



Graduado en Ingeniería Informática

Universidad a distancia de Madrid

Departamento de Ingeniería Informática

TRABAJO FIN DE GRADO
Curso 2018 - 19

Sistema big data para mejorar los
rendimientos agrícolas en Castilla y León

Autor: Francisco Javier Moreno Hermosilla

Director: Juan Alfonso Lara Torralbo

MADRID, FEBRERO 2019

Agradecimientos

Agradezco a mi tutor Juan Alfonso por guiarme durante el trabajo y por darme los ánimos suficientes para sacarlo adelante.

Dar las gracias a mi mujer y a mi hija por su paciencia y su apoyo en todo momento, ellas me han ayudado a alcanzar la meta.

Gracias.

Resumen

Actualmente se generan infinidad de datos dentro de cualquier área de conocimiento y para poder sacar partido de ellos se han desarrollado nuevas herramientas. Big data es el término que ha surgido para referirse a la rama de la informática que nos permite solucionar problemas a partir de grandes cantidades de datos y de manera rápida. El objetivo final es tomar decisiones que permitan mejorar los resultados de la organización. Cualquier sector de la economía puede beneficiarse del big data.

El presente trabajo de fin de grado tiene como objetivo el desarrollo e implementación de un sistema big data que mejore los rendimientos agrícolas en Castilla y León. Este sistema permite cargar los datos, procesarlos, visualizarlos y obtener modelos capaces de predecir la cantidad de precipitaciones que caerán el próximo año. Este conocimiento va a permitir tomar decisiones sobre el tipo de cultivo que será más rentable para el agricultor.

La memoria recoge el proyecto desde el comienzo, en el capítulo dos se hace un repaso de las principales herramientas de big data y del proceso de KDD incidiendo en su fase principal que es la minería de datos. Se han estudiado los principales cultivos de secano de Castilla y León y sus necesidades hídricas. El capítulo tres recoge la implementación del sistema, el cuatro los resultados obtenidos. En el capítulo cinco se describen las fases de desarrollo de este proyecto y su cronograma en el capítulo seis se hace un estudio del costo real de la implementación del sistema y en el capítulo siete las conclusiones y el trabajo futuro.

Palabras clave: Big data, cultivos, precipitaciones, procesado de datos, minería de datos, series temporales.

Abstract

At present, huge number data are generated within any area of knowledge and to be able to take advantage of them new tools have been developed. Big data is the term that has appeared to refer the field of computing that allows us to solve problems based on large amounts of data and quickly. The ultimate objective is to make decisions that improve the results of the organization. Any sector of the economy can benefit from big data.

The objective of this dissertation is the development and implementation of a big data system that improves agricultural performance in Castilla y León. This system allows to load the data, process them, visualize them and obtain models capable of predicting the amount of precipitations that will fall next year. This knowledge will allow taking decisions about the type of crop that will be more profitable for the farmer.

The report includes the project from the starting point, in chapter two, a review of the main tools of big data and the KDD process stressing its main phase, which is data mining. The main rainfed crops of Castilla y León have been studied and as well as their water needs have been studied. Chapter three includes the implementation of the system, chapter four, the results obtained. In chapter five the phases of development of this project and its timeline are described, in chapter six a study is made of the real cost of implementing the system is performed and chapter seven, the conclusions and future work.

Keywords: Big data, crops, precipitation, data processing, data mining, time series.

Tabla de contenido

1.	CAPÍTULO I. OBJETIVOS Y METODOLOGÍA	1
1.1.	Introducción.....	1
1.2.	Objetivos.....	3
1.3.	Requisitos	4
1.4.	Organización del resto del documento	5
2.	CAPÍTULO II. ESTADO DE LA CUESTIÓN	7
2.1.	Big data.....	7
2.1.1.	Introducción.....	7
2.1.2.	Propiedades.....	7
2.1.3.	Arquitectura de datos.....	8
2.1.4.	Herramientas big data	10
2.1.4.1.	Hadoop.....	10
2.2.	Proceso KDD	14
2.3.	Minería de datos	17
2.3.1	Técnicas supervisadas o predictivas.....	17
2.3.1.1.	Clasificación	18
2.3.1.2.	Regresión	27
2.3.2.	Técnicas descriptivas o no supervisadas	28
2.3.2.1.	Agrupación o clústering.....	28
2.3.2.2.	Detección de atípicos	33
2.3.2.3.	Asociación	34
2.3.3.	Series temporales en data mining.....	34
2.3.3.1.	Técnicas de transformación de series temporales-.....	37
2.3.3.1.1.	Transformaciones numéricas.....	38

2.3.3.1.2. Transformaciones simbólicas.....	41
2.4. Agricultura.....	44
2.4.1. La agricultura en Castilla y León.....	44
2.4.2. Principales cultivos, características principales.....	49
2.4.2.1. Cereales.....	49
a) Trigo	49
b) Cebada.....	51
c) Avena	52
d) Centeno.....	52
e) Triticale	53
2.4.2.2. Leguminosas	54
a) Garbanzos.....	54
b) Guisante.....	55
c) Lentejas	57
d) Veza.....	58
e) Yeros	59
2.4.2.3. Oleaginosas.....	60
a) Colza.....	60
b) Girasol	61
3. CAPÍTULO III. SOLUCIÓN PROPUESTA	63
3.1. Diseño del sistema.....	63
3.2. Importación de datos	71
3.2.1 Introducción	71
3.2.2 Repositorio de datos	72
3.2.2.1. Codificación de los datos.....	75
3.2.3 Base de datos mongoDB	76
3.2.4 Implementación.....	80

3.2.4.1.	Conexión y descarga.....	83
3.2.4.2.	Descompresión de archivos	86
3.2.4.3.	Inserción de datos en MongoDB.....	88
3.3.	Filtrado de datos	91
3.4.	Visualización de datos	101
3.5.	Minería de datos	106
4.	CAPÍTULO IV. RESULTADOS	114
4.1.	Introducción.....	114
4.2.	Resultados.....	115
5.	CAPÍTULO V. PLANIFICACIÓN.....	124
5.1.	Fases de desarrollo	124
5.2.	Diagrama de Gantt.....	125
6.	CAPÍTULO VI. PRESUPUESTO	127
6.1.	Recursos.	127
6.2.	Presupuesto	127
7.	CAPÍTULO VII. CONCLUSIONES Y TRABAJO FUTURO	130
7.1.	Conclusiones.....	130
7.2.	Trabajo futuro.....	131
	Bibliografía.....	133
	Referencias electrónicas	136
	Anexo A. Código de la aplicación BigDataTFG.	137
	Anexo B. Cálculo de precipitaciones históricas 2003-2018	176
	Anexo C. Datos de rendimiento agrícola.....	180

Índice de figuras

Figura 1.1. Datos generados cada minuto en 2018	2
Figura 2.1. Logo de Hadoop	10
Figura 2.2. Arquitectura HDFS.....	13
Figura 2.3. El proceso de KDD.....	15
Figura 2.4. Estructura de una neurona	21
Figura 2.5. Estructura de la red de neuronas.....	22
Figura 2.6. Ejemplo de aprendizaje y clasificación k-vecinos más próximos	26
Figura 2.7. Ejemplo de regresión lineal.....	27
Figura 2.8. Ejemplo de regresión no lineal.....	28
Figura 2.9 Ejemplo de serie temporal	35
Figura 2.10. Transformación de una serie temporal utilizando PAA	40
Figura 2.11. Representación serie temporal.....	41
Figura 2.12. Representación de la serie temporal usando método SAX.....	44
Figura 2.13. Afiliados a la S.S. sector agrario en Castilla y León	45
Figura 2.14. Cereales grano. Evolución superficie.....	47
Figura 2.15. Cereales grano. Evolución producción.....	47
Figura 2.16. Variación del rendimiento del cereal grano.....	47
Figura 2.17. Imagen de trigo.....	49
Figura 2.18. Imagen de cebada.....	51
Figura 2.19. Imagen de avena.....	52
Figura 2.20. Imagen de centeno.....	53
Figura 2.21. Imagen de triticale.....	54
Figura 2.22. Imagen de granos de trigo, centeno y triticale.....	54
Figura 2.23. Imagen de garbanzos.....	54

Figura 2.25. Imagen de granos de lentejas.....	57
Figura 2.26. Imagen de veza.....	58
Figura 2.27. Imagen de yeros	59
Figura 2.28. Imagen de colza.....	60
Figura 2.29. Imagen de girasoles.....	61
Figura 3.1. Visión general del sistema BigDataTFG.....	63
Figura 3.2. Diagrama de casos de uso del sistema BigDataTFG.....	64
Figura 3.3. Logotipo página web de NOAA.....	72
Figura 3.4. Probabilidad de que nieve más de 1 pulgada el 25 de diciembre de 2018.	73
Figura 3.5. Estaciones que recogen datos en España.....	74
Figura 3.6. Situación de las 14 estaciones de Castilla y León	74
Figura 3.7. Datos importados en formato csv.....	75
Figura 3.8. Página web mongoDB.....	77
Figura 3.9. Página de descarga mongoDB.....	78
Figura 3.10. Proceso de instalación mongoDB.....	78
Figura 3.11. Directorio con los archivos de mongoDB.	79
Figura 3.12. Servidor de la base de datos mongoDB.....	79
Figura 3.13. Terminal para trabajar con mongoDB.....	80
Figura 3.14. NetBeans.	80
Figura 3.15. Página web de Oracle	81
Figura 3.16. Características principales del computador usado en este trabajo.....	81
Figura 3.17. Logo de Windows	81
Figura 3.18. Página de descarga de la aplicación Robo 3T	82
Figura 3.19. Interfaz gráfica	83
Figura 3.20. JLabel con el año del que se obtienen los datos.	83
Figura 3.21. Panel para seleccionar el directorio donde se almacena el archivo.	84
Figura 3.22. Página de NOAA desde el 21 de diciembre de 2018	85

Figura 3.23 y 3.24. Barra de progreso durante la descarga y al finalizar la descarga.....	85
Figura 3.25. Mensaje de error por fichero no encontrado.....	86
Figura 3.26. Botón para descomprimir los archivos descargados.....	86
Figura 3.27. Selección del archivo descargado para descomprimir.....	87
Figura 3.28 y 3.29. Barra de progreso de descompresión de fichero.....	87
Figura 3.30. Botones de inserción y visualización del número de documentos guardados	88
Figura 3.31. Sistema de archivos para seleccionar fichero que se importa a mongoDB	88
Figura 3.32. Documentos almacenados en la base de datos tfg.....	89
Figura 3.33. Archivo que contiene los datos de 2018.....	89
Figura 3.34. Evolución de los documentos importados.....	90
Figura 3.35. Fichero 2018 importado con más de 30 millones de documentos.....	90
Figura 3.36. Visualización desde mongoDB	90
Figura 3.37. Parte del sistema encargado del filtrado de datos.....	91
Figura 3.38. Opción refrescar lista.....	92
Figura 3.39 y 3.40. Opción filtrar y formatear fecha	93
Figura 3.41 y 3.42. Opción que agrupa datos y guardar resultado final	93
Figura 3.43. Representación gráfica de las series temporales	101
Figura 3.44. Selectores para generar gráfica.....	102
Figura 3.45. Serie temporal de las lluvias en Castilla y León el año 2018	102
Figura 3.46. Serie temporal de las lluvias en Castilla y León periodo 2000-2015.....	103
Figura 3.47. Serie temporal de las lluvias en Castilla y León periodo 1920-2018.....	103
Figura 3.48. Parte del interface gráfico dedicado a Data Mining	106
Figura 3.49. Genera fichero clima.arff	107
Figura 3.50. Fichero clima arff generado en directorio D:\tfg.....	107
Figura 3.51. Aviso porque los años están mal indicados.....	107
Figura 3.52. Fichero clima.arff	108
Figura 3.53. Selección de carpeta clima.arff para algoritmo GaussianProcesses.....	108

Figura 3.54. Predicción de los próximos doce meses. GaussianProcesses.....	109
Figura 3.55. GaussianProcesses, LinearRegression, SMReg y MultilayerPerceptron,...	109
Figura 3.56. Limpiar tabla.....	110
Figura 4.1. Resultado obtenido desde la aplicación BigDataTFG.....	115
Figura 4.2. Gráfica usando algoritmo de progresos gausianos para la regresión.....	116
Figura 4.3. Gráfica previsión usando algoritmo de regresión lineal.....	118
Figura 4.4. Gráfica previsión usando algoritmo de MVS.....	119
Figura 4.5. Gráfica previsión usando algoritmo MultilayerPerceptron.....	120
Figura 4.6. Previsión: Rojo: Regresión lineal, Azul: Procesos gausianos, Verde: MVS...	121
Figura 5.1. Diagrama de Gantt del TFG	126
Figura B.1. Precipitaciones medias en CyL entre 2003 y 2018 más previsión 2019.....	179
Figura B.2. Recoge los años del 2000 al 2015 el porcentaje de precipitación normal.	179
Figura C.1. Rendimiento en secano de trigo en Castilla y León.....	180
Figura C.2. Rendimiento en secano de cebada en Castilla y León	181
Figura C.3. Rendimiento en secano de avena en Castilla y León.....	181
Figura C.4. Rendimiento en secano de centeno en Castilla y León.....	182
Figura C.5. Rendimiento en secano de triticale en Castilla y León	182
Figura C.6. Rendimiento en secano de maíz en Castilla y León.	183
Figura C.7. Rendimiento en secano de maíz forrajero en Castilla y León.	183
Figura C.8. Rendimiento en secano de sorgo en Castilla y León	184
Figura C.9. Rendimiento en secano de judía seca en Castilla y León	184
Figura C.10. Rendimiento en secano de lenteja en Castilla y León.....	185
Figura C.11 Rendimiento en secano de garbanzo en Castilla y León.....	185
Figura C.12. Rendimiento en secano de guisante seco en Castilla y León.....	186
Figura C.13. Rendimiento en secano de veza en Castilla y León.....	186
Figura C.14. Rendimiento en secano de altramuz en Castilla y León	187
Figura C.15. Rendimiento en secano de yeros en Castilla y León	187

Figura C.16. Rendimiento en secano de girasol en Castilla y León.	188
Figura C.17. Rendimiento en secano de colza en Castilla y León.....	188
Figura C.18. Rendimiento en secano de alfalfa en Castilla y León.	189
Figura C.19. Rendimiento en secano de veza forraje en Castilla y León.	189

Índice de tablas

Tabla 2.1. Herramientas de Hadoop	11
Tabla 2.2. Alfabeto para SDL.....	42
Tabla 2.3. Alfabeto SDA	43
Tabla 2.4. Rendimiento, superficie y producción de cereales grano en C. y L. 2016.. ..	46
Tabla 2.5. Cereales grano. Evolución superficie y producción ..	46
Tabla 2.6. Rendimiento, superficie y producción de leguminosas grano en C. y L. 2016..	48
Tabla 2.7. Rendimiento, y superficie de cultivos industriales en C. y L. 2016.. ..	48
Tabla 3.1. Plantilla de descripción textual de casos de uso.	65
Tabla 3.2. Descripción textual de caso de uso: Descarga fichero datos aaaa.csv.g.....	66
Tabla 3.3. Descripción textual de caso de uso: Descomprimir aaaa.csv.	66
Tabla 3.4. Descripción textual de caso de uso: Import mongoDB	67
Tabla 3.5. Descripción textual de caso de uso: Ver evolución import.....	67
Tabla 3.6. Descripción textual de caso de uso: Refrescar lista colecciones	68
Tabla 3.7. Descripción textual de caso de uso: Filtra estaciones CyL.....	68
Tabla 3.8. Descripción textual de caso de uso: Filtra precipitaciones mensuales.....	69
Tabla 3.9. Descripción textual de caso de uso: Crear archivo arf.....	69
Tabla 3.10. Descripción textual de caso de uso: Genera gráfica ..	70
Tabla 3.11. Descripción textual de caso de uso: Borrar tabla.....	70
Tabla 3.12. Descripción textual de caso de uso: Data mining.....	71
Tabla 4.1. Algoritmo de progresos gausianos para la regresión.....	116
Tabla 4.2. Algoritmo de regresión lineal... ..	117
Tabla 4.3. Algoritmo MVS para la regresión.	119
Tabla 4.4. Algoritmo MultilayerPerceptron.	120
Tabla 4.5. Evaluación de progresos gausianos para la regresión.....	122

Tabla 4.6. Algoritmo de regresión lineal.....	122
Tabla 4.7. Evaluación MVS para la regresión.....	122
Tabla 6.1. Costes materiales del proyecto.....	128
Tabla 6.2. Tiempo dedicado por tareas.....	128
Tabla 6.3. Costo de los recursos humanos.....	129
Tabla 6.4. Coste total.....	129
Tabla B.1. Precipitaciones medias anuales Castilla y León.....	178

1. CAPÍTULO I. OBJETIVOS Y METODOLOGÍA

1.1. Introducción

Actualmente se está viviendo una época donde las tecnologías han ido avanzando de tal manera que es posible resolver problemas que hace unos años eran técnicamente inabordables.

Se hace referencia en este caso al término conocido como *big data* y que consiste fundamentalmente en la obtención de grandes cantidades de datos, su almacenamiento, tratamiento, visualización y análisis.

Las computadoras que actualmente están al alcance de la mayoría de la gente disponen de una potencia de procesamiento y capacidad de almacenamiento que permiten afrontar con éxito estas tareas.

Por otra parte, está el desarrollo de nuevas herramientas que hacen posible trabajar con grandes cantidades de datos para realizar las tareas necesarias y así obtener resultados fiables y rápidos.

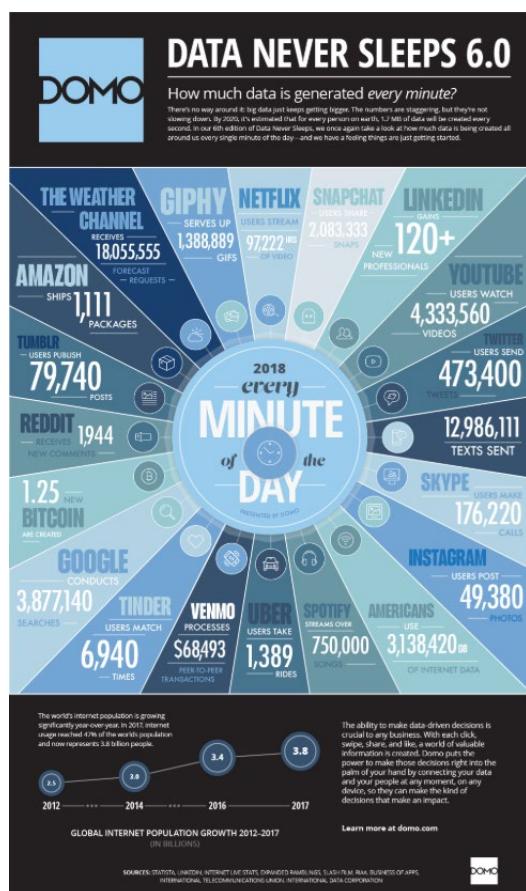
En la actualidad, muchos negocios han conseguido aumentar sus ingresos porque están aprovechando la ventaja competitiva que les da aplicar el conocimiento obtenido de esas grandes cantidades de datos que tienen en su poder, como consecuencia del desarrollo de su propio negocio, que pueden conseguir de fuentes propietarias o de las denominadas opendata.

Las redes sociales son una de las principales fuentes de generación de datos, muchas empresas son capaces de anticiparse a las necesidades de sus clientes. Conocen sus

preferencias para comprar ropa, sus coches preferidos, donde van a pasar las próximas vacaciones, qué programas de televisión ven, qué películas o series prefieren, qué música demandan. Toda esa información se la proporcionan los propios clientes, solamente tienen que analizar y transformar en información.

Los datos que se generan crecen día a día de manera exponencial y es necesario utilizar técnicas que permitan tratar esos datos de forma rápida y eficaz. El big data tiene por delante un futuro que promete grandes oportunidades de negocio.

Para hacerse una idea del volumen de datos que actualmente se mueve por internet en la figura 1.1 se muestra un esquema de la cantidad de datos generados cada minuto del día, durante el año 2018.



Cada minuto se envían 473.400 mensajes en **Twitter**, se visualizan 97.222 horas de streaming en **Netflix**, se envían 2.083.333 mensajes en **Snapchat**, **Amazon** envía 1.111 productos, se visualizan 4.333.560 vídeos en **Youtube**, se realizan 176.220 llamadas por **Skype**, se comparten 49.380 imágenes en **Instagram**, se escuchan 750.000 canciones en **Spotify** y **Google** ofrece 3.877.140 búsquedas en Internet.

Figura 1.1 - Datos generados cada minuto en 2018. Fuente <https://www.domo.com>

Big data se da en todas las áreas de la economía, pero como en otras ocasiones el campo siempre ha ido por detrás a la hora de adaptarse a las nuevas tecnologías. Por este motivo y para cubrir una necesidad que puede beneficiar a muchos agricultores es importante el diseño de un sistema fiable que sirva de ayuda para que puedan tomar decisiones acertadas a la hora de sembrar sus terrenos. Adecuar su decisión de siembra a las predicciones metereológicas puede aumentar los beneficios porque los cultivos que se van a sembrar serán seleccionados según el nivel de precipitaciones previstas para la próxima campaña. Como consecuencia la producción y el rendimiento de los cultivos aumentaría.

Se centra este trabajo en los agricultores de la comunidad castellanoleonesa, pero puede ampliarse a cualquier otra región del mundo llevando a cabo pequeñas adaptaciones.

1.2. Objetivos

Con este trabajo se va a desarrollar un sistema big data que recoja los datos de miles de estaciones meteorológicas dispersas por todo el mundo. La implementación de este sistema va a permitir el tratamiento de esos datos mediante herramientas capaces de manejarlos. Con el análisis de estos datos se va a predecir el nivel de precipitaciones que se producirán en una determinada zona del planeta.

Con esta información aplicada en el dominio de la agricultura, se planificará la próxima campaña de siembra en la comunidad de Castilla y León, decidiendo los cultivos que serán más productivos, teniendo en cuenta la cantidad de precipitaciones previstas. Esto a su vez va a permitir a los agricultores de la zona obtener mayores beneficios para su empresa.

Para conseguir el objetivo fundamental del proyecto se realizarán varios trabajos que serán también objetivos importantes:

1. Principal motivación del proyecto.
2. Estudio de las herramientas para el manejo de una gran base de datos.
3. Estudio de técnicas de minería de datos a aplicar al sistema.
4. Estudio de los principales cultivos en la comunidad de Castilla y León.
5. Estudio de las necesidades hídricas de los cultivos.
5. Diseño de un sistema big data que cumpla las necesidades del proyecto.
6. Implementación del sistema que cumpla especificaciones.
7. Visualización de resultados obtenidos.
8. Evaluación de resultados.
9. Conclusiones a extraer del trabajo.
10. Memoria y preparación de la defensa del trabajo realizado.

1.3. Requisitos

La principal tarea se centra en la búsqueda y el estudio de una base de datos que recoge la climatología a nivel mundial, almacenando datos de diferentes estaciones dispersas por todo el planeta. En Amazon Web Services (AWS) aparecen bases de datos que abarcan diferentes áreas de conocimiento y que son opendata, el acceso a los datos es en la dirección web: <https://registry.opendata.aws/>

Se ha seleccionado: NOAA Global Historical Climatology Network Daily (GHCN-D) que integra observaciones climáticas diarias de unas 30 fuentes diferentes. Actualmente se encuentra en la versión 3 que recoge datos de más de 90.000 estaciones terrestres de todo el mundo. Algunos datos tienen 175 años de antigüedad y están almacenados en formato CSV. El enlace de acceso a los datos es el siguiente link: <https://registry.opendata.aws/noaa-ghcn/>

Una vez seleccionada la base de datos es necesario realizar el proceso de KDD para obtener los modelos. Para ello se van a estudiar técnicas de minería de datos.

Por otra parte, se van a estudiar los principales cultivos que se siembran en la comunidad de Castilla y León, centrándose solamente en las zonas de secano.

Es importante conocer la cantidad de agua necesaria para que la producción de un determinado cultivo proporcione mayor rendimiento que otro con la misma cantidad de lluvia.

Con los modelos obtenidos y el estudio de los cultivos de la zona se va a ayudar a decidir a los agricultores de la zona, qué cultivos deben sembrar la próxima campaña. Consiguiendo de esta forma mayores beneficios como consecuencia de haber mejorado el rendimiento de su explotación.

1.4. Organización del resto del documento

En el capítulo dos de este documento se hace una introducción de lo que es el big data y sus propiedades. Se aborda la principal herramienta opensource (hadoop) actualmente disponible para trabajar con grandes volúmenes de datos, llegando a trabajar incluso con cantidades de datos del orden de petabytes.

Se recorren todas las etapas del proceso de KDD dando especial relevancia a la fase de data mining por ser la más importante. Dentro de la fase de minería de datos se da un repaso a las técnicas más importantes y los algoritmos que hacen posible extraer los modelos.

Para llevar a cabo el propósito principal de este trabajo va a ser necesario utilizar técnicas de minería de datos sobre series temporales por eso se repasan algunas de las técnicas para hacer transformaciones de series temporales.

Se dedica el apartado 2.4 para el estudio de los principales cultivos y sus necesidades hídricas para conseguir maximizar el rendimiento de las cosechas en la comunidad de Castilla y León.

El capítulo tres constituye la parte central del trabajo y está dedicado a describir el sistema BigDataTFG que se ha diseñado para automatizar todo el proceso desde la importación de los datos, el almacenamiento en una base de datos mongoDB, la obtención de gráficos de la serie temporal y las tareas de data mining.

En el capítulo cuatro se presentan los resultados obtenidos después de todo el proceso automatizado, exponiendo la predicción que el sistema ha hecho para las precipitaciones previstas en la comunidad de Castilla y León durante los próximos doce meses. Los resultados han sido calculados con cuatro algoritmos de aprendizaje diferentes, para el modelado de series temporales.

En el capítulo cinco se describen las fases de desarrollo de este proyecto y su cronograma y en el seis se hace un estudio del costo real de la implementación del sistema.

El capítulo siete resume las conclusiones finales de este trabajo y qué posibilidades hay de continuar en el futuro para mejorar los resultados.

2. CAPÍTULO II. ESTADO DE LA CUESTIÓN

2.1. Big data

2.1.1. Introducción

Cuando se habla de big data, este término se refiere a grandes cantidades de datos que se encuentran de manera no estructurada, a diferencia de los datos tradicionales.

Al tratarse de una rama de la informática relativamente joven no es fácil encontrar una definición, pero una aproximación sería definirla como la rama de la computación que va a proporcionar ayuda para resolver problemas a partir de un gran volumen de datos, con mucha variedad y que deben ser procesados a gran velocidad. Fue en 2005 cuando Doug Laney introdujo el teorema de las 3 V's para referirse al big data.

2.1.2. Propiedades

Las 3 V's:

- **Volumen:** para saber si se está ante un problema de big data es necesario que aparezcan una gran cantidad de datos. Todos los registros que guarda Amazon de las personas que se han conectado a su web es un ejemplo de gran volumen de datos. Como se representó en la figura 1.1, Amazon realiza 1.111 envíos cada minuto, pero además almacena mucha más

información de todos los movimientos de sus clientes, aunque no lleguen a completar la compra.

- **Velocidad:** es necesario que un sistema big data sea capaz de realizar sus tareas a mucha velocidad. De nada sirve que para resolver un problema el sistema se demore en analizar tan alto volumen de datos. Siguiendo con el ejemplo anterior, si un cliente está interesado en comprar un producto como una bici, de nada le sirve a Amazon conocer los deseos de compra de su cliente si tarda en analizar los datos, porque para cuando Amazon le ofrezca sus productos, el cliente ha podido realizar la compra en otra tienda. También existen sistemas que necesitan respuesta en tiempo real donde está claro que se precisa de muy alta velocidad de procesamiento, en caso de sufrir retrasos podrían crearse situaciones catastróficas.
- **Variedad:** esta propiedad se da en los datos cuando se reciben de diferentes orígenes y además los datos que se reciben poseen diferentes formatos. Actualmente la mayoría de los datos tratados son no estructurados a diferencia de épocas anteriores.

Hay otras dos propiedades más que determinan las 5V's.

- **Veracidad:** se refiere a los datos que recibe el sistema deben poseer esta característica porque si no es así las conclusiones obtenidas van a ser erróneas.
- **Valor:** es lo que debe de proporcionar el sistema big data como salida para obtener beneficios en el negocio. Si los resultados obtenidos no generan valor no son necesarios.

2.1.3. Arquitectura de datos

Cada proyecto es diferente, pero todos están formados por bloques que difieren según el proyecto para el que se han diseñado. Estos bloques denominados etapas de datos,

tienen relación con las fases, llamadas prácticas de datos y que pueden variar, pero las principales son: de integración, procesamiento, explotación y visualización de datos.

- **Integración.** Es la primera toma de contacto con los datos e incluye los procesos de extracción, transformación y carga. Otros procesos como aseguramiento de calidad, gobierno y seguridad de los datos también deben tenerse en cuenta en esta fase.
- **Procesamiento.** Es la práctica donde se almacenan los datos y donde se aplican tecnologías asociadas al big data como Hadoop o Spark.
- **Eplotación.** En esta fase se realizan los procesos de predicción, optimización y clasificación.
- **Visualización.** Representa mediante gráficas de fácil comprensión grandes cantidades de datos. Mostrando los resultados obtenidos de forma clara.

Los bloques o etapas de datos están más enfocados a la implementación y despliegue que las fases o prácticas antes enumeradas, aunque parecen coincidir por sus nombres, no tienen por qué hacerlo. Se indican las etapas:

- **Ingestión de datos.** Comprende la recolección de datos de diferentes orígenes y su almacenaje en un repositorio central. Hay que tener en cuenta si se trabaja con bases de datos legacy, si hay que limpiar, transformar los datos para almacenarlos o cambiar su formato.
- **Almacenamiento.** En esta etapa se determina dónde y cómo se deben almacenar los datos. Se pueden almacenar de forma local, distribuida, en la nube. Prestar atención al volumen de datos, uso de los mismos y a su seguridad.
- **Procesamiento.** Dependerá de los datos, del tiempo disponible para procesarlos y del coste, esto permite decidir qué tecnologías se van a utilizar.
- **Visualización.** Los datos son mostrados de forma que permitan descubrir el valor que aportan a los usuarios. Es importante poder trabajar con los datos de manera eficiente y sencilla.

2.1.4. Herramientas big data

Para desarrollar técnicas de big data se necesitan herramientas muy potentes para poder trabajar con un volumen muy grande de datos y hacerlo con prontitud, en algunos casos en tiempo real. Unas de las principales herramientas se describen a continuación.

2.1.4.1. Hadoop

Apareció por primera vez en 2003 en un proyecto de código abierto llamado Nutch y a partir de aquí ha ido en constante evolución. Fue creado por Doug Cutting y Mike Cafarella.

Según el propio desarrollador, el proyecto Apache™ Hadoop® desarrolla software de código abierto para computación confiable, escalable y distribuida. Es la Apache software Foundation la que se encarga de mantener Hadoop.



Figura 2.1 - Logo de Hadoop. Fuente <https://hadoop.apache.org/>

Apache Hadoop es una biblioteca de software para procesar grandes cantidades de datos mediante computadoras que se encuentran distribuidas. Su diseño permite trabajar desde servidores individuales hasta miles de máquinas. Cada máquina puede procesar y almacenar los datos de manera local.

Entre las ventajas destacan: su bajo coste ya que no requiere para ejecutarse un hardware de alto rendimiento y se pueden añadir nodos al clúster cuando se necesiten o

eliminarlos si no son necesarios. Flexibilidad en el almacenamiento y en el procesado de datos, pudiendo ser estos no estructurados o semi-estructurados. Es de código abierto y cuenta con la colaboración de muchos desarrolladores. Escalable y altamente tolerable a fallos y permite el análisis de datos muy complejos.

Hadoop permite realizar tareas de text mining, log processing, business intelligence /data warehousing, análisis de imágenes de video, reconocimiento de patrones y evaluación de riesgos entre otros usos.

Es enorme el número de instituciones que usan este software, se indica a continuación una pequeña muestra para hacerse una idea de su importancia y dimensiones.

- Facebook: Un clúster de 1100 máquinas con 8800 cores y aproximadamente 12 PB de almacenamiento en bruto y otro clúster de 300 máquinas con 2400 cores y aproximadamente 3 PB de almacenamiento en bruto.
- ebay: un clúster de 532 nodos (8 * 532 núcleos, 5.3PB).
- Spotify: un clúster de 1650 nodos: 43.000 cores virtualizados, 70TB de RAM, 65 PB de almacenamiento.

Es un ecosistema bastante complejo, un clúster Hadoop puede estar formado por miles de nodos y si su manejo se realiza manualmente sería muy complicado, pero dispone de varias herramientas que facilitan su uso. Se pueden ver en la tabla 2.1.

Tabla 2.1 – Herramientas de Hadoop.

Capa	Herramienta
Sistema de ficheros distribuidos.	Apache HDFS
Sistema de procesamiento distribuido	Apache MapReduce, Apache Hive, Apache Pig y Apache Spark
Bases de datos NoSQL	Apache HBase
Ingestión de datos	Apache Flume, Apache Sqoop, Apache Storm
Servicios de programación	Apache Zookeeper.
Programación	Apache Oozie
Aprendizaje automático	Apache Mahout
Implementación de sistema	Apache Ambari

Utiliza una arquitectura basada en computación distribuida soportada por HDFS y procesamiento paralelo por medio de MapReduce.

Desde agosto de 2018 está disponible la versión 3.1.1. El proyecto incluye los siguientes módulos:

- **Hadoop Common:** conjunto de utilidades comunes admitidas por el resto de módulos de Hadoop.
- **Sistema de archivos distribuidos de Hadoop (HDFS™):** es un sistema de archivos distribuidos para acceder a los datos de la aplicación.
- **Hadoop Yarn:** proporciona un marco para planificar las tareas y gestión de recursos del clúster.
- **Hadoop MapReduce:** es un sistema basado en Yarn que permite el procesamiento paralelo de grandes cantidades de datos.
- **Hadoop Ozone:** es un almacén de objetos para Hadoop.

La arquitectura de Hadoop tiene tres características fundamentales, es flexible, escalable y tolerante a fallos y su arquitectura se divide en dos partes: el almacenamiento y el procesamiento, del primero se encarga el Sistema de Archivos Distribuido de Hadoop (HDFS) y del segundo MapReduce.

Sistema de Archivos Distribuido de Hadoop HDFS

Es el sistema por defecto de almacenamiento de ficheros de Hadoop, funciona como maestro-esclavo. Sus principales características son, escalabilidad pudiendo llegar al orden de peta bytes, es flexible pues permite añadir o quitar nodos. Alta fiabilidad y tolerancia a fallos, los datos se almacenan en múltiples nodos y aun estando un nodo caído se puede acceder a sus datos desde otro nodo disponible. Coherencia de datos mediante WORM (escribir una vez, leer muchas) modelo que permite un alto rendimiento. Recuperación de fallos hardware, si se producen fallos en algún nodo del clúster HDFS permite la recuperación de datos y manejar la recuperación de fallos hardware. Potabilidad en diferentes hardware y software. Cómputo más cerca de los datos.

Arquitectura. HDFS es administrado por los procesos demonio o procesos no interactivos que se ejecutan en segundo plano sin precisar de la intervención del usuario.

- **NameNode:** Es el proceso maestro encargado de todas las operaciones relacionadas con el almacenamiento, lectura y escritura. Trabaja con memoria RAM para no ralentizar el proceso. Mantiene los dos archivos de metadatos, Fsimage que es todo el espacio de nombres del sistema de archivos y Editlog que es todo cambio que se produce en los metadatos del sistema de archivos.
- **DataNode:** Contiene los datos en HDFS, crea, borra o replica los bloques siguiendo las instrucciones del NameNode. Envía mensajes de hearbeats a NameNode a intervalos periódicos y en caso de que NameNode no reciba hearbeats marcará al nodo como muerto.
- **Checkpoint NameNode o NameNode Secundaria:** Es un nodo que contiene puntos de control de datos de archivos frecuentes.
- **BackupNode:** mantiene la copia actualizada de FsImage en la memoria RAM y está sincronizado con NameNode.

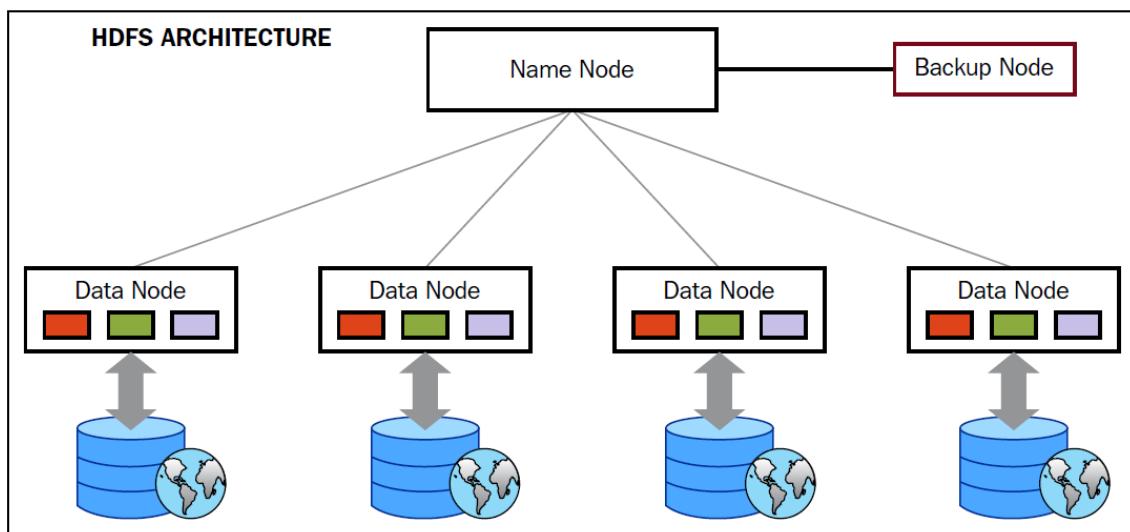


Figura 2.2 - Arquitectura HDFS Fuente. Achari, Shiva. (2015) Hadoop Essentials

MapReduce

MapReduce es una técnica de procesamiento paralelo masivo para sistemas distribuidos basado en Java y diseñado en 2004 por Google. Las tareas principales consisten en mapear y reducir. El mapeo divide un conjunto de datos en conjuntos más pequeños formados por pares de clave-valor. La función reducir a partir de los conjuntos antes obtenidos en el mapeo, los combina y los convierte en un conjunto de datos de tamaño más reducido. Trabaja en una arquitectura de maestro-esclavo.

Arquitectura. MapReduce se compone de dos procesos demonio, JobTracker que es el proceso maestro y TaskTracker como proceso esclavo.

- **JobTracker** es el proceso demonio encargado de coordinar y completar un trabajo MapReduce. Sus funciones son la gestión de recursos, así como identificar el TaskTracker para realizar ciertas tareas, supervisando el proceso y estado de la tarea.
- **TaskTracker** realiza la tarea asignada por JobTracker, le envía mensajes de heartbeat periódicamente indicando ranuras libres y comprueba las tareas a realizar y envía el estado a JobTracker sobre la tarea.

2.2. Proceso KDD

El proceso KDD (Knowledge Discovery in Databases) o descubrimiento del conocimiento en bases de datos consiste en la extracción del conocimiento en Bases de Datos. El objetivo es obtener conocimiento de manera automática que sea útil, no trivial, implícito, previamente desconocido y de calidad a partir de datos almacenados. Se trata de un proceso iterativo e interactivo.

El proceso de KDD consta de cinco fases:

1. Selección de datos o fase donde se identifican las fuentes de datos, así como el tipo de datos con los que se va a trabajar. Una vez hecho esto se almacenan todos los datos en un lugar que se denomina DataWarehouse, teniendo así una base de datos única y consistente. Una técnica de explotación del DataWarehouse es Query & Reporting, Online analytical processing (OLAP). Un sistema OLAP o de procesamiento analítico en línea realiza un análisis complejo de grandes cantidades de datos, es multidimensional y el usuario realiza una navegación asistida sobre los datos. Se caracteriza por su alta velocidad de respuesta, sobre todo al ejecutar comandos SQL de tipo SELECT.

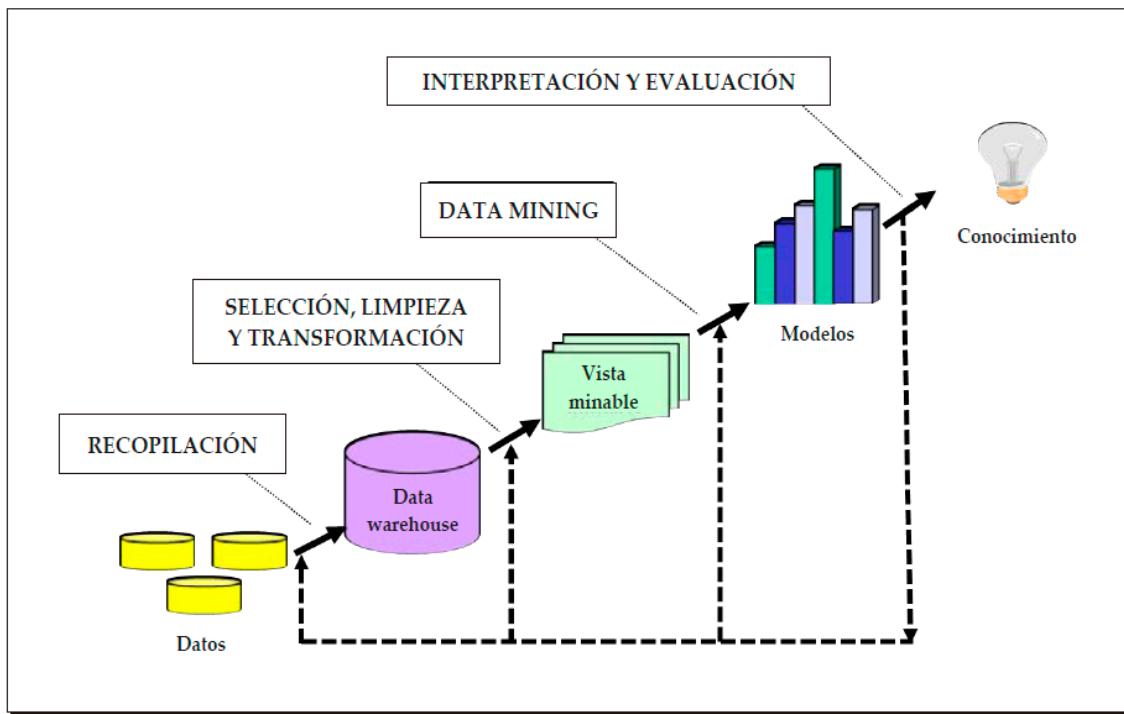


Figura 2.3 - El proceso de KDD. Fuente: Lara J.A. (2014).

2. Selección, Limpieza y transformación o fase donde se preparan los datos para poder trabajar adecuadamente con ellos. La finalidad es obtener una “vista minable” de los datos. En esta fase se lleva a cabo un proceso de selección que consiste en eliminar datos que se consideran irrelevantes.

Los datos pueden presentarse con anomalías sintácticas, semánticas o de contexto. Para realizar la limpieza de datos hay alternativas diferentes, se puede ignorar el ejemplo completo en caso de pérdida de un dato, indicar con una marca que ese dato se ha perdido o se puede llenar ese dato a partir de un cálculo hecho sobre el valor que toma ese dato en el resto de ejemplos.

En esta etapa se realiza también la transformación de los datos para permitir trabajar con ellos en la siguiente fase. Las técnicas de transformación más utilizadas son la de agregación de atributos a partir de otros, la normalización de los valores de los atributos que es habitual asignarles valores dentro del intervalo [0,1], la discretización de valores, la numerización o reducción de dimensionalidad.

3. Minería de datos o fase donde se extraen los patrones de los datos obteniendo modelos, mediante la aplicación de algoritmos. Es la fase más importante del proceso KDD.

En esta etapa es cuando se determina qué problema es el que se pretende solucionar. En segundo lugar, se aplica el algoritmo para construir el modelo y finalmente se representa la búsqueda del conocimiento.

4. Interpretación y evaluación o fase donde se muestran los resultados obtenidos en el análisis de los modelos. Hay que tener en cuenta el criterio que se va a aplicar, la comprensión del modelo, la precisión y la novedad o el nivel de interés.

Una técnica para evaluar los modelos obtenidos en la fase de data mining consiste en separar los datos en dos conjuntos, uno denominado conjunto de entrenamiento y otro conjunto de prueba. Los datos del conjunto de entrenamiento se utilizan para extraer conocimiento y los del conjunto de prueba se usan para validar el modelo.

El principal método utilizado para la validación del modelo es la n-validación cruzada. Consiste en dividir los datos en n particiones equitativas de las cuales una formará el conjunto de prueba y el resto serán los datos de entrenamiento.

Las medidas de evaluación se obtienen de las validaciones dadas a los datos según las tareas hechas en la fase de minería:

- Clasificación: Utiliza como medida la precisión predictiva que es igual al número de errores del modelo entre el número total de ejemplos probados.
- Regresión. Se utiliza el error cuadrático medio.
- Agrupamiento. La medida utilizada es la que tiene en cuenta la cohesión y la separación entre los distintos grupos creados.
- Reglas de asociación. Cobertura, confianza, etc.

2.3. Minería de datos

La minería de datos es una fase del proceso de KDD, es la fase más importante donde se reciben datos ya preparados en formato de “vista minable” y aplicando algoritmos, se obtienen modelos o patrones que posteriormente serán interpretados y evaluados.

Los dos objetivos fundamentales de la minería de datos son la predicción y la descripción. Partiendo de un conjunto de datos, relativamente grande, la predicción consiste en adivinar los valores de una variable desconocida a partir de otras conocidas de antemano. La descripción consiste en encontrar unos modelos o patrones de datos que se puedan entender y se descubran o perciban sus tendencias. Las principales técnicas de minería de datos se indican a continuación.

2.3.1 Técnicas supervisadas o predictivas

Son las que generan modelos que se utilizan para predecir el valor de uno o más atributos desconocidos.

2.3.1.1. Clasificación

Consiste en predecir el valor desconocido de un atributo de tipo cualitativo denominado de clase, que puede tomar un número finito de valores denominados clases. Para predecir el atributo de clase se dividen los datos en grupos excluyentes quedando en el mismo grupo los datos más cercanos. Se indican las principales técnicas:

- a) **Basadas en árboles de decisión.** También denominados árboles de clasificación.

Se consideran de tipo anticipativo o impaciente, porque el modelo que generan sirve para clasificar nuevos ejemplos. Son muy utilizadas porque es muy fácil interpretar el modelo que generan, aunque no son la técnica de mayor capacidad predictiva. Está formado por nodos hoja o terminales que representan regiones clasificadas, los nodos de decisión o internos representan condiciones donde hay que decidir a qué subregión pertenece el elemento que ha llegado al nodo. Para clasificar un elemento se comienza por el nodo raíz y según las condiciones que va cumpliendo se desplazará por las ramas hasta llegar al nodo terminal donde queda finalmente clasificado. La profundidad máxima del árbol de decisión es la cantidad máxima de condiciones que hay que resolver hasta alcanzar el nodo terminal que determina la clase a la que pertenece el elemento.

Para construir un árbol de decisión:

1. Se asignan todos los elementos del conjunto de entrenamiento al nodo raíz.
2. Se divide el árbol según una determinada heurística.
3. Repetir el paso anterior hasta alcanzar un nodo terminal.
4. Realizar la poda del árbol eliminando hojas con ruido u hojas que durante la creación del árbol menos aportan.

Los algoritmos más conocidos son:

- ID3 (Iterative Dichotomiser 3). Para construir el árbol utiliza como método de expansión una función basada en la entropía, la teoría de la información y la ganancia de información. El pseudocódigo del algoritmo para construir el árbol sería:

Paso 1. Seleccionar el atributo A_i que maximice la ganancia $G(A_i)$.

Paso 2. Crear un nodo para ese atributo con tantos sucesores como valores tenga.

Paso 3. Introducir los ejemplos en los sucesores según el valor que tenga el atributo A_i .

Paso 4. Por cada sucesor:

- a). Si solo hay ejemplos de una clase, C_k , entonces etiquetarlo con C_k .
- b). Si no, llamar a ID3 con una tabla formada por los ejemplos de ese nodo, eliminando la columna del atributo A_i

- C4.5. Es similar al anterior. Clasifica valores continuos y valores discretos. Utiliza para la división del árbol el concepto ganancia de información. Se permite realizar poda.
 - C5.0. Es una evolución comercial del algoritmo anterior.
 - CART. Los árboles que genera este algoritmo son binarios y trabaja con atributos con valores tanto cualitativos como cuantitativos, esto permite resolver problemas de clasificación y de regresión.
- b) **Basadas en inducción de reglas.** Es similar a los árboles de decisión, pero con diferente representación y al igual que los árboles de decisión son de tipo anticipativo. Son más comprensibles para el ser humano. Se basan en la estructura siguiente:

Si <condición> entonces <clase>

Algoritmos más utilizados:

- 1R. Propuesto por Robert C. Holte en 1993. Se caracteriza por su sencillez y consiste en generar un árbol de decisión de un solo nivel expresado por reglas. Se selecciona un atributo como nodo raíz del que parte una rama por cada valor terminando en un nodo hoja con la clase más probable obtenida de los ejemplos de entrenamiento. El pseudocódigo de este algoritmo es muy sencillo:

Para cada atributo (A)

Para cada valor del atributo (Ai)

Contar el número de apariciones de cada clase con Ai

Obtener la clase más frecuente (Cj)

Crear una regla de tipo $Ai \rightarrow Cj$

Calcular el error de las reglas del atributo A

Escoger las reglas con menor error

- PRISM. Es un algoritmo muy sencillo, asume siempre que los datos no llevan ruido. La entrada son los ejemplos y la clase sobre la que se hace el análisis, obteniendo un conjunto de reglas que se interpretan en el orden que indica el algoritmo.
 - PART. Este algoritmo en una primera fase crea las reglas de clasificación y en la segunda fase las mejora.
- c) **Redes neuronales artificiales.** Técnica que se basa en las redes neuronales del cerebro humano para modelar mediante computadoras el aprendizaje humano. Son de tipo anticipativo o paciente. Pueden solucionar problemas muy complejos, pero es difícil comprender los modelos que generan, así como extraer reglas de estos modelos. Las redes neuronales pueden detectar y aprender complejos patrones y características dentro de los datos. Son capaces de aprender

de la experiencia acumulada para resolver nuevos problemas gracias al adiestramiento (training). Poseen ambos tipos de aprendizaje, supervisado y no supervisado.

Están formadas por al menos tres niveles o capas formadas por nodos o neuronas: entrada, procesamiento u oculta y salida.

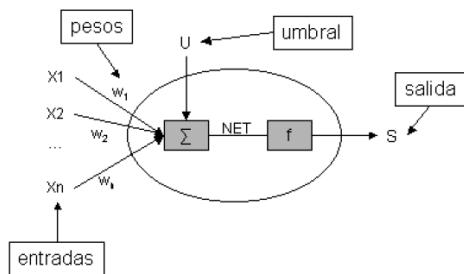


Figura 2.4 - Estructura de una neurona. Fuente: García, J., Molina, J.M. (2012)

Al umbral y los pesos se le da un valor constante al inicio, posteriormente durante el aprendizaje se cambia ese valor y la salida se define con las ecuaciones siguientes:

$$NET = \sum_{i=1}^N X_i w_i + U$$

$$S = f(NET)$$

La función f suele ser una función sigmoidal definida entre 0 y 1

$$f(x) = \frac{1}{1 + e^{-x}}$$

O bien entre -1 y 1

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Cada neurona es una unidad de procesamiento de la información que recibe y está conectada por medio de los pesos o dendritas a todas las neuronas de la capa anterior y posterior. La información recibida se procesa aplicando la función de

activación y se trasmite a las otras neuronas sólo si el resultado supera un determinado umbral U.

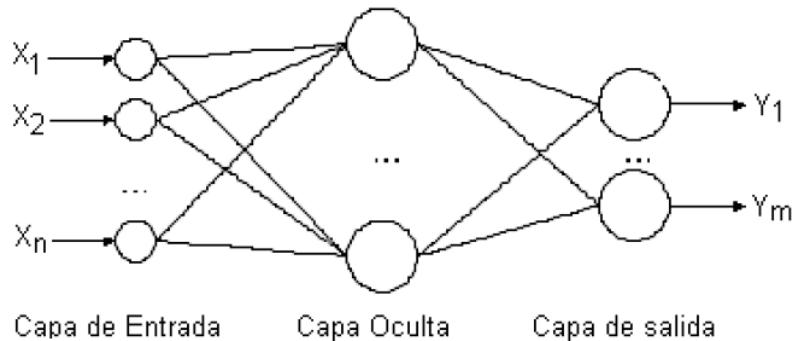


Figura 2.5 - Estructura de la red de neuronas. Fuente. García, J., Molina, J.M. (2012)

Los principales parámetros a tener en cuenta a la hora de diseñar una red neuronal es determinar el número de neuronas. Las correspondientes a las capas de entrada y salida las determina el propio problema y las de la capa o capas ocultas se hace a base de prueba y error.

Las redes neuronales tienen como objetivo aprender los valores más adecuados para los pesos, que son modificados durante el proceso de adiestramiento. Este proceso de adiestramiento viene determinado por el algoritmo y uno de los más conocidos se detalla a continuación:

- Backpropagation o retroprogramación. Según la diferencia entre la salida obtenida y la que se debería haber obtenido así se variarán los pesos. El método de descenso de gradiente modifica los parámetros de la red siguiendo la dirección negativa del gradiente del error. Para esto se hace con la siguiente fórmula:

$$w^{nuevo} = w^{anterior} + \alpha \left(-\frac{\delta e}{\delta w} \right) = w^{anterior} - \alpha \frac{\delta e}{\delta w} \quad (1)$$

El peso pasa de $w^{anterior}$ a w^{nuevo} que es el peso a modificar y α es la razón de aprendizaje encargada de controlar cuánto se desplazan los pesos en la dirección negativa del gradiente.

El algoritmo Backpropagation se calcula aplicando el método de descenso del gradiente a las redes de neuronas y tiene el pseudocódigo siguiente:

Paso 1. Inicialización aleatoria de los pesos y umbrales.

Paso 2. Dado un patrón del conjunto de entrenamiento $((x, t(x))$, se presenta el vector x a la red y se calcula la salida de la red para dicho patrón, $y(x)$.

Paso 3. Se evalúa el error $e(x)$ cometido por la red.

Paso 4. Se modifican todos los parámetros de la red utilizando la fórmula del descenso de gradiente (1).

Paso 5. Se repiten los pasos 2,3, y 4 para todos los patrones de entrenamiento, completando así un ciclo de aprendizaje.

Paso 6. Se realizan n ciclos de aprendizaje (pasos 2, 3, 4, y 5) hasta que se verifique el criterio de parada establecido.

Para establecer el criterio de parada se calcula la suma de los errores en los patrones de entrenamiento. Cuando el error sea constante de un ciclo a otro los parámetros no se modifican y se obtiene el error mínimo. Tras n ciclos de aprendizaje aparece el error en los patrones de validación, si este error evoluciona de forma favorable se continua con el proceso de aprendizaje, pero si el error no desciende se para el aprendizaje.

d) **Técnicas bayesianas.** Se basan en el teorema de Bayes para predecir la probabilidad condicional de la pertenencia de un objeto a una clase. El teorema de Bayes indica la probabilidad condicionada de un suceso aleatorio dado otro suceso. Son de tipo retardada o perezosa porque actúan en el momento de realizar la predicción para determinar la clase. A continuación, se expone un ejemplo de este tipo de algoritmo:

- Clasificador Naive Bayesiano. Consiste en conocer la hipótesis más probable utilizando el teorema de Bayes. Para hacer este cálculo se utiliza el teorema de Bayes:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

La probabilidad de que la hipótesis h sea cierta condicionada a los datos $P(h|D)$ va a ser igual a la probabilidad de los datos condicionada a la hipótesis $P(D|h)$ multiplicado por la probabilidad de la hipótesis $P(h)$, y dividido todo por la probabilidad de los datos $P(D)$.

El procedimiento a seguir para clasificar un nuevo ejemplo es, primero considerar tantas hipótesis como clases existan. Para cada hipótesis de pertenencia de el ejemplo a tratar a las diferentes clases posibles se calcula la probabilidad con la fórmula del teorema de Bayes. Finalmente, la clase cuya hipótesis tenga una probabilidad mayor será la clase a la que se asigna.

Los clasificadores Naive Bayesianos consideran que el valor de un atributo en una clase es independiente del valor del resto de atributos, esto se denomina independencia condicional de clase.

Una ventaja de este algoritmo es que, en caso de valores perdidos, cuando en un ejemplo aparece un atributo sin ningún valor este atributo no va a participar en el producto usado para el cálculo de probabilidades.

e) **Técnicas basadas en casos.** Para clasificar un nuevo ejemplo se compara con ejemplos ya clasificados dentro del conjunto de entrenamiento. Son de tipo retardada. Esa comparación se hace midiendo el parecido o la proximidad entre los ejemplos. Se considera más confiable cuando se trata de atributos numéricos que con atributos simbólicos. A continuación se enumeran algunos de los algoritmos más conocidos:

- K-vecinos más próximos (k-nearest-neighbors). Para clasificar un nuevo ejemplo se compara con los k-vecinos más próximos del conjunto de entrenamiento, la clase predominante de los k-vecinos será la clase del nuevo ejemplo.

Para ejecutar este algoritmo es necesario medir la distancia entre los objetos a clasificar. La medida de distancia entre dos instancias x_i y x_j dado un ejemplo de la n-tupla $(a_1(x), a_2(x), \dots, a_n(x))$, donde $a_r(x)$ es el valor de la instancia para el atributo a_r viene dada por la distancia euclídea:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^n (x_{il} - x_{jl})^2}$$

Se muestra en la figura 2.6 un ejemplo donde se ve el proceso de aprendizaje como almacena los ejemplos de entrenamiento, unos pertenecen a la clase roja y otros a la clase verde. Para realizar la clasificación del elemento estrella se mide la distancia con los 9 vecinos más próximos, porque se toma en este caso $k = 9$, cuatro elementos pertenecen a la clase roja y cinco a la clase verde, el elemento estrella se clasifica en la clase verde.

En este algoritmo también se puede asignar un peso o coste de los atributos que intervienen de esta forma los atributos poco relevantes se pueden anular. También permite considerar atributos sin valor.

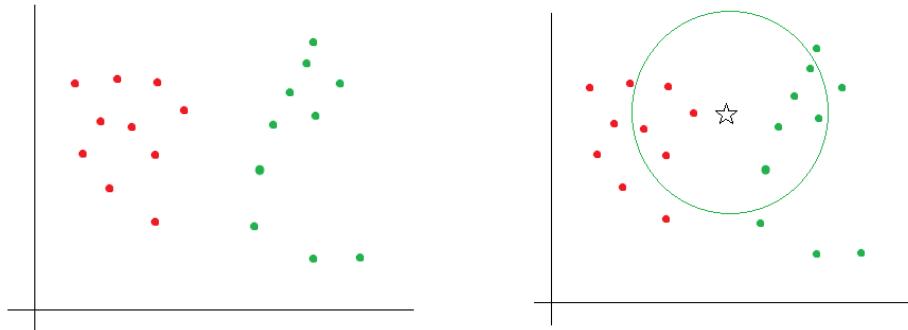


Figura 2.6 - Ejemplo de aprendizaje y clasificación k-vecinos más próximos

K^* . El algoritmo k estrella usa como medida de distancia entre ejemplares la teoría de la información. De otra forma se puede decir que la distancia entre dos ejemplares es directamente proporcional a la complejidad para transformar uno en otro.

- f) **Lógica borrosa.** (Fuzzy logic). Permite establecer clases donde la frontera entre unas y otras no es disjunta sino difusa, utiliza técnicas que permiten barreras difusas o suaves entre cada grupo clasificado utilizando el concepto matemático de distribución de posibilidad.
- g) **Técnicas genéticas.** Se basan en la biología, representando en un modelo matemático como los cromosomas, siguiendo la evolución, alcanzan la mejor estructura para la supervivencia. Los algoritmos genéticos usan técnicas biológicas de mutación y cruce para resolver problemas. Mutación es cuando un gen cambia aleatoriamente o de forma controlada mediante funciones y cruce sucede cuando la nueva solución procede de la contribución de otra solución padre. Los algoritmos genéticos transforman los problemas de búsqueda y

optimización de soluciones como un proceso de evolución de la solución de partida.

2.3.1.2. Regresión

Tarea predictiva que a diferencia de la clasificación el atributo a predecir es de tipo cuantitativo. Se utiliza la función de regresión para predecir el valor del atributo desconocido a partir de los valores conocidos de sus otros atributos. Hay dos tipos según el modelo generado: lineal y no lineal.

Regresión lineal. Los datos se modelan usando una línea recta. Se usa una variable aleatoria o variable respuesta y otra variable aleatoria llamada predictora, la primera es función lineal de la segunda.

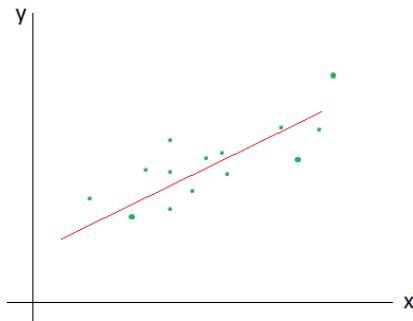


Figura 2.7 - Ejemplo de regresión lineal

La función es de la forma $y=a+bx$, donde a y b son los coeficientes de regresión que indican el corte con el eje de ordenadas y la pendiente de la recta respectivamente. Para su cálculo se utiliza el método de los mínimos cuadrados minimizando el error entre la recta y los datos reales.

Regresión no lineal. Se da cuando las variables tienen dependencia no lineal. Esta relación de las variables puede ser polinómica, en cuyo caso es necesario convertir la función a una forma lineal.

Dada la función polinómica: $y = a + b_1x + b_2x^2 + b_3x^3$

Aplicando:

$$x_1 = x, x_2 = x^2, x_3 = x^3$$

Se obtiene la ecuación:

$$y = a + b_1 x_1 + b_2 x_2 + b_3 x_3$$

La ecuación resultante ya se puede tratar como regresión lineal y predecir el valor del atributo desconocido.

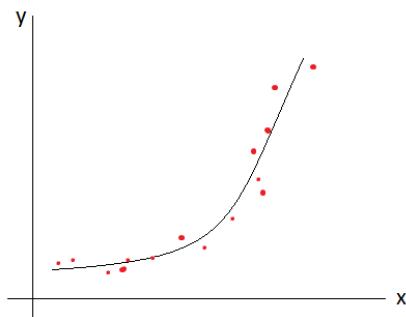


Figura 2.8- Ejemplo de regresión no lineal

2.3.2. Técnicas descriptivas o no supervisadas

Técnicas de minería de datos que generan modelos que describen los datos. A continuación, se explica cada una de ellas.

2.3.2.1. Agrupación o clústering

Consiste en identificar objetos pertenecientes a un grupo que sean semejantes entre sí y diferentes a objetos pertenecientes a otros grupos. Cada grupo de objetos similares se denomina clúster. Es una técnica muy utilizada en minería de datos ya que abarca gran cantidad de campos como el marketing, los seguros, economía, clasificación de documentos, astronomía entre otros.

Las principales características que se pide a los algoritmos de clústering son la escalabilidad, capacidad de trabajar con distintos tipos de variables, que pueda trabajar con datos con ruido y atípicos, que el orden en que están almacenados los datos no le influyan, que pueda operar con registros con muchos atributos y que sea fácil interpretar los resultados.

Para realizar técnicas de clústering se utilizan medidas de similaridad o distancia en el caso de atributos cualitativos y para atributos numéricos previamente se estandarizan para trabajar con ellos. Una medida de distancia muy utilizada es la distancia euclídea que se vio anteriormente al hablar del algoritmo de clasificación K-vecinos más próximos. También son utilizadas la distancia City- Block o Manhattan y la distancia Minkowski.

Los algoritmos de clústering aplican de diferente forma la medida de distancia al realizar la segmentación, esta característica origina grupos diferentes.

a) Clústering particional

- K-means (K-medias). Se trata de un algoritmo que se basa en particiones, es sencillo y muy utilizado. Consiste en elegir el número de clúster que se van a formar (k) y se seleccionan k elementos llamados centroides y que son el centro de cada clúster, esta selección puede hacerse al azar. Se van asignando todos los objetos al clúster donde la distancia entre el objeto y el centroide es menor. Una vez todos objetos están asignados se calcula el nuevo centroide de cada clúster, esto se hace calculando la media de los objetos de cada clúster si son numéricos o la moda si son simbólicos. El proceso es iterativo hasta que se llega al punto en que los centroides no cambian. El pseudocódigo del algoritmo K-medias consta de cuatro pasos:

Paso 1. Elegir k centroides para k número de clúster.

Paso 2. Asignar cada elemento al clúster cuyo centroide está más próximo.

Paso 3. Recalcular el centroide de cada clúster.

Paso 4. Si no se converge con dos iteraciones que no cambian la clasificación volver al paso 2.

- K-medoids. Al igual que el anterior pertenece al tipo de clústering particional y difiere del anterior en que cada clúster tiene un objeto llamado medoid que es el que ocupa la posición más centrada del clúster. También se puede seleccionar al inicio de forma aleatoria. El pseudocódigo de este algoritmo es:

Paso 1. Elegir k medoid de forma aleatoria.

Paso 2. Asignar cada elemento al medoid más cercano aplicando medida de distancia. Cada medoid junto con los elementos más próximos formar un clúster.

Paso 3. Calcular la distancia media de cada elemento al resto del clúster siendo el nuevo medoid el que minimice dicha distancia.

Paso 4. Finaliza si no se aprecian cambios en los medoids si cambian repetir pasos 2 y 3.

Este algoritmo se comporta mejor que el k-medias en caso de ruido o de elementos atípicos.

b) Clústering jerárquico

- Diana (DIvisive ANAlysis). Este algoritmo se encuadra dentro de las técnicas de clústering jerárquico que segmentan los datos a partir de una sucesión ordenada de clústers con estructura jerárquica que en caso de este algoritmo es descendente o divisiva porque va aumentando el número de clústers en cada iteración usando un proceso analítico llamado top-down. Se inicia con un solo clúster con todos elementos y se va segmentando hasta terminar con un objeto cada clúster.

- Agnes (DIvisive ANAlysis). Algoritmo localizado dentro de las técnicas de clústering jerárquico de tipo aglomerativo, ascendente o acumulativo. Sigue unos pasos inversos al divisivo, se inicia con un solo elemento en cada clúster para ir disminuyendo en cada paso el número de clústers. Es de tipo bottom-up, va construyendo clúster cada paso que son combinación del clúster más similar del anterior paso. El pseudocódigo de este algoritmo consta de los siguientes pasos:

Paso 1. Se asigna cada uno de los elementos a un clúster.

Paso 2. Se localizan el par de clústers más parecidos y se agrupan.

Paso 3. Calcular la distancia entre el clúster recién formado y el resto.

Paso 4. Repetir los pasos 1 y 2 hasta que quede un único clúster.

Las medidas de distancia utilizadas pueden ser:

- **Enlace simple o single link.** La distancia entre dos clústers será la distancia mínima entre todos los pares de objetos posibles de ambos clústers.
- **Enlace completo o complete-link.** La distancia será la distancia máxima entre los pares de objetos posibles.
- **Enlace promedio o average-link.** La distancia entre dos clústers es la media entre los pares posibles de objetos.

- c) **Clústering basado en densidad.** La segmentación se hace a partir de la densidad, es decir la cantidad de objetos alcanzables desde un objeto y un determinado radio. Se llama vecindad de un punto al número de objetos alcanzables desde ese punto y con un radio determinado.

Los parámetros principales de los algoritmos basados en densidad son radio o épsilon (Eps) y número mínimo de vecinos (MinPts).

Los objetos se clasifican en tres tipos:

- Puntos nucleares (core points), son los puntos que en su vecindario de radio Eps tienen igual o más vecinos a MinPts.
 - Puntos fronterizos (border points), tienen en su vecindario de radio Eps menor que MinPts pero que están en la vecindad de algún punto nuclear.
 - Puntos de ruido (noise points), el resto de puntos.
-
- DBSCAN es un algoritmo basado en densidad. El pseudocódigo de este algoritmo se muestra a continuación:

DBSCAN recibe el conjunto de objetos D, Eps y MinPts

C = 0

Para cada objeto P sin visitar del conjunto D

Marcar P como visitado

N = vecinos de P radio Eps

Si tamaño de N < MinPts

Marcar P como RUIDO

Entonces

C = siguiente clúster

expandClúster (P, N, C, Eps, MinPts)

expandClúster (P, N, C, Eps, MinPts)

Añadir P al clúster C

Para cada punto P' de N

Si P' no está marcado como visitado

Marcar P' como visitado

N' = vecinos de P', radio Eps)

Si tamaño N' >= MinPts

N = N U N'

Si P' no pertenece a ningún clúster

Añadir P' al clúster C

El algoritmo DBSCAN tiene buen comportamiento en presencia de ruido y no le influye el tamaño y forma de los clusters.

2.3.2.2. Detección de atípicos

Localiza objetos dentro de un conjunto que sean muy diferentes al resto de objetos del mismo grupo. En ocasiones es necesario detectar los objetos atípicos antes de realizar minería de datos para no perturbar los modelos obtenidos. Las técnicas utilizadas pertenecen a estos grupos:

- a) Aproximaciones estadísticas. Estas técnicas se basan en crear un modelo estadístico para representar una población. Comparando el objeto con el modelo creado se determina si ese objeto se considera atípico.
- b) Aproximaciones basadas en proximidad. Se considera que un objeto es atípico si está muy alejado del resto de objetos. Para aplicar estas técnicas se utilizan medidas de distancia. El algoritmo k-vecinos más próximos, visto anteriormente utiliza esta técnica.
- c) Aproximaciones basadas en densidad. Se considera atípico el objeto situado en zonas de baja densidad. El algoritmo DBSCAN aplica esta técnica.
- d) Aproximaciones basadas en clústering. Utiliza inicialmente técnicas de segmentación y una vez obtenidos los clústers se determinan los objetos atípicos. Su algoritmo consiste en identificar los objetos aislados en las zonas frontera de cada clúster para obtener los objetos atípicos.

2.3.2.3. Asociación

Tarea de minería de datos que crea reglas para relacionar los diferentes atributos existentes en el conjunto de datos, se denominan reglas de asociación. Consiste en obtener de un conjunto de datos todas las reglas de asociación que tengan un soporte y una confianza mayores a un umbral establecido por el usuario, estos ítems se denominan itemsets frecuentes o large ítems. El segundo paso es analizar los large ítems para generar las reglas de asociación.

Dados los large ítems ABCD y AB la confianza de la regla $AB \Rightarrow CD$ se calcula:

$$\text{Confianza}(AB \Rightarrow CD) = \frac{\text{soporte}(ABCD)}{\text{soporte}(AB)}$$

Se considera una regla válida aquella que la confianza supera el valor umbral establecido para ello.

- A priori. Es uno de los principales algoritmos que utiliza técnicas de asociación. Se basa en una propiedad llamada a priori y que dice que, si un conjunto de elementos de tamaño n no es frecuente, tampoco lo es un conjunto de tamaño n+1 que lo contenga.

2.3.3. Series temporales en data mining

A la hora de tomar decisiones el factor más importante que se utiliza es el de la predicción. Cuando un comerciante almacena un stock de productos en su tienda lo hace porque prevé que esos productos los va a vender en los próximos días, y para hacer esa predicción se basará en las ventas en días pasados, tendrá en cuenta los días festivos, y otras tantas variables que harán que su predicción sea más o menos certera.

Una serie temporal es una serie de datos de una determinada variable tomados sus valores en orden cronológico y a intervalos regulares que pueden ser horas, minutos, días, etc.

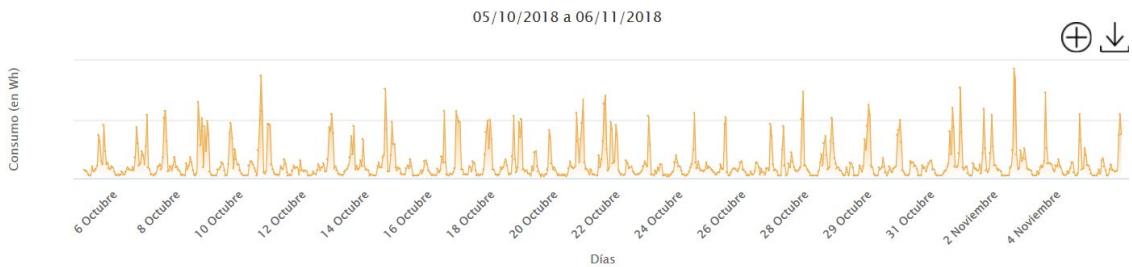


Figura 2.9 - Ejemplo de serie temporal. Fuente <https://www.iberdroladistribucionelectrica.com>

Para realizar un análisis de series temporales primero es necesario construir un modelo que representa una serie en el tiempo y segundo utilizar ese modelo para realizar las predicciones futuras de esa variable.

Para poder estudiar el comportamiento de una serie temporal a partir de un conjunto de datos obtenidos a lo largo del tiempo, se representa en un eje cartesiano, el tiempo es el eje de abscisas y en el eje de ordenadas se ponen los valores de la variable a estudiar, se unen todos los puntos y así se obtiene la representación gráfica de todos los valores tomados a lo largo del tiempo. El estudio de esta gráfica aporta información del comportamiento de dicha serie.

Una serie temporal se puede descomponer en otras componentes básicas para poder realizar su estudio:

- Tendencia. Refleja la evolución de la serie a largo plazo.
- Componente estacional. Refleja oscilaciones de la serie en períodos inferiores o iguales a un año.
- Componente cíclica. Refleja las oscilaciones periódicas cuya amplitud es superior a un año. Es necesario disponer de una serie histórica bastante grande.
- Componente aleatorio o irregular. Refleja las oscilaciones de la serie que son debidas a acontecimientos imprevisibles. Como bien se indica es imprevisible y en todas series temporales aparece.

Las cuatro componentes pueden seguir un esquema aditivo, multiplicativo o mixto para conformar la componente global de la serie.

- Esquema aditivo: se produce cuando la serie temporal es el resultado de la suma de las cuatro componentes, esto significa que las componentes son todas independientes.
- Esquema multiplicativo: se presenta cuando todas las componentes tienen relación entre sí porque no son independientes.
- Esquema mixto: se produce este esquema cuando alguna componente no es independiente. Normalmente la componente aleatoria es imprevisible, por tanto, siempre que exista se da un esquema mixto.

Para la descomposición de una serie temporal existen diferentes métodos que son propios del análisis descriptivo, el método clásico de descomposición y el método de descomposición STL.

Las principales técnicas de minería de datos se centran en resolver problemas que presentan las series temporales y se indican en los siguientes puntos:

- Comparación de dos series temporales. Consiste en representar el grado de parecido entre dos series temporales, utilizando medidas de distancia o similaridad. Dependiendo de las técnicas utilizadas se pueden tener los siguientes enfoques:
 - Enfoque basado en el uso de transformadas matemáticas. Consiste en representar las series por medio de los coeficientes de alguna transformada matemática. Estos coeficientes deben ajustarse lo máximo posible a la serie temporal. Comparando los coeficientes de la transformada de ambas series está la solución al problema.
 - Enfoque basado en el uso de transformaciones en las series temporales. Consiste en aplicar transformaciones a una serie

temporal hasta conseguir alcanzar el máximo parecido con la serie a comparar. Entonces se consideran las dos series temporales similares.

- Análisis de subsecuencias. Subsecuencia es un fragmento dentro de una secuencia de una serie temporal. Existen técnicas que localizan subsecuencias que se encuentran dentro de una serie o subsecuencias que se repiten dentro de un conjunto de series temporales.
- Obtención de modelos de referencia. Consiste en representar mediante una serie temporal un conjunto de series temporales de la forma más parecida. Se puede utilizar la técnica de transformada matemática calculando sus coeficientes para luego calcular la media y construir el modelo de referencia.
- Análisis de eventos. Consiste en centrarse solamente en una región de una serie temporal que contiene determinados eventos que son los que tienen un interés principal. Una técnica consiste en dividir la serie temporal en subsecuencias, repitiendo este proceso hasta cumplir una condición de parada.

El proceso de modelización de series temporales puede variar según la metodología que se vaya a utilizar. En todos casos el primer paso consiste en dibujar la gráfica como se comentó al principio, para estudiar comportamiento, componentes y determinar si es estacionaria o no lo es. La siguiente etapa consiste en identificar el posible modelo a usar, la siguiente fase es la de estimación del modelo y la última fase la validación del modelo.

Una vez superada la fase de validación del modelo se pasará a la fase de explotación del modelo.

2.3.3.1. Técnicas de transformación de series temporales-

Se denomina serie temporal agregada a la serie obtenida de una transformación y si se asocia a un alfabeto finito se llama serie temporal simbólica. Una vez realizadas estas

transformaciones se pueden aplicar técnicas de minería de datos como clasificación, asociación, clústering y detección de atípicos.

Las condiciones que debe cumplir una transformación son tres: primero, regla de la reducción de datos, segundo los extremos coinciden y tercero la distancia entre la serie original y la aproximada debe ser menor que el umbral.

Clasificando las técnicas de transformación según la secuencia que generan habrá dos categorías: estrategias de representación de series temporales numéricas y de series temporales simbólicas.

2.3.3.1.1. Transformaciones numéricas.

I. Transformada de Fourier discreta (DFT).

Basada en el teorema de Parseval que dice que la energía de una señal es la misma si se mide en el dominio de la frecuencia que en el dominio del tiempo. La distancia euclíadiana de una serie temporal medida en el dominio del tiempo es igual a la distancia de la misma serie medida en el dominio de la frecuencia.

$$\text{Teorema de Parseval} \quad \sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |\vec{x}_f|^2$$

Es interesante teniendo en cuenta el supuesto que afirma que solo las frecuencias de los primeros índices representan a la serie temporal con un alto grado de aproximación logrando una alta reducción de la dimensionalidad.

Dada una serie temporal $X = \{x_0, x_1, \dots, x_n\}$

$$\text{su DFT es } \vec{x}_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t e^{-j2\pi ft/n} \quad f = 0, 1, \dots, n$$

Siendo la fórmula para la reconstrucción de la serie original en el dominio del tiempo $x_t = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} \vec{x}_f e^{-j2\pi ft/n} \quad t = 0, 1, \dots, n$

$\vec{X} = \{\vec{x}_0, \vec{x}_1, \dots, \vec{x}_n\}$ representa X en el dominio de la frecuencia y $j = \sqrt{-1}$

Las series obtenidas aplicando esta transformación son de menor dimensionalidad que la original, manteniendo un alto grado de aproximación.

Variantes de esta transformación son la Transformada de Coseno Discreta (TCD) que utiliza la función coseno para su definición y la Transformada de Seno Discreta (TSD) que utiliza la función seno para su definición.

II. Transformada Wavelet discreta (DWT).

Es una transformación que toma las muestras de onda (wavelets) en periodos discretos de tiempo manteniendo la resolución temporal.

Las ventajas de DWT son:

- Ofrece buena aproximación a la serie original tomando los primeros coeficientes.
- El tiempo computacional para el cálculo es proporcional al tamaño de la serie temporal.
- Mantiene la distancia euclíadiana.
- Captura información relacionada con el tiempo y con la frecuencia.

III. Descomposición de valor sencillo (SVD).

Es una técnica de transformación que interviene en todo el conjunto de datos y lo rota de manera que sus ejes ortogonales entre sí, produzcan las máximas varianzas posibles.

Representa un conjunto de series temporales como una matriz N x M donde N es el número de series temporales y M el tamaño de cada una. Se trata de agrupar series temporales similares (dimensión N) y momentos parecidos (dimensión M). Este algoritmo crea grupos correspondientes a patrones, cada grupo reúne momentos que representan buena característica para usar con alto poder de discriminación y ortogonal al resto de grupos.

Su principal ventaja es la eficacia y eficiencia recuperando resultados, pero como desventaja tiene el alto consumo de recursos al construir el índice ya que debe

reconstruirse cada vez que se modifica la base de datos. El coste computacional es de $O(n^3)$.

IV. Piecewise Aggregate Approximation (PAA).

Consiste en dividir la serie en un conjunto finito de segmentos del mismo tamaño almacenando las medias de los valores que corresponden con los puntos de cada segmento.

En la figura 2.10 se aprecia la reducción de dimensionalidad utilizando PAA, se pasa de los 128 valores de la serie original a los 8 valores de la transformada.

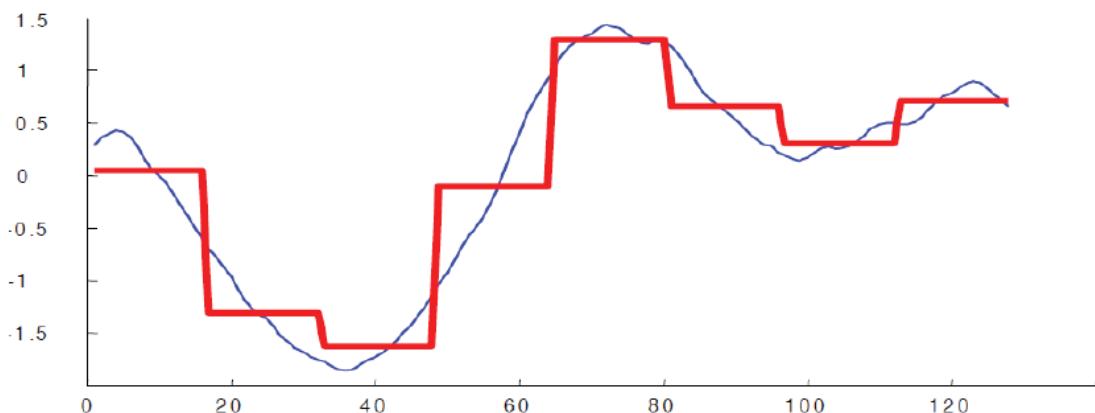


Figura 2.10 - Transformación de una serie temporal utilizando PAA. Fuente: Valencia, B. (2018).

V. Adaptive Piecewise Constant Approximation (APCA).

Se diferencia de PAA que los intervalos de agregación pueden tener longitudes variables. Esto repercute en una reducción de dimensionalidad de la serie transformada.

2.3.3.1.2. Transformaciones simbólicas

Transforman una serie temporal con valores numéricos en una serie temporal de símbolos, se presentan a continuación algunos de los métodos más representativos.

I. Shape Definition Language (SDL).

Método que extrae el comportamiento de la serie temporal en un tiempo dado, muestra si la serie está estable, crece, decrece, u otro comportamiento. Esta técnica permite al usuario preocuparse de la serie temporal y no de otros detalles específicos.

En la figura 2.11 Representación de una serie temporal que indica doce valores a lo largo de un año.

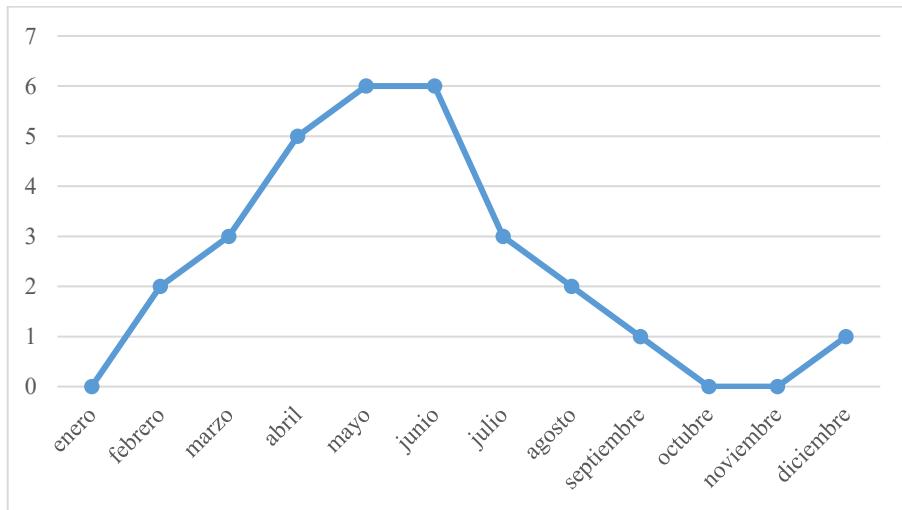


Figura 2.11 - Representación serie temporal

El alfabeto que se va a utilizar para la representación SDL de la serie temporal es el indicado en la tabla 2.2, indicando las columnas ‘li’ y ‘ls’ los límites inferior y superior del valor de la variación desde el valor inicial al final de la transacción. Las columnas ‘vi’

y ‘vf’ especifican las restricciones del valor inicial y final de la transición pudiendo tomar los valores cero, no cero y cualquier valor.

Tabla 2.2 - Alfabeto para SDL

Símbolo	Descripción	li	ls	vi	vf
zero	Valor inicial y final =0	0	0	zero	zero
stable	Valor inicial parecido al final	-0,5	0,5	anyvalue	anyvalue
down	Disminución ligera	-1,9	-0,6	anyvalue	anyvalue
Down	Disminución grande	-10	-2	anyvalue	anyvalue
up	Aumento ligero	0,6	1,9	anyvalue	anyvalue
Up	Aumento grande	2	10	anyvalue	anyvalue
appears	Transición de valor 0 a no 0	0	10	zero	non zero
disappears	Transición de valor no 0 a 0	-10	0	non zero	zero

La representación de la serie temporal aplicando el método SDL y utilizando el alfabeto indicado en la tabla 2.2 será:

appears up Up up stable Down down down disappears zero appears

II. Shape Description Alphabet (SDA).

Está basado en el mismo enfoque que el método SDL, transforma las series temporales en caracteres que podrán ser tratados como texto. Los pasos a seguir son, primero se traduce la serie en una cadena de texto y segundo se ejecuta a través de un generador de archivos de firma que crea un archivo de firma del texto deslizando una ventana por toda la cadena, mapeando la cadena para cada ventana, en unos bits de firma.

Se trabaja con un alfabeto basado en el SDL que se denomina Shape Description Alphabet (SDA) que puede ser según se especifica en la tabla 2.3, donde los valores lvalue y hvalue dependen del dominio de la aplicación y sirven para mapear los símbolos según el valor de cada atributo se incrementa en cada instante de tiempo.

Tabla 2.3 - Alfabeto SDA

Símbolo	Descripción	lvalue	hvalue
a	Transición altamente creciente	2	-
u	Transición ligeramente creciente	1	1,9
s	Transición estable	-0,99	0,99
d	Transición ligeramente decreciente	-1,9	-1
e	Transición altamente decreciente	-	-2

Aplicando el alfabeto SDA a la figura 2.10 del método anterior la representación SDA será: **a u a u s e d d d s u.**

III. Aproximación Agregada Simbólica (SAX).

Su objetivo es transformar una serie temporal en una secuencia de símbolos formando una palabra de longitud predeterminada. Tiene la ventaja frente a las vistas anteriormente que reduce la dimensionalidad de las series temporales mientras conserva los principales caracteres de forma. Se procede en dos pasos:

- Se transforma a una secuencia de segmentos mediante PAA. De forma que una serie con n elementos numéricos se transforma en una de p elementos donde cada elemento vale la media de los valores n/p presentes en el rango.
- Se transforma a símbolos discretos los segmentos obtenidos de PAA. Se hace dividiendo el espacio en regiones con igual probabilidad, según una distribución de Gauss, esto se hace por medio de una función que recibe dos parámetros: 'w' que es el número de dimensiones de la serie transformada denominada longitud de la palabra y 'a' que es el número de regiones, por tanto, habrá 'a-1' puntos de corte.

Si se toma la serie temporal de la figura 2.10 en la que se ha utilizado PAA para su transformación se puede ver que el método SAX proporciona la siguiente representación:

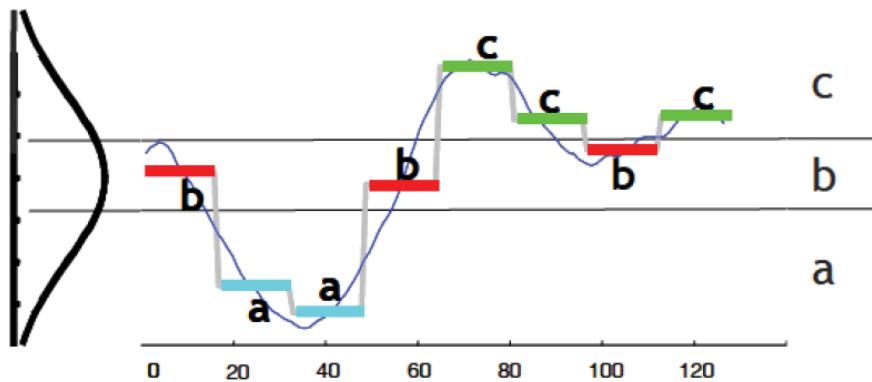


Figura 2.12 - Representación de la serie temporal usando método SAX. Fuente: Valencia, B. (2018).

El algoritmo recibe los dos parámetros, ‘word’ que es el tamaño de la palabra final resultante de la transformación y ‘a’ que es el número de regiones, en este caso vale 3. Se identifica cada intervalo de la serie transformada con PAA y se le asigna una letra resultando en el ejemplo la siguiente secuencia: **b a a b c c b c**.

2.4. Agricultura

2.4.1. La agricultura en Castilla y León

Castilla y León es una comunidad autónoma compuesta por nueve provincias: Ávila, Burgos, León, Palencia, Salamanca, Segovia, Soria, Valladolid y Zamora. Es la comunidad autónoma más extensa de España, con una superficie de 94 226 km², y tiene 2.425.801 habitantes según datos del INE del año 2017.

La superficie agrícola es de 5.783.831 hectáreas que es más de la mitad de la superficie total, el 90 por ciento dedicado a cultivo de secano. El terreno mayormente es árido y seco debido al clima y las pocas lluvias, pero fértil.

Hay alrededor de 90.000 explotaciones agrícolas, lo cual representa aproximadamente un 10 por ciento de la población activa de toda la comunidad, es por esto un sector muy importante.

Según datos procedentes del Ministerio de Empleo y Seguridad Social el número de afiliados a la seguridad social en el sector agrario en Castilla y León es muy variable según la época del año, se representa en la figura 2.13.

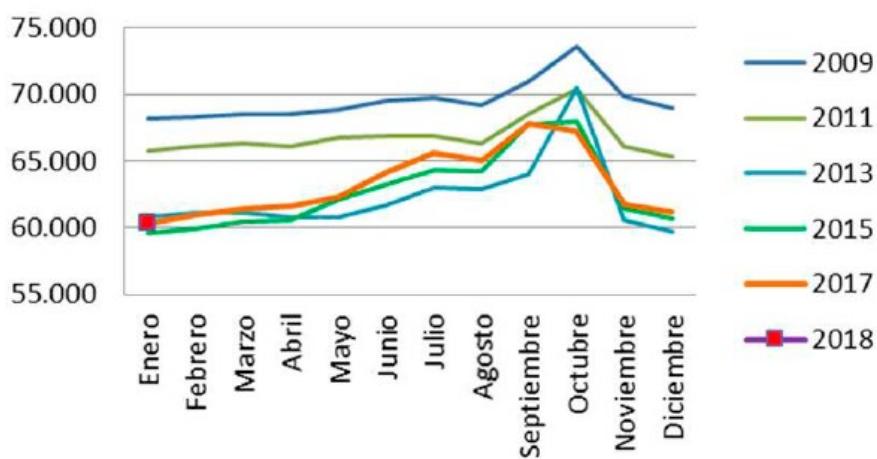


Figura 2.13 - Afiliados a la S.S. sector agrario en Castilla y León. Fuente: www.jcyl.es

En cuanto a la producción agrícola de esta comunidad representa el 15 por ciento del total de la nación. Tradicionalmente se ha considerado a Castilla el granero de España.

Los principales cultivos por número de hectáreas cultivadas y producción son: trigo, cebada, centeno, avena y triticale. Otros cultivos menos extendidos, pero con importancia son el girasol y la colza y las leguminosas como lentejas, garbanzos, guisantes, yeros y veza.

El cultivo del viñedo ha descendido en las últimas décadas, rondando las 56.330 hectáreas de superficie cultivada.

En la tabla 2.4 se observa como en el año 2016 la mayor parte de la superficie de secano se dedicó a la siembra de trigo con 804.972 hectáreas seguido de cebada con 717.870 hectáreas, seguido por otros cereales como avena, centeno y triticale a distancias muy considerables. La producción de trigo ese año rondó los 3,5 millones de toneladas

con un rendimiento de 4.231 kg por hectárea, teniendo en cuenta sólo la superficie de secano. La producción de cebada fue cercana a los 3 millones de toneladas en las tierras de secano con un rendimiento de 4.097 kg por hectárea.

Los rendimientos del resto de cereales son menores, para la avena de secano es de 3.990 kg por hectárea, 2.693 para el centeno y 3.605 para el triticale.

Tabla 2.4 - Rendimiento, superficie y producción de cereales grano en Castilla y león 2016.

	Superficie (ha)			Rendimiento (Kg/ha)			Producción (t)	
	Sedano	Regadío	Total	Secano	Regadío	Grano	Biocombustible	Paja cosechada
CEREALES DE INVIERNO								
Trigo duro	2.544	1.903	4.447	4.000	5.421	20.493	-	11.990
Trigo semiduro y blando	802.428	98.028	900.456	4.232	5.752	3.959.810	29.057	2.283.324
TRIGO TOTAL	804.972	99.931	904.903	4.231	5.746	3.980.303	29.057	2.295.314
Cebada de 6 carreras	58.398	4.228	62.626	3.789	4.660	240.989	-	135.910
Cebada de 2 carreras	659.472	64.228	723.700	4.125	5.153	3.050.959	-	1.780.751
CEBADA TOTAL	717.870	68.456	786.326	4.097	5.122	3.291.948	-	1.916.661
Avena	77.803	9.391	87.194	3.390	3.927	300.638	-	157.997
Centeno	96.543	5.116	101.659	2.693	4.165	281.299	-	149.035
Triticale	28.805	1.916	30.721	3.605	4.907	113.252	-	62.520
CEREALES DE PRIMAVERA								
Maíz	-	102.053	102.053	-	10.567	1.078.409	10.360	-
Sorgo	72	132	204	3.538	6.815	1.154	-	-
OTROS CEREALES	438	393	831	-	-	2.621	-	1.228
TOTAL CEREALES	1.726.503	287.388	2.013.891	-	-	9.049.625	39.417	4.582.755

Es importante destacar cómo evoluciona la producción de estos cultivos cada año. Estas variaciones son consecuencia directa de la climatología.

Tabla 2.5 - Cereales grano. Evolución superficie y producción

Años	Superficie (miles ha)	Producción (miles t)
2003	2.190	7.012
2004	2.224	8.188
2005	2.291	5.186
2006	2.130	6.081
2007	2.122	8.948
2008	2.367	10.396
2009	1.967	4.969
2010	1.981	7.042
2011	2.000	7.803
2012	2.014	5.993
2013	2.037	8.747
2014	2.064	6.689
2015	1.990	7.002
2016	2.014	9.050

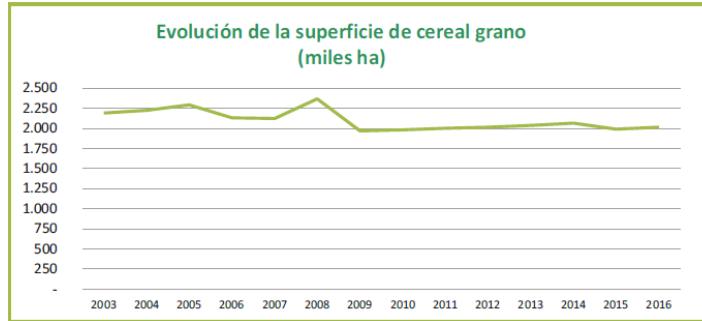


Figura 2.14 - Cereales grano. Evolución superficie cultivada. Fuente: www.jcyl.es

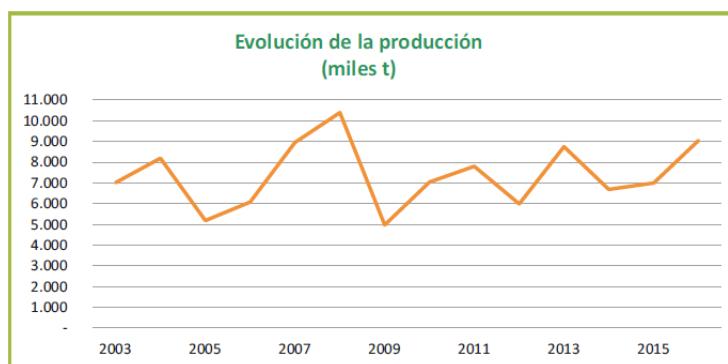


Figura 2.15 - Cereales grano. Evolución producción. Fuente: www.jcyl.es

En la figura 2.16 se observa cómo ha variado el rendimiento de los cereales grano entre los años 2003 al 2016. Se aprecia claramente la diferencia de producción de unos años a otros.

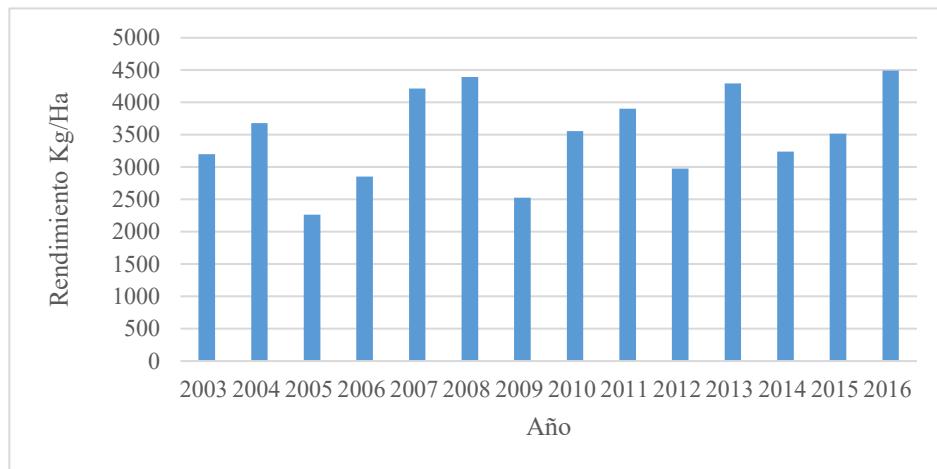


Figura 2.16 - Variación del rendimiento del cereal grano.

En las tablas 2.6 y 2.7 aparecen otros cultivos de menos relevancia, a excepción del girasol que ha ocupado en 2016 unas 220.000 hectáreas. La colza, guisantes secos, yeros y veza ocupan superficies bastante más reducidas.

Tabla 2.6 - Rendimiento, superficie y producción de leguminosas grano en Castilla y león 2016.

	Superficie (ha)			Rendimiento (Kg/ha)			Producción (t)	
	Secano	Regadio	Total	Secano	Regadio	Grano	Paja cosechada	
Judías secas	-	4.521	4.521	-	2.283	10.323	2.708	
Habas secas	1.138	67	1.205	2.469	2.379	2.969	10	
Lentejas	5.820	304	6.124	1.198	1.382	7.393	1.092	
Garbanzos	5.601	523	6.124	813	1.177	5.171	1.266	
Guisantes secos	34.970	4.573	39.543	2.206	3.143	91.508	8.567	
Veza	40.680	2.378	43.058	1.307	1.680	57.147	17.668	
Altramuz	531	3	534	570	1.103	306	7	
Yeros	10.590	306	10.896	1.433	2.196	15.846	2.200	
OTRAS LEGUMINOSAS	14.185	404	14.589			20.393	4.282	
TOTAL LEGUMINOSAS	113.515	13.079	126.594	-	-	211.056	37.800	

Tabla 2.7 - Rendimiento, superficie y producción de cultivos industriales en Castilla y león 2016.

	Superficie (ha)			Rendimiento (Kg/ha)		Producción	
	Secano	Regadio	Total	Secano	Regadio	(t)	
AZUCARERAS	-	22.494	22.494	-	-	2.037.242	
Remolacha	-	22.494	22.494	-	90.568	2.037.242	
OLEAGINOSAS	256.160	36.675	292.835	-	-	370.721	
Girasol	220.112	28.418	248.530	878	2.215	256.190	
Cártamo	907	52	959	786	1.823	808	
Soja	21	262	283	3.000	2.828	804	
Colza	34.516	7.916	42.432	2.413	3.659	112.272	
Camellina	604	27	631	1.016	1.226	647	
CONDIMENTOS	45	8	53	700	3.613	60	
Pimiento para pimentón	-	8	8	-	3.613	29	
Anís	45	-	45	700	-	32	
VARIOS	1.593	2.406	3.999	3.615	52.025	13.009	
Tabaco	-	60	60	-	3.190	191	
Lúpulo	-	496	496	-	1.791	888	
Achicoria (raíz en verde)	-	130	130	-	42.054	5.467	
Lavanda y lavandin	514	63	577	2.378	2.738	1.395	
Adormidera	1.079	1.657	2.736	1.237	2.252	5.067	
OTROS CULTIVOS INDUSTRIALES	6	1	7			20	
TOTAL	257.804	61.584	319.388	-	-	2.421.052	

2.4.2. Principales cultivos, características principales

Cada cultivo necesita unas condiciones que influirán en su crecimiento y que ofrecen rendimientos según las características del terreno, temperaturas, precipitaciones, abonado, semilla, etc. De todas ellas se reflejan aquí el nivel de precipitaciones y la temperatura pues son unas características que a priori, al realizar la siembra, son totalmente desconocidas para el agricultor.

Aclarar antes el concepto de integral térmica que se va a utilizar posteriormente y que se refiere a la suma de los grados necesarios para que la planta complete su estado fenológico o ciclo completo, su unidad es grados-día.

2.4.2.1. Cereales

a) Trigo

Es el cereal que ocupa más superficie en la tierra y fue de los primeros que adoptó el ser humano, comenzó en Oriente Próximo para extenderse por Europa, la India y finalmente por el resto del mundo.



Figura 2.17 - Imagen de trigo. Fuente: www.freepik.es

Principales condiciones para su crecimiento:

Exigencias de temperatura y agua.

Necesita de al menos 2ºC-3ºC para su germinación y ahijamiento, por debajo de cero grados no germina. La temperatura óptima sería entre 20ºC y 22ºC.

La integral térmica del trigo es variable y oscila entre 1850 ºC y 2375 ºC. En la siembra nascencia rondaría los 121 ºC.

Las temperaturas idóneas para el encañado son 10ºC-12ºC, para la floración 15ºC y para la maduración entre 18ºC y 20ºC.

El descenso de temperatura en invierno favorece el desarrollo de las raíces. Las altas temperaturas son perjudiciales.

La siembra es un periodo crítico ya que es necesario el agua, durante el ahijado el agua no es importante, incluso puede ser perjudicial si se produce encharcamiento.

En la época del encañado, floración y desarrollo del grano es necesario la presencia de agua. Este periodo denominado meseta comprende desde que el grano está lechoso hasta que se vuelve pastoso y la presencia de agua es determinante para obtener un buen rendimiento.

El trigo tiene un coeficiente de transpiración de 450-550, esto indica que son necesarios 450-550 litros de agua para elaborar 1 Kg de materia seca. Con precipitaciones entre 300 y 400 mm durante años secos la planta se desarrolla correctamente si el reparto se produce con pocas precipitaciones en invierno, pero más abundantes en primavera.

Una cantidad de lluvia entre 500 a 600 mm sería la correcta, siempre que en invierno sea más escasa y más abundante en primavera.

b) Cebada

Este cereal es el más cultivado en España, el segundo en Europa y el cuarto a nivel mundial tras trigo, arroz y maíz. Su cultivo se inició al igual que el trigo en Oriente Próximo nueve mil años antes de Cristo.

Teniendo en cuenta las hileras con granos que tiene la espiga se distingue cebada de dos carreras y de seis carreras. Según el uso que se haga de ella podrán ser cebadas cerveceras o de pienso.

Exigencias de temperatura y agua.

Es un cereal de invierno y aguanta el frío peor que el trigo, en cuanto al calor necesita menos que el trigo, respecto a esta característica se encuentra entre el trigo y la avena.



Figura 2.18 - Imagen de cebada. Fuente: <https://www.flickr.com>

Posee un coeficiente de transpiración mayor al del trigo, pero su ciclo es más corto, sus mayores necesidades de agua se dan en primavera cuando se encuentra en desarrollo, y necesita menos agua en las últimas fases. Este cultivo aguanta mejor las zonas áridas y la sequía que el trigo y la avena.

c) Avena

El cultivo de avena ha ido perdiendo peso desde el último cuarto del siglo pasado fundamentalmente debido a la disminución de la ganadería equina. Posee raíces muy desarrolladas en anchura y longitudinalmente. Posee un bajo peso específico, unos 1.000 granos de avena pesan entre 25 y 35 gramos. En cuanto al ciclo vegetativo es parecido a otros cereales de invierno.

Exigencias de temperatura y agua.

Se cultiva en zonas templadas, frescas y húmedas y es menos resistente que el trigo si se dan temperaturas bajas. Se siembra normalmente en primavera y precisa temperaturas más bajas que el trigo.



Figura 2.19 - Imagen de avena. Fuente: <https://www.flickr.com>

Precisa de climas frescos y húmedos y necesita bastante cantidad de agua, es menos resistente a la sequía que la cebada o el trigo, además cuando se produce el llenado de los granos si no se aporta suficiente agua puede sufrir daños.

d) Centeno

Cultivo cuyo origen fue Oriente Próximo y el área mediterránea oriental. Apareció posteriormente al trigo y la cebada. Tiene menor relevancia a nivel mundial que el trigo,

cebada, arroz y maíz y en España ha disminuido últimamente la superficie cultivada de centeno. El grano es más estrecho y alargado que el de trigo. Se puede recolectar como forraje o cuando está seco. Del grano de centeno se saca harina para hacer pan, aunque de peor calidad que la de trigo pero que tiene la ventaja de conservarse fresco dos días. Se cultiva en suelos pobres y pedregosos. Aunque las producciones suelen ser bajas son mejores que si se cultivase otro cereal en la misma zona y mismas condiciones.

Exigencias de temperatura y agua.

Se trata de un cereal de invierno. Es resistente a las bajas temperaturas y su exigencia térmica es menor que la del trigo o la cebada.

Se caracteriza por ser más resistente a la sequía que el trigo.



Figura 2.20 - Imagen de centeno

e) Triticale

Es un cereal creado a partir del cruce de trigos con centeno. Aunque sus comienzos fueron en el siglo XIX no fue hasta el último cuarto del siglo XX cuando se comenzaron a obtener buenos resultados. Concretamente se obtienen de hibridación de trigos con centeno para posteriormente realizar duplicación cromosómica.

Este cultivo proporciona rendimientos iguales e incluso superiores al del trigo salvo que se destina a la alimentación animal ya que no alcanzan buena calidad para panificación.



Figura 2.21 - Imagen de triticale. Fuente: <https://www.flickr.com>



Figura 2.22 - Imagen de granos de trigo, centeno y triticale. Fuente: <http://www.wikiwand.com>

Exigencias de temperatura y agua.

Este cereal suma las buenas características del trigo y del centeno, en cuanto al clima es menos exigente que el trigo y se adapta al frío, altitud y acidez del suelo mejor que el trigo. Su rendimiento puede superar al del trigo o al del centeno.

2.4.2.2. Leguminosas

a) Garbanzos



Figura 2.23 - Imagen de garbanzos. Fuente: <https://www.publicdomainpictures.net>

Pertenece a las leguminosas, sus primeros cultivos fueron en el Medio Oriente en la zona del Creciente Fértil hace más de tres mil años antes de Cristo, aunque hay rastros de su existencia hace más de cinco mil años antes de Cristo. Se considera un arbusto. Se destinan prácticamente, en su totalidad para consumo humano.

Exigencias de temperatura y agua.

Soportan en invierno -10 °C durante periodos de 3 a 4 días, para su desarrollo las temperaturas diurnas deben rondar los 21°C a 26,5 °C y las nocturnas entre 18°C y 21 °C.

Su necesidad hídrica está entre 300 mm y 450 mm Permitiendo su cultivo en zonas secas, en cuanto a otras leguminosas si las precipitaciones son por debajo de las indicadas permitiría el cultivo de lentejas o yeros y si fuesen superiores permitiría el cultivo de habas.

b) Guisante

Su cultivo se inició entre los milenios seis y siete antes de Cristo por las tierras del Cercano Oriente, en épocas parecidas al trigo, cebada y lenteja. Su uso como guisante verde comenzó en el siglo XVI en Inglaterra.

Cada vaina puede alojar entre cuatro y nueve granos dependiendo de la variedad y de las condiciones climáticas y pueden tener colores diversos como amarillo, crema o verde y las formas varían ya que puede ser redonda o irregular.

En verde se utiliza fundamentalmente para el consumo humano y en grano seco para consumo animal.



Figura 2.24 - Imagen de guisante. Fuente: <https://www.flickr.com>

Exigencias de temperatura y agua.

Se cultiva en zonas con temperaturas entre 7 °C y 30 °C. Sus temperaturas óptimas para el desarrollo y la reproducción se sitúan entre 16°C y 21°C las diurnas y entre 10 °C y 16°C para las nocturnas. Entre 4°C y 5°C no hay crecimiento y las temperaturas altas, superiores a 30°C son perjudiciales sobre todo mermando la calidad del grano verde.

La integral térmica de este cultivo es de 650-700 grados-día para los cultivos tempranos y de 900 a 1050 grados-día para los tardíos.

En zonas húmedas su periodo de cultivo es de unos 150 días y en zonas semiáridas se reduce a un periodo comprendido entre 80 y 100 días.

En casos de sequía el tamaño del guisante apenas varía, pero el número de vainas se reduce en la misma cantidad de superficie. Para el uso hortícola se suele utilizar regadío dejando el cultivo en grano seco para tierras de secano.

El periodo transcurrido desde la floración hasta el llenado de vainas es cuando es más necesaria la aportación de agua y en caso de no producirse será muy perjudicial para la planta.

c) Lentejas

Esta leguminosa se cultivó en sus orígenes en la zona del Creciente Fértil y el sur de Turquía en la misma época que el trigo y la cebada. Se dedican para consumo humano.

Exigencias de temperatura y agua.

Se cultiva en climas templados y cálidos. Es necesario para su madurez un periodo de al menos 110 días que no se produzcan heladas. Se adapta bastante bien a climas frescos posponiendo la siembra a la primavera cuando los inviernos son muy fríos.



Figura 2.25 - Imagen de granos de lentejas

Fuente: <https://www.freepik.es>

Para que germine la semilla la temperatura debe superar los 15°C siendo temperaturas entre 18°C y 25°C cuando la germinación es más favorable.

Es un cultivo de secano que necesita más cantidad de agua durante la floración. Para las semillas grandes es necesario más aporte hídrico que para las de tamaño menor.

d) Veza

Leguminosa de dudosa procedencia se piensa que es originaria del Cáucaso. Se recolecta en grano seco o en verde como forraje. Se destina para la alimentación de animales tanto en grano como en forraje.

Exigencias de temperatura y agua.

Si se cultiva en zonas cálidas se usan variedades de ciclo corto para la producción en grano y variedades de ciclo medio-largo para producción de forraje.

Se cultiva en climas templados-húmedos, la planta es sensible al frío, cuando más en el estado de plántula y en el estado desarrollado. Si la temperatura es inferior a -5°C la planta comienza a sufrir daños. Entre 15°C y 20°C es su mejor temperatura para el desarrollo de la planta. Durante la floración los golpes de calor son perjudiciales para la planta.



Figura 2.26 - Imagen de veza

Fuente: www.wikimedia.org

Cuando los inviernos son muy fríos la siembra se realiza a primeros de abril, pero si son suaves la siembra se puede realizar entre septiembre y noviembre

Estas leguminosas de invierno se siembran en terrenos de secano y sus necesidades hídricas son de 325 a 450 mm incluso con precipitaciones más bajas, pero si las precipitaciones son mayores el rendimiento también aumenta.

e) Yeros

Sus orígenes son el Creciente Fértil, y fue uno de los cultivos más antiguos hechos por el hombre, posteriormente se extendió a Egipto y se extendió a Europa por los romanos.

Exigencias de temperatura y agua.

Se cultiva en zonas cálidas y templadas y soporta bastante bien la sequía incluso en primavera. La planta es similar a la de la lenteja. Su uso principal es en grano y para la alimentación de especies animales.

La siembra se lleva a cabo a principios de febrero en zonas con inviernos fríos, si los inviernos son suaves se puede realizar a partir de octubre. Una de sus virtudes es que resiste las bajas temperaturas.



Figura 2.27 - Imagen de yeros

Fuente: <https://www.juntadeandalucia.es>

Habitualmente esta leguminosa se siembra en tierras de secano. Necesitan entre 250 y 300 mm de agua, es más resistente a la sequía que el resto de leguminosas que se han estudiado.

2.4.2.3. Oleaginosas

a) Colza

Se trata de una planta oleaginosa que se aprovecha para extraer aceite de consumo humano, harina, forraje y en los últimos años ha crecido su uso para fabricar combustible o biodiesel.

Exigencias de temperatura y agua.

Se trata de una semilla muy pequeña que se siembra muy superficialmente por ello necesita de humedad en las capas superiores para germinar. La temperatura desde el periodo de nascencia hasta el estado de cuatro hojas no debe bajar de -2°C. Posteriormente soporta temperaturas de hasta -10°C, aunque las temperaturas bajas retrasan la maduración.



Figura 2.28 - Imagen de colza

Fuente: <https://www.juntadeandalucia.es>

Las necesidades de agua oscilan entre 450 y 500 mm para el ciclo vegetativo. Le perjudica si se produce encharcamiento durante varios días. Cuando aparecen los botones florales hasta la maduración es imprescindible el aporte hídrico.

b) Girasol

El girasol es la planta oleaginosa más cultivada en España. La siembra en terrenos de secano tiene un bajo rendimiento pero que se compensa por el bajo coste de siembra y mantenimiento, muchas veces son sembrados en terrenos que están destinados al barbecho en zonas de cultivo de cereal.



Figura 2.29 - Imagen de girasoles. Fuente: www.wikimedia.org

Exigencias de temperatura y agua.

En el momento de siembra la temperatura idónea a cinco centímetros de profundidad del suelo es entre 7°C y 10°C, esto significa que la siembra se producirá entre finales de febrero y principios de junio dependiendo de la climatología de la zona. La nascencia será más favorable si hay humedad en el suelo y la temperatura es elevada.

El girasol tiene unas necesidades hídricas que van de 500 a 650 mm para producir altos rendimientos de producción. Esta planta tiene muy baja eficiencia en el consumo de agua, pero en caso de falta de agua esta eficiencia puede aumentar entre un 20 y un 50 por ciento. En condiciones de sequía esta planta asegura una producción, aunque sea baja

donde otros cultivos apenas darían producción. Esta planta tiene la ventaja de soportar grandes períodos de sequía si finalmente recibe una alta aportación de agua. La fase crítica donde necesita el aporte de agua va desde que el botón floral mide entre tres y cinco centímetros de diámetro hasta pasados 10 a 15 días después de la floración.

3.CAPÍTULO III. SOLUCIÓN

PROPUESTA

3.1. Diseño del sistema

Después de analizar las tecnologías existentes y de hacer un estudio de los principales cultivos que se siembran en la región castellanoleonesa, se puede comenzar con la implementación del sistema BigDataTFG.

El sistema se ha diseñado para que un usuario pueda realizar la descarga e importación de datos, el filtrado, la visualización gráfica y el modelado de los datos, como se puede ver en la figura 3.1.

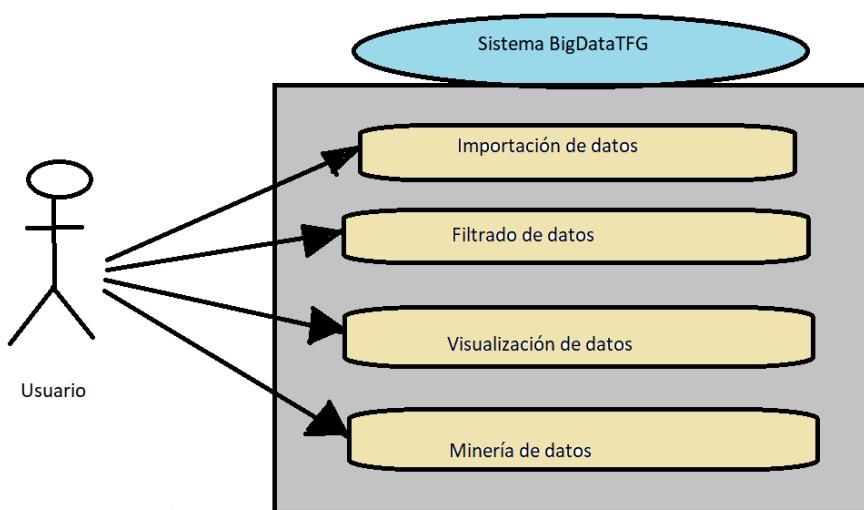


Figura 3.1 – Visión general del sistema BigDataTFG.

El conjunto de requisitos funcionales o funcionalidades que el sistema proporciona a los actores externos se pueden ver en el diagrama de casos de uso de la figura 3.2. donde se representan las interacciones entre el usuario del sistema y la aplicación.

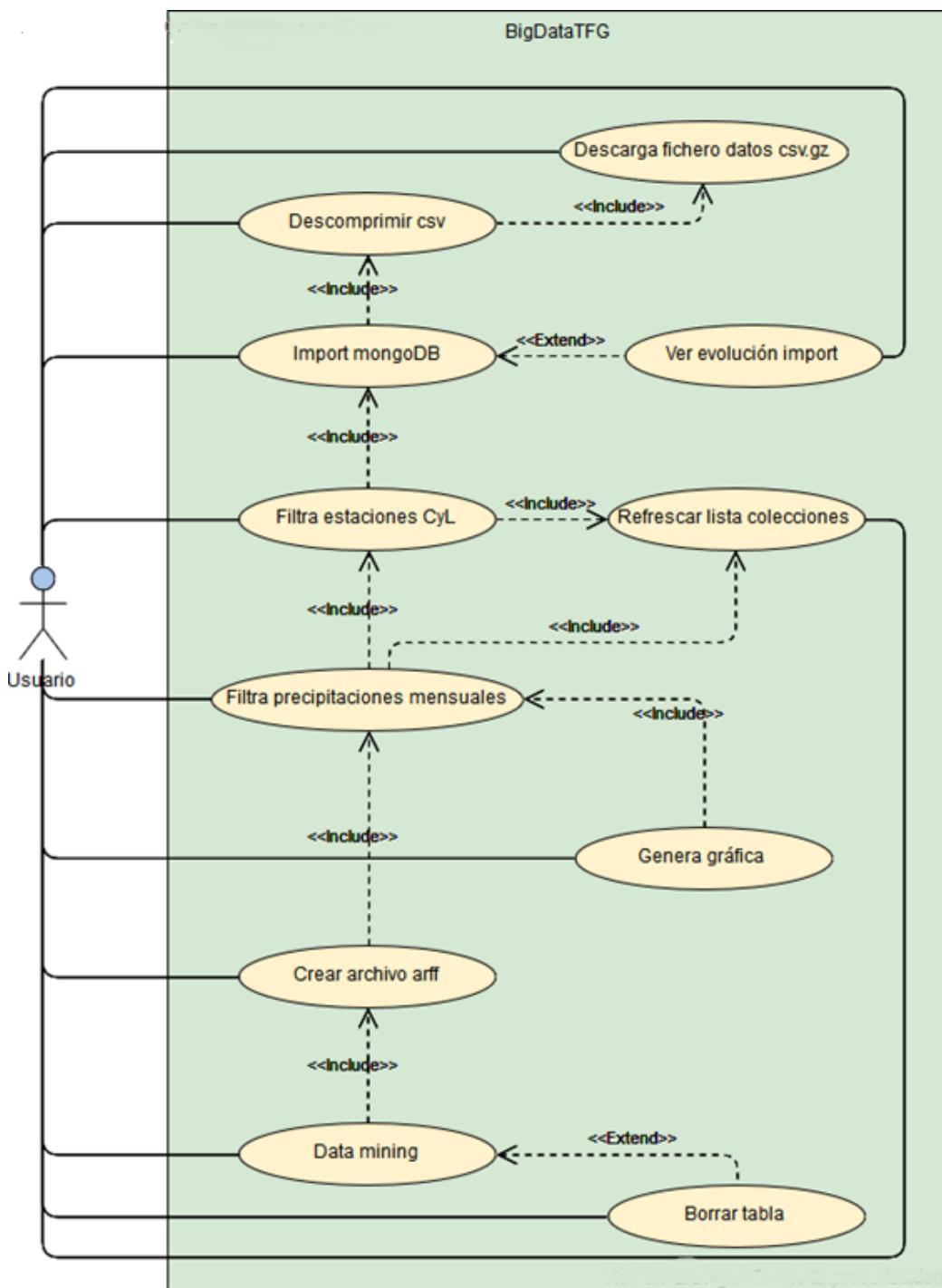


Figura 3.2 – Diagrama de casos de uso del sistema BigDataTFG.

Se completa el diagrama de casos de uso con la descripción textual del mismo, de esta forma se presenta una explicación más formal y con más detalle.

En la tabla 3.1 se ve la plantilla utilizada para la descripción textual de los casos de uso y una descripción de la nomenclatura utilizada.

Tabla 3.1. Plantilla de descripción textual de casos de uso

Caso de uso		
Actores		
Resumen		
Precondiciones		
Postcondiciones		
Incluye		
Extiende		
Hereda de		
Flujo de eventos		
Actor	Sistema	

En la primera línea aparece el caso de uso, debajo va el actor que interacciona con el sistema, en la siguiente fila se hace un resumen con una descripción breve del objetivo, debajo en precondiciones se indican las condiciones previas necesarias para realizar la operación.

La línea de postcondiciones indica el estado del sistema después de haber realizado la operación. Las tres filas siguientes especifican las relaciones con otros casos de uso que pueden ser de inclusión, extensión y herencia.

En la última fila se indican las interacciones entre el actor y el sistema especificando los diferentes pasos que incluye la funcionalidad del caso de uso.

Tabla 3.2. Descripción textual de caso de uso: Descarga fichero datos aaaa.csv.gz

Caso de uso	Descarga fichero datos aaaa.csv.gz
Actores	Usuario
Resumen	El usuario descarga el fichero comprimido con datos anuales de las estaciones de todo el mundo
Precondiciones	El sistema debe estar conectado a internet
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario solicita fichero de datos (año)	
2.El usuario pulsa ‘descargar’	
4.El usuario selecciona directorio de descarga D:\tfg\tfg_descargas y pulsa ‘guardar’	3.El sistema presenta el sistema de ficheros para guardar fichero de descarga
	5.El sistema presenta la barra de progreso de la descarga
	6.El sistema almacena el fichero descargado

Tabla 3.3. Descripción textual de caso de uso: Descomprimir aaaa.csv

Caso de uso	Descomprimir aaaa.csv
Actores	Usuario
Resumen	El usuario descomprime el fichero descargado con datos anuales
Precondiciones	El fichero seleccionado debe estar en formato .gz
Postcondiciones	Debe haber suficiente espacio de almacenamiento en el disco duro de la computadora
Incluye	Descarga fichero datos aaaa.csv.gz
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario pulsa el botón ‘descomprimir’	
3.El usuario selecciona fichero comprimido en D:\tfg\tfg_descargas y pulsa ‘abrir’	2.El sistema presenta el sistema de ficheros
	4.El sistema presenta la barra de progreso de descompresión
	5.El sistema almacena el fichero descargado

Tabla 3.4. Descripción textual de caso de uso: Import mongoDB

Caso de uso	Import mongoDB
Actores	Usuario
Resumen	El usuario importa el fichero descomprimido a la base de datos 'tfg' de mongoDB
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	-
Incluye	Descomprimir aaaa.csv
Extiende	Ver evolución import
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario pulsa el botón 'csv a mongoDB'	2.El sistema presenta el sistema de ficheros
3.El usuario selecciona fichero descomprimido en D:\tfg\tfg_descomprimido y pulsa 'abrir'	4.El sistema importa documentos del fichero aaaa.csv a base de datos mongoDB

Tabla 3.5. Descripción textual de caso de uso: Ver evolución import

Caso de uso	Ver evolución import
Actores	Usuario
Resumen	El usuario visualiza los documentos almacenados en la base de datos 'tfg' de mongoDB
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario pulsa el botón 'documentos guardados'	2.El sistema imprime por pantalla el número de documentos guardados en la colección actual

Tabla 3.6. Descripción textual de caso de uso: Refrescar lista colecciones

Caso de uso	Refrescar lista colecciones
Actores	Usuario
Resumen	El usuario refresca la lista de colecciones en el desplegable ‘colecciones en base de datos tfg’
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario pulsa el botón ‘Filtrado & Refresco’	2.El sistema refresca la lista del desplegable con las colecciones de la base de datos ‘tfg’

Tabla 3.7. Descripción textual de caso de uso: Filtra estaciones CyL

Caso de uso	Filtra estaciones CyL
Actores	Usuario
Resumen	El usuario selecciona una colección con formato ‘aaaa’ siendo las 4 aes el año, de la lista desplegable en ‘colecciones en la base de datos tfg’
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	
Incluye	Refrescar lista colecciones, Import mongoDB
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1.El usuario selecciona del desplegable la colección con formato ‘aaaa’ siendo las 4 aes el año	
2.El usuario pulsa el botón ‘Filtrado & Refresco’	3.El sistema crea una colección llamada ‘aaaa_CyL’ con los documentos filtrados (estaciones de C y L, decilitros, fecha)

Tabla 3.8. Descripción textual de caso de uso: Filtra precipitaciones mensuales

Caso de uso	Filtra precipitaciones mensuales
Actores	Usuario
Resumen	El usuario selecciona una colección con formato ‘aaaa_CyL’ siendo las 4 aes el año, de la lista desplegable en ‘colecciones en base de datos tfg’
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	
Incluye	Filtra estaciones CyL
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1. El usuario selecciona del desplegable la colección con formato ‘aaaa_CyL’ siendo las 4 aes el año 2. El usuario pulsa el botón ‘Filtrado & Refresco’	3. El sistema guarda los 12 documentos con las medias mensuales en una colección llamada ‘CyL’ (campos: año, mes , litros mensuales)

Tabla 3.9. Descripción textual de caso de uso: Crear archivo arff

Caso de uso	Crear archivo arff
Actores	Usuario
Resumen	El usuario selecciona los años que quiere guardar datos en el archivo clima.arff
Precondiciones	Debe estar conectado el servidor mongoDB, debe estar creado el directorio D:/tfg/
Postcondiciones	
Incluye	Filtra precipitaciones mensuales
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1. El usuario selecciona el selector ‘generar fichero arff’ y selecciona los años de inicio y de final 2. El usuario pulsa el botón ‘enviar’	3. El sistema guarda D:/tfg/clima.arff con los datos mensuales medios de todos los años indicados

Tabla 3.10. Descripción textual de caso de uso: Genera gráfica

Caso de uso	Genera gráfica
Actores	Usuario
Resumen	El usuario selecciona los años y genera gráfica sobre precipitaciones medias durante esos años
Precondiciones	Debe estar conectado el servidor mongoDB
Postcondiciones	
Incluye	Filtra precipitaciones mensuales
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1. El usuario selecciona el selector ‘generar gráfica’ y selecciona los años de inicio y de final 2. El usuario pulsa el botón ‘enviar’	3. El sistema genera una gráfica de las precipitaciones medias mensuales de los años seleccionados

Tabla 3.11. Descripción textual de caso de uso: Borrar tabla

Caso de uso	Borrar tabla
Actores	Usuario
Resumen	El usuario borra todos los datos de la tabla de predicciones
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de eventos	
Actor	Sistema
1. El usuario selecciona el selector ‘Limpiar tabla’ 2. El usuario pulsa el botón ‘enviar’	3. El sistema borra todos los datos de predicción de la tabla

Tabla 3.12. Descripción textual de caso de uso: Data mining

Caso de uso	Data mining
Actores	Usuario
Resumen	El usuario selecciona el algoritmo de predicción y se imprime resultado en la tabla
Precondiciones	Debe haber datos almacenados en archivo clima.arff
Postcondiciones	-
Incluye	Crear archivo arff
Extiende	Borrar tabla
Hereda de	-
Flujo de eventos	
Actor	Sistema
1. El usuario selecciona un selector de los cuatro algoritmos de predicción 2. El usuario pulsa el botón ‘Predecir Serie Temporal’	3. El sistema imprime por pantalla la predicción de la serie temporal para los 12 próximos meses, según algoritmo seleccionado

3.2. Importación de datos

3.2.1 Introducción

Para realizar este trabajo se ha seguido un proceso que consiste en un primer paso donde se realiza la descarga de datos, el segundo paso es descomprimir los ficheros descargados para posteriormente cargarlos en una base de datos. Una vez estén los datos almacenados se podrán realizar consultas y también se va a construir modelos que permitan realizar predicciones aplicando algoritmos de minería de datos.

3.2.2 Repositorio de datos

Los datos necesarios para este trabajo se han descargado de GHCN (Global Historical Climatology Network) que es una base de datos integrada de resúmenes de datos climáticos diarios, obtenidos de estaciones de superficie terrestre en todo el mundo. Los registros climáticos diarios han ido aumentando a la vez que van aumentando las fuentes que se integran, estos datos son sometidos a revisiones y controles de calidad.



Formerly the National Climatic Data Center (NCDC)... [more about NCEI »](#)

Figura 3.3 Logotipo página web de NOAA

Los datos se obtienen de más de 20 fuentes. Algunos datos tienen más de 175 años, mientras que otros tienen menos de una hora. GHCN es el conjunto de datos archivado oficial y sirve como un producto de consulta para los conjuntos de datos más antiguos mantenidos por NCEI.

Dichos datos son gestionados por la Administración Nacional Oceánica y Atmosférica (National Oceanic and Atmospheric Administration, NOAA) que es una agencia científica del Departamento de Comercio de los Estados Unidos que se dedica a realizar actividades para mejorar las condiciones de los océanos y la atmósfera. Además, NOAA avisa cuando se producen eventos meteorológicos que pueden ser peligrosos y realiza multitud de actividades para mejorar la comprensión y administración del medio ambiente.

Un ejemplo de predicción meteorológica realizado por esta agencia se muestra en la figura 3.4 y presenta un mapa que indica las zonas de Estados Unidos con la probabilidad de que haya al menos una pulgada de nieve el día de Navidad. Esta predicción está basada en los datos almacenados los años comprendidos entre 1981 y 2010 en los Centros Nacionales de Información Ambiental (NCEI) de NOAA.

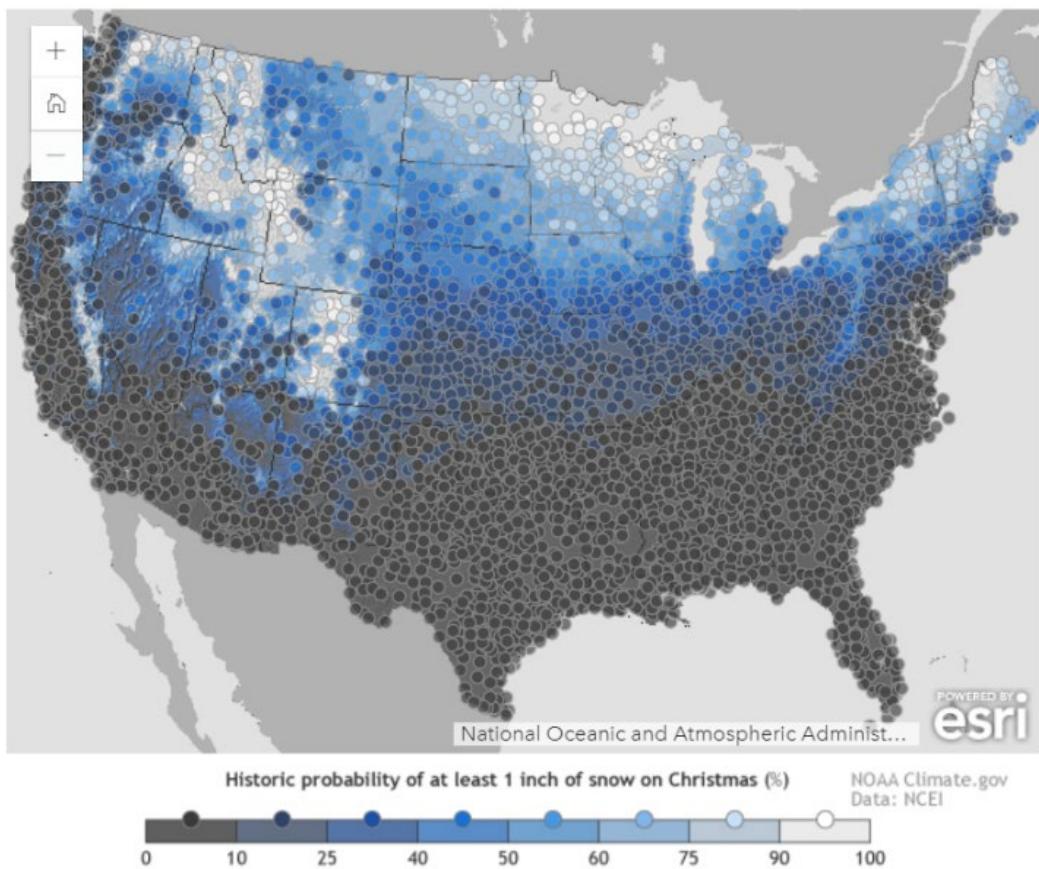


Figura 3.4. Probabilidad de que nieve más de 1 pulgada el 25 de diciembre de 2018.

Fuente: <https://www.ncdc.noaa.gov/sotc/>

Para poder comprender la dimensión de los datos que se almacenan en GHCN hay que conocer que guarda registros de más de 100,000 estaciones pertenecientes a 180 países. NCEI recoge variables como la temperatura máxima y mínima, la precipitación diaria total, las nevadas y la profundidad de la nieve, aunque prácticamente la mitad de las estaciones solo recogen la cantidad de lluvia caída. Existen estaciones que reportan datos de hace 175 años mientras otras se fueron incorporando posteriormente. En los ficheros descargados los datos más antiguos son del año 1763.

Los datos almacenados por NOAA son a nivel mundial, en España recogen datos de 207 estaciones, ver figura 3.5.



Figura 3.5. Estaciones que recogen datos en España.

En Castilla y León las estaciones de las que se recogen datos son las 14 que se muestran en la figura 3.6.

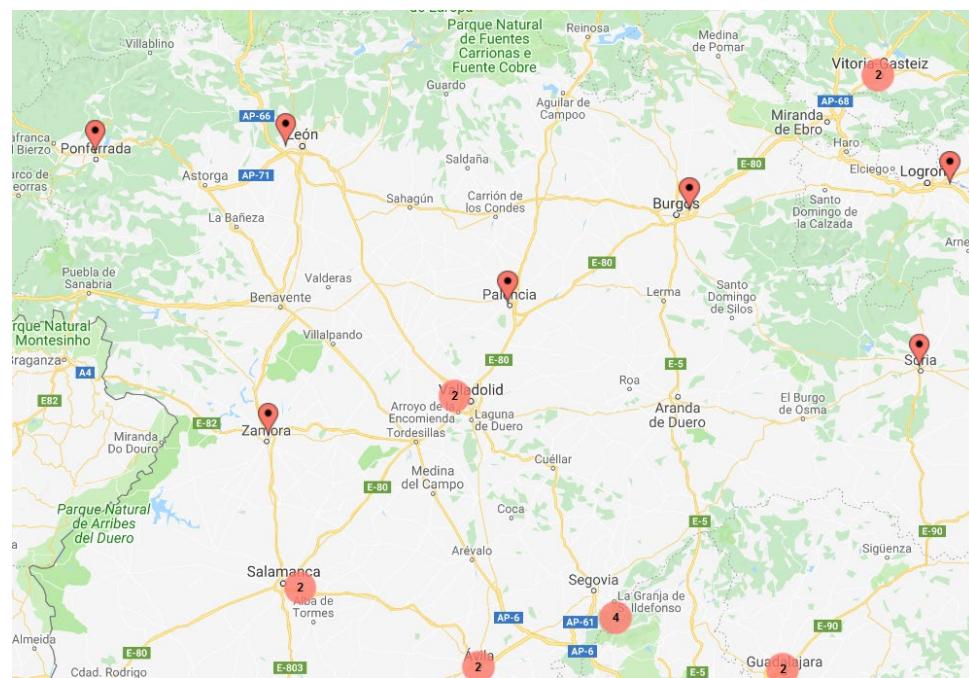


Figura 3.6. Situación de las 14 estaciones de Castilla y León.

3.2.2.1. Codificación de los datos

Los datos descargados son archivos comprimidos en gz que una vez descomprimidos están en formato csv, no traen en la cabecera el nombre de los campos. En la figura 3.7 se ve una muestra de los datos.

```
ASN00015643,ZUUUU1U1,TMAX,35U,,,a,  
ASN00015643,20000101,PRCP,0,,,a,  
ASN00085296,20000101,TMAX,14,,,a,  
ASN00085296,20000101,TMIN,79,,,a,  
ASN00085296,20000101,PRCP,10,,,a,  
ASN00085280,20000101,TMAX,187,,,a,  
ASN00085280,20000101,TMIN,79,,,a,  
ASN00085280,20000101,PRCP,6,,,a,  
ASN00040209,20000101,TMAX,290,,,a,  
ASN00040209,20000101,TMIN,202,,,a,  
ASN00040209,20000101,PRCP,0,,,a,  
ASN00084106,20000101,PRCP,130,,,a,  
CA007045400,20000101,TMAX,-43,,,C,  
CA007045400,20000101,TMIN,-255,,,C,  
CA007045400,20000101,PRCP,28,,,C,  
CA007045400,20000101,SNOW,28,,,C,  
CA007045400,20000101,SNWD,200,,,C,  
CA007020860,20000101,TMAX,30,,,C,  
CA007020860,20000101,TMIN,-180,,,C,  
CA007020860,20000101,PRCP,10,,,C,  
CA007020860,20000101,SNOW,10,,,C,  
CA007020860,20000101,SNWD,80,,,C,  
ASN0009661,20000101,PRCP,0,,,a,  
ASN00075176,20000101,PRCP,0,,,a,  
ASN00014814,20000101,PRCP,98,,,a,  
ASN00036096,20000101,PRCP,0,,,a,
```

Figura 3.7. Datos importados en formato csv.

Campo 1= ID: 11 caracteres que indican el código de la estación.

Campo 2= fecha: 8 caracteres en formato YYYYMMDD

Campo 3= elemento: 4 caracteres que indican tipo de elemento (precipitación, temperatura máxima...)

Campo 4= valor: 5 caracteres que indican el valor que toma el elemento indicado en el campo 3.

Campo 5= M-FLAG: 1 carácter codificado según tabla.

Campo 6= Q-FLAG: 1 carácter codificado según tabla.

Campo 7= S-FLAG = 1 carácter codificado según tabla.

Campo 8= OBS-TIME: 4 caracteres indican el momento de la observación en horas-minutos, 0700 =7:00 am)

Los campos necesarios para este trabajo son la identidad de la estación, la fecha y la cantidad de precipitación.

3.2.3 Base de datos mongoDB

Es un sistema de base de datos NoSQL. Los datos no se guardan en tablas tal y como lo hacen los sistemas relacionales, sino que se guardan en forma de documentos, de manera similar a JSON, es decir siguiendo un esquema dinámico.

Es un software gratuito y de código abierto que se ha desarrollado a partir de octubre de 2007 por la compañía 10gen.1, actualmente llamada MongoDB Inc. Se trata de un sistema multiplataforma ya que el código binario está disponible para los sistemas operativos Windows, Linux, OS X y Solaris.

MongoDB proporciona alto rendimiento, alta disponibilidad y escalado automático. La compatibilidad con los modelos de datos integrados reduce la actividad de entrada/salida de la base de datos. Gracias a los índices que incluye proporciona consultas mucho más rápidas. Alta disponibilidad gracias al conjunto de réplicas que se pueden almacenar en diferentes servidores proporcionando redundancia de datos y aumento de la disponibilidad.

Los datos son almacenados en documentos que a su vez forman colecciones que pertenecen a una base de datos. Esta forma de organizar los datos es muy sencilla.

Potente lenguaje de consulta para realizar operaciones de lectura y escritura (CRUD), agregación de datos, búsqueda de textos y consultas geoespaciales.

Un registro en MongoDB es un documento que tiene la forma <campo>: >valor> muy semejante a los objetos JSON. El valor del campo puede incluir nuevos documentos, arrays y arrays de documentos.

Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.

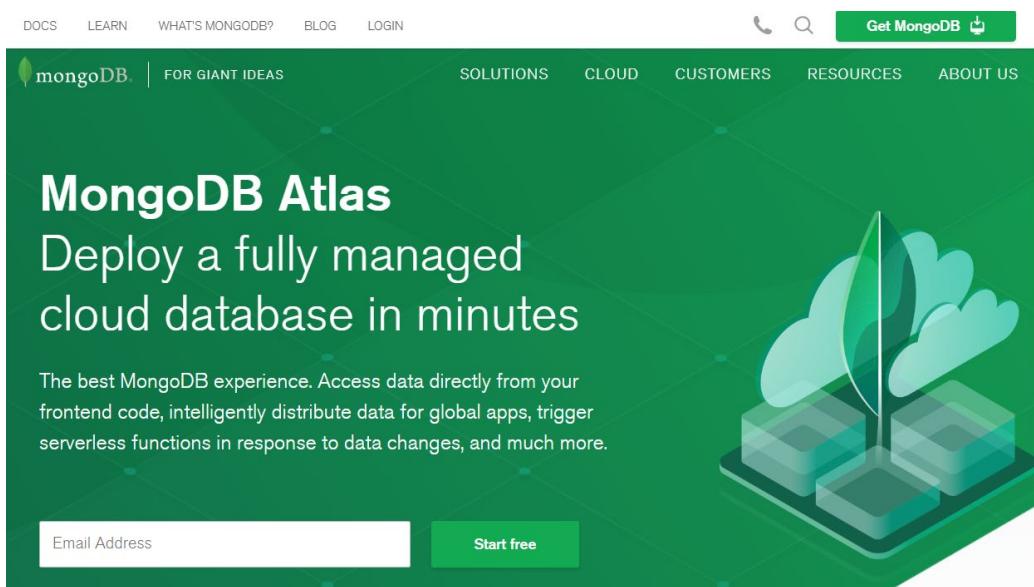


Figura 3.8. Página web mongoDB.

Un ejemplo de un documento de MongoDB puede ser de la forma:

```
{  
    nombre: "Antonio",  
    Edad: 21,  
    Peso: 80,  
    Aficiones: ["deporte", "cine"]  
}
```

A diagram illustrating a JSON document structure. The document is enclosed in curly braces. Inside, there are four key-value pairs: 'nombre: "Antonio"', 'Edad: 21', 'Peso: 80', and 'Aficiones: ["deporte", "cine"]'. An arrow points from the 'Aficiones' key to a rectangular box containing five 'campo:valor' entries, suggesting that the array value can contain multiple items.

Otra ventaja de trabajar con documentos es que éstos se corresponden con tipos de datos nativos en los lenguajes de programación.

MongoDB proporciona librerías para que desde otros lenguajes poder interactuar con la base de datos. Esto permite elegir entre las diferentes opciones de trabajo: mongo Shell, Node.JS, Java, C++, C#, Python y Ruby.

Para realizar este trabajo se utiliza MongoDB como base de datos así que se procede a su descarga e instalación en una máquina con el sistema operativo Windows 10. Se realiza la descarga de la versión más reciente, la 4.0.4 para el sistema operativo Windows 10 de 64 bit, desde la página oficial de MongoDB.

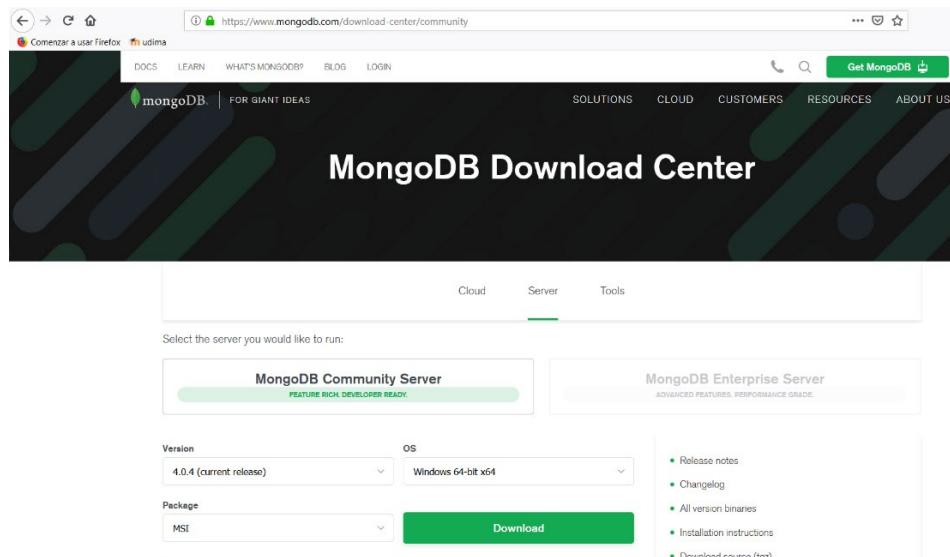


Figura 3.9. Página de descarga mongoDB.

Una vez descargado el instalador se ejecuta con doble click y se procede con su instalación.

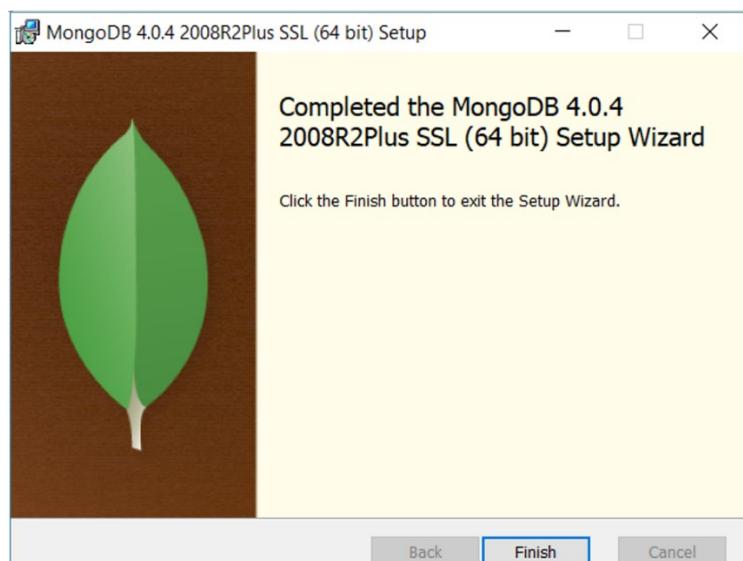


Figura 3.10. Proceso de instalación mongoDB.

Una vez finalizada la instalación hay que ir al directorio donde están los archivos mongo.exe y mongod.exe que en este caso es la siguiente carpeta:

C:\Program Files\MongoDB\Server\4.0\bin

C:\Program Files\MongoDB\Server\4.0\bin	
Nombre	Fecha de modifica...
bsondump.exe	06/11/2018 19:49
InstallCompass.ps1	06/11/2018 20:15
libeay32.dll	03/04/2018 19:58
mongo.exe	06/11/2018 20:11
mongod.cfg	03/12/2018 19:19
mongod.exe	06/11/2018 20:17
mongod.pdb	06/11/2018 20:17
mongodump.exe	06/11/2018 19:53
mongoexport.exe	06/11/2018 19:51
mongoimport.exe	06/11/2018 19:51
mongofiles.exe	06/11/2018 19:51
mongorestore.exe	06/11/2018 19:52
mongorestore.exe	06/11/2018 19:52

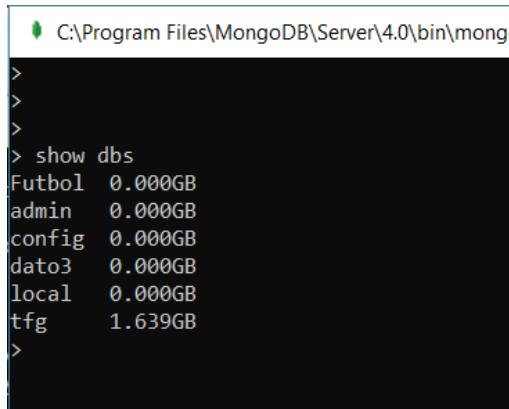
Figura 3.11. Directorio con los archivos de mongoDB.

Para arrancar el servidor, click sobre mongod.exe y se abre un Shell, que se muestra en la figura 3.12, donde se indica el proceso de arranque y que finalmente queda conectado por el puerto 27017, como en este caso el servidor está en la misma máquina que el cliente la ip será localhost. El servidor debe estar arrancado siempre para poder establecer conexión con la base de datos.

```
C:\Program Files\MongoDB\Server\4.0\bin>mongod.exe
n-recover: Set global recovery timestamp: 0
2018-12-20T21:52:46.564+0100 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2018-12-20T21:52:46.681+0100 I CONTROL [initandlisten]
2018-12-20T21:52:46.681+0100 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-12-20T21:52:46.681+0100 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2018-12-20T21:52:46.682+0100 I CONTROL [initandlisten]
2018-12-20T21:52:46.683+0100 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-12-20T21:52:46.683+0100 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2018-12-20T21:52:46.684+0100 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2018-12-20T21:52:46.684+0100 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2018-12-20T21:52:46.686+0100 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired,
2018-12-20T21:52:46.686+0100 I CONTROL [initandlisten] ** start the server with --bind_ip 127.0.0.1 to disable this warning.
2018-12-20T21:52:46.687+0100 I CONTROL [initandlisten]
2018-12-20T21:52:47.207+0100 W FTDC [initandlisten] Failed to initialize Performance Counters for FTDC: WindowsPdhError: PdhExpandCounterPathW failed with 'El objeto especificado no se encontró en el equipo.' for counter '\Memory\Available Bytes'.
2018-12-20T21:52:47.207+0100 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2018-12-20T21:52:47.212+0100 I NETWORK [initandlisten] waiting for connections on port 27017
```

Figura 3.12. Servidor de la base de datos mongoDB.

Una vez arrancado el servidor ya se puede arrancar mongo para interactuar con la base de datos. Haciendo click sobre mongo.exe se abre otro terminal por donde se realizarán los accesos necesarios a la base de datos. En la figura 3.13 se ha pedido que muestre las bases de datos creadas, mostrando las existentes en ese momento.



```
C:\Program Files\MongoDB\Server\4.0\bin\mong>
>
>
>
> show dbs
Futbol 0.000GB
admin 0.000GB
config 0.000GB
dato3 0.000GB
local 0.000GB
tf3 1.639GB
>
```

Figura 3.13. Terminal para trabajar con mongoDB.

3.2.4 Implementación



Figura 3.14. NetBeans.

Para la programación del sistema se ha utilizado NetBeans como entorno de desarrollo. Se trata de un producto gratuito sin restricciones de uso, y se ha utilizado con el lenguaje de programación Java versión 1.8.0_191.

Características principales del entorno de programación que se ha usado:

Product Version: NetBeans IDE 8.2 (Build 201610071157)

Actualizaciones: NetBeans IDE se ha actualizado a la versión NetBeans 8.2 Patch 2

Java: 1.8.0_191; Java HotSpot(TM) 64-Bit Server VM 25.191-b12

Runtime: Java(TM) SE Runtime Environment 1.8.0_191-b12



Figura 3.15. Página web de Oracle.

Edición de Windows

Windows 10 Home

© 2018 Microsoft Corporation. Todos los derechos reservados.

Sistema

Procesador: Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz 2.39 GHz

Memoria instalada (RAM): 16,0 GB (15,9 GB utilizable)

Tipo de sistema: Sistema operativo de 64 bits, procesador x64

Figura 3.16. Características principales del computador usado en este trabajo.



Sistema operativo
Windows 10

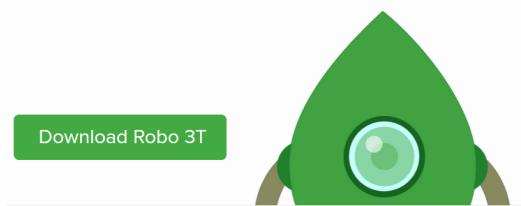
Figura 3.17. Logo de Windows

Para visualizar de manera más fluida y gráfica la base de datos mongoDB y para ir comprobando rápidamente las operaciones que se han ido realizando en la base de datos se ha utilizado el programa Robo 3T.

Robo 3T

Robo 3T (formerly Robomongo) is the free lightweight GUI for MongoDB enthusiasts.

- MongoDB GUI with embedded shell



Robomongo is Robo 3T

Figura 3.18. Página de descarga de la aplicación Robo 3T.

Para importar los datos con los que se va a trabajar se ha diseñado un sistema compuesto por tres módulos que automatizan los siguientes procesos:

- Conexión y descarga de los ficheros que contienen los datos climatológicos.
- Descompresión de los archivos descargados.
- Inserción de los datos en mongoDB.

La interface gráfica del sistema está formada por un JFrame que presenta los tres módulos que se muestran en detalle en los siguientes puntos.

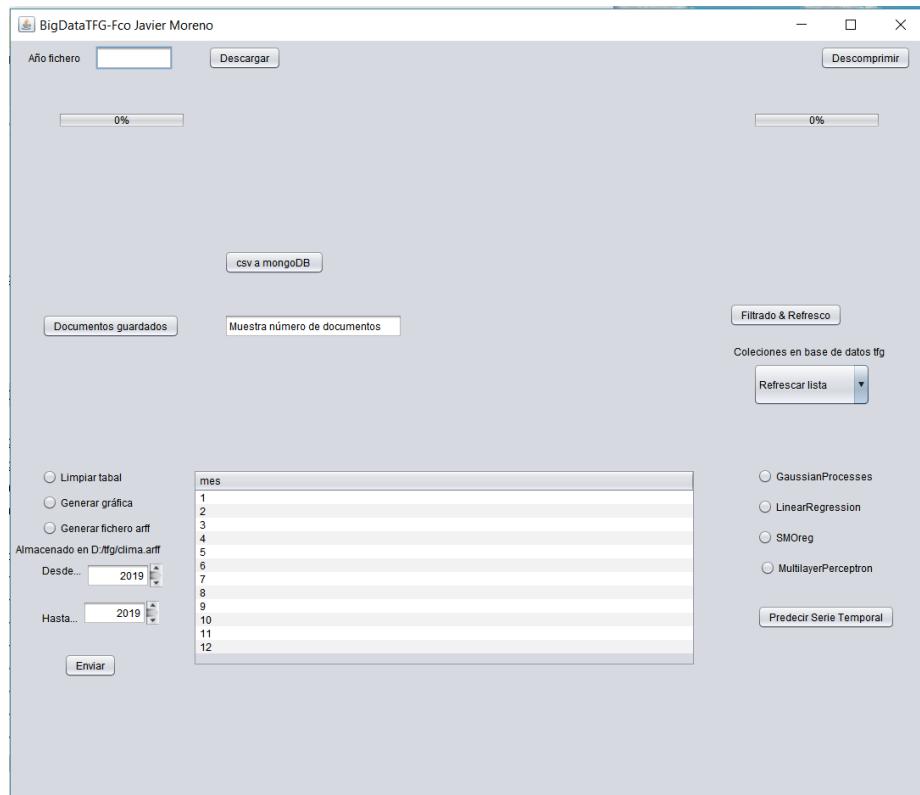


Figura 3.19. Interfaz gráfica.

3.2.4.1. Conexión y descarga

En la interface gráfica, está situado en la parte superior izquierda y está formada por un JLabel, un JTextField, un JButton y un JProgressBar, como se ve en la figura 3.20.



Figura 3.20. JLabel donde se introduce el año del que se obtienen los datos.

La etiqueta “Año fichero” indica que se introduzca en el área de texto el año de los datos que se quieren descargar. Debe introducirse un valor comprendido entre 1763 y 2019 que son los años en los cuales existen datos.

Una vez indicado el año con cuatro cifras en el área de texto se pulsa el botón “Descargar” y aparece un cuadro para seleccionar la carpeta donde se guarda el archivo. Una vez seleccionado click en “descarga” y comienza la descarga del fichero de ese año con los datos de climatología registrados en todo el mundo.

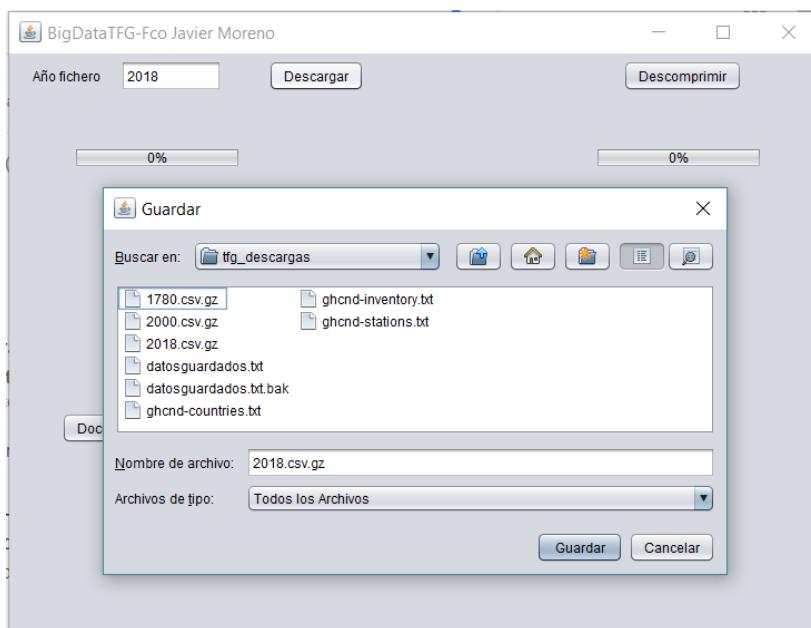


Figura 3.21. Panel para seleccionar el directorio donde se almacena el archivo.

La aplicación se conecta a la página web poniendo el año pedido en lugar de las cuatro “x” en la URL.

https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/xxxx, csv.gz

Actualmente la página saca el mensaje que aparece en la figura 3.22 debido al cierre del gobierno americano que mantiene cerrados los servicios públicos no considerados esenciales desde el 21 de diciembre de 2018.



The website you are trying to access is not available at this time due to a lapse in appropriation.

NOAA.gov and specific NOAA websites necessary to protect lives and property are operational and will be maintained during this partial closure of the U.S. Government.

See [weather.gov](#) for forecasts and critical weather information.

NOAA Federal Employees: Go to the [NOAA Furlough information page](#) for information, forms and other resources related to the shutdown.

Figura 3.22. La página de NOAA desde el 21 de diciembre de 2018 permanece con este mensaje.

Se cambia por <http://noaa-ghcn-pds.s3.amazonaws.com/csv.gz/xxxx.csv.gz> que actualmente sí que permite descargar los ficheros.

En la figura 3.23 y 3.24 la barra de progreso va mostrando en todo momento cómo va la descarga.

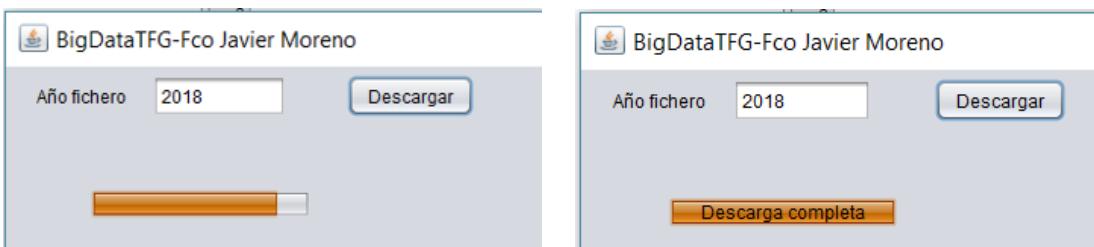


Figura 3.23 y 3.24. Barra de progreso durante la descarga y al finalizar la descarga.

Los ficheros anuales son de tamaño que superan 1 GB y contienen del orden de los 30 millones de documentos por lo que se descargan comprimidos. Los ficheros descargados son del tipo 'aaaa.csv.gz', donde las 4 aes representan el año, contienen datos en formato csv, pero comprimidos. Repitiendo este proceso se pueden descargar todos los ficheros que se deseen, en caso de pedir un año que no exista en la página web aparecerá un error como el mostrado en la figura 3.25.



Figura 3.25. Mensaje de error por fichero no encontrado.

3.2.4.2. Descompresión de archivos

La parte superior derecha de la interface gráfica se encarga de descomprimir los archivos descargados para dejarlos en formato csv.

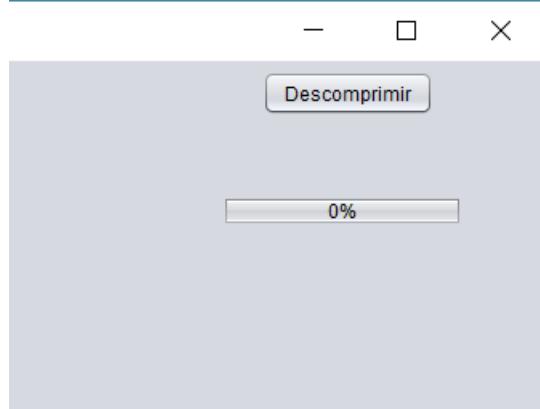


Figura 3.26. Botón para descomprimir los archivos descargados.

Consta de un JButton y una JProgressBar, al pulsar el botón “Descomprimir” se abre el cuadro de sistema de archivos mostrado en la figura 3.27 para seleccionar, el archivo ya descargado con anterioridad, que se quiere descomprimir.

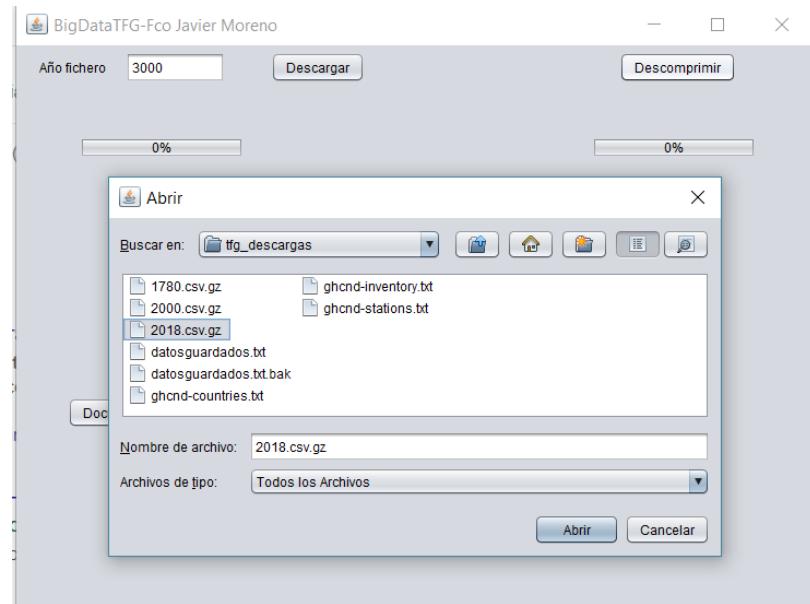


Figura 3.27. Selección del archivo descargado para descomprimir.

Pulsando en “abrir” comienza la descompresión dejando en el directorio “D:\tfg\tfg_descomprimido” el archivo descomprimido con el mismo nombre, pero sin la extensión .gz de la forma ‘aaaa.csv’.

La barra de progreso nos da una idea de cómo va el proceso de descompresión.

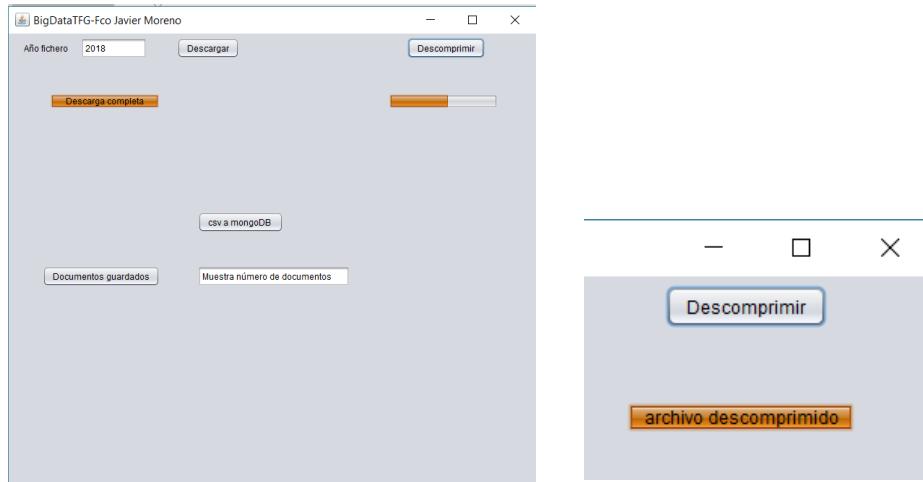


Figura 3.28 y 3.29. Barra de progreso de descompresión de fichero

3.2.4.3. Inserción de datos en MongoDB

Es imprescindible tener arrancado el servidor mongoDB para poder trabajar con la base de datos.

La parte central de la interface gráfica tiene dos JButton y un JTextField y es la encargada de insertar los documentos en la base de datos de mongoDB.

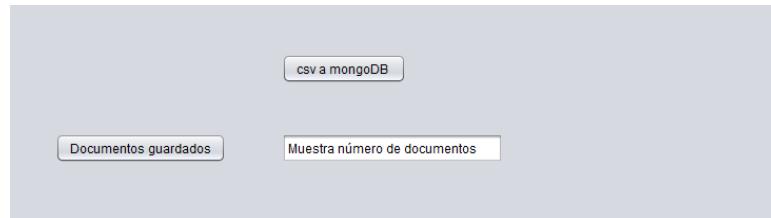


Figura 3.30. Botones para inserción de documentos y visualización del número de documentos guardados.

Al pulsar el botón “csv a mongoDB” se va a abrir el sistema de archivos para seleccionar el fichero que se quiere insertar en la base de datos.

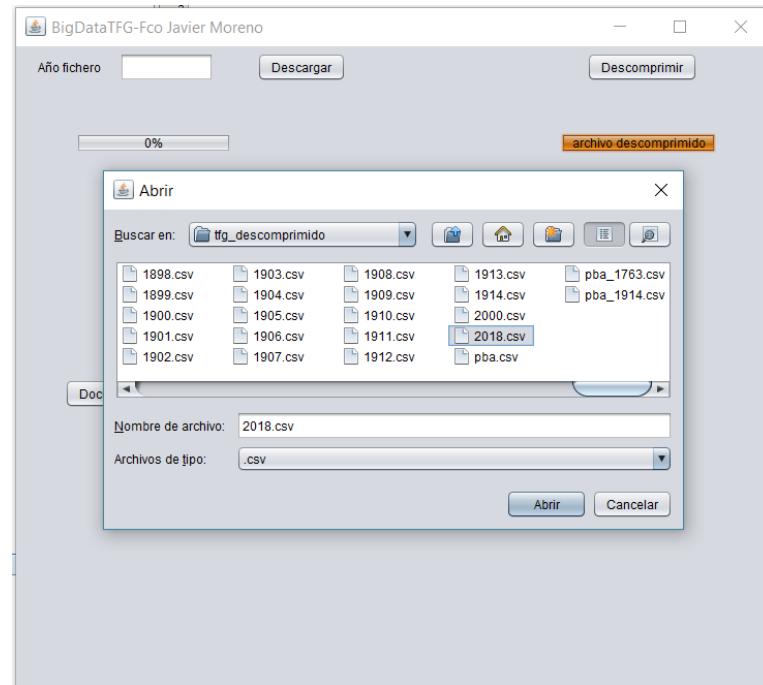


Figura 3.31. Sistema de archivos para seleccionar fichero que se importa a mongoDB.

Una vez seleccionado se pincha en “abrir” y la aplicación crea el comando para realizar la inserción de documentos:

```
mongoimport --db <nombre de la base de datos> --collection <nombre de la colección> --type csv -file d:\tfg_descomprimido\1765.csv --fields ID_estacion, fecha, elemento, valor_dato,M_flag, q_flag, s_flag,observ_time
```

La herramienta mongoimport de mongoDB importa documentos que estén en formato CSV o TSV.

La aplicación va a rellenar los campos “nombre de la base de datos” con “tfg” y “nombre de la colección” con “aaaa” que es el año de los datos que contiene el fichero. Además, añade el nombre de todos los campos ya que los ficheros vienen sin él.

Para saber los documentos que se han insertado en mongoDB, pulsando el botón “Documentos guardados”, en el área de texto se imprime el número de documentos almacenados hasta el momento, cuando no aumente este número es que ha finalizado la importación. Se puede ver en la figura 3.32.



Figura 3.32. Documentos almacenados en la base de datos tfg.

Aunque es un proceso muy rápido se debe tener en cuenta que los archivos descargados pueden ser muy grandes. En este caso el archivo con los datos del año 2018 es de más de 1 GB.



Figura 3.33. Archivo que contiene los datos de 2018.



Figura 3.34. Evolución de los documentos importados.

Si se vuelve a consultar los documentos guardados irán aumentando hasta terminar la importación de todos los documentos.



Figura 3.35. Fichero 2018 importado con más de 30 millones de documentos.

El archivo 2018.csv contiene más de 31 millones de documentos y ha tardado unos siete minutos en insertarlos todos en mongoDB. Se comprueba desde el terminal de mongoDB y coinciden los documentos guardados, se ve en la figura 3.36.

```
> db.climatologia.find().count()
31473090
>
```

Figura 3.36. Visualización desde mongoDB

Una vez finalizado el proceso de importación de datos estarán almacenados en mongoDB todos los documentos de los años 1920 hasta 2019 en una base de datos llamada “tfg” y en 100 colecciones diferentes, una por año. Esto se ha hecho en diferentes fases para poder ir eliminando las colecciones innecesarias ya que las 100 colecciones suponen cerca de 100 GB de disco duro.

3.3. Filtrado de datos

La primera parte de la fase de filtrado consiste en eliminar los campos no necesarios para este trabajo.

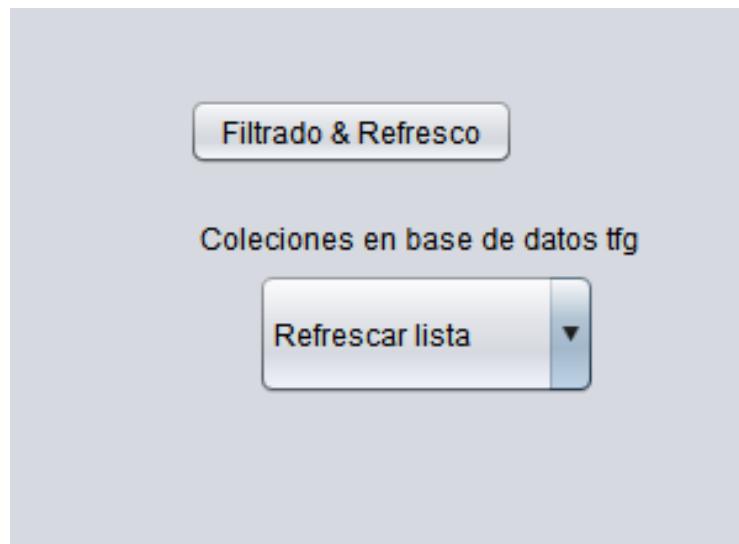


Figura 3.37. Parte del sistema encargado del filtrado de datos

De los datos suministrados por NOAA en el archivo csv, las precipitaciones diarias de cada estación vienen en el campo que se ha denominado “elemento” si su valor es “PRCP” en el campo “valor_dato” vendrá la cantidad de lluvia registrada en decilitros y si el campo “elemento” lleva el valor “SNOW” entonces el campo “valor_dato” indica milímetros de nieve registrados. Se ha comprobado que si la estación registra la cantidad

de nieve caída también ha registrado esa misma cantidad como “PRCP” por tanto el documento con “elemento” de tipo “SNOW”, no debe tenerse en cuenta.

El botón “Filtrado & Refresco” ofrece tres operaciones diferentes según el elemento seleccionado en el desplegable.

Primera operación. Si se pulsa el botón “Filtrado & Refresco” con la opción “Refrescar lista”, seleccionada en el desplegable, como se ve en la figura 3.38, se va a realizar una consulta a mongoDB para que nos diga todas las colecciones que tiene almacenadas en la base de datos “tfg”.

Si se abre el desplegable “refrescar lista” aparecen todas las colecciones que están almacenadas en la base de datos “tfg”.

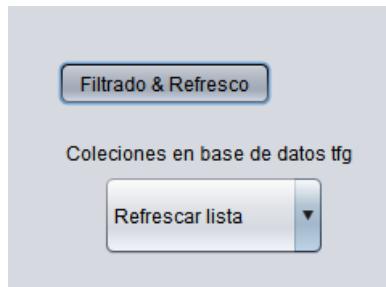


Figura 3.38. Opción refrescar lista.

Segunda operación. Si en el desplegable se selecciona una colección con el nombre “aaaa”, siendo las 4 aes el año, tal y como se indica en la figura 3.39 y 3.40, el comportamiento de la aplicación será el de filtrar las estaciones y los campos no necesarios y formatear la fecha almacenando todos los documentos en una nueva colección llamada “aaaa_CyL”.

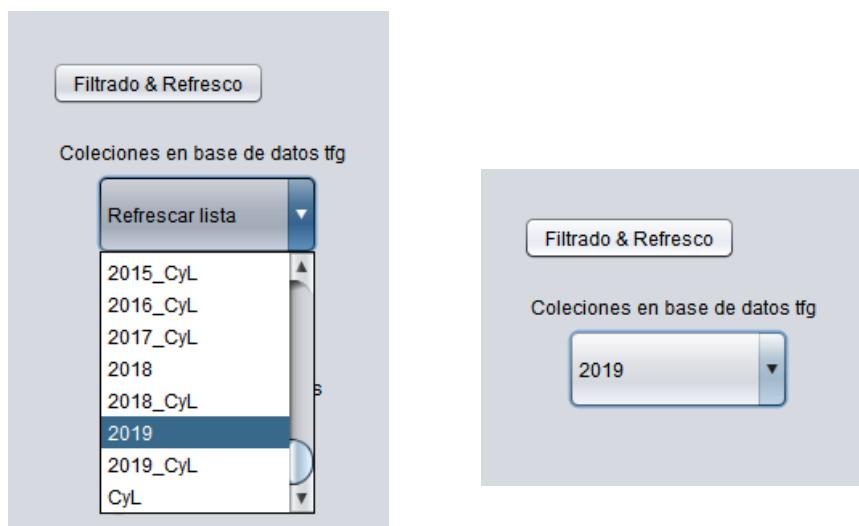


Figura 3.39 y 3.40. Opción filtrar y formatear fecha

Tercera operación. Si en el desplegable se selecciona una colección del tipo “aaaa_CyL”, como se muestra en las figuras 3.41 y 3.42 la aplicación realiza una consulta agrupando por meses, que devuelve la cantidad de litros mensuales caídos en la comunidad de Castilla y León, aplicando la media aritmética de los litros caídos en todas las estaciones se calcula los litros mensuales y se almacenan en una nueva colección llamada CyL. Esta colección almacena los resultados de todos los años desde 1920 hasta 2019 ya que son los años que reportan datos de precipitaciones caídas en Castilla y León.

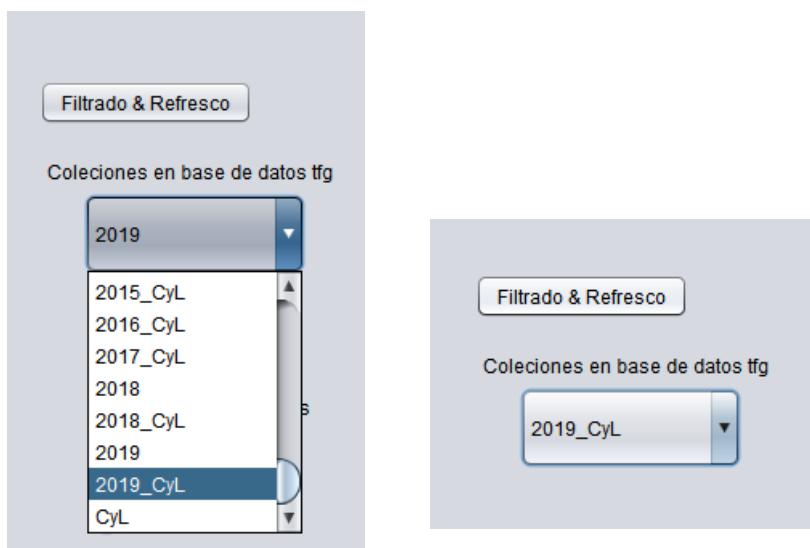


Figura 3.41 y 3.42. Opción que agrupa datos y guardar resultado final

La primera operación indicada consiste en un refresco de la lista de colecciones que hay en la base de datos “tfg”, su implementación se lleva a cabo en la clase VerColecciones la cual tiene el método listaComboBox() que recibe jComboBox1 que es el JComboBox donde se mostrará el resultado.

Se crea un ArrayList donde se almacenan todas las colecciones. (1).

Se guardan todas las colecciones como String (2) y se guardan en el ArrayList (3).

Bucle que añade las colecciones al jComboBox1. (4).

El código utilizado para la implementación es:

Paso (1): *ArrayList<String> colecionCombo;*

Paso (2): *Set coltfg = db.getCollectionNames();*

Set<String> colecciones = new TreeSet<>(coltfg);

Paso (3): *colecciones.stream().map((ss) -> {*

return ss;

}).filter((ss) -> (!"system.js".equals(ss))).forEachOrdered((ss) -> {

colecionCombo.add(ss);

});

Paso (4): *int x=colecionCombo.size();*

for(int y=0;y<x;y++){

jComboBox1.addItem(colecionCombo.get(y));

Segunda operación, cada colección contiene todos los documentos con los datos procedentes de las estaciones de todo el mundo. Seleccionando un año se selecciona esa colección. Pulsando el botón “filtrado y refresco” la aplicación va a crear una nueva colección llamada “aaaa_CyL” donde las cuatro aes corresponden al año. Esta colección que se crea hace un filtrado donde se eliminan todos los documentos de las estaciones que no están situadas en Castilla y León”, de las estaciones que pasan este filtro sólo se almacenan las que contienen “PRCP” en el campo “ID_estación”. De todos los documentos filtrados se presentan sólo los campos “_id”, “ID_estacion”, “fecha” y “valor_dato”. Además, se cambia el formato del campo “fecha” que originalmente es de tipo int y tiene el formato “yyyymmdd”, convirtiéndole en tipo Date con el formato “yyyy-mm-ddThh:mm:ss.mmmZ”.

El código implementado en la aplicación BigDataTFG para realizar el filtrado de datos se describe a continuación.

La clase ColecionFinal es la encargada de hacer el filtrado de documentos. En esta clase se obtiene un objeto FindIterable a partir de aplicar el método find() a la colección que corresponde al año a filtrar.

El método find() va a filtrar por el campo "ID_estacion", las estaciones que están en el array estacionCod que contiene las 14 estaciones de Castilla y León y además en el campo "elemento" tienen el valor "PRCP". (5)

El método projection() va a devolver los campos que serán visibles de todos los documentos encontrados por el método find(). (6)

Se crea un ArrayList docs y se guardan en él todos los documentos guardados en el anterior paso en FindIterable it. (7)

Se guardan todos los documentos filtrados en una nueva colección llamada ‘aaaa_CyL’ siendo el año las cuatro aes. (8) y (9).

Paso (5): *FindIterable it = collection.find(and (in ("ID_estacion",*

estacionCod),eq("elemento","PRCP"))).projection (fields(include (

```
"fecha", "ID_estacion", "valor_dato"), excludeId());
```

Paso (7): *ArrayList<Document> docs = new ArrayList();*

```
it.into(docs);
```

Paso (8): *MongoCollection <Document> collection1 = database.getCollection(coleccionFinal);*

Paso (9): *docs.stream().map((doc) -> {collection1.insertOne(*

```
(org.bson.Document) (Document)doc);
```

```
return doc;
```

Ahora los documentos de la colección ‘aaaa’ están almacenados en la nueva colección ‘aaaa_CyL’ pero el campo “fecha” está como tipo int y formato ‘aaaammdd’ y se les quiere guardar como tipo ISODate. Para este proceso se ha implementado la clase FormatFecha.

El método FormatFecha() recibe un String con el nombre de la colección, se crea un objeto MongoCursor que con el método iterator() recorre toda la colección pues el método find() al ir sin parámetros no realiza ningún filtrado sobre los documentos. (10).

Se hace un bucle que recorra todos los documentos (11) y se guardan en ArrayList (12).

Se recoge el valor del campo “fecha” (13).

Se actualiza cada uno de los documentos de la colección metiendo en el campo fecha el valor obtenido del método FechaIso(x) saliendo del bucle cuando ya no quedan documentos en la colección. (14)

El método FechaIso(x) recibe la fecha con tipo de dato int y la devuelve en tipo ISODate con formato yyyy-MM-ddTHH:mm:ss.mmmZ. (15).

Paso (10): *MongoCursor<Document> cur = col.find().iterator();*

Paso (11): *while (cur.hasNext()) { Document doc = cur.next();*

Paso (12): *List list = new ArrayList(doc.values());*

Paso (13): *int x=(int) list.get(2);*

Paso (14): *col.updateOne (doc, new Document ("\$set", new Document ("fecha",
FechaISO(x))));*

Paso (15): *final String FechaISO(int fecha) throws ParseException {*

String fechaString = Integer.toString(fecha);

String dateSt=fechaString;

Date traduceDate = new SimpleDateFormat("yyyyMMdd",

Locale.ENGLISH).parse(dateSt);

String dateISO= new SimpleDateFormat ("yyyy-MM-dd'T'HH

:mm:ss.mmm'Z'",

Locale.ENGLISH).format(traduceDate); System.out.print(dateISO);

return dateISO;

}

Desde la aplicación Robo 3T se puede ver un documento cualquiera de la colección “2019_CyL”.

```
{  
  "_id" : ObjectId("5c3d17a95321c12884bd4946"),  
  "ID_estacion" : "SPE00120593",  
  "fecha" : "2019-01-01T00:00:00.000Z",  
  "valor_dato" : 0  
}
```

La tercera operación consiste en agrupar todos los datos por meses de todas las estaciones y obtener la cantidad media diaria de precipitaciones en la comunidad de Castilla y León. El resultado es la cantidad diaria que multiplicándola por 30 se obtiene la media mensual y dividiendo por 10 se pasan los decilitros a litros. Este código se implementa en la clase Consulta.

Para realizar esta consulta se hace uso de la agregación que permite encadenar varias operaciones en las que se utiliza la salida de una como la entrada de la siguiente operación.

Primero se realiza una operación ‘projection’ para seleccionar los campos que se van a mostrar, en este caso “valor_dato”, “ID_estación” y “fecha” (16).

Segundo se realiza el agrupamiento por meses, sumando todos los documentos que corresponden con todas las estaciones y calculando la media aritmética del campo “valor_dato” que será la cantidad media de decilitros caídos al día. (17).

La tercera y última operación de agregación es ordenar los documentos por el campo _id que en este caso es el mes. (18).

Se almacenan en una lista el pipeline de operaciones de agregación. (19).

Con el método aggregate() se realizan las operaciones de agregación. Creando un documento que se recorre con un cursor. (20).

Selección de la colección CyL para insertar los documentos. (21).

Comienza bucle while que recorre todos los documentos realizando las operaciones:

Cálculo de media mensual de litros de lluvia caídos. (22).

Creación del documento con los campos “Año”, “Mes” y “litros”. (23).

Inserción de cada documento en la colección “CyL”. (24).

El código implementado en este método es:

Paso (16): *Document project1=new Document ("valor_dato", true).append*

```
("ID_estacion", ).append ("fecha",newDocument ("$dateFromString",
```

```
new Document("dateString","$fecha")));
```

```
Document project = new Document("$project", project1);
```

Paso (17): *Document groupFields = new Document ("_id", new Document*

```
("$month","$fecha")).append ("Documentos", new Document ( "$sum",
```

```
1)).append("litros", new Document ("$sum","$valor_dato")).append
```

```
("media",new Document ("$avg","$valor_dato"));
```

```
Document group = new Document("$group", groupFields);
```

Paso (18): *Document sort = new Document("\$sort", new Document("_id",1));*

Paso (19): *List<Document> pipeline = Arrays.asList(project,group,sort);*

Paso (20): *MongoCursor<Document> cur = col.aggregate(pipeline).iterator();*

Paso (21): *MongoCollection<Document> colCyL = database.getCollection("CyL");*

Paso (22): *while (cur.hasNext())*

```
{ Document doc = cur.next();
```

```
 List list = new ArrayList(doc.values());
```

```
double litros= (double) list.get(3);

if (litros!=0){

    litros=(double) list.get(3);

    litros=litros*3;

}
```

Paso (23): *Document document = new Document("Año",ANO).append ("Mes",*

```
list.get(0)).append("Litros",litros);
```

Paso (24): *colCyL.insertOne(document);*

Desde la aplicación Robo 3T se puede ver un documento cualquiera de la colección “CyL” donde aparece el campo _id, año, mes y litros.

```
{

    "_id" : ObjectId("5c3b42135321c11ac46eacd7"),

    "Año" : 2018,

    "Mes" : 12,

    "Litros" : 23.8582677165354

}
```

3.4. Visualización de datos

Una vez están todos los datos almacenados en mongoDB se pueden presentar por pantalla las gráficas con las precipitaciones históricas desde 1920 hasta la actualidad.

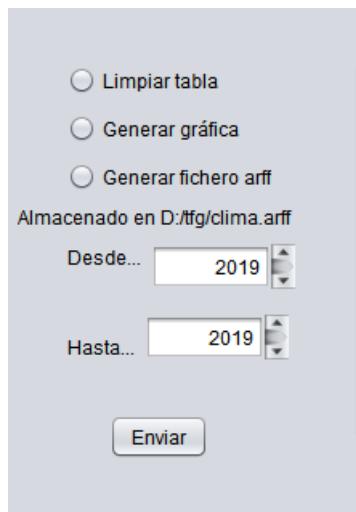


Figura 3.43. Representación gráfica de las series temporales.

En la figura 3.43 se ve la parte de la aplicación BigDataTFG que se encarga de dibujar la serie temporal.

Para generar la gráfica de la serie temporal se utilizan los dos JYearChooser del panel donde se seleccionan los años de los que se desean obtener el gráfico de la serie temporal. Como indica la figura 3.44 una vez seleccionados los años se selecciona el botón “generar gráfica” y se pulsa el botón “Enviar”.

Se pueden seleccionar series temporales con un rango de años que se deseé, la única limitación es que haya datos almacenados en la base de datos para esos años.

De las estaciones de Castilla y León existen datos almacenados desde el año 1920.

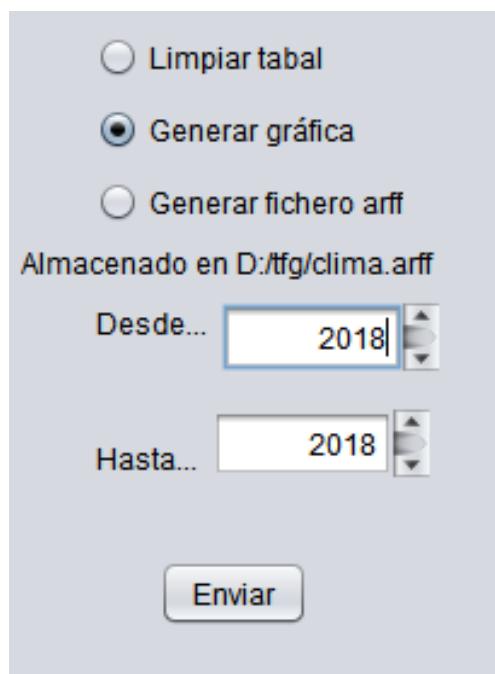


Figura 3.44. Selectores para generar gráfica.

En la figura 3.44 se ha seleccionado desde 2018 hasta 2018 de forma que la gráfica representa las lluvias caídas en la comunidad en un solo año, el 2018. Como se ve en la figura 3.45.

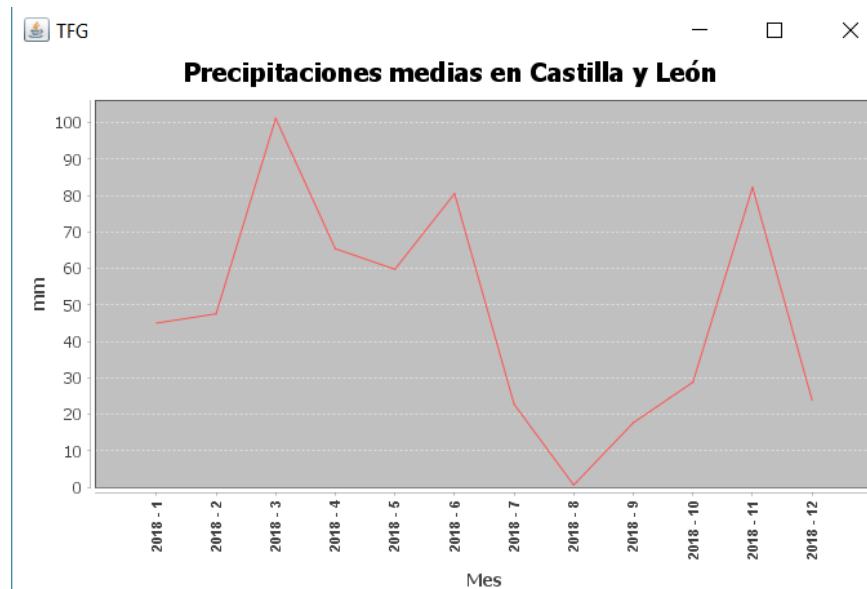


Figura 3.45. Serie temporal de las lluvias en Castilla y León el año 2018.

En la figura 3.46. Se representa gráficamente una serie temporal con las precipitaciones caídas en la comunidad de Castilla y León entre los años 2000 y 2015,

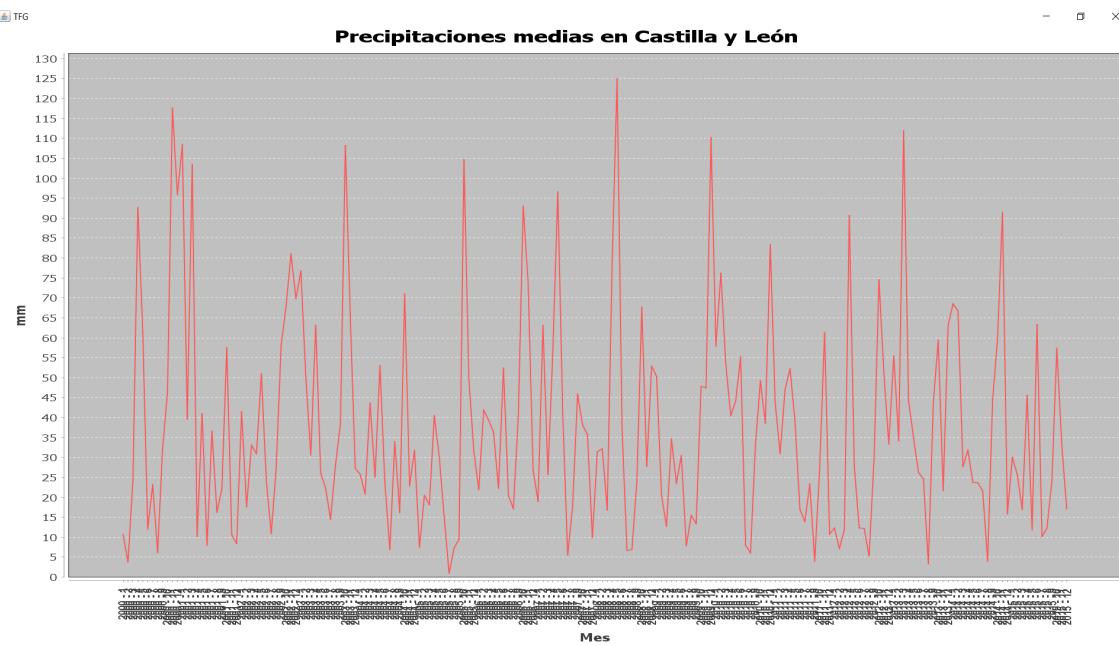


Figura 3.46. Serie temporal de las lluvias en Castilla y León periodo 2000-2015.

La serie temporal completa con todos los datos registrados desde el año 1920 hasta 2018 se representa en la figura 3.47.

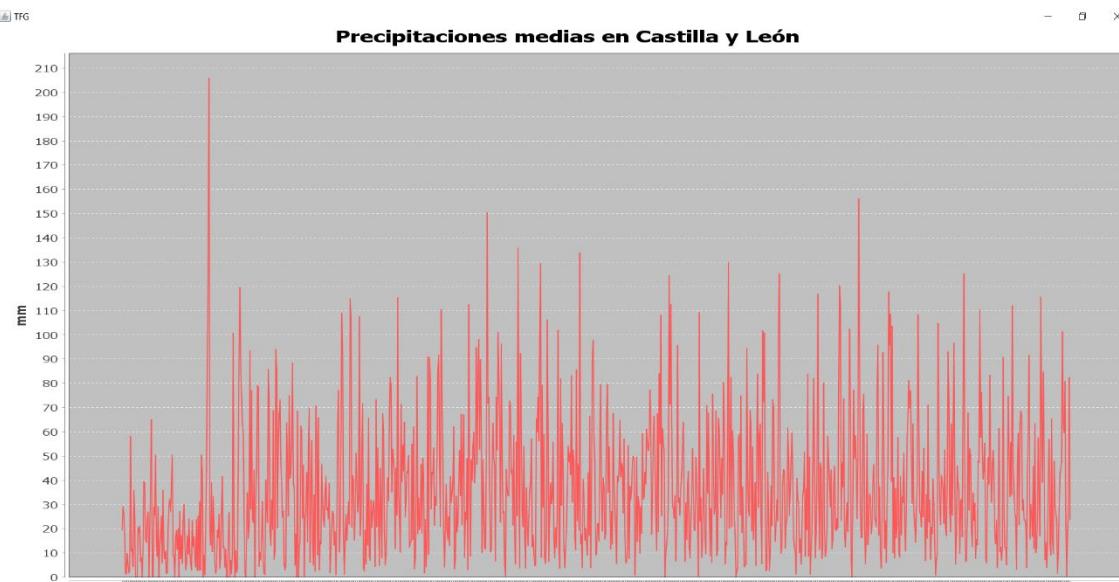


Figura 3.47. Serie temporal de las lluvias en Castilla y León periodo 1920-2018.

Para la implementación del código de este apartado se ha utilizado la librería JFreeChart que es una librería Java gratuita para la realización de gráficos.

La interface gráfica está compuesta por un JRadioButton con el que se selecciona la acción a realizar, es decir generar la gráfica, los dos JYearChooser para seleccionar los años de inicio y de fin, un JButton para ejecutar la acción y los tres JLabel informativos para facilitar el uso de la aplicación.

La clase encargada de realizar la gráfica de la serie temporal es ConsultaGraficar, que tiene el método ConsultaGraficar() para realizar la representación gráfica de la serie. El código principal es el siguiente:

El método ConsultaGraficar() recibe como parámetros el año inicial y el año final que se van a representar y se conecta a la base de datos ‘tfg’ a la colección ‘CyL’ donde están los datos.

Se crea un filtro con los años solicitados para hacer la gráfica. (25).

Con ese filtro se crea un documento que creará un objeto del tipo MongoCursor al que se le aplica el método iterator() para recorrerle. (26).

Los datos se almacenan en un arreglo DefaultCategoryDataset mediante un bucle while. (27)

Finalmente se dibuja la gráfica de líneas. (28).

Se indican características de tipo de letra y posición de etiquetas. (29) y se hace visible. (30).

Se escribe el código a continuación:

Paso (25): `Bson filtro = new Document("$gte", yearIni).append("$lte", yearFin);`

Paso (26): `Document findDocument = new Document("Año", filtro);`

`MongoCursor<Document> cur= col.find(findDocument).iterator();`

Paso (27): `DefaultCategoryDataset dataset = new DefaultCategoryDataset();`

```
while (cur.hasNext()) {  
  
    Document doc = cur.next();  
  
    List list = new ArrayList(doc.values());  
  
    String ano=Integer.toString ((int) list.get(1));  
  
    Double litros=(Double) list.get(3);  
  
    dataset.addValue(litros,"mm",ano+" - "+(Comparable) list.get(2));  
  
}
```

Paso (28): *JFreeChart chart = ChartFactory.createLineChart(*

```
"Precipitaciones medias en Castilla y León",  
"Mes",  
"mm", // Etiqueta de valores  
dataset, // Datos  
PlotOrientation.VERTICAL, // orientacion  
false, // Incluye leyenda  
true, // Incluye tooltips  
false // urls  
);
```

Paso (29): *final CategoryAxis domainAxis = plot.getDomainAxis();*

```
domainAxis.setTickLabelFont(new Font("Arial", Font.BOLD, 11));  
domainAxis.setCategoryLabelPositions (CategoryLabelPositions.  
UP_90);
```

Paso (30): *ChartFrame frame = new ChartFrame("TFG", chart);*

frame.pack();

frame.setVisible(true);

3.5. Minería de datos

La parte principal de la aplicación BigDataTFG es la dedicada al data mining, la mitad inferior de la interface gráfica es la encargada de realizar las operaciones y presentar los datos. En la figura 3.48 se muestra la interface gráfica.



Figura 3.48. Parte de la interface gráfica dedicado a Data Mining.

Antes de realizar las operaciones de minería se van a almacenar en un fichero los datos necesarios para realizar la predicción. Para ello primero se seleccionan en los dos desplegables, el año desde el cual se quieren almacenar los datos de la serie temporal hasta el año final, como se ve en la figura 3.49. Se selecciona el botón de “Generar fichero arff” y se pulsa el botón “enviar”, esta acción genera un fichero llamado ‘clima.arff’ en el directorio ‘D:\tfg’. Ver figura 3.50.

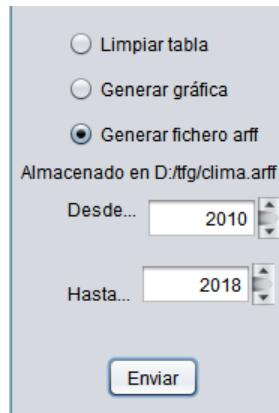


Figura 3.49. Genera fichero clima.arff.

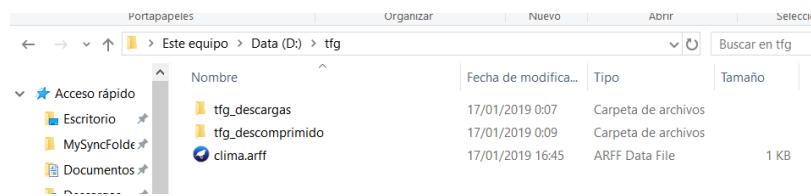


Figura 3.50. Fichero clima.arff generado en directorio D:\tfg.

Si la selección está mal formada porque el año inicial es mayor que el año final aparecerá un mensaje de aviso como se ve en la figura 3.51.

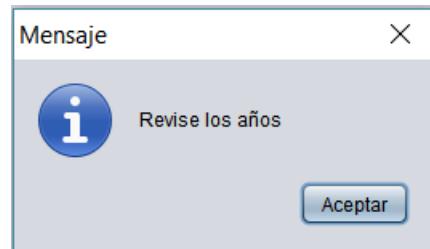


Figura 3.51. Aviso porque los años están mal indicados.

El fichero que se ha generado al pulsar el botón enviar contiene los datos de las series temporales correspondientes a los años seleccionados y tiene el formato que se puede ver en la figura 3.52.

```

@relation tfg
@attribute Date date 'yyyy-MM-dd'
@attribute litros numeric
@data
1933-1-01,20.129032258064516
1933-2-01,13.392857142857142
1933-3-01,38.90322580645161
1933-4-01,10.7000000000000001
1933-5-01,33.193548387096776
1933-6-01,19.3
1933-7-01,5.709677419354839
1933-8-01,1.741935483870968
1933-9-01,4.199999999999999
1933-10-01,19.548387096774196
1933-11-01,16.3
1933-12-01,25.838709677419356
1934-1-01,3.774193548387097
1934-2-01,8.25
1934-3-01,28.06451612903226
1934-4-01,41.5
1934-5-01,12.29032258064516
1934-6-01,12.5
1934-7-01,4.161290322580646
1934-8-01,11.612903225806452
1934-9-01,0.5
1934-10-01,0.0
1934-11-01,15.1
1934-12-01,26.516129032258064

```

Figura 3.52. Fichero clima.arff.

Las tareas de predicción se van a realizar sobre estos datos que se han almacenado en el fichero ‘clima.arff’. Se puede generar un nuevo fichero si se desea realizar las tareas de predicción con otros datos.

Una vez creado el fichero ‘clima.arff’ se pueden aplicar los algoritmos de aprendizaje para la predicción de series temporales y hay cuatro opciones. Al seleccionar una de las cuatro se abre una ventana con el sistema de archivos para cargar el fichero ‘clima.arff’ como se indica en la figura 3.53.

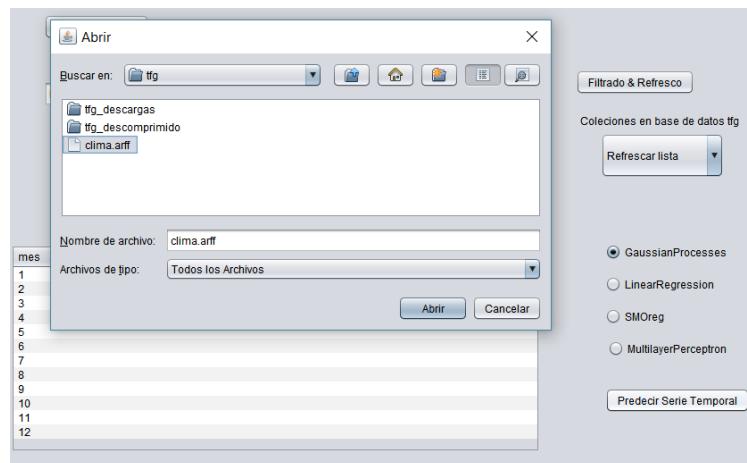


Figura 3.53. Búsqueda de la carpeta clima.arff para predicción usando el algoritmo GaussianProcesses.

Pulsando el botón de “abrir” se selecciona dicho fichero y se muestra el resultado en la tabla situada en la interface. En breves instantes aparece en la tabla la previsión de los próximos doce meses, correspondientes al año 2019. En la figura 3.54 se muestra la previsión usando GaussianProcesses.

mes	predicción
1	36.928580028515086
2	28.3196898952773
3	33.55313899791029
4	42.50296112265436
5	50.52926287250232
6	39.82308101464608
7	13.532044736367522
8	10.336182636412884
9	31.440973896079957
10	45.331969293839116
11	51.66101136902192
12	46.51606874196975

Figura 3.54. Predicción de los próximos doce meses. GaussianProcesses.

Repetiendo estos pasos con los otros tres algoritmos se presentan en la misma tabla en la siguiente columna la predicción LinearRegression, la SMoReg y MultilayerPerceptron. En la figura 3.55 aparecen las predicciones con los cuatro métodos.

mes	GaussianProcesses	LinearRegression	SMoReg	MultilayerPerceptron
1	36.9285800285150...	38.533059765964...	35.680412275066125	170.245745891379
2	28.3196898952773	32.837097541995...	24.759844956569154	-44.39107632394
3	33.55313899791029	33.204618204617...	29.598521981261527	232.59489565282897
4	42.50296112265436	45.380120665306...	41.71166793985091	-192.20282220170046
5	50.52926287250232	53.419892975174...	45.64124274006388	381.63793344619114
6	39.82308101464608	43.4478381148835	31.94457606024149	-19.48940023588969
7	13.5320447363675...	18.132558691345...	10.432022991993875	326.8409634312685
8	10.3361826364128...	9.1943975304840...	12.33262633292331	-203.9778118176637
9	31.4409738960799...	34.461354130172...	33.12810358943698	332.5957560014541
10	45.3319692938391...	49.8101704435209	42.72578731493995	-62.3582906942826
11	51.66101136902192	53.418817713739...	46.021961749949696	372.5118042504871
12	46.51606874196975	49.968284746171...	36.54049898605824	-260.7437831042324

Figura 3.55. Predicción 12 meses. GaussianProcesses, LinearRegression, SMoReg y MultilayerPerceptron,

Si se quieren hacer nuevas predicciones basta con limpiar la tabla seleccionando el botón “limpiar tabla” y pulsando el botón “enviar”, situados en la parte inferior izquierda de la interface como se muestra en la figura 3.56.

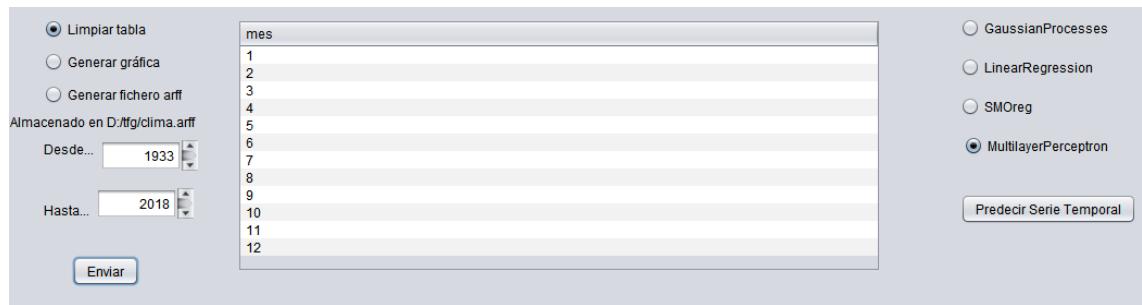


Figura 3.56. Limpiar tabla.

El código utilizado para la implementación se explica a continuación:

Para escribir el fichero ‘clima.arff’ se ha creado la clase FicheroArff, el método ficheroArff() recibe dos enteros que son el año de inicio y el año final sobre los que se generan los datos.

Se crea el archivo ‘clima.arff’. (31) y se escribe la cabecera (32).

Tras conectarse a la base de datos ‘tfsg’ y a la colección ‘CyL’ se crea un filtro con los años que llevan datos y que se escribirán en el archivo. (33).

Se recorre el objeto Mongocursor (34) y mediante un bucle while se van escribiendo los datos en un ArrayList para escribirlos en el fichero. (35).

Paso (31): *File archivo=new File("D:/tfsg/clima.arff");*

escribir = new FileWriter(archivo,false);

Paso (32): *escribir.write(saludo+atributo1+atributo2+dato);*

Paso (33): *Bson filtro = new Document("\$gte", anoIni).append("\$lte", anoFin);*

```
Document findDocument = new Document("Año", filtro);
```

Paso (34): *MongoCursor<Document> cur = col.find(findDocument).iterator();*

Paso (35): *{while (cur.hasNext()) {*

```
    Document doc = cur.next();
```

```
    List list = new ArrayList(doc.values());
```

```
    escribir.write(list.get(1)+"-");
```

```
    escribir.write(list.get(2)+"-");
```

```
    escribir.write("01,");
```

```
    escribir.write(list.get(3)+"\n");
```

```
}
```

```
    escribir.close();
```

```
}
```

Una vez almacenado el archivo con los datos ya se puede procesar para realizar las tareas de predicción. Para ello se ha utilizado la librería timeseriesForecasting de weka que es opensource y de libre disposición.

La clase TimeSeriesTFG implementa el método timeSeriesTFG() que recibe dos parámetros, uno un String con el método que se va a usar para hacer la predicción y otro de tipo File con el fichero de datos que es el ‘clima.arff’.

Lo primero es leer el fichero con los datos a procesar. (36) y se cargan (37).

Se crea un objeto WekaForecaster pronosticador y se le entregan los datos leídos del fichero ‘clima.arff’. (38).

Se establecen los objetivos a pronosticar. (39).

Para la selección del algoritmo: se aplican procesos gaussianos para el cálculo de regresión, GaussianProcesses() (40), LinearRegression() (41), SMOrreg() (42) y MultilayerPerceptron() (43).

Se añade un campo indicador del mes del año. (44):

Se añade un campo indicador del trimestre del año. (45):

Se lleva a cabo la construcción del modelo. (46).

Entregar al pronosticador los datos (47).

Se guarda el pronóstico para los doce meses siguientes a los datos entregados. (48).

Salida de las predicciones. (49).

Se detalla el código utilizado:

Paso (36): *File pathToWineData =archivo;*

Paso (37): *Instances clima = new Instances(new BufferedReader(new
FileReader(pathToWineData)));*

Paso (38): *WekaForecaster forecaster = new WekaForecaster();*

Paso (39): *forecaster.setFieldsToForecast("litros");*

Paso (40): *if(methodo=="GaussianProcesses") {*

forecaster.setBaseForecaster(new GaussianProcesses());

}else

Paso (41): *if(methodo=="LinearRegression"){*

forecaster.setBaseForecaster(new LinearRegression());

}else

Paso (42) *if(methodo=="SMOrreg"){*

forecaster.setBaseForecaster(new SMOreg());

}else

Paso (43): *forecaster.setBaseForecaster(new MultilayerPerceptron());*

forecaster.getTSLagMaker().setTimeStampField("Date");

forecaster.getTSLagMaker().setMinLag(1);

forecaster.getTSLagMaker().setMaxLag(12);

Paso (44): *forecaster.getTSLagMaker().setAddMonthOfYear(true);*

Paso (45): *forecaster.getTSLagMaker().setAddQuarterOfYear(true);*

Paso (46): *forecaster.buildForecaster(clima, System.out);*

Paso (47): *forecaster.primeForecaster(clima);*

Paso (48): *List<List<NumericPrediction>> forecast = forecaster.forecast(12,*

System.out);

Paso (49): *for (int i = 0; i < 12; i++) {*

List<NumericPrediction> predsAtStep = forecast.get(i);

for (int j = 0; j < 1; j++) {

NumericPrediction predForTarget = predsAtStep.get(j);

System.out.print("'" + predForTarget.predicted() + " ");

datos [i] =predForTarget.predicted();

}

}

4.CAPÍTULO IV. RESULTADOS

4.1. Introducción.

Las series temporales permiten la descripción de fenómenos que suceden en el tiempo de forma clara y fácil de comprender para las personas. La minería de datos construye modelos que explican su evolución y predicen los valores que tomarán en el futuro.

Las series temporales que reflejan la cantidad de lluvia caída en una determinada zona son estacionarias porque oscilan alrededor de un nivel constante además tienen un comportamiento que se repite a lo largo del tiempo, por esta característica se las denomina series estacionales.

Se aprecia en las series como las precipitaciones disminuyen los meses de verano y aumentan en invierno. También hay series que son relativamente altas y otras son bajas coincidiendo con años lluviosos y con años más secos. Cuando el valor medio de las precipitaciones caídas depende del mes se dice que esa serie tiene estacionalidad.

Un modelo se denomina univariante cuando representa la evolución de una serie temporal y predice su comportamiento en el futuro teniendo en cuenta una variable. Las hipótesis utilizadas para predecir el valor que tomarán en el futuro es que las condiciones futuras se asemejarán a las pasadas.

La librería utilizada para la predicción de series temporales está desarrollada por Pentaho Community y se denomina timeseriesForecasting y está disponible para Weka en versiones posteriores a la 3.7.3.

Las predicciones hechas con esta librería realizan un aprendizaje automático de extracción de datos para modelar las series temporales, transformando los datos para que los algoritmos de aprendizaje proposicional habituales los puedan procesar.

Trabaja eliminando el orden temporal de los ejemplos de entrada codificando la dependencia del tiempo con campos de entrada adicionales que denomina variables retrasadas. Además, los campos se pueden calcular automáticamente para permitir que los algoritmos modelen tendencia y estacionalidad, por ejemplo, en caso de periodicidad mensual se crean automáticamente los campos año y trimestre.

Con los datos transformados ya se puede aplicar cualquiera de los algoritmos de regresión para aprender el modelo, como regresión lineal múltiple, máquinas de vectores de soporte para regresión, árboles de decisión con funciones de regresión lineal en las hojas. Estas técnicas pueden mejorar a las más tradicionales técnicas estadísticas como ARMA y ARIMA.

4.2. Resultados.

Los resultados obtenidos de la aplicación de los cuatro algoritmos de aprendizaje para modelar la serie temporal con datos recogidos entre los años 1933 y 2018 han sido los siguientes:

mes	GaussianProcesses	LinearRegression	SMOreg	MultilayerPerceptron
1	36.9285800285150...	38.533059765964...	35.680412275066125	170.245745891379
2	28.3196898952773	32.837097541995...	24.759844956569154	-44.39107632394
3	33.55313899791029	33.204618204617...	29.598521981261527	232.59489565282897
4	42.50296112265436	45.380120665306...	41.71166793985091	-192.20282220170046
5	50.52926287250232	53.419892975174...	45.64124274006388	381.63793344619114
6	39.82308101464608	43.4478381148835	31.94457606024149	-19.48940023588969
7	13.5320447363675...	18.132558691345...	10.432022991993875	326.8409634312685
8	10.3361826364128...	9.1943975304840...	12.33262633292331	-203.9778118176637
9	31.4409738960799...	34.461354130172...	33.12810358943698	332.5957560014541
10	45.3319692938391...	49.8101704435209	42.72578731493995	-62.3582906942826
11	51.66101136902192	53.418817713739...	46.021961749949696	372.5118042504871
12	46.51606874196975	49.968284746171...	36.54049898605824	-260.7437831042324

Figura 4.1. Resultado obtenido desde la aplicación BigDataTFG.

- Procesos gaussianos para la regresión (GaussianProcesses). Este algoritmo implementa procesos gaussianos para la regresión sin ajuste de hiperparámetros. Para facilitar la elección de un nivel de ruido adecuado, esta implementación aplica la normalización / estandarización al atributo objetivo,

así como a los otros atributos (si la normalización / estandarización está activada). Los valores que faltan se reemplazan por la media. Se han usado 1031 instancias. La tabla 4.1 muestra la previsión y la figura 4.2. su gráfica.

Tabla 4.1. Algoritmo de progresos gaussianos para la regresión.

Año 2019	Litros Previstos
Enero	38,53
Febrero	32,84
Marzo	33,20
Abril	45,38
Mayo	53,42
Junio	43,45
Julio	18,13
Agosto	9,19
Septiembre	34,46
Octubre	49,81
Noviembre	53,42
Diciembre	49,97
Total	461,81

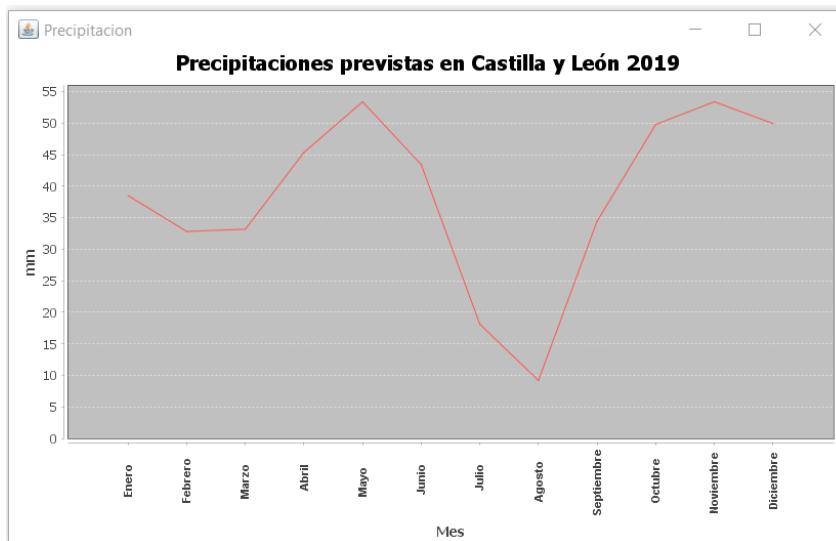


Figura 4.2. Gráfica previsión usando algoritmo de progresos gaussianos para la regresión.

- Regresión lineal (LinearRegression). Este algoritmo hace uso de regresión lineal para la predicción. Utiliza el criterio de Akaike para la selección de modelos, y es capaz de trabajar con instancias ponderadas.

El criterio de información de Akaike /AIC) ofrece una medida de calidad relativa sobre un modelo estadístico para un conjunto de datos. Para seleccionar un modelo se basa en la entropía de la información.

$$AIC = 2k - 2 \ln(L)$$

Siendo k el número de parámetros en el modelo estadístico y L el valor máximo de la función de verosimilitud para el modelo estimado.

Para tamaños de muestra finitos se suele usar la corrección:

$$AIC_C = AIC + \frac{2k^2 + 2k}{n - k - 1}$$

La tabla 4.2 muestra la previsión y la figura 4.3 su gráfica.

Tabla 4.2. Algoritmo de regresión lineal.

Año 2019	Litros Previstos
Enero	36,93
Febrero	28,32
Marzo	33,55
Abril	42,50
Mayo	50,53
Junio	39,82
Julio	13,53
Agosto	10,34
Septiembre	31,44
Octubre	45,33
Noviembre	51,66
Diciembre	46,52
Total	430,48

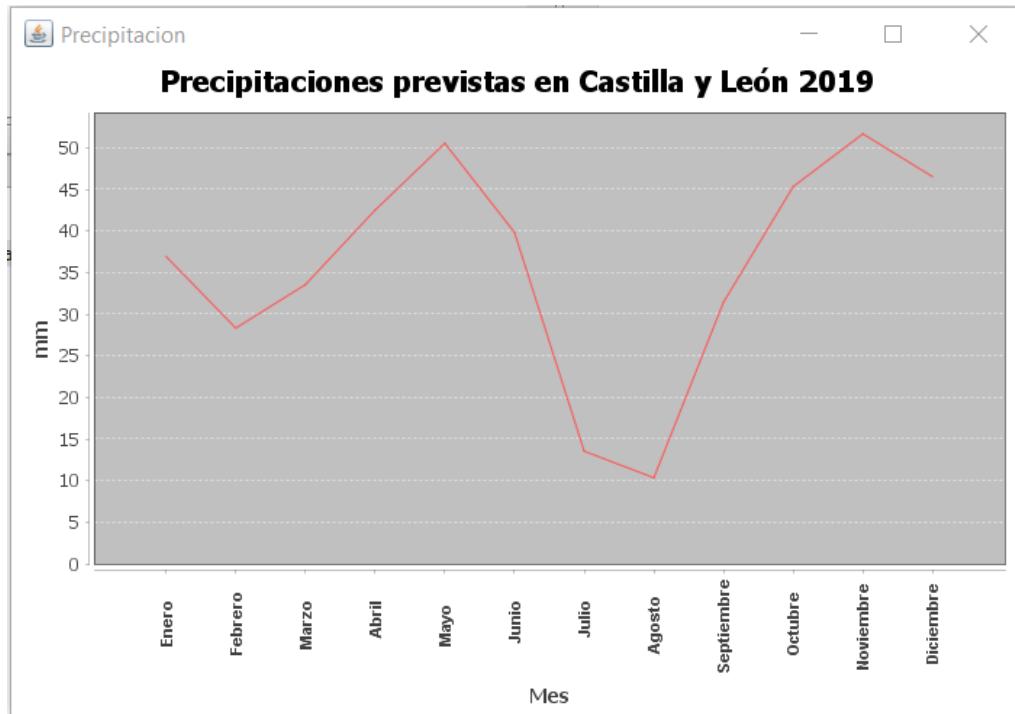


Figura 4.3. Gráfica previsión usando algoritmo de regresión lineal.

- Máquina de vectores de soporte para la regresión (SMOreg). La Máquina de Vector Soporte (MVS) fue desarrollada inicialmente por Vapnik para construir clasificadores. Se basa en la teoría estadística del aprendizaje que permite seleccionar clasificadores que establecen una cota superior para el error de prueba minimizándola. Se caracterizan por que consiguen clasificadores que modelizan correctamente sobre datos que aún no se han visto. Producen modelos que no sobreajustan los datos. Construyen regiones de decisión no lineales de una manera discriminativa mediante la introducción de una función núcleo. Los resultados de la aplicación de este algoritmo se presentan en la tabla 4.3 y en la figura 4.4.

Tabla 4.3. Algoritmo MVS para la regresión.

Año 2019	Litros Previstos
Enero	35,68
Febrero	24,76
Marzo	29,60
Abril	41,71
Mayo	45,64
Junio	31,94
Julio	10,43
Agosto	12,33
Septiembre	33,13
Octubre	42,73
Noviembre	46,02
Diciembre	36,54
Total	390,52

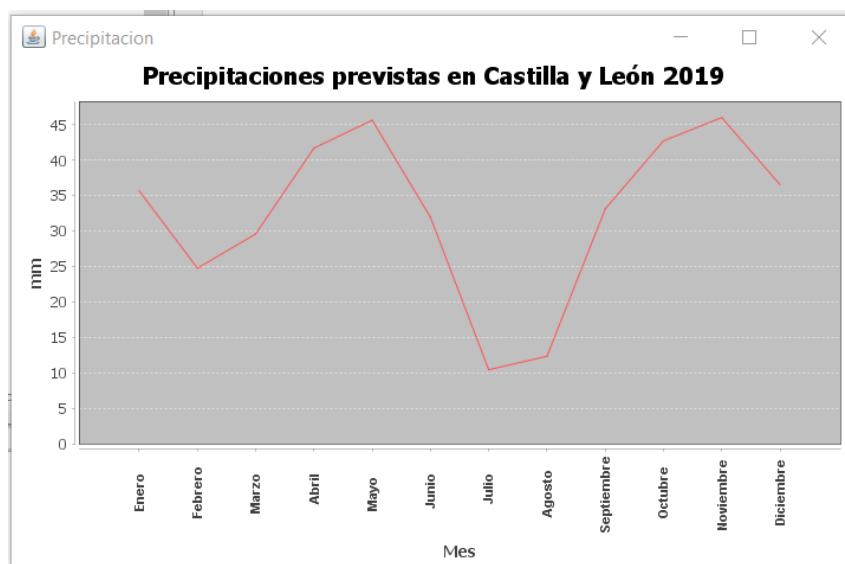


Figura 4.4. Gráfica previsión usando algoritmo de MVS.

- Perceptrón multicapa (MultilayerPerceptron). Un clasificador que utiliza la propagación hacia atrás para aprender un perceptrón de múltiples capas para clasificar instancias. La red puede construirse a mano o configurarse mediante

una heurística simple. Los parámetros de la red también pueden ser monitoreados y modificados durante el tiempo de entrenamiento. Los nodos en esta red son todos sigmoides (excepto cuando la clase es numérica, en cuyo caso los nodos de salida se convierten en unidades lineales sin umbral). En la tabla 4.4 y en la figura 4.5 están los resultados obtenidos con este algoritmo.

Tabla 4.4. Algoritmo MultilayerPerceptron.

Año 2019	Litros Previstos
Enero	170,25
Febrero	-44,39
Marzo	232,59
Abrial	-192,20
Mayo	381,64
Junio	-19,49
Julio	326,84
Agosto	-203,98
Septiembre	332,60
Octubre	-62,36
Noviembre	372,51
Diciembre	-260,74
Total	1033,26

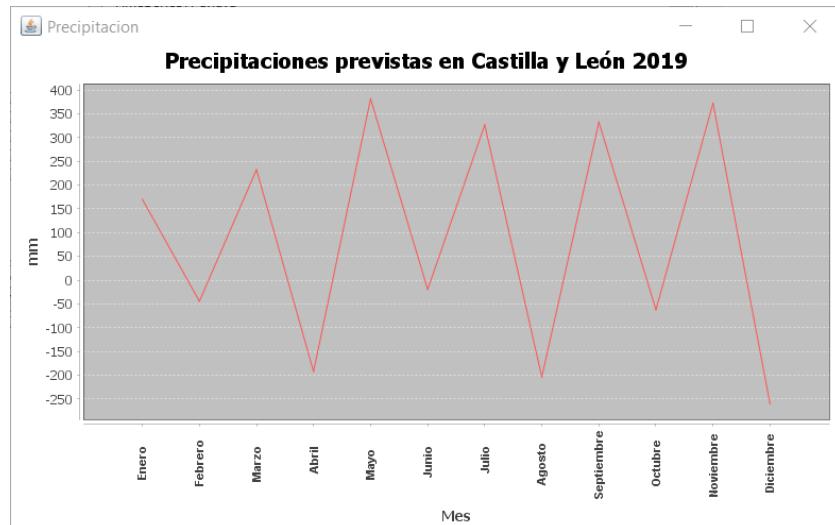


Figura 4.5. Gráfica previsión usando algoritmo MultilayerPerceptron.

- Si se comparan los resultados obtenidos y descartando el algoritmo perceptrón multicapa por presentar unos resultados que no son aceptables para el estudio de la serie temporal de este trabajo, en la figura 4.6 se ve la gráfica que compara los resultados de los tres algoritmos restantes. Se comprueba que las tres líneas que representan la predicción son muy semejantes.

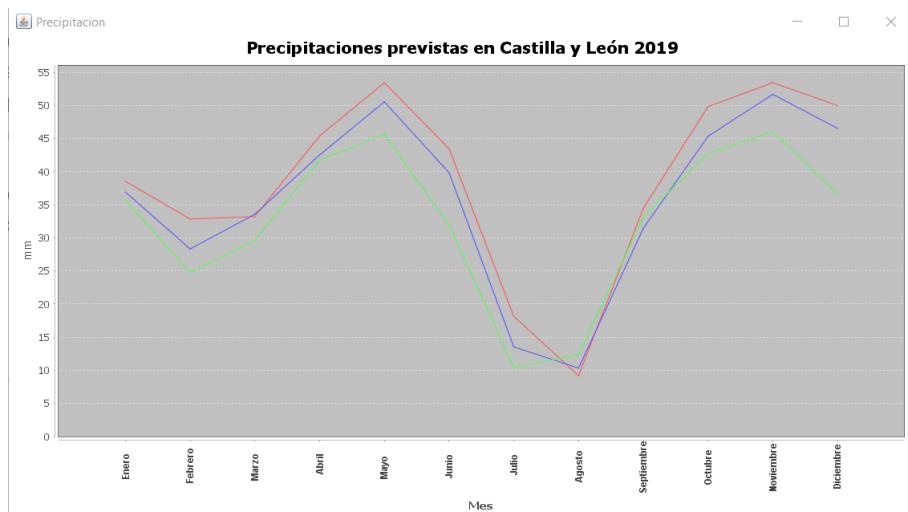


Figura 4.6. Gráfica previsión precipitaciones. Rojo: Regresión lineal, Azul: Procesos gaussianos, Verde: MVS.

- Para poder comprobar los resultados de los algoritmos que se han utilizado, se han extraído mediante la herramienta Weka los resultados para las diferentes métricas de evaluación.
 1. Error absoluto medio (MAE): $\text{suma}(\text{abs}(\text{pronosticado} - \text{real})) / N$
 2. Error cuadrático medio (MSE): $\text{suma}((\text{pronosticada} - \text{real})^2) / N$
 3. Error cuadrático medio (RMSE): $\text{sqrt}(\text{suma}((\text{predicho} - \text{real})^2) / N)$
 4. Error porcentual absoluto medio (MAPE): $\text{suma}(\text{abs}((\text{predicho} - \text{real}) / \text{actual})) / N$
 5. Precisión de dirección (DAC): $\text{recuento}(\text{signo}(\text{actual_current} - \text{actual_previous}) == \text{signo}(\text{pred_current} - \text{pred_previous})) / N$
 6. Error absoluto relativo (RAE): $\text{suma}(\text{abs}(\text{pronosticado} - \text{real})) / \text{suma}(\text{abs}(\text{previous_target} - \text{actual}))$

7. Error cuadrático relativo (RRSE): $\text{sqrt}(\text{suma}((\text{predicho} - \text{real})^2) / N)$
 $/ \text{sqrt}(\text{suma}(\text{previous_target} - \text{actual})^2) / N)$

Tabla 4.5. Evaluación de progresos gaussianos para la regresión.

Target	1020	1019	1018	1017	1016	1015	1014	1013	1012	1011	1010	1009
Algoritmo Gaussian processes												
Mean absolute error	19,38	19,65	19,68	19,75	19,78	19,79	19,80	19,83	19,86	19,91	19,88	19,87
Root relative squared error	72,29	73,04	64,88	63,04	63,62	63,16	64,22	63,59	62,82	63,70	65,48	68,24
Direction accuracy	53,58	60,61	60,57	60,73	60,30	59,76	59,33	59,68	59,94	59,41	59,07	59,13
Relative absolute error	74,25	75,35	65,17	63,60	64,04	64,15	65,10	65,13	62,99	64,04	64,62	69,66
Mean absolute percentage error	215,55	212,50	214,50	216,48	218,92	223,31	223,90	225,42	223,05	216,93	218,53	218,54
Root mean squared error	25,04	25,30	25,31	25,36	25,40	25,42	25,43	25,48	25,50	25,57	25,58	25,58
Mean squared error	627,11	639,93	640,70	643,18	645,03	645,99	646,53	649,14	650,21	653,97	654,12	654,22

Tabla 4.6. Algoritmo de regresión lineal.

Algoritmo LinearRegression	1020	1019	1018	1017	1016	1015	1014	1013	1012	1011	1010	1009
Mean absolute error	19,37	19,63	19,66	19,74	19,76	19,76	19,76	19,77	19,80	19,85	19,83	19,82
Root relative squared error	72,45	73,17	64,99	63,14	63,69	63,17	64,22	63,61	62,82	63,73	65,51	68,28
Direction accuracy	52,70	59,73	60,47	58,96	59,01	59,07	59,13	59,09	59,35	59,41	59,46	59,42
Relative absolute error	74,22	75,24	65,10	63,58	63,97	64,06	64,98	64,94	62,79	63,86	64,45	69,48
Mean absolute percentage error	218,61	215,78	218,01	219,81	220,79	220,53	220,82	223,54	222,93	216,67	216,74	216,83
Root mean squared error	25,09	25,34	25,35	25,40	25,42	25,42	25,43	25,49	25,50	25,58	25,59	25,59
Mean squared error	629,75	642,25	642,86	645,16	646,38	646,25	646,53	649,56	650,18	654,49	654,64	654,87

Tabla 4.7. Evaluación MVS para la regresión.

Algoritmo SMOreg	1020	1019	1018	1017	1016	1015	1014	1013	1012	1011	1010	1009
Mean absolute error	18,83	19,14	19,13	19,21	19,22	19,24	19,27	19,30	19,37	19,44	19,46	19,48
Root relative squared error	74,08	75,12	66,74	65,03	65,70	65,19	66,19	65,48	64,53	65,58	67,46	70,29
Direction accuracy	54,27	60,12	60,37	60,33	60,00	60,85	60,32	59,98	60,04	59,90	60,06	58,83
Relative absolute error	72,19	73,37	63,35	61,86	62,23	62,38	63,35	63,40	61,43	62,55	63,26	68,28
Mean absolute percentage error	192,59	180,41	182,91	182,02	182,85	186,54	189,22	190,07	187,35	178,95	180,14	180,30
Root mean squared error	25,65	26,02	26,04	26,16	26,23	26,23	26,21	26,24	26,19	26,33	26,35	26,34
Mean squared error	658,14	676,97	677,97	684,44	687,85	688,23	686,86	688,36	686,02	693,04	694,13	694,03

Comparando los resultados obtenidos, el algoritmo SMOreg tiene mejores métricas en la evaluación de errores por tanto los resultados serán más confiables que los reflejados por el resto de algoritmos.

5. CAPÍTULO V. PLANIFICACIÓN

5.1. Fases de desarrollo

Para realizar este trabajo se han dedicado tres meses y medio distribuyendo el tiempo de la siguiente manera:

1. Planteamiento del problema o necesidad que se pretende solucionar. Se proyecta crear un sistema software que integre las fases del proceso de KDD, recopilación de datos, almacenamiento en una base de datos, filtrado de datos, modelado de los datos e interpretación y evaluación. En esta fase se ha forjado la idea principal del trabajo y se ha localizado la fuente de datos que se ha utilizado. El tiempo estimado ha sido de siete días.
2. Estudio de las tecnologías big data más utilizadas actualmente, profundización en las tareas de minería de datos y de series temporales. Ampliar el conocimiento sobre la agricultura en la comunidad de Castilla y León, principales cultivos de secano y sus necesidades hídricas. Tiempo estimado doce días.
3. En esta fase se han definido los requisitos del sistema y la arquitectura de los componentes que lo van a formar. Tiempo cinco días.
4. Configuración y descarga del software necesario para la implementación del sistema. Se ha instalado mongoDB, se ha actualizado Java y netbeans, se han instalado herramientas auxiliares para pruebas de comprobación como Robo 3T y Weka. Además, ha sido necesario un estudio profundo de la base de datos mongoDB y de sus principales operaciones CRUD (insertar, buscar, actualizar y eliminar documentos). Se ha profundizado en el estudio del driver java para mongoDB ya que la implementación de las consultas a la base de datos se ha hecho desde java. Tiempo estimado veinte días.

5. Implementación del código. Esta tarea ha sido la que más tiempo de trabajo ha ocupado, unos treinta días.
6. Configuración del entorno de pruebas. Descarga de los datos en computadora y carga de todos ellos en la base de datos local. Ha supuesto una descarga aproximada de 100 GB en total. Tiempo estimado tres días.
7. Pruebas. Una vez escrito el código se han hecho todas las pruebas necesarias hasta corregir los fallos que se han producido. Además, se han estudiado las conclusiones obtenidas de la aplicación del sistema a los datos obtenidos. Tiempo estimado siete días.
8. Redacción de la memoria que se ha ido escribiendo durante todas las fases del trabajo, con un tiempo estimado de veintiún días.

Todas estas fases se han realizado con el apoyo del director del TFG.

5.2. Diagrama de Gantt

Se puede ver en la figura 5.1 el diagrama de Gantt mostrando el tiempo que se ha dedicado a cada una de las fases del trabajo indicadas en el punto anterior.

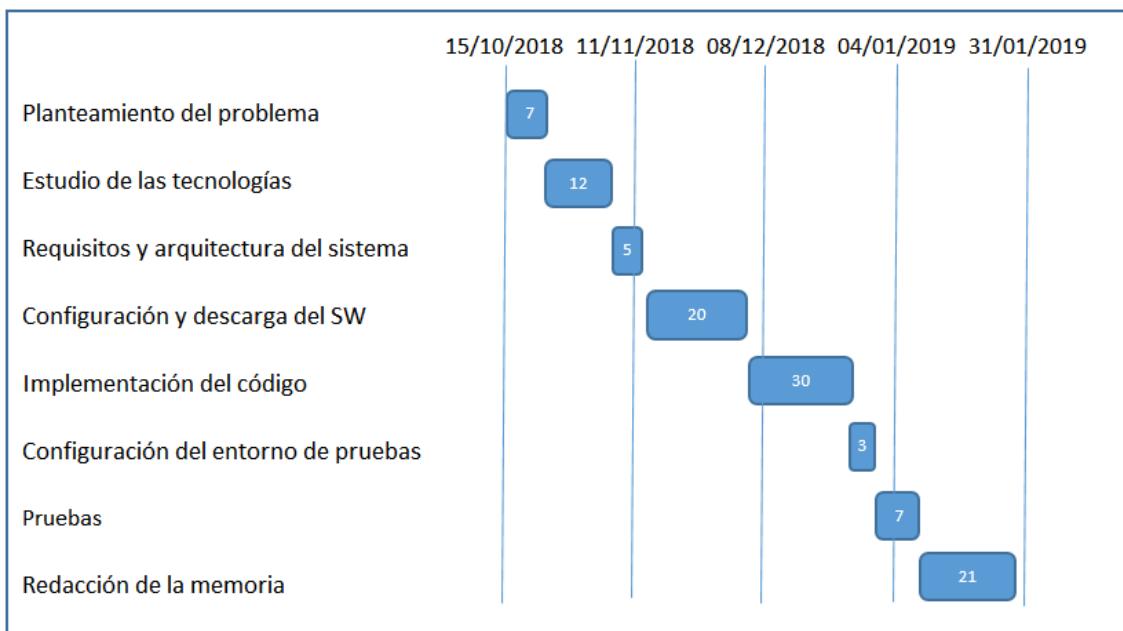


Figura 5.1. Diagrama de Gantt del TFG.

6. CAPÍTULO VI. PRESUPUESTO

6.1. Recursos.

Realizar este trabajo en un entorno real no difiere prácticamente del realizado para el trabajo de fin de grado. Los recursos para desarrollar este proyecto serían:

- Recursos humanos: Un ingeniero informático y un ingeniero agrónomo.
- Recursos materiales: un PC portátil, conexión a internet.
- Software propietario: Windows 10, Office Professional Plus 2016.
- Software gratuito: Netbeans, mongoDB, Java, Robo 3T, Weka.

6.2. Presupuesto

Autor y departamento:

-Francisco Javier Moreno Hermosilla

-Ingeniería Informática

Proyecto:

- Título: Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.

- Duración: 4 meses

Presupuesto del proyecto: 18974 €

Impuestos 21% IVA: 3984.54 €

Presupuesto IVA incluido: 22958.54 €

Desglose del presupuesto:

La tabla 6.1 desglosa los costes materiales del proyecto.

Tabla 6.1. Costes materiales del proyecto.

Concepto	Coste €	Tiempo—meses	Coste Imputable 10%
PC portátil	800	4	80
Office 2016	140	4	14
Conexión internet	80	4	320
Total			414

Para calcular el coste de los recursos humanos empleados se suman las horas empleadas y luego se multiplica por el precio unitario, se desglosa en la tabla 6.2.:

Tabla 6.2. Tiempo dedicado por tareas.

	Días	Horas Ing. Informático	Horas Ing. Agrónomo
Planteamiento del problema	7	28	28
Estudio de las tecnologías	12	48	
Requisitos y arquitectura del sistema	5	20	
Configuración y descarga del SW	20	80	
Implementación del código	30	120	
Configuración del entorno de pruebas	3	12	
Pruebas	7	28	
Redacción de la memoria	21	84	16
Total		420	44

Tabla 6.3. Costo de los recursos humanos.

Concepto	Horas	Coste por hora	Total
Ingeniero informático	420	40	16800
Ingeniero agrícola	44	40	1760
total			18560

En la tabla 6.3 se han desglosado los costes de los recursos humanos y en la tabla 6.4 se han sumado los costes de recursos materiales más recursos humanos para obtener el coste total:

Tabla 6.4. Coste total.

Coste recursos humanos	18.560
Coste recursos materiales	414
Total	18.974

7. CAPÍTULO VII. CONCLUSIONES Y TRABAJO FUTURO

7.1. Conclusiones.

El pronóstico del tiempo es complicado cuando se trata de medio y largo plazo. Son habituales las predicciones a corto plazo que no van más allá de 5 a 10 días donde la predicción sí que es bastante acertada, pero si se trata de pronosticar todo un año ya es menos fiable.

En Castilla y León la agricultura es una de las principales fuentes de ingresos y la climatología tiene importante peso en la economía de la región. El clima es un factor productivo muy importante junto con otros como el suelo o la tecnología utilizada. Por esto el conocimiento del clima es información importante para mejorar la producción agrícola.

A partir de todos los datos almacenados en la base de datos se ha recopilado información de las precipitaciones medias de los últimos años en la región. En el anexo B se puede ver la gráfica de barras desde el año 2003 hasta el 2018 y la predicción para el año 2019 utilizando el algoritmo de máquina de vectores soporte para la regresión (MVS). Los datos históricos de precipitaciones de los años 2003 al 2018 han sido calculados según se indica en anexo B.

Un estudio realizado por la junta de Castilla y León analiza periodos con características necesarias para ser considerados como de sequía. El Porcentaje de Precipitación Normal (PPN) es la relación entre la precipitación acumulada en un periodo de tiempo y la precipitación media anual para una región. La precipitación media anual histórica se obtiene a partir del promedio de las precipitaciones anuales en 30 años (mínimo). Siendo la situación normal: 100%. De la observación de ese gráfico se

consideran años secos entre el año 2000 y 2015 los que se sitúan por debajo del nivel 100. Los años 2001, 2004, 2005, 2007, 2009, 2011, 2012 y 2015 serán considerados secos.

Teniendo en cuenta las previsiones de lluvia para el 2019 se predice que será un año seco y viendo las producciones obtenidas por los cultivos en el intervalo de años que van del 2003 al 2016, adjuntas en anexo C de esta memoria, donde se disponen de datos de rendimiento de diferentes cultivos se pueden tomar decisiones. También habría que tener en cuenta factores de otro tipo como las subvenciones que reciben los agricultores por su explotación, esta variable también puede influir a la hora de decidir el cultivo a sembrar, así como el precio de venta del producto.

Según las tablas de rendimiento y la predicción obtenida a partir de este trabajo, se deduce que la producción de triticale en años secos supera a la de otros cereales, por tanto, es una de las mejores opciones. Otros cereales como el centeno también mantienen su rendimiento en años con menos lluvias sustituyendo a cereales como el trigo y la cebada que en caso de sequía disminuye la producción en mayor medida. La alfalfa y la veza forrajera también son cultivos que tienen buenos rendimientos en años con precipitaciones semejantes a las previstas para 2019.

En el anexo C se pueden ver las figuras con los rendimientos de los principales cultivos de secano en Castilla y León los años comprendidos entre 2003 y 2016.

7.2. Trabajo futuro.

Este trabajo es solamente el comienzo teniendo unas posibilidades de evolución prácticamente ilimitadas.

El uso de otras variables mejora las predicciones de las series univariantes porque se pueden construir modelos que cuenten con las dependencias entre variables. Se denominan modelos de regresión dinámica o de función de transferencia y su uso mejoraría los resultados obtenidos en este trabajo.

Castilla y León es una región muy extensa y el clima es muy diferente de unas zonas a otras, realizar un estudio teniendo en cuenta estos desequilibrios dentro de la región va a permitir obtener resultados más fiables.

No solo las diferencias climáticas son importantes dentro de cada región, sino que además la riqueza del terreno varía también de unas zonas a otras. Su estudio aportaría también una mejoría considerable de los resultados.

La gran posibilidad de evolución para trabajos futuros reside en poder realizar predicciones climáticas de cualquier zona del mundo ya que hoy en día las bases de datos permiten almacenar cantidades de datos tan grandes que la capacidad de almacenamiento no es un inconveniente.

Si aún no es suficiente y se quiere ampliar el campo del conocimiento, teniendo en cuenta variables climáticas como temperatura, precipitación, viento y radiación el estudio se podría extender a la industria de la energía. Cada vez va cogiendo mayor importancia la generación de energías renovables para las cuales el estudio de elementos como el viento y las radiaciones solares permitirían determinar las zonas mejor situadas para su explotación.

Esto convertiría muchos campos con bajos rendimientos agrícolas en productores de energía eólica o solar, reportando ingresos extra a los agricultores.

Bibliografía

- Achari, Shiva. (2015) *Hadoop Essentials*, Packt Publishing Ltd. ProQuest Ebook Central, <http://ebookcentral.proquest.com/lib/udima-ebooks/detail.action?docID=2039889>.
- Aranda, Y. R., & Sotolongo, A. R. (2013). Integración de los algoritmos de minería de datos 1R, PRISM e ID3 a PostgreSQL. *JISTEM - Journal of Information Systems and Technology Management*, 10(2), 389-406. doi:10.4301/S1807-17752013000200012
- Castilla y León. (2018). En *Wikipedia, la enciclopedia libre*. Recuperado de https://es.wikipedia.org/w/index.php?title=Castilla_y_Le%C3%B3n&oldid=111672359
- Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2), 171-209. doi:<http://dx.doi.org/10.1007/s11036-013-0489-0>
- Fayyad, U. M., Piatetsky-Shapiro, G. y Smyth, P. (1996). From data mining to knowledge discovery: an overview. En U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, y R. Uthurusamy (eds.), *Advances In Knowledge Discovery And Data Mining* (pp. 37-54). Menlo Park, CA: AAAI Press/The MIT Press.
- García, D. J. C. (2016). *Predicción en el dominio del tiempo: análisis de series temporales para ingenieros*. Retrieved from <https://ebookcentral.proquest.com>
- García, J., Molina, J.M. (2012) *Técnicas de análisis de datos aplicaciones prácticas utilizando Microsoft excell*. Recuperado de <http://ocw.uc3m.es/ingenieria-informatica/analisis-de-datos/libroDataMiningv5.pdf> .
- García, J. R. (2009). *Guía de cultivo de la colza*. Retrieved from <https://www.juntadeandalucia.es>
- Gironés, J., Casas, J., y Minguillón, J. (2017). *Minería de datos*. UOC.

Goddard, J., Gerardo, S., Pérez, B.R., Gutiérrez, M.A. (2000). un algoritmo para el entrenamiento de máquinas de vector soporte para regresión. *Revista de Matemática: Teoría y Aplicaciones* 7(1-2): 107–116 descargado de <https://core.ac.uk/download/pdf/67715344.Pdf>

Gómez-Arnau, J., (1988). *El cultivo del girasol*. Retrieved from <https://www.mapa.gob.es/es/>

Instituto Nacional de Estadística. (Spanish Statistical Office). (s. f.). Recuperado 1 de noviembre de 2018, de <http://www.ine.es/>

Lara, J. A. (2010). *Marco de descubrimiento de conocimiento para datos estructurados complejos con énfasis en el análisis de eventos en series temporales*. (Tesis doctoral). Universidad Politécnica de Madrid. España.

Lara J.A. (2014). *Business Intelligence*, CEF.

Lara J.A. (2014). *Minería de datos*, CEF.

León, J. de C. y. (s. f.). 2. *Estadísticas agrícolas 2016*. Recuperado 1 de noviembre de 2018, de https://agriculturaganaderia.jcyl.es/web/jcyl/AgriculturaGanaderia/es/Plantilla100/1284777102415/_/_/

López, M. J. J., & Zarza, G. (2017). *La ingeniería del big data: cómo trabajar con datos*. Retrieved from <https://ebookcentral.proquest.com>

Menne, M.J., I. Durre, B. Korzeniewski, S. McNeal, K. Thomas, X. Yin, S. Anthony, R. Ray, R.S. Vose, B.E. Gleason, and T.G. Houston, (2012). *Global Historical Climatology Network - Daily (GHCN-Daily)*, Version 3.x. NOAA National Climatic Data Center. <http://doi.org/10.7289/V5D21VHZ> access octubre de 2018.

Molina, M.E. (2017). *Modelo de descubrimiento de patrones en series temporales simbólicas*. (Tesis doctoral). Universidad Politécnica de Madrid. España.

Nadal, M. S., González, V. C.I., Román, D. B. & Córdoba, G. E. (2009). *Guías de cultivo*. Retrieved from <https://www.juntadeandalucia.es>.

- Nadal, M. S., Moreno, Y. M. T., & Cubero, S. J. I. (2004). *Las leguminosas grano en la agricultura moderna*. Retrieved from <https://ebookcentral.proquest.com>
- Osca, L. J. M. (2013). *Cultivos herbáceos extensivos: cereales*. Retrieved from <https://ebookcentral.proquest.com>
- Peña, D. (2010). *Análisis de series temporales*. Alianza editorial. Madrid
- Valencia, B. (2018). *Método para la búsqueda de patrones similares en series temporales simbólicas*. (Trabajo fin de máster). Escuela Politécnica de Madrid.

Referencias electrónicas

<http://ualmtorres.github.io/howtos/MongoDBJava/>. Consultado diciembre 2018.

<http://www.jfree.org/jfreechart/>. Consultado enero 2019.

<https://es.wikipedia.org>. Consultado enero 2019.

<https://hadoop.apache.org/>. Consultado noviembre 2018.

<https://netbeans.org>. Consultado noviembre 2018.

<https://online.visual-paradigm.com>. Consultado enero 2019.

<https://robomongo.org/>. Consultado noviembre 2018.

<https://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka>. Consultado enero 2019

<https://www.domo.com/learn/data-never-sleeps-6>. Consultado noviembre 2018.

<https://www.easymapmaker.com/map/>. Consultado noviembre 2018

<https://www.java.com/es/>. Consultado noviembre 2018.

<https://www.mongodb.com>. Consultado noviembre 2018.

<https://yuml.me/diagram/scruffy/usecase/draw>. Consultado enero 2019.

Anexo A. Código de la aplicación BigDataTFG.

Clase BigDataTFG

```
package bigdatatfg;

import java.awt.HeadlessException;
import java.io.File;
import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JProgressBar;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author javi
 */

// Interface gráfico para el proyecto BigDataTFG desde aquí se accede a todas las acciones
// que realiza el programa

public class BigDataTFG extends JFrame {

    private final JFileChooser selectorArchivoDescarga;// declaración selector de archivos
    private File archivo; // guarda archivo seleccionado
    static final String BASE_DATOS= "tfg"; // declaración de la base de datos mongoDB usada
    String colecion=""; // declaración colecciones usadas en mongoDB
    Object []datos=new Object[12]; // array guarda la predicción de los algoritmos timeseriesforecasting
    Object []mes={1,2,3,4,5,6,7,8,9,10,11,12}; // columna mes de la tabla con las predicciones
    DefaultTableModel modelo; // modelo de tabla con las predicciones
    /**
     * Creates new form BigDataTFG3
     */
    public BigDataTFG() {
```

```
initComponents();  
this.setLocationRelativeTo(null); // sitúa en el centro de la pantalla el JPanel  
VerColecciones.listaComboBox(jComboBox1); // presenta colecciones en jComboBox1  
selectorArchivoDescarga = new JFileChooser(); // crea selector de archivos  
// añade los jRadioButton a grupo de botones para solo poder seleccionar 1 simultaneamente  
// grupo de botones para graficar y crear archivo arff  
buttonGroup1.add(jRadioButton1);  
buttonGroup1.add(jRadioButton2);  
buttonGroup1.add(jRadioButton3);  
// añade los jRadioButton a grupo de botones para solo poder seleccionar 1 simultaneamente  
// grupo de métodos de predicción  
buttonGroup2.add(jRadioButton4);  
buttonGroup2.add(jRadioButton5);  
buttonGroup2.add(jRadioButton6);  
buttonGroup2.add(jRadioButton7);  
// tabla que presenta predicciones  
modelo=new DefaultTableModel(); // creación  
modelo.addColumn("mes",mes); // columna mes  
this.jTable1.setModel(modelo); // se hace visible  
}// fin de método constructor  
  
/**  
 * This method is called from within the constructor to initialize the form.  
 * WARNING: Do NOT modify this code. The content of this method is always  
 * regenerated by the Form Editor.  
 */  
@SuppressWarnings("unchecked") // crea componentes JFrame  
// código generado automáticamente por netbeans  
// <editor-fold defaultstate="collapsed" desc="Generated Code">  
private void initComponents() {  
  
    buttonGroup1 = new javax.swing.ButtonGroup();  
    jTextField3 = new javax.swing.JTextField();  
    buttonGroup2 = new javax.swing.ButtonGroup();  
    jLabel1 = new javax.swing.JLabel();  
    jTextField1 = new javax.swing.JTextField();  
    jButton1 = new javax.swing.JButton();  
    jProgressBar1 = new javax.swing.JProgressBar();  
    jButton2 = new javax.swing.JButton();  
    jProgressBar2 = new javax.swing.JProgressBar();  
    jButton3 = new javax.swing.JButton();
```

```
jButton4 = new javax.swing.JButton();
jTextField2 = new javax.swing.JTextField();
jButton5 = new javax.swing.JButton();
jComboBox1 = new javax.swing.JComboBox<>();
jLabel2 = new javax.swing.JLabel();
jRadioButton1 = new javax.swing.JRadioButton();
jRadioButton2 = new javax.swing.JRadioButton();
jRadioButton3 = new javax.swing.JRadioButton();
jButton6 = new javax.swing.JButton();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jScrollPane1 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jRadioButton4 = new javax.swing.JRadioButton();
jRadioButton5 = new javax.swing.JRadioButton();
jRadioButton6 = new javax.swing.JRadioButton();
jRadioButton7 = new javax.swing.JRadioButton();
jButton7 = new javax.swing.JButton();
jLabel5 = new javax.swing.JLabel();
jYearChooser3 = new com.toedter.calendar.JYearChooser();
jYearChooser4 = new com.toedter.calendar.JYearChooser();

jTextField3.setText("jTextField3");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("BigDataTFG-Fco Javier Moreno");

jLabel1.setText("Año fichero");

jButton1.setText("Descargar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jProgressBar1.setStringPainted(true);

jButton2.setText("Descomprimir");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
jButton2ActionPerformed(evt);
}
});

jProgressBar2.setStringPainted(true);

jButton3.setText("csv a mongoDB ");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton4.setText("Documentos guardados");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jTextField2.setText("Muestra número de documentos");

jButton5.setText("Filtrado & Refresco");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

jLabel2.setText("Colecciones en base de datos tfg");

jRadioButton1.setText("Limpiar tabla");
jRadioButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton1ActionPerformed(evt);
    }
});

jRadioButton2.setText("Generar gráfica");

jRadioButton3.setText("Generar fichero arff");
```

```
jButton6.setText("Enviar");
jButton6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton6ActionPerformed(evt);
    }
});

jLabel3.setText("Hasta...");

jLabel4.setText("Desde...");

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        new String [] {
            "Mes", "Predicción"
        }
    },
    new Class[] {
        java.lang.Integer.class, java.lang.Double.class
    };
    boolean[] canEdit = new boolean [] {
        false, false
    };
);

public Class getColumnClass(int columnIndex) {
    return types [columnIndex];
}

public boolean isCellEditable(int rowIndex, int columnIndex) {
    return canEdit [columnIndex];
}

jScrollPane1.setViewportView(jTable1);

jRadioButton4.setText("GaussianProcesses");
jRadioButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton4ActionPerformed(evt);
    }
});
```



```
.addComponent(jButton1)
.addComponent(jButton2)
.addGap(18, 18, 18))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
.addGroup(layout.createSequentialGroup()
.addComponent(jButton4)
.addGap(53, 53, 53)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
.addComponent(jButton3, javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 386,
Short.MAX_VALUE)
.addComponent(jButton5)
.addGap(62, 62, 62)))
.addGroup(layout.createSequentialGroup()
.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jButton5)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jLabel2)
.addComponent(jLabel3)
.addComponent(jRadioButton2)
.addComponent(jRadioButton1)
.addGroup(layout.createSequentialGroup()
.addComponent(jLabel3)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jYearChooser4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jButton6)
.addComponent(jButton3, javax.swing.GroupLayout.Alignment.TRAILING))
```

```
.addGroup(layout.createSequentialGroup()
    .addComponent(jLabel4)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jYearChooser3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE, 192,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGap(17, 17, 17)
    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 593,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jButton7)
            .addComponent(jButton6, javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jButton5, javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jButton4, javax.swing.GroupLayout.Alignment.LEADING))
        .addComponent(jButton7)))
    .addGap(37, 37, 37))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel1)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton1)
            .addComponent(jButton2)))
        .addGap(50, 50, 50)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(50, 50, 50)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
);
```

```

.addGap(144, 144, 144)
.addComponent(jButton3)
.addGap(47, 47, 47)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
.addComponent(jButton4)))
.addGroup(layout.createSequentialGroup()
.addGap(206, 206, 206)
.addComponent(jButton5)))
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addGap(8, 8, 8)
.addComponent(jLabel2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGap(74, 74, 74)
.addComponent(jRadioButton4)
.addGap(18, 18, 18)
.addComponent(jRadioButton5)
.addGap(18, 18, 18)
.addComponent(jRadioButton6)
.addGap(18, 18, 18)
.addComponent(jRadioButton7)
.addGap(35, 35, 35)
.addComponent(jButton7)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 155,
Short.MAX_VALUE)
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 232,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addGroup(layout.createSequentialGroup()
.addComponent(jRadioButton1)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jRadioButton2)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
.addComponent(jRadioButton3)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(jLabel5),           javax.swing.GroupLayout.PREFERRED_SIZE,      21,
        javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel4)
        .addComponent(jYearChooser3),      javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(14, 14, 14)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel3)
        .addComponent(jYearChooser4,      javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(28, 28, 28)
        .addComponent(jButton6)))
        .addGap(142, 142, 142)))
    );

    pack();
}// </editor-fold>

// Acciones al pulsar botón descargar ...
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
// Obtiene el posible nombre del archivo a descargar del jTextField1

String ficheroAno= jTextField1.getText(); // cadena con el año del fichero a descargar
if ("all_programador".equals(ficheroAno)) {// si se pone all_programador descarga los que pongo en bucle for
for(int ano=1986 ;ano <= 2019;ano++){ // código reservado para programador
String enteroString = Integer.toString(ano);
String urlDatos = "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/" + enteroString + ".csv.gz";
String posibleNombreArchivo = enteroString + ".csv.gz";
String ruta = "D:/tfg_descargas/" + enteroString + ".csv.gz";
DownLoadFichero descarga = new DownLoadFichero(urlDatos, ruta, this);
Thread thread = new Thread(descarga);
thread.start();
} //fin bucle for con los años a descargar esto no lo implementare
} //fin if de código implementado sólo para el programador
else{ // descarga el fichero del año que se pide

// http://noaa-ghcn-pds.s3.amazonaws.com/csv/1788.csv la otra ya no funciona
// http://noaa-ghcn-pds.s3.amazonaws.com/csv.gz/1788.csv.gz o esta

```

```
// https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/ por la huelga de la administración está cerrada

String urlDatos = "http://noaa-ghcn-pds.s3.amazonaws.com/csv.gz/" + jTextField1.getText() + ".csv.gz";
//String urlDatos = "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/" + jTextField1.getText() + ".csv.gz";
= "https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/" + jTextField1.getText() + ".csv.gz";

String nombreArchivo = jTextField1.getText() + ".csv.gz"; // renombra el fichero con el año + extensión
.csv.gz

// Muestra el diálogo para indicar el lugar donde se descargará el archivo
File archivoDescarga = new File(nombreArchivo); // crea fichero con nombre año.csv.gz
selectorArchivoDescarga.setSelectedFile(archivoDescarga); // selector ruta descarga
int opcion = selectorArchivoDescarga.showSaveDialog(this); //
// instancia de DownloadFichero que hace la descarga de la página web
if (opcion == JFileChooser.APPROVE_OPTION) {
    String ruta = selectorArchivoDescarga.getSelectedFile().getPath(); // ruta de descarga
    DownLoadFichero descarga = new DownLoadFichero(urlDatos, ruta, this); // descarga fichero de url a ruta
descarga
    Thread thread = new Thread(descarga); // hilo descarga
    thread.start();
} // fin if hilo de descarga
} // fin else que hace la descarga
// si el fichero no es válido nos da error de fichero no encontrado
}

// escuchador botón descomprimir, descomprime los ficheros descargados de la web
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
try {
// muestra el cuadro de diálogo de archivos, para que el usuario pueda elegir el archivo a abrir
JFileChooser selectorArchivos = new JFileChooser();
selectorArchivos.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
// indica cual fue la acción de usuario sobre el jfilechooser
int resultado = selectorArchivos.showOpenDialog(this);
archivo = selectorArchivos.getSelectedFile(); // obtiene el archivo seleccionado
// muestra error si es inválido
if ((archivo == null) || (archivo.getName().equals("")))) {
    JOptionPane.showMessageDialog(this, "Nombre de archivo inválido", "Nombre de archivo inválido",
JOptionPane.ERROR_MESSAGE);
} // fin de if
else{
// quita la extensión .gz al nuevo nombre del archivo quedando con extensión .csv
String nombreArchivoConExtension = archivo.getName();

```

```
String[] token = nombreArchivoConExtension.split(".gz");
int cantidadToken = token.length;
String nombreArchivoSinExtension = token[cantidadToken - 1];
String rutaDescarga = "D:/tfg/tfg_descomprimido/" + nombreArchivoSinExtension;

// comprueba extensión de fichero
int index = nombreArchivoConExtension.lastIndexOf('.');
String extension = nombreArchivoConExtension.substring(index + 1);
if(extension.equals("gz")){

// hilo que descomprime el archivo
Descompresor descomprime = new Descompresor(archivo, rutaDescarga, this); // descomprime archivo
Thread thread = new Thread(descomprime); // hilo descarga
thread.start();
// si no es .gx saca emergente
} else JOptionPane.showMessageDialog(this, "Nombre de archivo inválido", "Nombre de archivo inválido",
JOptionPane.ERROR_MESSAGE);
}

// catch para informar del error al descomprimir
} catch (HeadlessException ex) {
Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);
}// fin catch// fin catch// fin catch// fin catch

}

// importa fichero csv a mongoDB lo hace todo el archivo a la vez en vez de documento a documento
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
// muestra el cuadro de diálogo de archivos, para que el usuario pueda elegir el archivo a abrir
JFileChooser selectorArchivosCSV = new JFileChooser();
//para que muestre solo archivos de extensión csv
FileNameExtensionFilter filter = new FileNameExtensionFilter(".csv", "csv");
selectorArchivosCSV.setFileFilter(filter);
// indica cual fue la acción de usuario sobre el jfilechooser si es cancelar o error saca emergente
int resultado = selectorArchivosCSV.showOpenDialog(this);
if((resultado==1)|| (resultado==-1)) { // 1 =cancelar , -1 =error
// System.exit(0); // se cierra aplicación no se implementa
JOptionPane.showMessageDialog(this, "Debe seleccionar archivo csv"); // saca emergente con fallo
} else{
File archivoCSV = selectorArchivosCSV.getSelectedFile(); // obtiene el archivo seleccionado
```

```
// para crear colección con nombre del año ej 2000
String colecion1 =archivoCSV.getName();
String[] token = colecion1.split(".csv");
int cantidadToken = token.length;
colecion = token[cantidadToken - 1];
// comprueba extensión de fichero
int index = colecion1.lastIndexOf('.');
String extension=colecion1.substring(index + 1);
if(extension.equals("csv")){
// si el fichero tiene extensión .csv le insertará en base de datos tfg con comando mongoimport
// crea instancia de MongoDB
MongoDB a =new MongoDB(archivoCSV,BASE_DATOS,colecion,this);
} else // si no tiene extensión .csv saca emergente de error
JOptionPane.showMessageDialog(this,"Debe seleccionar archivo csv");// saca emergente con fallo
}//fin else

}

// nos muestra por pantalla el número de documentos que hay en la base de datos tfg colecion mongoimport
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
LeerMongo longitud=new LeerMongo(BASE_DATOS,colecion,this );
jTextField2.setText(longitud.getLongitud());// escribe el número de documentos de la colección guardados
}

// boton para filtrar de cada colecion por años- Crea una colección con los datos de estaciones
// de CyL y cambia el formato fecha a ISODate queda colecion con nombre aaaa_CyL a es año
// tiene tres comportamientos según se seleccione colección xxxx, colección aaaa_CyL o refrescar lista
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
// llama al metodo para poner las colecciones de base de datos tfg en el desplegable
String filtrarColecion = (String) jComboBox1.getSelectedItem(); //texto seleccion del combobox
if(filtrarColecion != "Refrescar lista"){ // si es distinto de refrescar puede ser
// aaaa o aaaa_CyL
String[] tokens =filtrarColecion.split("_");
String ano=tokens[0]; // lo saco para pasarlo a consulta
int cantidadTokens = tokens.length;
if(tokens.length<2){ // si es aaaa
// la colección seleccionada es aaaa por tanto hay que filtrar datos para enviarla
// a la colección aaaa_CyL
```

```

    ColecionFinal a =new ColecionFinal(filtrarColecion,this);
    try {
        // el campo fecha de cada documento se formates de int a Date
        FormatFecha b =new FormatFecha(filtrarColecion+"_CyL");

    } catch (ParseException ex) {
        Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);
    }
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    // en caso de seleccionar aaaa_CyL los documentos de la colección ya están filtrados
    // se llevan a la colección CyL donde están todas las colecciones de todos los años
    // con la media mensual. Estos documentos ya sirven para gráficas y predicciones
    }else{ String anoCyL=filtrarColecion;
    // se llama la método con aaaa_CyL y con aaaa
    Consulta.consulta(anoCyL,ano);
}
} else {
    jComboBox1.removeAllItems();// elimina todos item para no repetir las colecciones
    VerColecciones.listaComboBox(jComboBox1); //presenta colecciones en jcombox
}
}

//boton enviar de los 3 radio botons paramongo graficar, limpiar tabla y crear archivo arff
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
int yearIni;
int yearFin;
String mensaje="Revise los años"; // mensaje para emergente
yearIni=jYearChooser3.getValue(); //recoge año seleccionado por jYearChooser1
yearFin=jYearChooser4.getValue(); //recoge año seleccionado por jYearChooser2
if (jRadioButton1.isSelected()){ //jRadioButton1 de borrar columnas tabla (limpiar tabla)
    // mensaje="Ha seleccionado Quincenal";
}
}

```

Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.

```
modelo.setColumnCount(1); // borra las columnas de la tabla excepto la primera del mes
} else
if (jRadioButton2.isSelected()) { // jRadioButton2 de años a graficar

    if (yearIni <= yearFin) // comprueba que la fecha inicial es menor que la final
        ConsultaGraficar.consultaGraficar(yearIni, yearFin);
    else JOptionPane.showMessageDialog(this, mensaje); // saca emergente para revisar fechas
// botón para generar fichero arff con los datos de los años que se seleccione
} else
if (jRadioButton3.isSelected()) { // jRadioButton3 genera fichero arff
try {

    if (yearIni <= yearFin)
        FicheroArff.ficheroArff(yearIni, yearFin); // crear gráfica
    else JOptionPane.showMessageDialog(this, mensaje); // saca emergente para comprobar fechas
} catch (ParseException ex) {
    Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);
}
}

// jRadioButton1ActionPerformed del radioBotton limpiar tabla

private void jRadioButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

// jRadioButton4ActionPerformed del radioBotton predicción Gaussian Processes
private void jRadioButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

// botón para predecir por uno de los cuatro métodos, controla los cuatro radiobots
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
// muestra el cuadro de diálogo de archivos, para que el usuario pueda elegir el archivo a abrir
JFileChooser selectorArchivosArff = new JFileChooser();
selectorArchivosArff.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
// indica cual fue la acción de usuario sobre el jfilechooser
selectorArchivosArff.showOpenDialog(this);
archivo = selectorArchivosArff.getSelectedFile(); // obtiene el archivo seleccionado
```

```

String ruta =archivo.getName();           // path del archivo
// compruebo si es arff el archivo
String[] token = ruta.split("\\."); // separa archivo
int cantidadToken = token.length;
String extension = token[cantidadToken - 1];
if("arff".equals(extension)){
    String metodo;
    // GaussianProcesses
    if (jRadioButton4.isSelected()){
        metodo="GaussianProcesses";
        datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this);
    // ConsultaGraficar.consultaGraficar(datos); no funciona era para graficar la prevision
    // modelo.addColumn("mes",mes);
        modelo.addColumn(metodo, datos);
    }else
        //LinearRegression
        if (jRadioButton5.isSelected()){
            metodo="LinearRegression";
            datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this);
        // modelo.addColumn("mes",mes);
            modelo.addColumn(metodo, datos);
        }else // SMOreg
            if (jRadioButton6.isSelected()){
                metodo="SMOreg";
                datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this);
        // modelo.addColumn("mes",mes);
                modelo.addColumn(metodo, datos);
            }else //MultilayerPerceptron
                {metodo="MultilayerPerceptron";
                datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this);
        // modelo.addColumn("mes",mes);
                modelo.addColumn(metodo, datos);
            }
    }else
        JOptionPane.showMessageDialog(this,"El archivo no es -arff");// saca emergente con el error
}

// método get retorna barra de progreso para la clase DownLoadFichero
public JProgressBar getPgrAvance() {
    return jProgressBar1;
}

```

```
// método get retorna barra de progreso para la clase Descompresor
public JProgressBar getPgrAvance1() {
    return jProgressBar2;
}

/**
 * @param args the command line arguments
 * Método main
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException | InstantiationException | IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(BigDataTFG.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }

    // crea y visualiza JPanel BigDataTFG
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            new BigDataTFG().setVisible(true);
        }
    });
}
// fin método main

// Variables declaration - do not modify
```

```
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
public javax.swing.JComboBox<String> jComboBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JProgressBar jProgressBar1;
private javax.swing.JProgressBar jProgressBar2;
private javax.swing.JRadioButton jRadioButton1;
private javax.swing.JRadioButton jRadioButton2;
private javax.swing.JRadioButton jRadioButton3;
private javax.swing.JRadioButton jRadioButton4;
private javax.swing.JRadioButton jRadioButton5;
private javax.swing.JRadioButton jRadioButton6;
private javax.swing.JRadioButton jRadioButton7;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private com.toedter.calendar.JYearChooser jYearChooser3;
private com.toedter.calendar.JYearChooser jYearChooser4;
// End of variables declaration
}//fin clase
```

```
class ColecionFinal
```

```
package bigdatatfg;
```

```
import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import static com.mongodb.client.model.Filters.and;
import static com.mongodb.client.model.Filters.eq;
import static com.mongodb.client.model.Filters.in;
import static com.mongodb.client.model.Projections.excludeId;
import static com.mongodb.client.model.Projections.fields;
import static com.mongodb.client.model.Projections.include;
import java.util.ArrayList;
import javax.swing.JOptionPane;
import org.bson.Document;
```

```
/**
```

```
*
```

```
* @author javi
```

```
*/
```

```
// guarda en la colección aaaa_CyL los ficheros de estaciones de CyL con elemento valor_dato PRCP
```

```
// y fecha en formato Date
```

```
public class ColecionFinal {
```

```
// Estación de CyL
```

```
String estacionCod[] = {"SP000008202", "SPE00120422", "SPE00119837", "SPE00119846", "SPE00119909",
    "SPE00120224", "SPE00120233", "SPE00120386", "SPE00120485", "SPE00120494",
    "SPE00120521", "SPE00120593", "SPE00120602", "SPE00120620"};
```

```
private final BigDataTFG parent; // objeto
```

```
private final String filtrarColeccion; // colección climatología
```

```
ColecionFinal(String filtrarColeccion, BigDataTFG parent) {
```

```
this.filtrarColeccion = filtrarColeccion;
```

```
this.parent = parent;
```

```
// 1. conexión a mongoDB localhost
```

```
try { MongoClient mongoClient = new MongoClient();
```

```
// conexión a base de datos 'tfsg'
```

```
MongoDatabase database = mongoClient.getDatabase("tfg");
// Acceso a colección aaaa siendo las aes el año
MongoCollection<Document> collection = database.getCollection(filtrarColecion);
// campos ID_estacion , elemento valor_dato y fecha se guardan en colección aaaa_CyL
FindIterable it = collection.find(and(in("ID_estacion", estacionCod), eq("elemento", "PRCP")));
// elemento que se guardan son lo projection
projection(fields(include("fecha", "ID_estacion", "valor_dato"), excludeId()));
ArrayList<Document> docs = new ArrayList();
String colecionFinal=filtrarColecion+"_CyL";
it.into(docs);
MongoCollection<Document> collection1 = database.getCollection(colecionFinal);
docs.stream().map((doc) -> {
// inserta los documentos en la colección aaaa_CyL
collection1.insertOne((org.bson.Document) (Document) doc);
return doc;
}).forEachOrdered((doc) -> {
// System.out.println(doc); // comentario para que no lo imprima por pantalla
});
mongoClient.close(); // cerrar conexión
} catch (Exception ex) { // error
JOptionPane.showMessageDialog(parent, ex.getMessage(), "Error", consultar, "mongodb",
JOptionPane.ERROR_MESSAGE);
}
}
```

Clase Consulta

```
package bigdatatfg;
import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import org.bson.Document;
/**
 * @author javi
 */
// En esta clase a partir de la colección aaaa_CyL se calcula la media
//mensual de precipitaciones y se insertan documentos en la colección CyL
// se insertan doce documentos por colección (12 meses de cada año)
public class Consulta {
    static int ANO; // es el año de la colección
    int litros; // son los litros media mensual

    // el método constructor recibe la colección aaaa_CyL y aaaa
    public static void consulta(String anoCyL, String ano) {
        ANO = Integer.parseInt(ano);
        try {
            //para intentar actualizar a iSOJSON la fecha y no en Date
            //Conexión a mongoDb base de datos 'tfg' colección 'aaaa_CyL'
            MongoClient mongoClient = new MongoClient("localhost", 27017);
            MongoDatabase database = mongoClient.getDatabase("tfg");
            MongoCollection<Document> col = database.getCollection(anoCyL);
            // primero se define la projection con los campos que interesan
            Document project1 = new Document("valor_dato", true).append("ID_estacion", true).append("fecha",
                new Document("$dateFromString", new Document("dateString", "$fecha")));
            //projection
            Document project = new Document("$project", project1);
            // para la agrupación por meses
            Document groupFields = new Document("_id", new Document("$month", "$fecha")).append("Documentos",
                new Document("$sum", 1)).append("litros", new Document("$sum", "$valor_dato"))
                .append("media", new Document("$avg", "$valor_dato"));
            //group
            Document group = new Document("$group", groupFields);
```

```
//sort para ordenar
Document sort = new Document("$sort", new Document("_id",1));
// tubería de los tres projection, group y sort
List<Document> pipeline = Arrays.asList(project,group,sort);
// objeto iterator
MongoCursor<Document> cur = col.aggregate(pipeline).iterator();
// colección donde irán los nuevos documentos: 'CyL'
MongoCollection<Document> colCyL = database.getCollection("CyL");
// bucle que calcula para cada documento la media mensual de litros
// multiplicando por los días (30) y dividendo por 10 porque son decilitros
// por tanto multiplica por tres
// luego cada documento es insertado en la nueva colección 'CyL'
while (cur.hasNext()) {
    Document doc = cur.next();
    List list = new ArrayList(doc.values());
    /* System.out.print(list.get(0)); // se comentan para no imprimir
    System.out.print(": ");
    System.out.print(list.get(1));
    System.out.print(": ");
    System.out.print(list.get(2));
    System.out.print(": ");
    System.out.println(list.get(3));
    */
    /*double litros= (double) list.get(3);
    if(litros!=0){
        litros=(double) list.get(3);
        litros=litros*3;
    }
    */
    // Crea los documentos que se van a insertar
    Document document = new Document("Año",ANO).
        append("Mes",list.get(0)).
        append("Litros",litros);
    // Inserta los documentos en la colección 'CyL'
    colCyL.insertOne(document);
}mongoClient.close();
}//fin try
}//fin método consulta()
}//fin clase Consulta
```

Clase ConsultaGraficar

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import java.awt.Font;
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;
import org.bson.conversions.Bson;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
/* * @author javi

// clase para dibujar la gráfica de los años seleccionados
public class ConsultaGraficar{
    int litros;
    // método constructor recibe año inicial y año final para dibujar gráfica
    public static void consultaGraficar(int yearIni,int yearFin) {
        Bson filtro = new Document("$gte", yearIni).append("$lte", yearFin); //filtro de búsqueda
        try ( //para intentar actualizar a iSOJSON la fecha y no en Date
            //this.colecion = colecion;
            MongoClient mongoClient = new MongoClient("localhost", 27017)) {
            MongoDB database = mongoClient.getDatabase("tfgr");
            MongoCollection<Document> col = database.getCollection("CyL");
            // Crea documento con criterio de búsqueda
            Document findDocument = new Document("Año", filtro);
            // Document to store query results
            MongoCursor<Document> cur= col.find(findDocument).iterator();
            //crea un arreglo que guarde los datos
            DefaultCategoryDataset dataset = new DefaultCategoryDataset();
            // Iterate over the results printing each document
            while (cur.hasNext()) {
```

```
//l hago de esa forma a ver
    Document doc = cur.next();
    List list = new ArrayList(doc.values());
    String ano=Integer.toString ((int) list.get(1));
    /* System.out.println(list.get(1)); no imprimir
    System.out.println(list.get(2));
    System.out.println(list.get(3));
    */
    Double litros=(Double) list.get(3);
//agregar datos .addValue(valor,"Concepto","Categoria")...
    dataset.addValue(litros,"mm",ano+" - "+(Comparable) list.get(2));
}
// crear gráfica...
JFreeChart chart = ChartFactory.createLineChart( //gráfica de lineas
"Precipitaciones medias en Castilla y León", // Titulo
"Mes", // Etiqueta de datos
"mm", // Etiqueta de valores
dataset, // Datos
PlotOrientation.VERTICAL, // orientacion
false, // Incluye leyenda
true, // Incluye tooltips
false // urls
);
// chart Customisation
CategoryPlot plot = (CategoryPlot) chart.getPlot();
//Indicar la etiqueta vertical posición y fuentes
final CategoryAxis domainAxis = plot.getDomainAxis();
domainAxis.setTickLabelFont(new Font("Arial", Font.BOLD, 11));
domainAxis.setCategoryLabelPositions(CategoryLabelPositions.UP_90);
//para mostrar gráfica
ChartFrame frame = new ChartFrame("TFG", chart);
frame.pack();
frame.setVisible(true);
mongoClient.close();// cerrar conexión mongoDB
}
}//fin método consultaGraficar()
}//fin clase
```

Clase Descompresor

```
package bigdatatfg;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.GZIPInputStream;
import javax.swing.JOptionPane;

/**
 *
 * @author javi
 */

// clase para descomprimir archivos .gz
public class Descompresor implements Runnable {
    private final File archivo ;
    private final String rutaDescarga; // ruta para descargar fichero
    private final BigDataTFG parent; // objeto
    // constructor recibe el archivo a descomprimir, la ruta de descarga y objeto BigDataTFG
    public Descompresor(File archivo, String rutaDescarga, BigDataTFG parent) {
        this.archivo = archivo;
        this.rutaDescarga = rutaDescarga;
        this.parent = parent;
    }
    // hilo de descomprimir
    @Override
    public void run() {
        try {
            //se crea instancia descomprimir fichero .gz
            try (GZIPInputStream gin = new GZIPInputStream(new FileInputStream(archivo))) {
                int length = (int) archivo.length() * 8; // ajusto la barra de progreso
                parent.getPgrAvance1().setString(""); // barra de progreso
                parent.getPgrAvance1().setMinimum(0);
                parent.getPgrAvance1().setMaximum(length);
                parent.getPgrAvance1().setValue(0);
            }
            //fichero descomprimido
            try (FileOutputStream fos = new FileOutputStream(rutaDescarga)) {
                byte[] buf = new byte[1024]; // buffer de descompresión
                int count;
                while ((count = gin.read(buf)) > 0) {
                    fos.write(buf, 0, count);
                }
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "Error al descomprimir el archivo");
        }
    }
}
```

```
int len;
int current = 0;
while ((len = gin.read(buf)) > 0) { // bucle descomprime fichero
    fos.write(buf, 0, len);
    parent.getPgrAvance1().setValue(current); // barra de progreso de descompresión de fichero
    current = current + len;
}
parent.getPgrAvance1().setValue(length); // barra de progreso
parent.getPgrAvance1().setString("archivo descomprimido");
} // fin buffer de descompresión
} // fin ajusto la barra de progreso
} catch (IOException ex) {
    Logger.getLogger(Descompresor.class.getName()).log(Level.SEVERE, null, ex);
    JOptionPane.showMessageDialog(parent, ex.getMessage(), "Error sin formato gz",
JOptionPane.ERROR_MESSAGE);
}
} // hilo descomprimir
}//fin clase
```

Clase DownLoadFichero

```
package bigdatatfg;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
import javax.swing.JOptionPane;

/**
 *
 * @author javi
 */

//clase de descarga de ficheros de la web
//http://noaa-ghcn-pds.s3.amazonaws.com/csv/aaaa.csv ha cambiado por la huelga administración americana
//https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/xxxx
public class DownLoadFichero implements Runnable {
    private final String direccion; // url
    private final String rutaDescarga; // ruta para descargar fichero
    private final BigDataTFG parent; // objeto

    // método constructor recibe url, path descarga y objeto BigDataTFG
    public DownLoadFichero(String direccion, String rutaDescarga, BigDataTFG parent) { //constructor
        this.direccion = direccion;
        this.rutaDescarga = rutaDescarga;
        this.parent = parent;
    }

    @Override //hilo de descarga
    // hilo
    public void run() {
        try {
            URL url = new URL(direccion); // url de descarga
            // establece conexión
            URLConnection conexion = url.openConnection();
            FileOutputStream fichero;
            try (InputStream stream = conexion.getInputStream()) {
                //para crear la barra de progreso de la descarga
                int length = conexion.getContentLength();
```

```
parent.getPgrAvance().setString("");
parent.getPgrAvance().setMinimum(0);
parent.getPgrAvance().setMaximum(length);
parent.getPgrAvance().setValue(0);
fichero = new FileOutputStream(rutaDescarga);
// Lectura del fichero de la web y escritura en fichero local
byte[] buffer = new byte[1024]; // buffer temporal de lectura.
int leido = stream.read(buffer);
int current = 0;
while (leido > 0) {           // descargando fichero
    fichero.write(buffer, 0, leido);
    leido = stream.read(buffer);
    parent.getPgrAvance().setValue(current); // barra de progreso
    current = current+leido;
} parent.getPgrAvance().setValue(length); // barra de progreso
parent.getPgrAvance().setString("Descarga completa");
// cierre de conexion y fichero.
fichero.close(); // cerrar fichero
} catch (IOException ex) { // error devuelve url que ha fallado
JOptionPane.showMessageDialog(parent, ex.getMessage(), "Error fichero no encontrado",
JOptionPane.ERROR_MESSAGE);
}
}//fin hilo
}//fin clase
```

Clase FicheroArff

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import java.text.ParseException;
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;
import org.bson.conversions.Bson;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author javi
 */
// clase que genera fichero .arff con los datos de los años seleccionados
public class FicheroArff{

    // método constructor recibe año inicial y año final para generar fichero
    public static void ficheroArff(int anoIni,int anoFin) throws ParseException{
        FileWriter escribir=null;
        try {
            //cabecera del fichero arff
            String saludo="@relation tfg\n";
            String atributo1="@attribute Date date \'yyyy-MM-dd\'\n";
            String atributo2="@attribute litros numeric\n";
            String dato="@data\n";
            try // conexión a mongoDB, base de datos y colección
                (MongoClient mongoClient = new MongoClient("localhost", 27017)) {
                    MongoDatabase database = mongoClient.getDatabase("tfg");
                    MongoCollection<Document> col = database.getCollection("CyL");
                    //Crear un objeto File clima.arff
                    File archivo=new File("D:/tfg/clima.arff");

```

```
//Crear objeto FileWriter que sera el que nos ayude a escribir sobre archivo
escribir = new FileWriter(archivo,false); // false sobreescribe y true añade
//Escribe en el archivo con el metodo write
escribir.write(saludo+atributo1+atributo2+dato);
//filtro de busqueda de documentos de los años comprendidos entre ambos pedidos
Bson filtro = new Document("$gte", anoIni).append("$lte", anoFin);
// Crea documento con criterio de busqueda
Document findDocument = new Document("Año", filtro);
MongoCursor<Document> cur = col.find(findDocument).iterator();
{
    // va escribiendo los datos leidos de colección 'CyL'
    while (cur.hasNext()) {
        Document doc = cur.next();
        List list = new ArrayList(doc.values());
        escribir.write(list.get(1)+"-");
        escribir.write(list.get(2)+"-");
        escribir.write("01,");
        escribir.write(list.get(3)+"\n");
    } //fin bucle
    escribir.close(); //Cierra fichero
} mongoClient.close(); //Cierra la conexion
}
} catch (IOException ex) {
    Logger.getLogger(FicheroArff.class.getName()).log(Level.SEVERE, null, ex);
}
} //fin método
} //fin clase
```

Clase FormatFecha

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import org.bson.Document;

// clase para descarga todos los documentos de la colecion les modifica formato fecha
// de int a ISODate y los actualiza en la colecion
public final class FormatFecha{
    private final String colecion; //colecion climatologia

    //método constructor recibe la colección
    FormatFecha(String colecion) throws ParseException{
        this.colecion = colecion;

        try (MongoClient mongoClient = new MongoClient("localhost", 27017)) {
            MongoDatabase database = mongoClient.getDatabase("tfsg");
            MongoCollection<Document> col = database.getCollection(colecion);
            //recorre todos los documentos para modificar fecha
            try (MongoCursor<Document> cur = col.find().iterator()) {
                while (cur.hasNext()) {
                    Document doc = cur.next();
                    List list = new ArrayList(doc.values());
                    /* System.out.print(list.get(1));// no imprimir
                    System.out.print(": ");
                    System.out.println(list.get(2));
                    */
                    int x=(int) list.get(2);
                    // va guardando todos los documentos con la fecha en formato ISODATE
                    col.updateOne(doc, new Document("$set", new Document("fecha", FechaISO(x))));
                }
            }mongoClient.close();
        }
    }
}
```

```
}

//convierte fecha de int a ISODate
final String FechaISO(int fecha) throws ParseException {
String fechaString = Integer.toString(fecha);
String dateSt=fechaString;
Date traduceDate = new SimpleDateFormat("yyyyMMdd", Locale.ENGLISH).parse(dateSt);
String dateISO = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.mmm'Z'", Locale.ENGLISH).format(traduceDate);
System.out.print(dateISO);
return dateISO;
}//fin método FechaIso()
}//fin clase
```

Clase LeerMongo

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import javax.swing.JOptionPane;
import org.bson.Document;

/**
 *
 * @author javi
 */

// clase para leer e imprimir los documentos de la base de datos 'tfg'
public class LeerMongo {

    private final String baseDatos; // base de datos de mongoDB tfg
    private final String colecion; //colección climatología
    private final BigDataTFG parent; // objeto
    String documento;

    // método constructor recibe nombre base de datos , colección y objeto BigDataTFG
    public LeerMongo(String baseDatos, String colecion, BigDataTFG parent) { // muestra cantidad de documentos
        guardados

        this.parent = parent;
        this.baseDatos = baseDatos;
        this.colecion = colecion;

        //Conexión a la base de datos
        try { // PASO 1: Conexión al Server de MongoDB Pasandole el host y el puerto
            MongoClient mongoClient = new MongoClient("localhost", 27017);
            // PASO 2: Conexión a la base de datos
            MongoDatabase db = mongoClient.getDatabase(baseDatos);
            // PASO 3: se obtiene una colección para trabajar con ella
            MongoCollection<Document> collection = db.getCollection(colecion);
            long contadorDocumento = collection.countDocuments(); // cuenta documentos
            System.out.println("Documentos totales: " + contadorDocumento);
            documento = Long.toString(contadorDocumento); // convierto long a String
            mongoClient.close(); // cierre de base de datos
        } catch (Exception ex) { // error devuelve si no hay conexión a base de dstos
    }
}
```

```
JOptionPane.showMessageDialog(parent, ex.getMessage(),  
        "Error no estás conectado a mongodb", JOptionPane.ERROR_MESSAGE);  
    }  
  
}  
  
//método devuelve el número de documentos insertados hasta el momento  
public String getLongitud(){  
    return documento;  
}//fin método get  
//fin de clase
```

Clase MongoDB

```
package bigdatatfg;

import java.io.File;
import java.io.IOException;
import javax.swing.JOptionPane;
/***
 * @author javi
 */
// guarda ficheros csv en la base de datos tfg colecion aaaa siendo las aes el año
public class MongoDB{
    BigDataTFG parent;
    private File archivoCSV;//fichero csv con datos de climatologia de un año
    private String baseDatos; // base de datos de mongoDB tfg
    private String colecion; //colecion climatologia
    Process p = null;

    // constructor recibe el fichero csv, la colección y objeto BigDataTFG
    MongoDB(File archivoCSV, String baseDatos, String colecion, BigDataTFG parent) {
        this.archivoCSV = archivoCSV;
        this.baseDatos = baseDatos;
        this.colecion = colecion;
        this.parent = parent;
        // comando mongoimport que importa un fichero csv a mongoDB
        String command = "mongoimport --db " + baseDatos + " --collection " + colecion + " --type csv --file "
                + archivoCSV + " --fields ID_estacion,fecha,elemento,valor_dato,M_flag,q_flag,s_flag,observ_time";
        try { // ruta de mongo mas comando mongoimport
            p = Runtime.getRuntime().exec("C:\\Program Files\\MongoDB\\Server\\4.0\\bin\\" + command);
            // System.out.println("Reading csv into Database");
            // System.out.println(command);
        } catch (IOException e) { // no salen estos errores
            System.out.println("EXCEPTION: " + e.getMessage());
            JOptionPane.showMessageDialog(parent, e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    }catch (Exception ex) { // error devuelve url que ha fallado
        JOptionPane.showMessageDialog(parent, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
} // fin método constructor
}//fin clase
```

Clase TimeSeriesTFG

```
package bigdatatfg;

import java.io.*;
import java.util.List;
import javax.swing.JOptionPane;
import weka.core.Instances;
import weka.classifiers.functions.GaussianProcesses;
import weka.classifiers.evaluation.NumericPrediction;
import weka.classifiers.functions.LinearRegression;
import weka.classifiers.functions.MultilayerPerceptron;
import weka.classifiers.functions.SMOreg;
import weka.classifiers.timeseries.WekaForecaster;

//clase para aplicar algoritmos de predicción
public class TimeSeriesTFG {
    static Object [] datos = new Object[12];// guarda predicción
    private BigDataTFG parent; // objeto
    // método constructor recibe el método de algoritmo a aplicar y el fichero arff
    public static Object[] timeSeriesTFG(String metodo, File archivo, BigDataTFG parent) {
        try {
            // path a clima.arffto
            File pathToWineData = archivo;
            // carga datos
            Instances clima = new Instances(new BufferedReader(new FileReader(pathToWineData)));
            // nuevo forecaster
            WekaForecaster forecaster = new WekaForecaster();
            // set the targets we want to forecast. This method calls
            // setFieldsToLag() on the lag maker object for us
            forecaster.setFieldsToForecast("litros");
            // gaussian processes for regression instead
            if(metodo=="GaussianProcesses") {
                forecaster.setBaseForecaster(new GaussianProcesses());
            }else
            //LinearRegression
            if(metodo=="LinearRegression"){
                forecaster.setBaseForecaster(new LinearRegression());
            }else
            //SMOreg
            if(metodo=="SMOreg"){


```

```

forecaster.setBaseForecaster(new SMOreg());
}else
//MultilayerPerceptron());
forecaster.setBaseForecaster(new MultilayerPerceptron());

forecaster.getTSLagMaker().setTimeStampField("Date"); // date time stamp
forecaster.getTSLagMaker().setMinLag(1); //longitud mínima de retraso
forecaster.getTSLagMaker().setMaxLag(12); //longitud máxima de retraso
// añade el campo mes del año
forecaster.getTSLagMaker().setAddMonthOfYear(true);
// añade el trimestre del año
forecaster.getTSLagMaker().setAddQuarterOfYear(true);
// construye el modelo
forecaster.buildForecaster(clima, System.out);
// prime the forecaster with enough recent historical data
// to cover up to the maximum lag. In our case, we could just supply
// the 12 most recent historical instances, as this covers our maximum
// lag period
forecaster.primeForecaster(clima);
//forecast para 12 unidades (meses) más allá de datos de entrenamiento
List<List<NumericPrediction>> forecast = forecaster.forecast(12, System.out);
// salida de la predicción
for (int i = 0; i < 12; i++) {
    List<NumericPrediction> predsAtStep = forecast.get(i);
    for (int j = 0; j < 1; j++) { // solo 1 atributo
        NumericPrediction predForTarget = predsAtStep.get(j);
        // System.out.print("'" + predForTarget.predicted() + " ");
        datos [i] =predForTarget.predicted();
        } //fin bucle
    } //fin bucle
} catch (Exception ex) {
// ex.printStackTrace();
// si error en predicción
 JOptionPane.showMessageDialog(parent, ex.getMessage(), "error al hacer predicción: ", JOptionPane.ERROR_MESSAGE);
    for (int i = 0; i < 12; i++) { //pone error en la tabla
        datos [i] ="error";
    }
}return datos;
} //fin método
}//fin clase

```

Clase VerColecciones

```
package bigdatatfg;

import com.mongodb.DB;
import com.mongodb.MongoClient;
import java.util.ArrayList;
import java.util.Set;
import java.util.TreeSet;
import javax.swing.JComboBox;

/**
 *
 * @author javi
 */

// clase para presentarnos en desplegable del combobox las colecciones en la base de datos
// tfg y seleccionar la que se va a hacer el filtrado de las estaciones de CyL
public class VerColecciones {
    // guardo las colecciones de la base de datos 'tfg'
    ArrayList<String> colecionCombo;

    //mostrar las colecciones en combobox recibe el jComboBox1
    public static void listaComboBox(JComboBox jComboBox1) {
        ArrayList<String> colecionCombo = new ArrayList<>();
        // PASO 2: Conexión a la base de datos 'tfg'
        try (// PASO 1: Conexión al Server de MongoDB Pasandole el host y el puerto
            MongoClient mongoClient = new MongoClient("localhost", 27017)) {
            // PASO 2: Conexión a la base de datos 'tfg'
            DB db = mongoClient.getDB("tfg");
            //nombres de todas colecciones guardadas en 'tfg'
            Set coltfg = db.getCollectionNames();
            Set<String> colecciones= new TreeSet<>(coltfg);
            colecciones.stream().map((ss) -> {
                //System.out.println(ss);
                return ss;
            });
            // guardo en array las colecciones de base de datos tfg para usar en combo box
            }).filter((ss) -> (!"system.js".equals(ss))).forEachOrdered((ss) -> {
                colecionCombo.add(ss);
            });
            // aparece en primer lugar y al pulsar filtrar refresca lista
            jComboBox1.addItem("Refrescar lista");
        }
    }
```

```
int x=colecionCombo.size();
//va añadiendo todas las colecciones en jComboBox1
for(int y=0;y<x;y++){
    jComboBox1.addItem(colecionCombo.get(y));
}
}//fin método
}//fin clase
```

Anexo B. Cálculo de precipitaciones históricas 2003-2018

Para realizar la gráfica con las lluvias en el intervalo de tiempo 2003-2018 en la comunidad de Castilla y León se han realizado consultas a la base de datos ‘tfg’ de mongoDB utilizando javascript y desde la aplicación ROBO 3T.

Se muestra para el año 2003, donde se hace una consulta sobre la colección 2003 que tiene los datos de todas las estaciones a nivel mundial y con los documentos tal y como se han importado a mongoDB como se muestra a continuación.

```
{  
    "_id" : ObjectId("5c4ca7b55055ab1330b3223b"),  
    "ID_estacion" : "ASN00040209",  
    "fecha" : 20030101,  
    "elemento" : "PRCP",  
    "valor_dato" : 0,  
    "M_flag" : "",  
    "q_flag" : "",  
    "s_flag" : "a",  
    "observ_time" : ""  
}
```

Para hacer esta consulta primero se ha convertido el formato de fecha que es de tipo ‘int’ a ‘ISODate’ y posteriormente usando la agregación se comienza con ‘match’ para filtrar las estaciones de Castilla y León, posteriormente se agrupa ‘group’ por año, sumando los documentos y calculando la media de decilitros caídos, finalmente se ordena ‘sort’.

```
fechaToString = {  
    $addFields: {  
        fechaString: { $toString: "$fecha" }  
    }  
};  
fechaToDate={
```

```

$project: {
    fechaString: {
        $dateFromString: {
            dateString: '$fechaString',
        }
    },
    valor_dato : "$valor_dato",
    elemento : "$elemento"
}
}

resultado=db.getCollection('2003').aggregate([
{$match:{ $or: [ {ID_estacion: "SP000008202"},{ID_estacion: "SPE00120422"},{ID_estacion: "SPE00119837"},{ID_estacion: "SPE00119846"},{ID_estacion: "SPE00119909"},{ID_estacion: "SPE00120224"},{ID_estacion: "SPE00120233"},{ID_estacion: "SPE00120386"},{ID_estacion: "SPE00120485"},{ID_estacion: "SPE00120494"},{ID_estacion: "SPE00120521"},{ID_estacion: "SPE00120593"},{ID_estacion: "SPE00120602"},{ID_estacion: "SPE00120620"} ]}),
{$match:{ $or: [ {elemento: "PRCP"}] }},
fechaToString.fechaToDate,
{
    $group: { _id: { $year: "$fechaString"}, // $week por semanas y $month lo agrupa por meses
        documentos: { $sum: 1 },
        litros:{$sum:'$valor_dato' },
        media:{$avg:'$valor_dato'}
    }
},
{$sort:{_id:1}}
])
db.getCollection('nueva2').insertMany(resultado.toArray())

```

Se obtiene el siguiente documento con las precipitaciones diarias durante el año 2003:

```
{
    "_id" : 2003,
    "documentos" : 8018.0,
    "litros" : 122426,
    "media" : 15.2688949862809
}
```

A partir de este resultado se calcula la media anual y se divide por 10 para pasar el resultado a litros o mm.

$$\frac{15,2689 \times 365}{10} = 557,31 \text{ litros}$$

El cálculo para los siguientes años es idéntico:

Tabla B.1. Precipitaciones medias anuales Castilla y León y previsión 2019

Año	Media anual litros
2003	557.31
2004	381.97
2005	340.87
2006	490.77
2007	456.94
2008	515.44
2009	420.38
2010	550.58
2011	378.05
2012	373.04
2013	531.96
2014	483.14
2015	351.04
2016	511.92
2017	312.64
2018	588.75
2019	390.52

A partir de todos los datos almacenados en la base de datos se ha recopilado información de las precipitaciones medias de los últimos años en la región. En la figura B.1. se puede ver la gráfica de barras desde el año 2003 hasta el 2018 y la predicción utilizando el algoritmo de máquina de vectores soporte para la regresión (MVS).

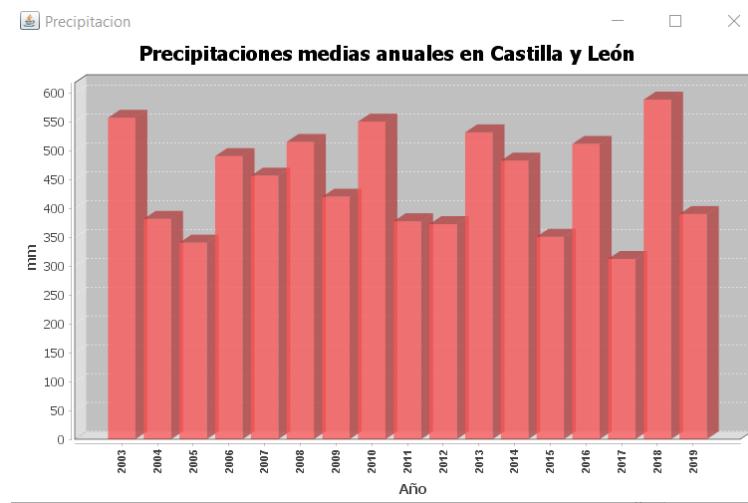


Figura B.1. Gráfica precipitaciones medias en CyL entre 2003 y 2018 más previsión 2019.

Por otra parte la figura B.2. obtenida de un estudio realizado por la junta de Castilla y León analiza periodos con características necesarias para ser considerados como de sequía. El Porcentaje de Precipitación Normal (PPN) es la relación entre precipitación acumulada en un periodo de tiempo y la precipitación media anual para una región. La precipitación media anual histórica se obtiene a partir del promedio de las precipitaciones anuales en 30 años (mínimo). (Situación normal: 100%). De la observación de este gráfico se ven los años considerados secos entre el año 2000 y 2015 situándose por debajo del nivel 100.

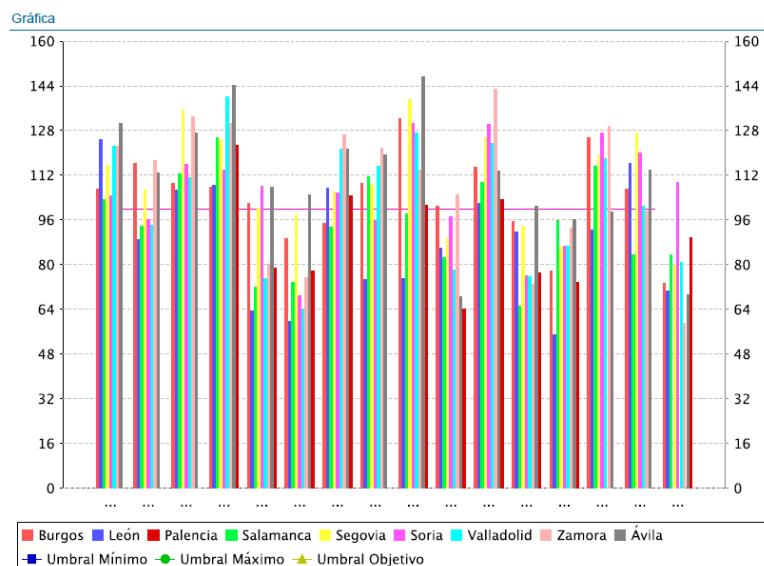


Figura B.2. Recoge los años del 2000 al 2015 el porcentaje de precipitación normal (PPN).

Anexo C. Datos de rendimiento agrícola.

La Consejería de Agricultura y Ganadería de la Junta de Castilla y León elabora el Anuario de Estadística Agraria de Castilla y León, que recoge tanto los resultados de las operaciones estadísticas incluidas en el área de agricultura y ganadería del Plan Estadístico de Castilla y León 2014-2017, como las que forman parte del Plan Estadístico Nacional.

Todos los datos de este anexo son extraídos de las tablas publicadas en dicho Anuario de estadística agraria de Castilla y León 2016 donde se muestra la superficie cultivada, la producción y el rendimiento de los principales cultivos de la región. A partir de dichas tablas se han elaborado las gráficas donde se representa el rendimiento por hectárea de los principales cultivos de secano de la región.

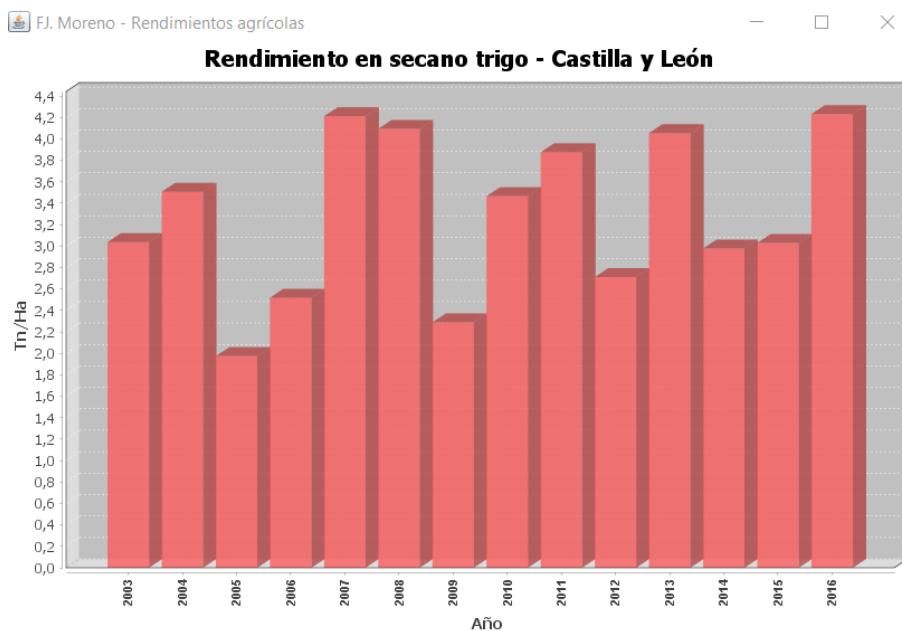


Figura C.1. Rendimiento en secano de trigo en Castilla y León.

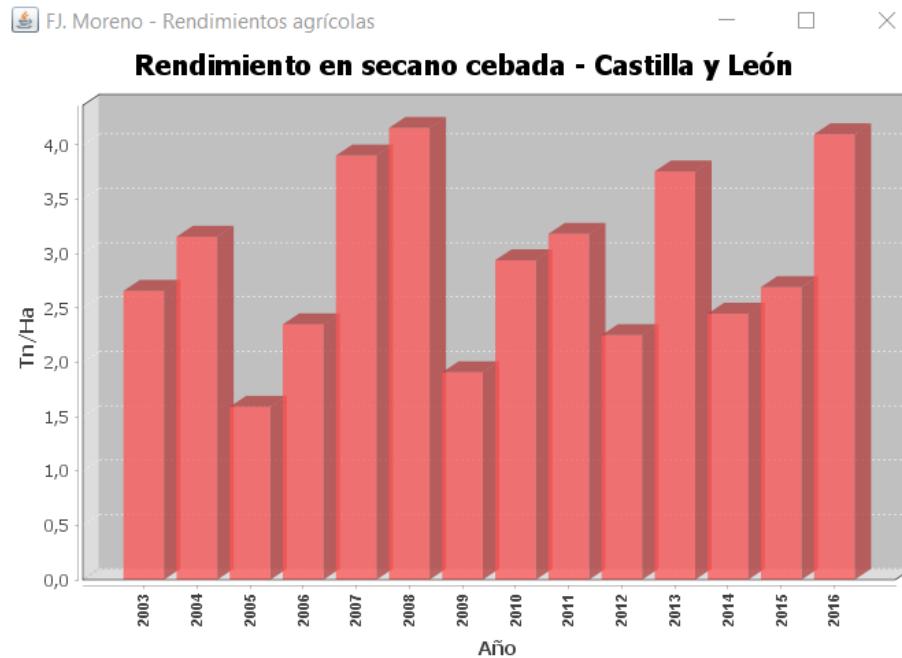


Figura C.2. Rendimiento en secano de cebada en Castilla y León.

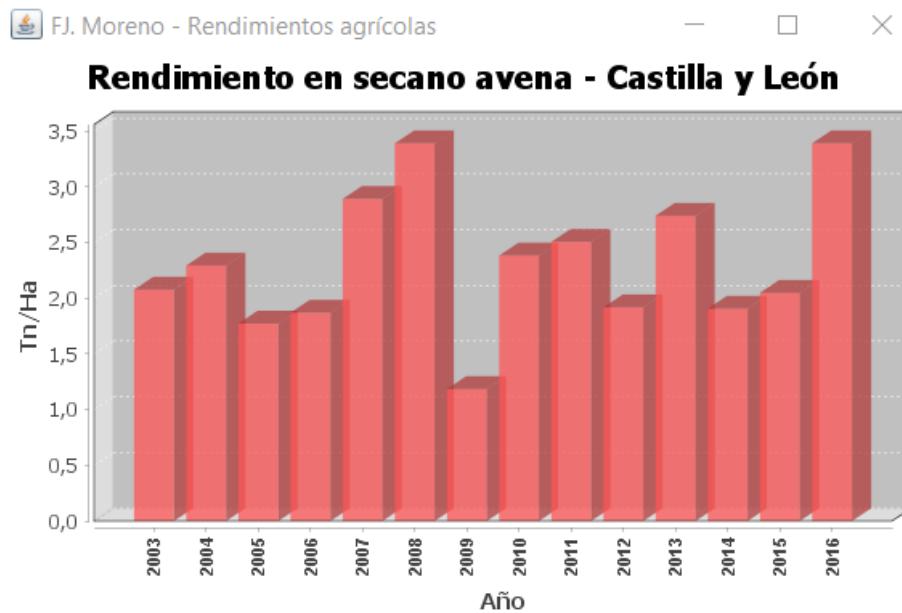


Figura C.3. Rendimiento en secano de avena en Castilla y León.

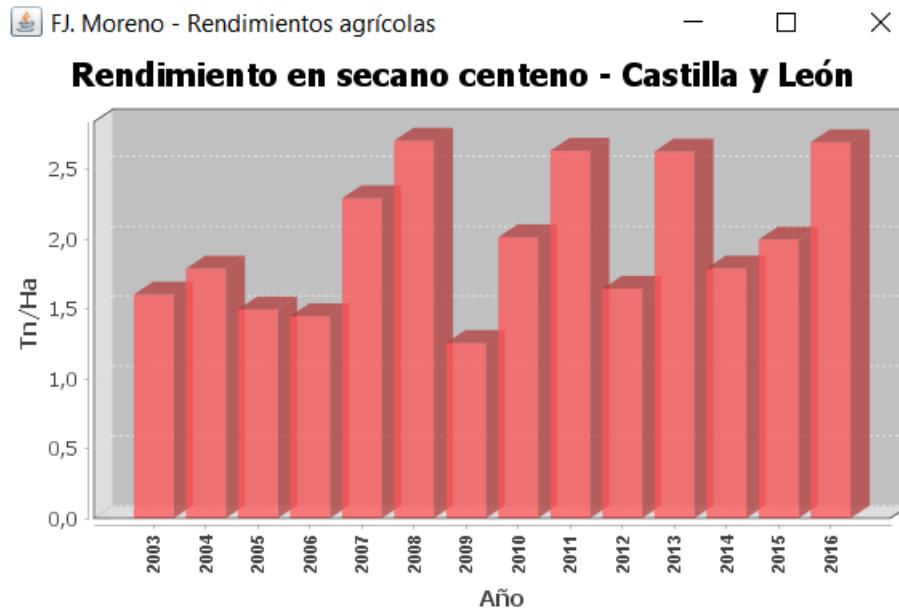


Figura C.4. Rendimiento en secano de centeno en Castilla y León.

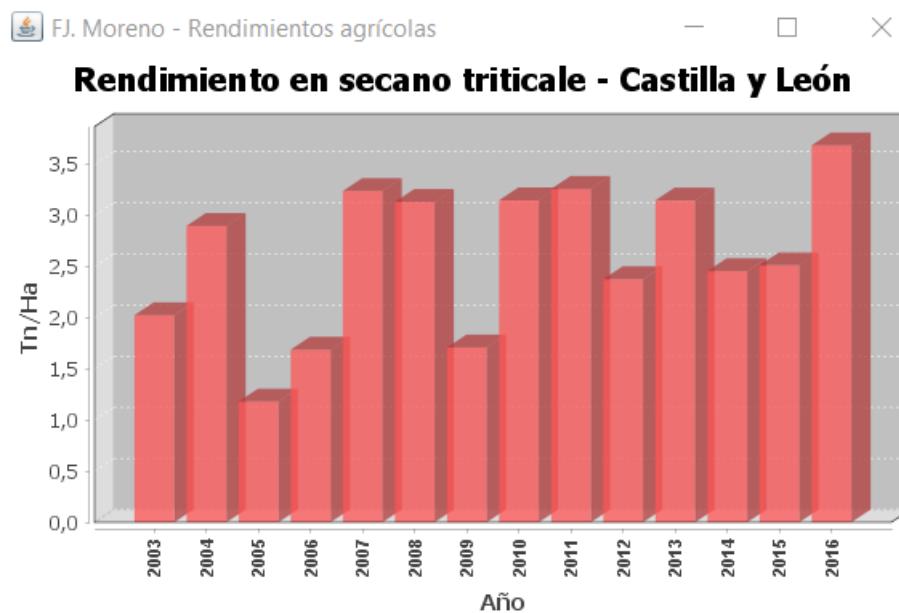


Figura C.5. Rendimiento en secano de triticale en Castilla y León.

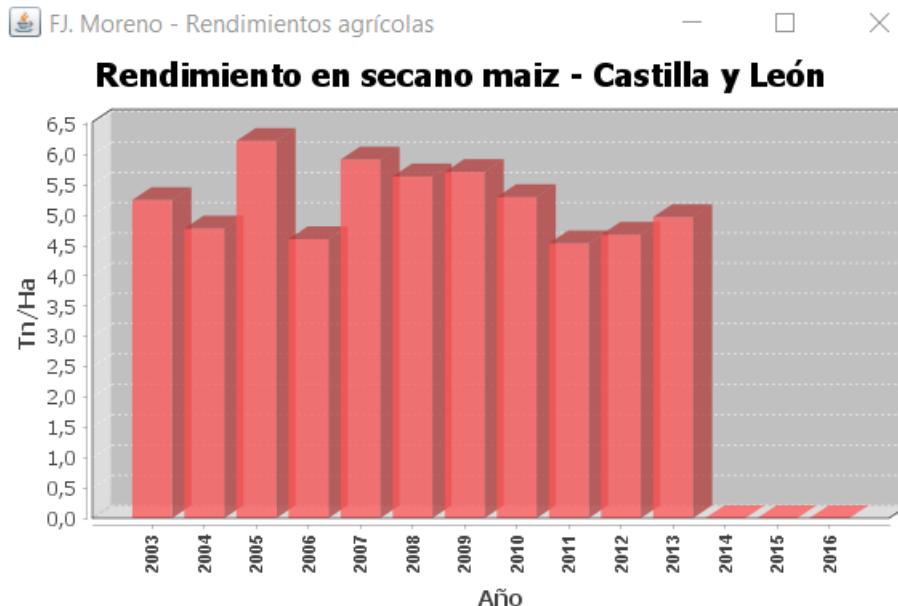


Figura C.6. Rendimiento en secano de maíz en Castilla y León.

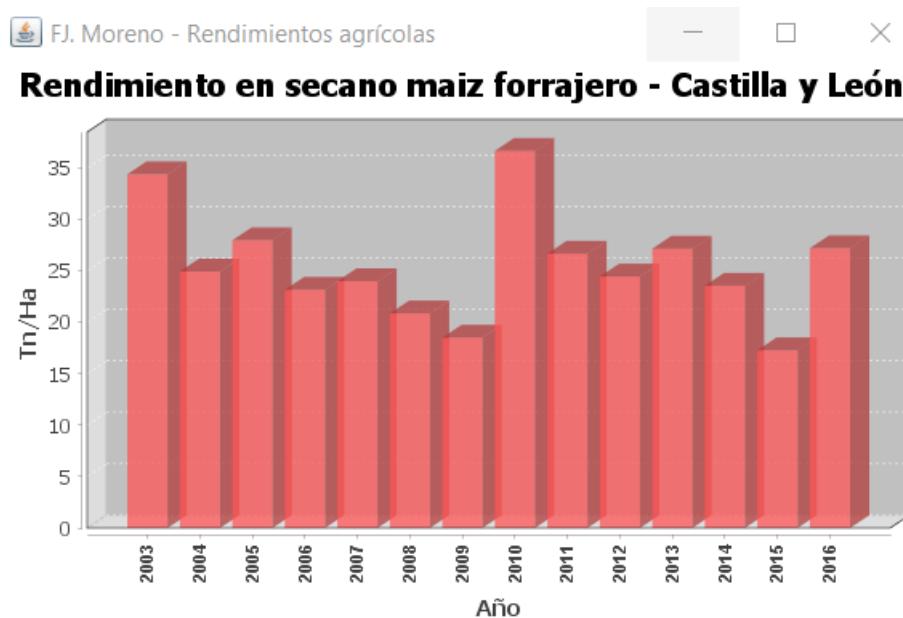


Figura C.7. Rendimiento en secano de maíz forrajero en Castilla y León.

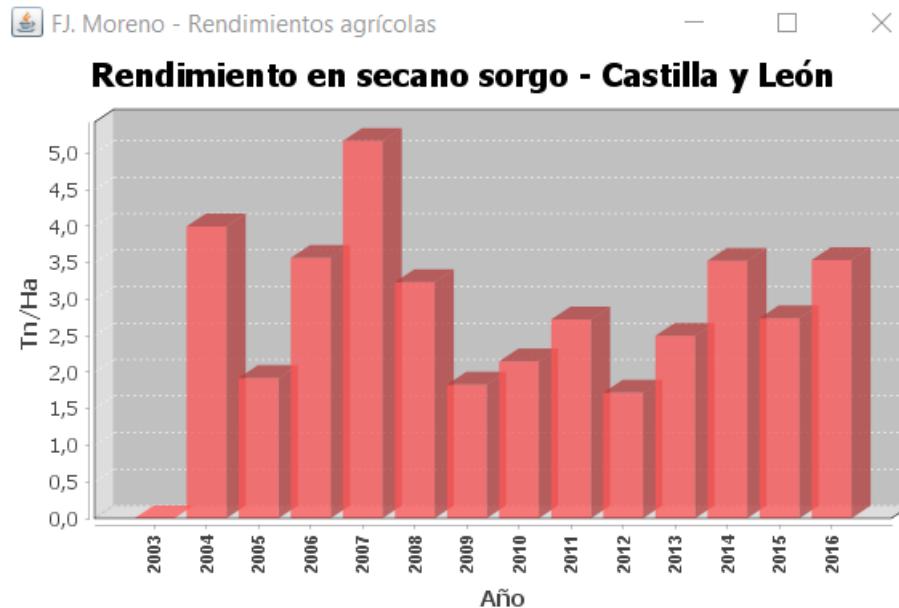


Figura C.8. Rendimiento en secano de sorgo en Castilla y León.

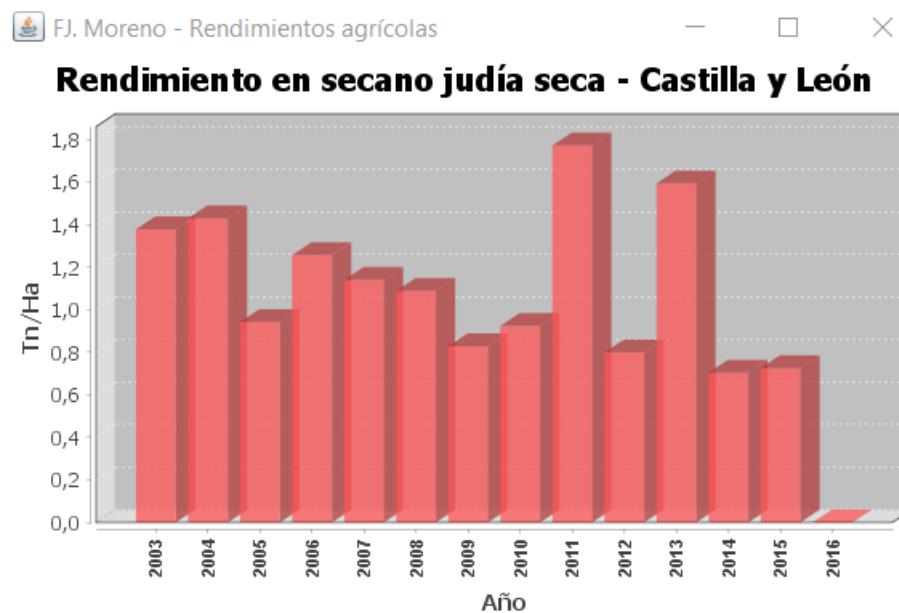


Figura C.9. Rendimiento en secano de judía seca en Castilla y León.

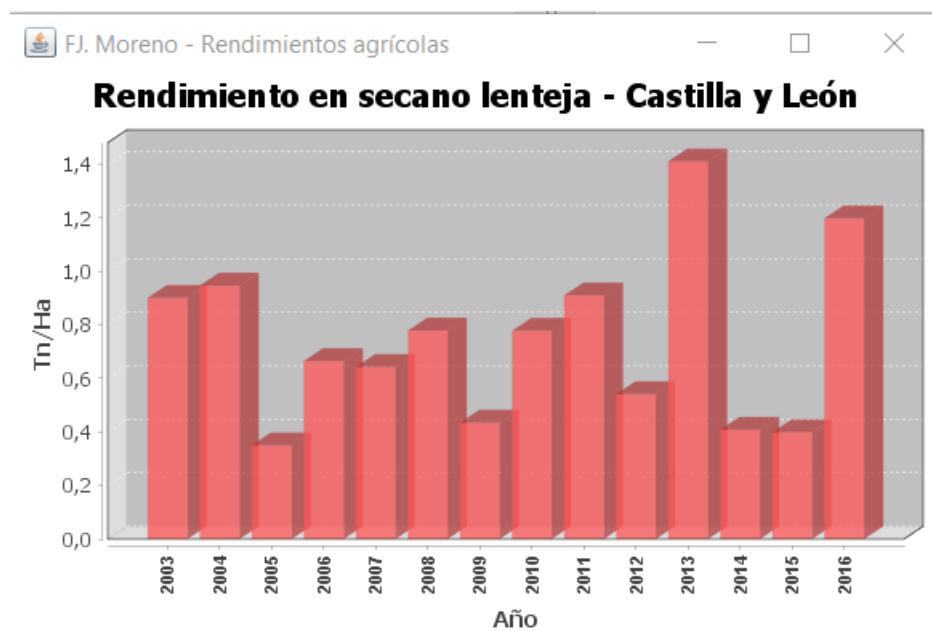


Figura C.10. Rendimiento en secano de lenteja en Castilla y León.

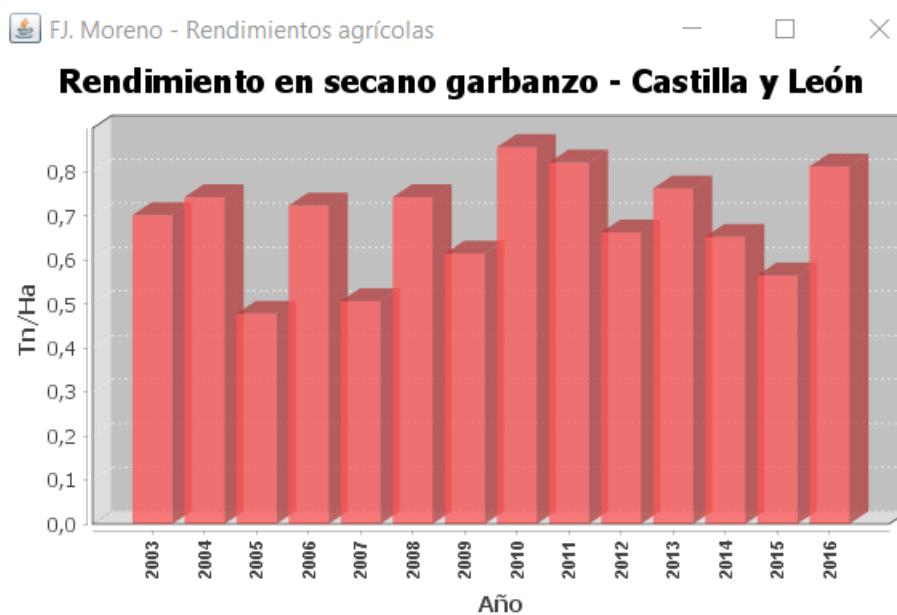


Figura C.11 Rendimiento en secano de garbanzo en Castilla y León.

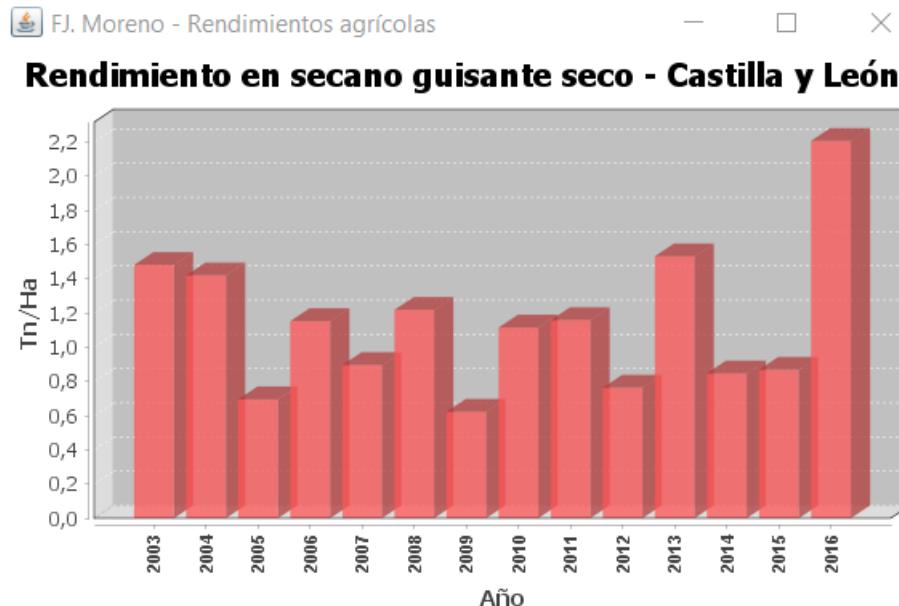


Figura C.12. Rendimiento en secano de guisante seco en Castilla y León.

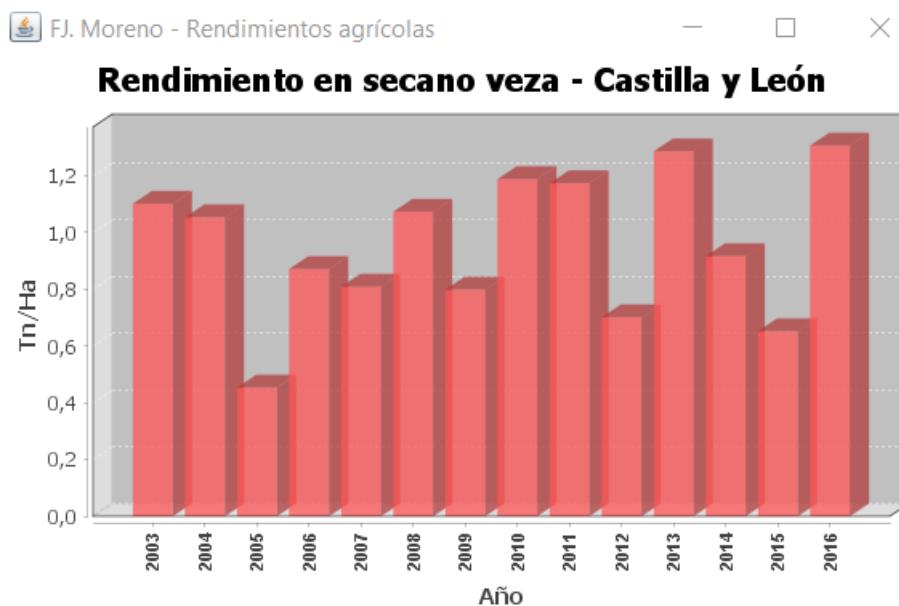


Figura C.13. Rendimiento en secano de veza en Castilla y León.

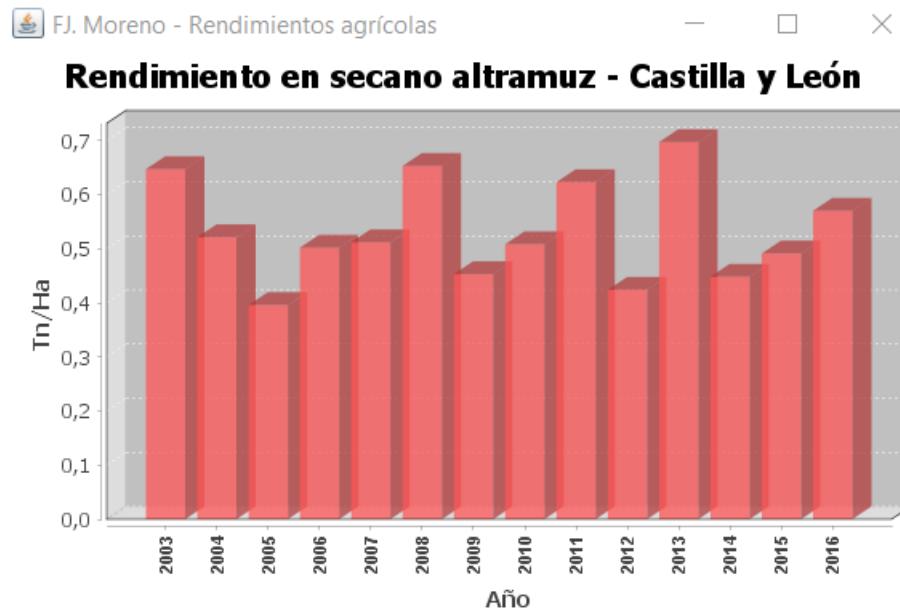


Figura C.14. Rendimiento en secano de altramuz en Castilla y León.

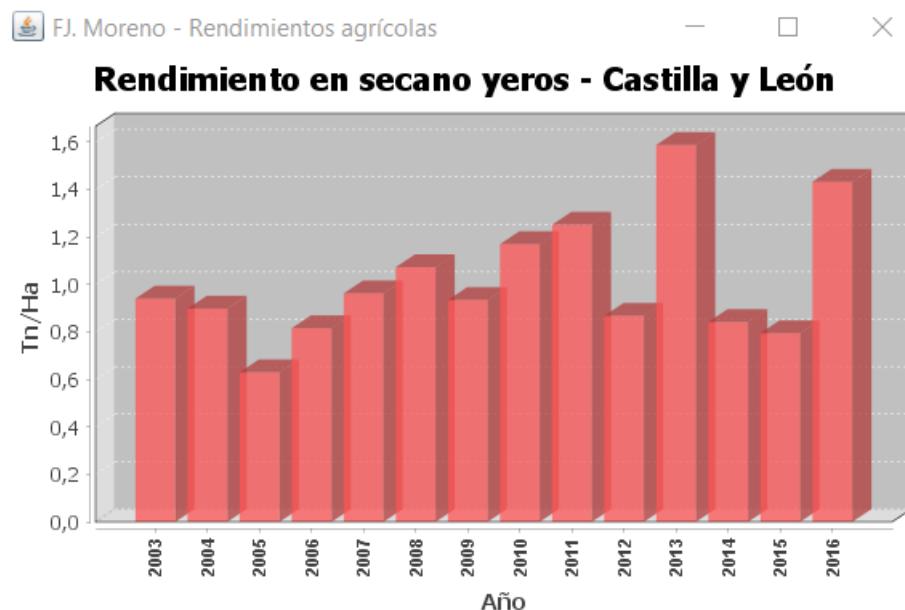


Figura C.15. Rendimiento en secano de yeros en Castilla y León.

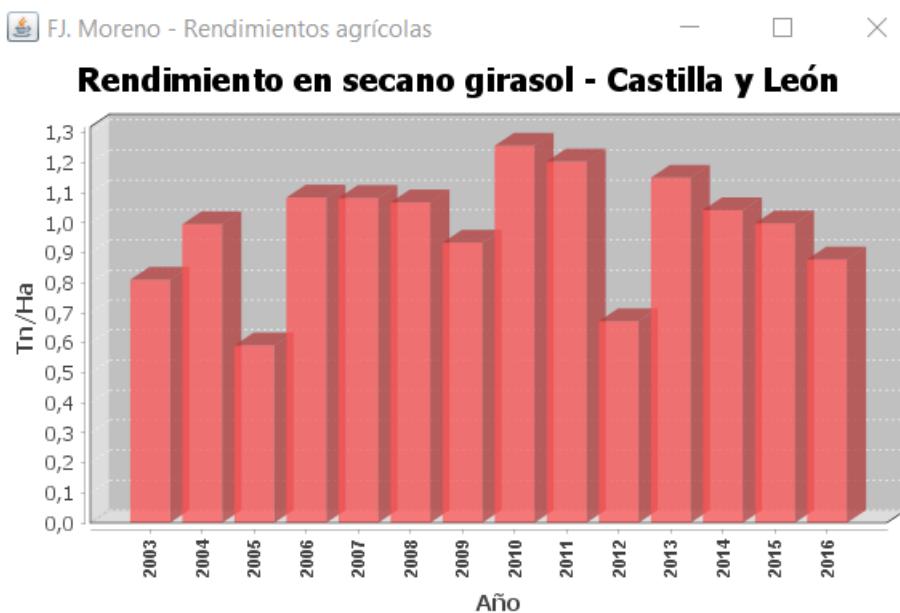


Figura C.16. Rendimiento en secano de girasol en Castilla y León.

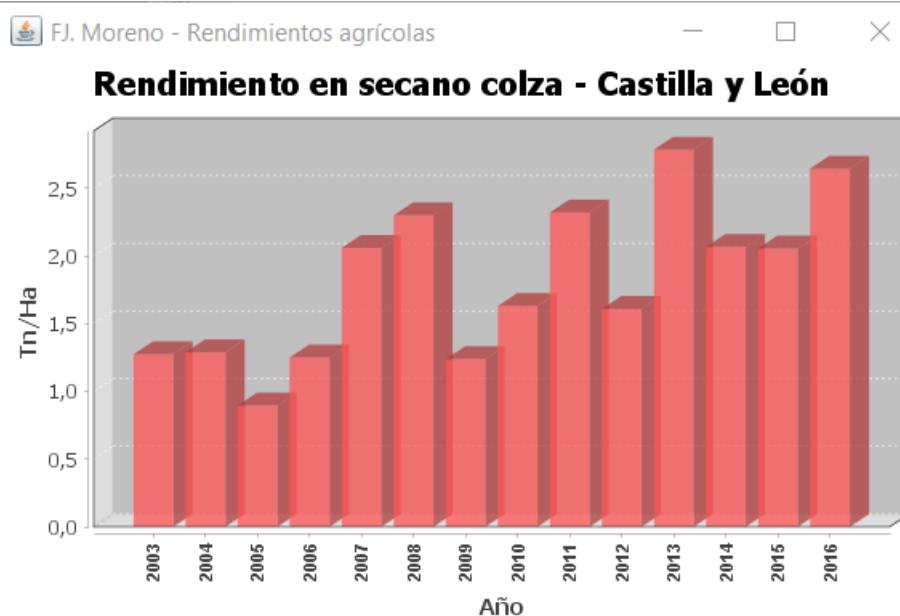


Figura C.17. Rendimiento en secano de colza en Castilla y León.

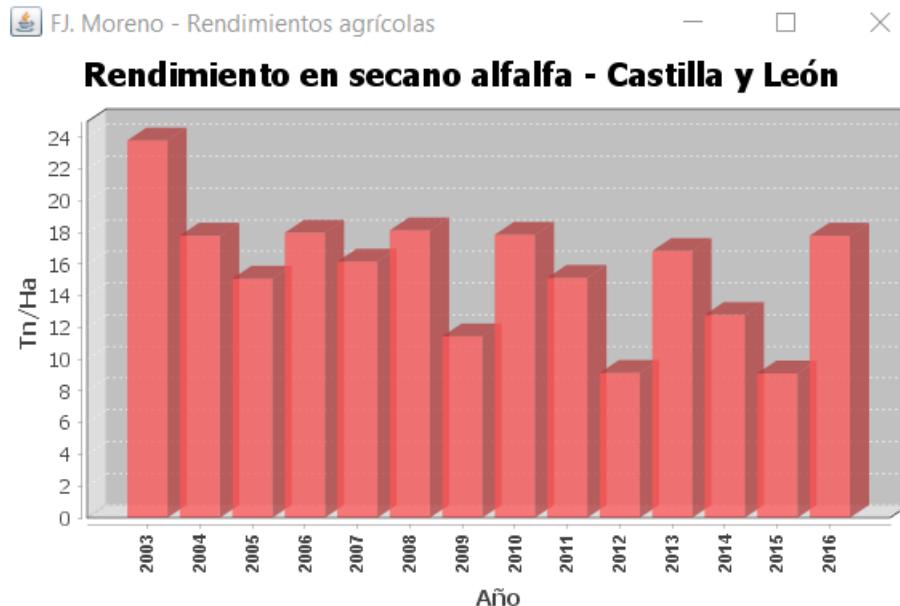


Figura C.18. Rendimiento en secano de alfalfa en Castilla y León.

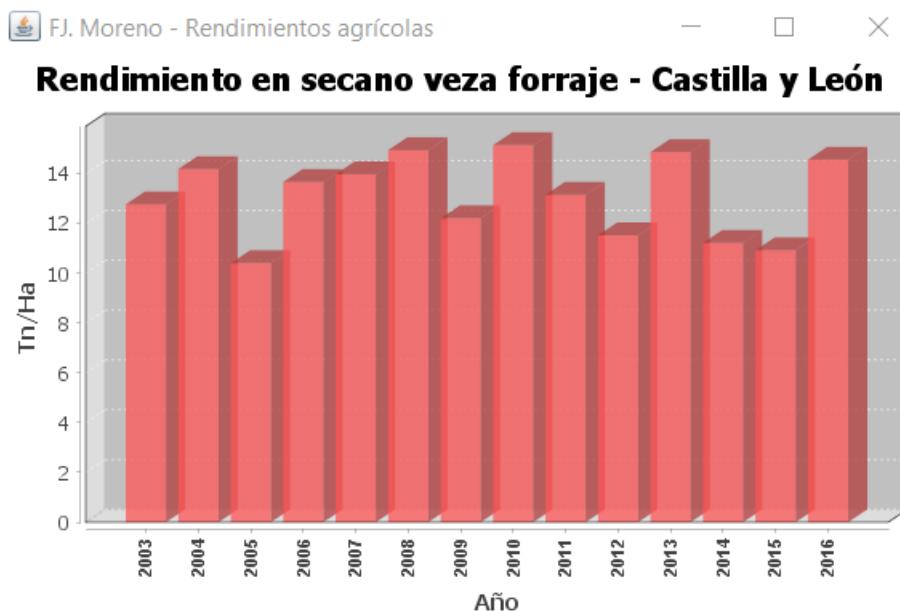


Figura C.19. Rendimiento en secano de veza forraje en Castilla y León.