13%. It also reduced the error of the worst predicted day by 8% and the error of the best predicted day by 13%.

5. Results for solar photovoltaic data

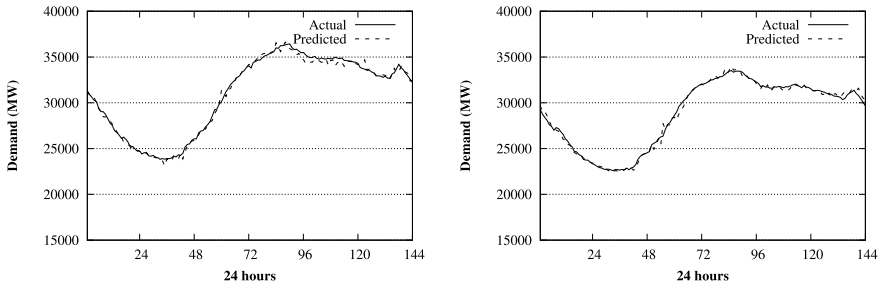
Solar energy is a very promising renewable energy source, which is still underutilized. However, in recent years there has been a considerable increase in production worldwide. Solar energy production depends on weather conditions such as solar radiation, cloud cover, rainfall and temperature. This dependence creates uncertainty where it is important to ensure a reliable supply of electricity, making it difficult to integrate solar energy into electricity markets. Therefore, the ability to predict the solar energy generated is a critical task for stakeholders in the energy sector.

In this section, we present and discuss the application of our proposed method for prediction of solar power data. We firstly describe the dataset in Section 5.1. The experimental setting is presented in Section 5.2. We present and analyse the results obtained by the different static and dynamic ensemble methods in Section 5.3.

5.1. Dataset description

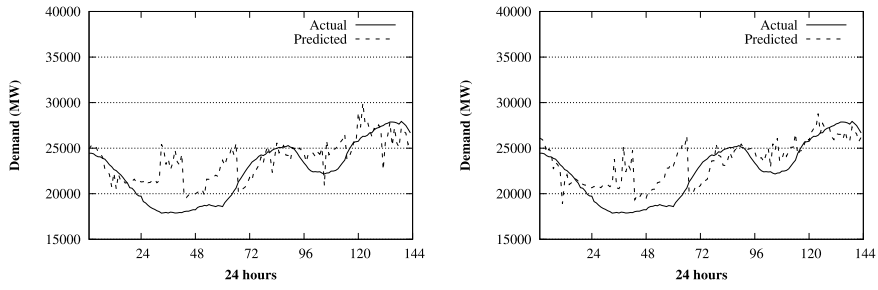
The time series used is related to Australian solar photovoltaic data for two years, from January 1st 2015 to 31 December[st] 2016. It is a time series with 30 min intervals, where each day has 20 measurements corresponding to the solar day from 7:00 to 17:00.

When using the proposed methodology with a prediction horizon of 10 h (h is set to 20 values), the dataset consists of 730 instances and 20 attributes. These 20 attributes correspond to a window w of 20 past values (10 h). This dataset is divided into a training set and a test set consisting of 70% and 30% of the data, respectively.



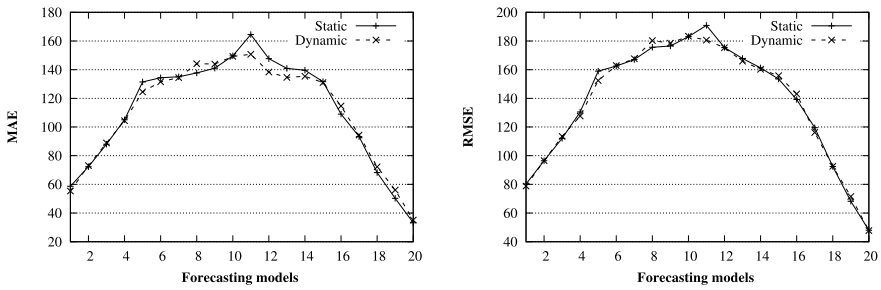
(a) Static ensemble (b) Dynamic ensemble

Fig. 9. Best predicted day — actual and predicted values .



(a) Static ensemble (b) Dynamic ensemble

Fig. 10. Worst predicted day — actual and predicted values .



(a) MAE comparison (b) RMSE comparison

Fig. 11. MAE and MRSE comparison for each time point of the prediction horizon.

The training set is divided again into a sub-training set – 30% used to generate the prediction model for each algorithm –and a validation set—the remaining 30% used to obtain the weights of the ensemble method.

In the case of dynamic ensemble, the weights of the ensemble are updated every two weeks (every 280 predicted values) sliding the training set 280 measurements forward while keeping the same training set size. In the same way, the prediction model is updated every two weeks.

5.2. Design of experiments

Deep learning (DL) has been used to forecast large datasets of solar energy data [46], comparing the performance with two other advanced forecasting methods published in [47]. In particular, Pattern Sequence-based Forecasting (PSF) based on pattern similarity [48] and an Artificial Neural Network (ANN). Static and dynamic ensembles are compared with these algorithms.

According to the referenced work to compare with, the experimental setting is summarized below:

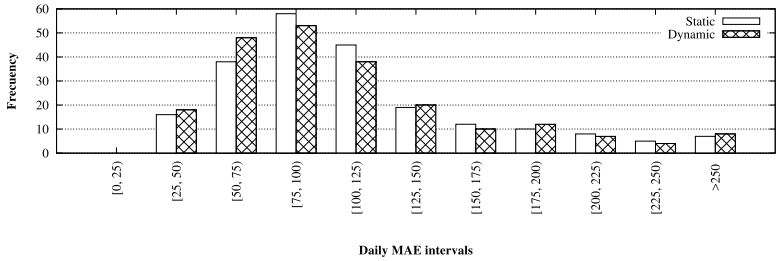


Fig. 12. Histogram of daily errors for static and dynamic ensemble models.

1. The size of the window w is formed by 20 past values, corresponding to 10 h. Given this number of past values, the goal is to predict the next 20 values.
2. The number of trees and the maximum depth of trees are input parameters in GBT and RF. Specifically, a depth of 8 has been used for both algorithms, 5 trees for GBT and 100 trees for RF.
3. The ensemble technique combines DT, GBT and RF. When a dynamic ensemble is applied, the weights are updated every two weeks.

The mean absolute error (MAE) and root mean squared error (RMSE) have been used as evaluation measures to compare the accuracy of the predictions obtained by the different prediction methods. MAE and RMSE are defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - a_i| \tag{11}$$

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(p_i - a_i)^2} \tag{12}$$

where a_i and p_i represent actual and predicted values of the time series, respectively, and n is the number of samples to be predicted.

5.3. Analysis of results

In this section, we present and discuss the accuracy of the ensemble prediction models – the daily errors along with the worst and best days and the average relative errors of the static and dynamic ensemble models on the test set –comparing them to the errors of the ANN, PSF and DL algorithms.

Recall that in the case of the static ensemble model, we build one prediction model by using 70% of the data as training set. For the dynamic ensemble model, the training set always retains the same size, but the model is updated every two weeks, i.e., every 280 values. Thus, the training set is slid forward 280 measurements and the weights of the ensemble model are calculated again from the new validation set. Then, a new updated model is obtained to predict the 280 next values.

5.3.1. Overall performance

Fig. 11 presents the MAE and RMSE for the static and dynamic ensemble, for every half hour of the 10-hour forecasting horizon. From Table 5 we can see that the most accurate prediction model is the dynamic ensemble. It outperforms the static ensemble and all other prediction models.

Fig. 11 shows that initially (during the first 1–3 h of the forecasting horizon) the accuracy is high. For hours 4–7, prediction is more difficult, because error increases. For the following hours of the forecasting horizon, both methods perform better again.

Table 5
Comparison of the performance of solar energy forecasts.

	Worst		Mean		Best	
Method	MAE	RMSE	MAE	RMSE	MAE	RMSE
ANN	**191.52**	**221.58**	114.64	154.16	58.87	106.88
PSF	252.77	279.12	117.17	147.52	31.72	36.15
DL	206.33	233	114.76	148.98	31.66	41.91
Static ensemble	303.83	353.45	111.59	130.98	**25.93**	**32.95**
Dynamic ensemble	329.60	362.60	**110.62**	**129.46**	26.72	34.04

These result are consistent for both MAE (Fig. 11(a)) and RMSE (Fig. 11(b)), where static and dynamic ensemble behaviours are similar.

5.3.2. Daily performance

To study the daily MAE and MRSE we group the predictions of each algorithm into groups of 20 values as the measurements are taken every 30 min and we predict next day ahead. Fig. 12 shows the histogram of the daily MAE and MRSE of the test set for the dynamic and static ensemble. The histogram represents the frequency of the daily MAE and MRSE in different intervals when predicting all days of the test set. We can see that the dynamic ensemble considerably reduces the error in the MAE intervals between 75 and 125, where the accuracy is the highest.

5.3.3. Worst and best days

It is also interesting to study the worst and best predicted days for the different forecasting methods. Table 5 presents the MAE and MRSE for the best and worst predicted day for ANN, PSF, DL and also the static and dynamic ensembles. As it can observed, the dynamic ensemble achieves the best accuracy by average, and the static ensemble the predicted best day.

Fig. 13 shows a graphical representation of the MAE and RMSE for the test set, and also the MRE corresponding to the days with the best and the worst prediction, for each ensemble model. We can see similar behaviours between dynamic and static ensembles.

It is also important to analyse the days with worst predictions since they contribute to increase the average errors. Fig. 14(a) shows the day with the worst prediction obtained with the static ensemble model, resulting in a MAE of 303.83. Fig. 14(b) shows the day with the worst prediction obtained with the dynamic ensemble model, resulting in MAE of 329.60.

Fig. 15 provides information about the predictive performance of the static and dynamic ensemble for the best day. Specifically, Fig. 15(a) shows the actual and predicted PV data for the day with the best prediction (MAE of 25.93), obtained by the static ensemble model. Fig. 15(b) shows the same information for the day with the best prediction (MAE of 26.72), obtained by the dynamic ensemble model.

The results show that these ensemble methods offer accurate predictions, obtaining better averaged results that the methods

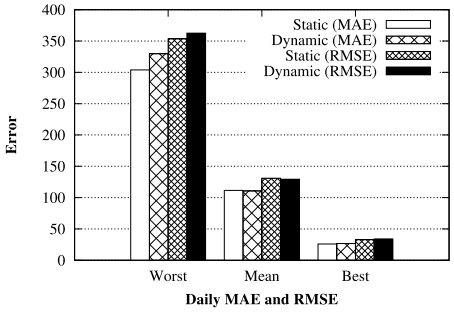


Fig. 13. Comparison of daily MAE and RMSE of static and dynamic ensemble models.

used for comparison. When the day to predict is very atypical, ensembles also have difficulty for obtaining accurate predictions. However, the best predicted days by the ensembles have a higher accuracy than ANN, PSF and DL algorithms. A bigger dataset with more training instances could help the dynamic ensemble.

6. Conclusions

In this paper, we proposed a novel approach based on ensemble learning for predicting big data time series (time series with a high sampling frequency and multi-step forecast horizon). We proposed an ensemble method which computes the weights for each ensemble member using a least square method, assigning higher weights to the more accurate ensemble members based on their past performance. We investigated two strategies for updating the weights, resulting in a dynamic and static ensemble. Although in our case study we have chosen to combine tree-based regression models (DT, GBT and RF), our method can be used to combine other type of prediction models. For the implementation of the prediction algorithms we have used the MLlib library of the Apache Spark framework, to ensure the scalability of our method and its suitability for big data. We conducted a comprehensive evaluation using Spanish electricity consumption data for 10 years consisting of about 500,000 measurements at 10-min intervals. Our results showed that both ensemble methods performed well, outperforming the individual ensemble members they combined, but the dynamic ensemble was the best method. It considerably outperformed the static ensemble, obtaining MRE of 2%. This is very competitive result, showing the viability of the proposed methodology for the prediction of big time series. In addition, Australian solar data have been used to compare static and dynamic ensembles with ANN, PSF and DL, improving the accuracy of these algorithms.

In future work, we plan study the addition of other types of prediction models to the ensemble, which are suitable for big data, in order to increase the diversity among the ensemble members. We will also investigate other machine learning strategies for determining the weights in a dynamic way. We also plan to evaluate the performance of our dynamic ensemble on other big datasets from different sources. Finally, we will develop models for forecasting special days.

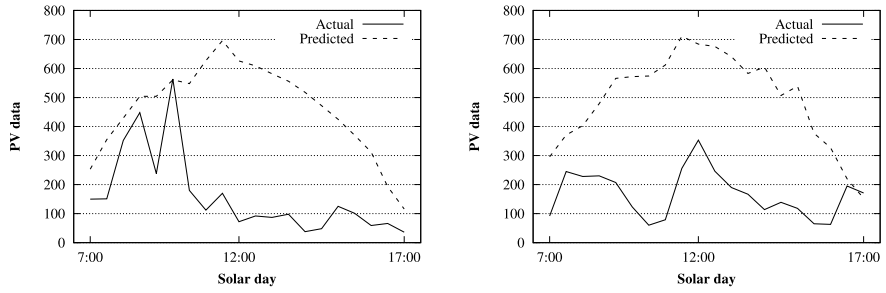


Fig. 14. Worst predicted day—actual and predicted values .

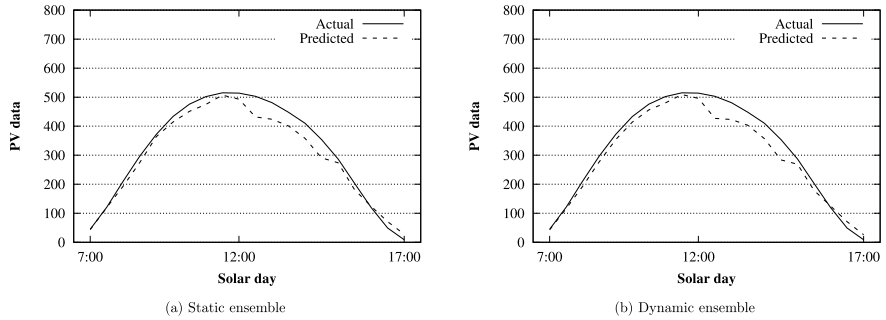


Fig. 15. Best predicted day—actual and predicted values .

Acknowledgements

The authors would like to thank the Spanish Ministry of Economy and Competitiveness and Junta de Andalucía for the support under projects TIN2017-8888209C2-1-R, TIN2014-55894-C2-R and P12-TIC-1728, respectively. Additionally, the authors want to express their gratitude to the T-Systems Iberia company since all experiments were carried out on its Open Telekom Cloud Platform based on the Open-Stack open source.

References

[1] M. Ge, H. Bangui, B. Buhnova, Big data for internet of things: A survey, Future Gener. Comput. Syst. (2018).
[2] X. Liu, P.S. Nielsen, A hybrid ICT-solution for smart meter data analytics, Energy 115 (2016) 1710–1722.
[3] Y. Sakurai, Y. Matsubara, C. Faloutsos, Mining and forecasting of big time-series data, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2015, pp. 919–922.
[4] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters.
[5] T. White, Hadoop, the Definitive Guide, O' Really Media, 2012.
[6] M. Hamstra, H. Karau, M. Zaharia, A. Knwinski, P. Wendell, Learning Spark: Lightning-Fast Big Analytics, O' Really Media, 2015.
[7] Machine learning library (mllib) for apache spark, 2016, On-line. http://spark.apache.org/docs/latest/mllib-guide.html.
[8] F. Hu, C. Yang, J.L. Schnase, D.Q. Duffy, M. Xu, M.K. Bowen, T. Lee, W. Song, ClimateSpark: An in-memory distributed computing framework for big climate data analytics, Comput. Geosci. 115 (2018) 154–166.
[9] M.E. Shafiee, Z. Barker, A. Rasekh, Enhancing water system models by integrating big data, Sustai. Cities Soc. 37 (2018) 485–491.
[10] S. Magana-Zook, J.M. Gaylord, D.R. Knapp, D.A. Dodge, S.D. Ruppert, Large-scale seismic waveform quality metric calculation using Hadoop, Comput. Geosci. 94 (2016) 18–30.
[11] D. Gomes-Mestre, C.E. Santos-Pires, D.C. Nascimento, A.R. Monteiro-de Queiroz, V. Borges-Santos, T. Brasileiro-Araujo, An efficient spark-based adaptive windowing for entity matching, J. Syst. Softw. 128 (2017) 1–10.
[12] T. Cerquitelli, Predicting large scale fine grain energy consumption, Energy Proc. 111 (2017) 1079–1088, 8th International Conference on Sustainability in Energy and Buildings, SEB-16, 11-13 2016, Turin, Italy.
[13] X. Zhang, U. Khanal, X. Zhao, S. Ficklin, Making sense of performance in in-memory computing frameworks for scientific data analysis: A case study of the spark system, J. Parallel Distrib. Comput. (2017).
[14] S. Caíno-Lores, A. Lapin, J. Carretero, P. Kropf, Applying big data paradigms to a large scale scientific workflow: Lessons learned and future directions, Future Gener. Comput. Syst. (2018).
[15] A. Galicia, J.F. Torres, F. Martínez-Álvarez, A. Troncoso, Scalable forecasting techniques applied to big electricity time series, in: Proceedings of the 14th International Work-Conference on Artificial Neural Networks, 2017, pp. 165–175.
[16] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, J.C. Riquelme, A survey on data mining techniques applied to electricity-related time series forecasting, Energies 8 (11) (2015) 13162–13193.
[17] G. Box, G. Jenkins, Time Series Analysis: Forecasting and Control, John Wiley and Sons, 2008.
[18] C.W. Tsai, C.F. Lai, H.C. Chao, A. Vasilakos, Big data analytics: a survey, J. Big Data 2 (1) (2015) 21.
[19] L. Zhou, S. Pan, J. Wang, A.V. Vasilakos, Machine learning on big data: opportunities and challenges, Neurocomputing 237 (2017) 350–361.
[20] G. Cavallaro, M. Riedel, M. Richerzhagen, J.A. Benediktsson, A. Plaza, On understanding big data impacts in remotely sensed image classification using support vector machine methods, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (2015) 4634–4646.
[21] J.L. Reyes-Ortiz, L. Oneto, D. Anguita, Big data analytics in the cloud: Spark on hadoop vs MPI/OpenMP on Beowulf, Proc. Comput. Sci. 53 (2015) 121–130.
[22] R. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, F. Martínez-Álvarez, A nearest neighbours-based algorithm for big time series data forecasting, in: Proceedings of the International Conference on Hybrid Artificial Intelligence Systems, 2016, pp. 174–185.
[23] J. González-López, S. Ventura, A. Cano, Distributed nearest neighbor classification for large-scale multi-label data on spark, Future Gener. Comput. Syst. 87 (2018) 66–82.
[24] I. Triguero, D. Peralta, J. Bacardit, S. García, F. Herrera, MRPR: A MapReduce solution for prototype reduction in big data classification, Neurocomputing 150 (2015) 331–345.
[25] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, P. Loos, Time series classification using deep learning for process planning: A case from the process industry, Proc. Comput. Sci. 114 (2017) 242–249.
[26] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, A.Y. Zomaya, I. Khalil, F. Sebti, A. Bouras, A.Y. Zomaya, S. Foufou, A. Bouras, A survey of clustering algorithms for big data: Taxonomy & empirical analysis, IEEE Trans. Emerg. Top. Comput. 5 (3) (2014) 267–279.
[27] H. Asri, H. Mousannif, H.A. Moatassime, Real-time miscarriage prediction with SPARK, Proc. Comput. Sci. 113 (2017) 423–428.
[28] R. Pérez-Chacón, R. Talavera-Llames, F. Martínez-Álvarez, A. Troncoso, Finding electric energy consumption patterns in big time series data, in: Proceedings of the International Conference on Distributed Computing and Artificial Intelligence, 2016, pp. 231–238.
[29] J.M. Luna-Romera, M.M. Martínez-Ballesteros, J. García-Gutiérrez, J.C. Riquelme-Santos, J.C. Riquelme, M.M. Martínez-Ballesteros, J.C. Riquelme, An approach to silhouette and dunn clustering indices applied to big data in spark, in: Progress in Artificial Intelligence, Springer, Cham, 2016, pp. 160–169.
[30] H. Hassani, E.S. Silva, Forecasting with big data: A review, Ann. Data Sci. 1 (2) (2015) 5–19.
[31] G. Asencio-Cortés, A. Morales-Esteban, X. Shang, F. Martínez-Álvarez, Earthquake prediction in California using regression algorithms and cloud-based big data infrastructure, Comput. Geosci. 115 (2018) 198–210.
[32] J.F. Torres, A.M. Fernández, A. Troncoso, F. Martínez-Álvarez, Deep Learning-Based Approach for Time Series Forecasting with Application to Electricity Load, Springer International Publishing, 2017, pp. 203–212.
[33] I. Rodríguez-Fdez, M. Mucientes, A. Bugarín, S-FRULER: Scalable fuzzy rule learning through evolution for regression, Knowl.-Based Syst. 110 (2016) 255–266.
[34] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, 2014.
[35] R. Polikar, Ensemble Learning, Springer US, 2012, pp. 1–34.
[36] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, Michał Woźniak, Ensemble learning for data stream analysis: A survey, Inf. Fusion 37 (2017) 132–156.
[37] L. Li, S. Bagheri, H. Goote, A. Hasan, G. Hazard, Risk adjustement of patient expenditures: A big data analytics approach, in: Proceedings of the IEEE International Conference on Big Data, 2013, pp. 12–14.
[38] B. Panda, J.S. Herbach, S. Basu, R.J. Bayardo, PLANET: massively parallel learning of tree ensembles with MapReduce, in: Proceedings of the Very Large Databases, 2009, pp. 1426–1437.
[39] K. Singh, S.C. Guntuku, A. Thakur, C. Hota, Big data analytics framework for peer-to-peer botnet detection using random forests, Inform. Sci. 278 (2014) 488–497.
[40] J.H. Abawajy, A. Kelarev, M. Chowdhury, Large iterative multitier ensemble classifiers for security of big data, IEEE Trans. Emerg. Top. Comput. 2 (3) (2014) 352–363.
[41] Y. Yan, Q. Zhu, M.L. Shyu, S.C. Chen, A classifier ensemble framework for multimedia big data classification, in: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), jul 2016, pp. 615–622.
[42] T. Do, F. Poulet, Random local SVMs for classifying large datasets, in: Proceedings of the International Conference on Future Data and Security Engineering, 2015, pp. 3–15.
[43] V. Estruch, C. Ferri, J. Hernández-Orallo, M.J. Ramírez-Quintana, Shared ensemble learning using multi-trees, in: Proceedings of the 8th Ibero-American Conference on Artificial Intelligence, 2002, pp. 204–213.
[44] A. Torres-Barrán, Á. Alonso, J.R. Dorronsoro, Regression tree ensembles for wind energy and solar radiation prediction, Neurocomputing (2017).
[45] M.A. Hassan, A. Khalil, S. Kaseb, M.A. Kassem, Exploring the potential of tree-based ensemble methods in solar radiation modeling, Appl. Energy 203 (Suppl. C) (2017) 897–916.
[46] J.F. Torres, A. Troncoso, I. Koprinska, Z. Wang, F. Martínez-Álvarez, Deep learning for big data time series forecasting applied to solar power, in: International Joint Conference SOCO'18-CISIS'18-ICEUTE'18, Springer International Publishing, 2019, pp. 123–133.
[47] Z. Wang, I. Koprinska, M. Rana, Solar power forecasting using pattern sequences, in: Artificial Neural Networks and Machine Learning – ICANN 2017, Springer International Publishing, 2017, pp. 486–494.
[48] F. Martínez-Álvarez, A. Troncoso, J.C. Riquelme, J.S. Aguilar Ruiz, Energy time series forecasting based on pattern sequence similarity, IEEE Trans. Knowl. Data Eng. 23 (8) (2011) 1230–1243.

Bibliografía

[1] A. Szalay and J. Gray. «Science in an exponential world». *Nature* 440 (2006), pp. 413–414. DOI: 10.1038/440413a.

[2] K.H. Choi and B.W. Ang. «A time-series analysis of energy-related carbon emissions in Korea». *Energy Policy* 29.13 (2001), pp. 1155–1161. ISSN: 0301-4215. DOI: 10.1016/S0301-4215(01)00044-1.

[3] J. Dean and S. Ghemawat. «MapReduce: Simplified Data Processing on Large Clusters». *OSDI'04: Sixth Symposium on Operating System Design and Implementation.* 2004, pp. 137–150.

[4] T. White. *Hadoop, The definitive guide.* O'Really Media, 2012.

[5] M. Hamstra, H. Karau, M. Zaharia, A. Knwinski, and P. Wendell. *Learning Spark: Lightning-Fast Big Analytics.* O' Really Media, 2015.

[6] Apache Software Foundation. «Machine Learning Library (MLlib) for Apache Spark». *http://spark.apache.org/docs/latest/mllib-guide.html* (2018).

[7] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «Scalable Forecasting Techniques Applied to Big Electricity Time Series». *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cadiz, Spain, June 14-16, 2017, Proceedings, Part II.* Springer International Publishing, 2017, pp. 165–175. DOI: 10.1007/978-3-319-59147-6_15.

[8] A. Galicia, J.F. Torres, F. Martínez-Álvarez, and A. Troncoso. «A novel Spark-based multi-step forecasting algorithm for big data time series». *Information Sciences* (2018). DOI: 10.1016/j.ins.2018.06.010.

[9] J.F. Torres, A. Galicia, A. Troncoso, and F. Martínez-Álvarez. «A scalable approach based on deep learning for big data time series forecasting». *Integrated Computer-Aided Engineering* 25 (2018), pp. 1–14. DOI: 10.3233/ICA-180580.

[10] A. Galicia, R. Talavera-Llames, A. Troncoso, I. Koprinska, and F. Martínez-Álvarez. «Multi-step forecasting for big data time series based on ensemble learning». *Knowledge-Based Systems* (2018). DOI: 10.1016/j.knosys.2018.10.009.

[11] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. «From data mining to knowledge discovery in databases». *AI magazine* 17.3 (1996), p. 37.

[12] H.B. Barlow. «Unsupervised learning». *Neural computation* 1.3 (1989), pp. 295–311.

[13] O. Nelles. «Unsupervised Learning Techniques». *Nonlinear System Identification*. Springer, 2001, pp. 137–155.

[14] C.W. Tsai, C.F. Lai, H.C. Chao, and A. Vasilakos. «Big data analytics: a survey». 2.1 (2015), p. 21.

[15] L. Zhou, S. Pan, J. Wang, and A.V. Vasilakos. «Machine Learning on big data: opportunities and challenges». *Neurocomputing* in press (2017).

[16] J.M. Luna-Romera, M. Martínez-Ballesteros, J. García-Gutierrez, and J.C. Riquelme. «An Approach to Silhouette and Dunn Clustering Indices Applied to Big Data in Spark». *Proceedings of the Conference of the Spanish Association for Artificial Intelligence*. 2016, pp. 160–169.

[17] R. Pérez-Chacón, R. Talavera-Llames, F. Martínez-Álvarez, and A. Troncoso. «Finding electric energy consumption patterns in big time series data». *Proceedings of the International Conference on Distributed Computing and Artificial Intelligence*. 2016, pp. 231–238.

[18] A. Fahad and N. Alshatri and Z. Tari and A. Alamri and A.Y. Zomaya and I. Khalil and F. Sebti and A. Bouras. «A Survey of Clustering Algorithms for Big Data: Taxonomy & Empirical Analysis». *IEEE Transactions on Emerging Topics in Computing* 5 (2014), pp. 267–279.

[19] R. Ding et al. «YADING: fast clustering of large-scale time series data». *Proceedings of the VLDB Endowment* 8.5 (2015), pp. 473–484.

[20] W. Zhao and H. Ma and Q. He. «Parallel K-Means Clustering Based on MapReduce». *Lecture Notes in Computer Science* 5391 (2009), pp. 674–679.

[21] M. Capó and A. Pérez and J.A. Lozano. «A Recursive k-means Initialization Algorithm for Massive Data». *Proceedings of the Spanish Association for Artificial Intelligence*. 2015, pp. 929–938.

[22] S.B. Kotsiantis, I. Zaharakis, and P. Pintelas. «Supervised machine learning: A review of classification techniques». *Emerging artificial intelligence applications in computer engineering* 160 (2007), pp. 3–24.

[23] G. Cavallaro, M. Riedel, M. Richerzhagen, J.A. Benediktsson, and A. Plaza. «On understanding big data impacts in remotely sensed image classification using support vector machine methods». *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8 (2015), pp. 4634–4646.

[24] N. Mehdiyev et al. «Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry». *Procedia Computer Science* 114 (2017), pp. 242–249. ISSN: 1877-0509. DOI: 10.1016/j.procs.2017.09.066.

[25] J.L. Reyes-Ortiz, L. Oneto, and D. Anguita. «Big Data Analytics in the Cloud: Spark on Hadoop vs MPI/OpenMP on Beowulf». *Procedia Computer Science* 53 (2015), pp. 121–130.

[26] I. Triguero, D. Peralta, J. Bacardit, S. García, and F. Herrera. «MRPR: A MapReduce solution for prototype reduction in big data classification». *Neurocomputing* 150 (2015), pp. 331–345. ISSN: 0925-2312.

[27] J. Maillo, S. Ramírez, I. Triguero, and F. Herrera. «kNN-IS: An Iterative Spark-based design of the k-Nearest Neighbors classifier for big data». *Knowledge-Based Systems* 117 (2017), pp. 3–15. ISSN: 0950-7051.

[28] B. Panda, J.S. Herbach, S. Basu, and R.J. Bayardo. «PLANET: massively parallel learning of tree ensembles with MapReduce». *Proceedings of the International Conference in Very Large Data Bases*. 2009, pp. 1426–1437.

[29] L. Li, S. Bagheri, H. Goote, A. Hasan, and G. Hazard. «Risk adjustement of patient expenditures: A big data analytics approach». *Proceedings of the IEEE International Conference on Big Data*. 2013, pp. 12–14.

[30] G. Asencio-Cortés, A. Morales-Esteban, X. Shang, and F. Martínez-Álvarez. «Earthquake prediction in California using regression algorithms and cloud-based big data infrastructure». *Computers & Geosciences* 115 (2018), pp. 198–210. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2017.10.011.

[31] H. Hassani and E.S. Silva. «Forecasting with Big Data: A Review». *Annals of Data Science* 1.2 (2015), pp. 5–19.

[32] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín. «S-FRULER: Scalable fuzzy rule learning through evolution for regression». *Knowledge-Based Systems* 110 (2016), pp. 255–266. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2016.07.034.

[33] R. Polikar. «Ensemble Learning». *Ensemble Machine Learning: Methods and Applications*. Springer US, 2012, pp. 1–34.

[34] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2014.

[35] K. Singh, S.C. Guntuku, A. Thakur, and C. Hota. «Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests». *Information Sciences* 278 (2014), pp. 488–497. ISSN: 0020-0255. DOI: 10.1016/j.ins.2014.03.066.

[36] J.H. Abawajy, A. Kelarev, and M. Chowdhury. «Large Iterative Multitier Ensemble Classifiers for Security of Big Data». *IEEE Transactions on Emerging Topics in Computing* 2.3 (2014), pp. 352–363. ISSN: 2168-6750. DOI: 10.1109/TETC.2014.2316510.

[37] Y. Yan, Q. Zhu, M.L. Shyu, and S.C. Chen. «A Classifier Ensemble Framework for Multimedia Big Data Classification». *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*. 2016, pp. 615–622. DOI: 10.1109/IRI.2016.88.

[38] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, and Michał Woźniak. «Ensemble learning for data stream analysis: A survey». *Information Fusion* 37 (2017), pp. 132–156. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2017.02.004.

[39] G. Box and G. Jenkins. *Time series analysis: forecasting and control*. Holden Day, 1970.

[40] I.H. Witten, E. Frank, M.A. Hall, and C.J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. 4th ed. Burlington, MA: Morgan Kaufmann, 2016.

[41] F. Martínez-Álvarez, A. Troncoso, G. Asencio-Cortés, and J.C. Riquelme. «A Survey on Data Mining Techniques Applied to Electricity-Related Time Series Forecasting». *Energies* 8.11 (2015), pp. 13162–13193.

[42] Sajal Ghosh and Anjana Das. «Short–run electricity demand forecasts in Maharashtra». *Applied Economics* 34.8 (2002), pp. 1055–1059.

[43] S.R. Khuntia, J.L. Rueda, and M.A.M.M. van-der Meijden. «Volatility in electrical load forecasting for long-term horizon - An ARIMA-GARCH approach». *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. 2016, pp. 1–6.

[44] P.F. Pai and W.C. Hong. «Support Vector Machines with Simulated Annealing Algorithms in Electricity Load Forecasting». *Energy Conversion Management* 46.17 (2005), pp. 2669–2688.

[45] S. Fan, C. Mao, J. Zhang, and L. Chen. «Forecasting Electricity Demand by Hybrid Machine Learning Model». *Lecture Notes in Computer Science* 4233 (2006), pp. 952–963.

[46] Y. Guo, D. Niu, and Y. Chen. «Support-Vector Machine Model in Electricity Load Forecasting». *Proceedings of the International Conference on Machine Learning and Cybernetics*. 2006, pp. 2892–2896.

[47] A. Abraham and B. Nath. «A Neuro-Fuzzy Approach for Forecasting Electricity Demand In Victoria». *Applied Soft Computing journal* 1.2 (2001), pp. 127–138.

[48] M. El-Telbany and F. El-Karmi. «Short-term forecasting of Jordanian electricity demand using particle swarm optimization». *Electric Power Systems Research* 78 (2008), pp. 425–433.

[49] M.S. Kıran, E. Özceylan, M. Gündüz, and T. Paksoy. «Swarm intelligence approaches to estimate electricity energy demand in Turkey». *Knowledge-Based Systems* 36 (2012), pp. 93–103. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2012.06.009.

[50] I. Samuel, T. Ojewola, A. Awelewa, and P. Amaize. «Short-Term Load Forecasting Using The Time Series And Artificial Neural Network Methods». *Journal of Electrical and Electronics Engineering* 11.1 (2016), pp. 72–81.

[51] A. Ghanbari, S.M.R. Kazemi, F. Mehmanpazir, and M.M. Nakhostin. «A Cooperative Ant Colony Optimization-Genetic Algorithm approach for construction of energy demand forecasting knowledge-based expert systems». *Knowledge-Based Systems* 39 (2013), pp. 194–206. ISSN: 0950–7051. DOI: 10.1016/j.knosys.2012.10.017.

[52] A. Baliyan, K. Gaurav, and S. Mishra. «A Review of Short Term Load Forecasting using Artificial Neural Network Models». 48 (2015), pp. 121–125.

[53] F. Razak, S. Mahendran, A.H. Hashim, and I.Z. Abidin. «Load Forecasting Using Time Series Models». *Jurnal Kejuruteraan* 21 (2009).

[54] G.S. Anitha and S. Kuldeep. «Short term load forecasting methods, a comparative study». *International Journal of Advance Research and Innovative Ideas in Education* 1 (2016), pp. 31–37.

[55] M.D.C. Ruiz-Abellón, A. Gabaldón, and A. Guillamón. «Dependency-aware clustering of time series and its application on energy markets». *Energies* 9.10 (2016). DOI: 10.3390/en9100809.

[56] D. Liebl et Al. «Modeling and forecasting electricity spot prices: A functional data perspective». *The Annals of Applied Statistics* 7.3 (2013), pp. 1562–1592.

[57] H.S. Guirguis and F.A. Felder. «Further Advances in Forecasting Day-Ahead Electricity Prices Using Time Series Models». *KIEE International Transactions on PE* 4-A.3 (2004), pp. 159–166.

[58] R.C. García, J. Contreras, M. van Akkeren, and J.B.C. García. «A GARCH forecasting model to predict day-ahead electricity prices». *IEEE Transactions on Power Systems* 20.2 (2005), pp. 867–874.

[59] J. Taylor. «Density forecasting for the efficient balancing of the generation and consumption of electricity». *International journal of Forecasting* 22 (2006), pp. 707–724.

[60] A. Troncoso, J.C. Riquelme, J.M. Riquelme, J.L. Martínez, and A. Gómez. «Electricity Market Price Forecasting Based on Weighted Nearest Neighbours Techniques». *IEEE Transactions on Power Systems* 22.3 (2007), pp. 1294–1301.

[61] F. Martínez-Álvarez, A. Troncoso, J.C. Riquelme, and J.S. Aguilar. «Energy time series forecasting based on pattern sequence similarity». *IEEE Transactions on Knowledge and Data Engineering* 23 (2011), pp. 1230–1243.

[62] N. Bokde, G. Asencio-Cortés, F. Martínez-Álvarez, and K. Kulat. «PSF: Introduction to R Package for Pattern Sequence Based Forecasting Algorithm.» *R Journal* 9.1 (2017).

[63] I. Koprinska, M. Rana, A. Troncoso, and F. Martínez-Álvarez. «Combining Pattern Sequence Similarity with Neural Networks for Forecasting Electricity Demand Time Series». *Proceedings of the International Joint Conference on Neural Networks*. 2013, pp. 940–947.

[64] M. Rana, I. Koprinska, A. Troncoso, and V.G. Agelidis. «Extended Weighted Nearest Neighbor for Electricity Load Forecasting». *Artificial Neural Networks and Machine Learning – ICANN 2016: 25th International Conference on Artificial Neural Networks, Spain, September 6-9, 2016, Proceedings, Part II*. Springer International Publishing, 2016, pp. 299–307.

[65] M. Rana, I. Koprinska, and A. Troncoso. «Forecasting hourly electricity load profile using neural networks». *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014, pp. 824–831. DOI: 10.1109/IJCNN.2014.6889489.

[66] L. Fortuna, G. Nunnari, and S. Nunnari. *Nonlinear modeling of solar radiation and wind speed time series*. Springer, 2016.

[67] W.Y. Chang. «A literature review of wind forecasting methods». *Journal of Power and Energy Engineering* 2.4 (2014).

[68] A. Yona et al. «Application of neural network to one-day-ahead 24 hours generating power forecasting for photovoltaic system». 2007. DOI: 10.1109/ISAP.2007.4441657.

[69] C. Monteiro, T. Santos, L.A. Fernandez-Jimenez, I.J. Ramirez-Rosado, and M.S. Terreros-Olarte. «Short-term power forecasting model for photovoltaic plants based on historical similarity». *Energies* 6.5 (2013), pp. 2624–2643. DOI: 10.3390/en6052624.

[70] M. Rana, I. Koprinska, and V.G. Agelidis. «Univariate and multivariate methods for very short-term solar photovoltaic power forecasting». *Energy Conversion and Management* 121 (2016), pp. 380–390. DOI: 10.1016/j.enconman.2016.05.025.

[71] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, and F. Martínez-Álvarez. «MV-kWNN: A novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting». *Neurocomputing* (2018).

[72] S. Comello, S. Reichelstein, and A. Sahoo. «The road ahead for solar PV power». *Renewable and Sustainable Energy Reviews* 92 (2018), pp. 744–756. ISSN: 1364-0321. DOI: 10.1016/j.rser.2018.04.098.

[73] S. Pfenninger. «Dealing with multiple decades of hourly wind and PV time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability». *Applied Energy* 197 (2017), pp. 1–13. ISSN: 0306–2619.

[74] C. Hartmann, M. Hahmann, W. Lehner, and F. Rosenthal. «Exploiting big data in time series forecasting: A cross-sectional approach». *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344786.

[75] R. Talavera-Llames, R. Pérez-Chacón, M. Martínez-Ballesteros, A. Troncoso, and F. Martínez-Álvarez. «A nearest neighbours-based algorithm for big time series data forecasting». *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems*. 2016, pp. 174–185.

[76] R. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, and F. Martínez-Álvarez. «Big data time series forecasting based on nearest neighbours distributed computing with Spark». *Knowledge-Based Systems* (2018). ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.07.026.

[77] J.F. Torres, A.M. Fernández, A. Troncoso, and F. Martínez-Álvarez. «Deep Learning-Based Approach for Time Series Forecasting with Application to Electricity Load» (2017), pp. 203–212.

[78] X. Zhang, U. Khanal, X. Zhao, and S. Ficklin. «Making sense of performance in in-memory computing frameworks for scientific data analysis: A case study of the spark system». *Journal of Parallel and Distributed Computing* (2017). ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2017.10.016.

[79] S. Caíno-Lores, A. Lapin, J. Carretero, and P. Kropf. «Applying big data paradigms to a large scale scientific workflow: Lessons learned and future directions». *Future Generation Computer*

Systems (2018). ISSN: 0167-739X. DOI: 10.1016/j.future.2018.04.014.

[80] F. Hu et al. «ClimateSpark: An in-memory distributed computing framework for big climate data analytics». *Computers & Geosciences* 115 (2018), pp. 154 –166. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2018.03.011.

[81] M.E. Shafiee, Z. Barker, and A. Rasekh. «Enhancing water system models by integrating big data». *Sustainable Cities and Society* 37 (2018), pp. 485–491. ISSN: 2210-6707. DOI: 10.1016/j.scs.2017.11.042.

[82] S. Magana-Zook, J.M. Gaylord, D.R. Knapp, D.A. Dodge, and S.D. Ruppert. «Large-scale seismic waveform quality metric calculation using Hadoop». *Computers & Geosciences* 94 (2016), pp. 18–30. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2016.05.012.

[83] D. Gomes-Mestre et al. «An efficient spark-based adaptive windowing for entity matching». *Journal of Systems and Software* 128 (2017), pp. 1–10. ISSN: 0164-1212. DOI: 10.1016/j.jss.2017.03.003.

[84] T. Cerquitelli. «Predicting Large Scale Fine Grain Energy Consumption». *Energy Procedia* 111 (2017). 8th International Conference on Sustainability in Energy and Buildings, SEB-16, 11-13 September 2016, Turin, Italy, pp. 1079 –1088. ISSN: 1876-6102. DOI: 10.1016/j.egypro.2017.03.271.

[85] K. Pearson and W. Gibson. «Stellar characters, further considerations of the correlations of». *Monthly Notices of the Royal Astronomical Society* 68 (1908), p. 415.

[86] J.H. Friedman. «Greedy function approximation: a gradient boosting machine». *Annals of statistics* (2001), pp. 1189–1232.

[87] L. Breiman. «Random forests». *Machine learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[88] J.F. Torres, A. Troncoso, I. Koprinska, Z. Wang, and F. Martínez-Álvarez. «Deep Learning for Big Data Time Series Forecasting Applied to Solar Power». *International Joint Conference SOCO'18-CISIS'18-ICEUTE'18*. Springer International Publishing, 2019, pp. 123–133. ISBN: 978-3-319-94120-2.