



Graduado en Ingeniería Informática

Universidad a Distancia de Madrid

Departamento de Ingeniería Informática

TRABAJO DE FIN DE GRADO

Ampliación de un sistema de Big data para mejorar los rendimientos agrícolas con objetivo de realizar previsiones de necesidades de agua tratada en países con escasez de recursos hídricos.

Autor: Paulina Liliana Pyzel

Director: Juan Alfonso Lara Torralbo

MADRID, JULIO DE 2019

Declaración de originalidad:

El autor de este trabajo, Paulina Liliana Pyzel, con DNI ATI667648, declara que el contenido de este trabajo de fin de grado es original, y en el caso de haber utilizado las ideas o contenidos de otros trabajos de otros autores están convenientemente citados, de acuerdo con las normas de citación establecidas.

El número de palabras de este trabajo, excluyendo los anexos es: <20.311>

Índice

Índice de ilustraciones	6
Índice de tablas	9
Resumen	11
Palabras Clave	11
Abstract.....	12
Key Words	12
CAPÍTULO 1: INTRODUCCIÓN	13
1.1. Ámbito del problema	13
1.2. Motivación.....	13
1.3. Objetivos.....	15
1.4. Organización del resto del documento	16
CAPÍTULO 2: TRABAJOS RELACIONADOS	17
2.1. <i>Big data y KDD.</i>	17
2.1.1. El gran reto de Big data.....	17
2.1.2. El proceso de KDD.....	19
2.2. Análisis de series temporales.....	24
2.2.1. Modelos estadísticos.....	25
2.2.2. Transformadas	26
2.2.3. Comparación entre series.	27
2.2.4. Análisis de subsecuencias.....	30
2.2.5. Análisis de eventos	30
2.3. Gestión de agua en países con escasez de recursos hídricos.	31
2.3.1. La escasez del agua como problema global.....	31

2.3.2. MENA- la región impactada por la escasez de recursos hídricos.	32
2.3.3. La importancia de la gestión de recursos hídricos en países con escasez de agua.....	34
2.4. Sistemas informáticos para la predicción de datos meteorológicos.	37
2.4.1. Análisis univariante de las precipitaciones y de la temperatura en el siguiente periodo anual- estudio comparativo de diferentes técnicas.	37
2.4.2. Data mining de datos espaciales y datos de precipitaciones para la observación de sequías.....	39
2.4.3. Data mining de datos meteorológicos con las técnicas de redes neuronales.	41
2.4.4. Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.	42
CAPÍTULO 3: PLANIFICACIÓN TEMPORAL Y COSTES.....	44
CAPÍTULO 4: PROPUESTA.....	47
4.1. Diseño de la ampliación de la aplicación.	47
4.2. Internacionalización de la aplicación.	60
4.2.1. Selección del idioma.....	60
4.2.2. Inclusión de estaciones internacionales en la Base de Datos.	63
4.3. Mejoras de usabilidad en las tareas de descarga, descompresión e importación. 65	
4.3.1. Portabilidad de la aplicación.....	65
4.3.2. Elementos del apoyo para guiar al usuario en la realización de las tareas... 66	
4.4. Creación de bases de datos de usuario.....	69
4.5. Filtrado de datos.	78
4.6. Visualización de datos.	83
4.7. Tareas de data mining.....	86
4.7.1. Análisis de series temporales.....	86

4.7.2. Clasificación.....	91
CAPÍTULO 5: EVALUACIÓN.....	97
CAPÍTULO 6: CONCLUSIONES Y LÍNEAS FUTURAS.....	118
Bibliografía.....	122
Anexos.....	125
Anexo 1: Las clases de la aplicación y su autoría	125
Anexo 2: Código de la clase BigDataTFG.....	127
Anexo 3: Código de la clase LeerMongoRunnable.....	201
Anexo 4: Código de la clase CrearBDMongo	205
Anexo 5: Código de la clase VerTabla.....	206
Anexo 6: Código de la clase ConsultaWhere	211
Anexo 7: Código de la clase VerBasesDeDatos	216
Anexo 8: Código de la clase ColeccionFinalN.....	218
Anexo 9: Código de la clase ConsultaGraficar.....	237
Anexo 10: Código de la clase VerColecciones	243
Anexo 11: Código de la clase Descompresor	246
Anexo 12: Código de la clase DownloadFichero.....	250
Anexo 13: Código de la clase FormatFecha.....	254
Anexo 14: Código de la clase MongoDB.....	257
Anexo 15: Código de la clase FicheroArff.....	259
Anexo 16: Código de la clase TimeSeriesTFG.	274
Anexo 17: Código de la clase Clasificar.....	281

Índice de ilustraciones

Ilustración 1. Ejemplos de Wavelets.....	27
Ilustración 2. Ejemplo de dos series temporales.....	28
Ilustración 3. Ejemplos de modelo "hombros y cabeza".	29
Ilustración 4. Regiones con escasez de agua.	32
Ilustración 5. Recursos de agua dulce internos renovables per cápita (metros cúbicos) en 2014.	33
Ilustración 6. Usos de agua dulce en Jordania (en %).	34
Ilustración 7. Comparación de previsión con datos reales de precipitaciones.	38
Ilustración 8. Comparación de previsión con datos reales de temperatura.	38
Ilustración 9. Metodología del proyecto “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”.....	39
Ilustración 10. Resultados del estudio “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”.....	40
Ilustración 11. Predicciones realizadas sobre la precipitación en la región Centro-Oeste de Brasil en comparación con los datos originales.....	41
Ilustración 12. Error cuadrático de las predicciones realizadas sobre la precipitación en la región Centro-Oeste de Brasil.	42
Ilustración 13. Interfaz gráfica del proyecto “Sistema big data para mejorar los rendimientos agrícolas en Castilla y León”.	43
Ilustración 14. Ejemplo de visualización de los datos: serie temporal de las lluvias en Castilla y León el año 2018.	43
Ilustración 15. Planificación temporal del TFG.	46
Ilustración 16. Nuevos casos de uso.	49
Ilustración 17. Diagrama de secuencia: caso de uso “Cargar los datos en la base de datos elegida”.....	51
Ilustración 18. Diagrama de secuencia: caso de uso “Realizar DM- clasificación”.....	53
Ilustración 19. Diagrama de clases de la aplicación.	59

Ilustración 20. Ventana emergente para la selección del idioma.....	60
Ilustración 21. Aplicación en español.....	60
Ilustración 22. Aplicación en inglés.	61
Ilustración 23. Colección creada en MongoDB con los datos de las estaciones.	64
Ilustración 24. Ventana emergente para selección de directorio.....	65
Ilustración 25. Ejemplos de ventanas emergentes.	67
Ilustración 26. Documentos almacenados en la base de datos tfg (solución anterior)....	67
Ilustración 27. Barra de progreso para la importación de datos en MongoDB.	68
Ilustración 28. Opción refrescar lista (solución anterior).	69
Ilustración 29. Los Combo Box para seleccionar base de datos a crear.....	70
Ilustración 30. Efecto de filtrado en el segundo Combo Box.....	70
Ilustración 31. Colección de los países en MongoDB.....	72
Ilustración 32. Colección de las regiones en MongoDB.	72
Ilustración 33. MongoDB antes de la creación de la base de datos.....	73
Ilustración 34. MongoDB después de la creación de la base de datos.	73
Ilustración 35. Botón para consultar la localización de las estaciones en el mapa.....	75
Ilustración 36. El mapa abierto por el botón.	75
Ilustración 37. Botón de filtrado.....	78
Ilustración 38. Resultado de tareas de filtrado.	79
Ilustración 39. Porcentaje de datos completos.	82
Ilustración 40. Opciones para la visualización.	83
Ilustración 41. Información sobre porcentaje de completitud del dato.	84
Ilustración 42. Gráfica con los datos seleccionados.	84
Ilustración 43. Opciones de selección para creación del fichero .arff.....	86
Ilustración 44. Resultados de análisis (predicción de precipitaciones sin números negativos).	87
Ilustración 45. Selección de técnica de Data Mining.	87
Ilustración 46. Selección de estaciones para el modelo de clasificación.....	92
Ilustración 47. Información sobre el error al intentar elegir una base de datos que no es estación.....	92

Ilustración 48. Solicitud de creación de fichero .arff.	92
Ilustración 49. Fichero .arff para la creación de modelo (fichero de entrenamiento). ...	93
Ilustración 50. Selección de técnica de Data Mining.	93
Ilustración 51. Selección de estación para la predicción.	94
Ilustración 52. Selección de ruta de archivo de entrenamiento.	94
Ilustración 53. Archivo de testeo.	95
Ilustración 54. Resultados de Data Mining- clasificación.	96

Índice de tablas

Tabla 1. Tareas, algoritmos y técnicas de Data Mining.....	20
Tabla 2. Coste (Recursos Humanos).....	44
Tabla 3. Coste (Recursos técnicos).....	45
Tabla 4. Costes totales del proyecto.	45
Tabla 5. Modificaciones de casos de uso.....	47
Tabla 6. Caso1: precipitaciones en Jordania. Completitud del dato.	99
Tabla 7. Caso1: precipitaciones en Jordania. Previsiones.	99
Tabla 8. Caso1: precipitaciones en Jordania. Medición de errores.	100
Tabla 9. Caso 2: precipitaciones en la estación JOM00040265. Completitud del dato.	101
Tabla 10. Caso 2: precipitaciones en la estación JOM00040265. Previsiones.....	101
Tabla 11. Caso 2: precipitaciones la estación JOM00040265. Medición de errores....	102
Tabla 12. Caso 3: precipitaciones en Israel. Completitud del dato.....	103
Tabla 13. Caso 3: precipitaciones en Israel. Previsiones.	103
Tabla 14. Caso 3: precipitaciones la Israel. Medición de errores.	104
Tabla 15. Caso 4: precipitaciones en la estación ISE00105694. Completitud del dato.	105
Tabla 16. Caso 4: precipitaciones en la estación ISE00105694. Previsiones.	105
Tabla 17. Caso 4: precipitaciones en la estación ISE00105694. Medición de errores.	106
Tabla 18. Caso 5. Temperatura mínima en Israel. Completitud del dato.....	107
Tabla 19. Caso 5. Temperatura mínima en Israel. Previsiones.	107
Tabla 20. Caso 5. Temperatura mínima en Israel. Medición de errores	108
Tabla 21. Caso 6. Temperatura mínima en Jordania. Completitud del dato.	109
Tabla 22. Caso 6. Temperatura mínima en Jordania. Previsiones.	109
Tabla 23. Caso 6. Temperatura mínima en Jordania. Medición de errores.	110
Tabla 24. Caso 7. Temperatura mínima en la estación JOM00040265. Completitud del dato.	111
Tabla 25. Caso 7. Temperatura mínima en la estación JOM00040265. Previsiones....	111
Tabla 26. Caso 7. Temperatura mínima en la estación JOM00040265. Medición de errores.	112

Tabla 27. Caso 8. Temperatura mínima en la estación ISE00105694. Completitud del dato.	113
.....
Tabla 28. Caso 8. Temperatura mínima en la estación ISE00105694. Previsiones.....	113
Tabla 29. Caso 8. Temperatura mínima en la estación ISE00105694. Medición de errores.	114
.....
Tabla 30. Resumen de mediciones de error cuadrático.	115

Resumen

El presente trabajo es la continuación del proyecto de sistema de *Big data* para el control de las precipitaciones en la región de Castilla y León de Francisco Javier Hermosilla. Se marca como el principal objetivo transformar una aplicación con un fin particular en una solución más universal que abarque todas las regiones del mundo y distintas variables meteorológicas. Asimismo, se desarrollan nuevos algoritmos de *data mining* y su rendimiento está contrastado en diferentes escenarios.

Si bien el sistema puede ser de utilidad en cualquier zona geográfica, las pruebas se centran en las estaciones meteorológicas de los países de Próximo Este, en especial en Jordania e Israel. Esta región se caracteriza por una escasez de recursos hídricos y se ha considerado que un sistema de predicciones de precipitaciones y temperatura sería especialmente interesante en este entorno. Poder pronosticar las condiciones meteorológicas del siguiente año permitiría a los poderes administrativos estimar las necesidades de agua de fuentes alternativas.

Adicionalmente, se dedica una parte importante de este trabajo al problema de la realización de tareas de minería de datos cuando éstos presentan un alto nivel de valores ausentes. Se ha propuesto uso de algoritmos alternativos a series temporales para mejorar la exactitud de las predicciones.

El presente proyecto constituye un gran avance respecto al sistema inicial. No obstante, se precisan desarrollos de la interfaz de usuario para mejorar la usabilidad y un estudio más detallado de los distintos algoritmos de minería de datos para lograr un sistema completo.

Palabras Clave

Big Data, Minería de Datos, NoSQL, Meteorología, Próximo Este

Abstract

This work is the continuation of the Big data system that improves agricultural performance in Castilla y León, authored by Francisco Javier Hermosilla. The project aims to transform the application that was made with a particular purpose into a more universal solution that would cover all the regions of the world as well as different meteorological variables.

Furthermore, new data mining algorithms are developed, and their performance is contrasted in different scenarios. Although the system can be useful in any geographical area, the tests focus on the weather stations of the Near East countries, especially Jordan and Israel. This region is characterized by water shortage and a system for rainfall and temperature forecast would be especially interesting in this environment. The possibility to predict weather condition for the next year would allow the public administration to estimate the needs of alternative water resources.

Additionally, an important part of this work is dedicated to the problem of data mining performance with a high level of missing values. Alternative algorithms, apart from time series, were developed to improve the accuracy of predictions.

This project represents a great advance in regard to the original system. However, the user interface needs to be improved in order to achieve full usability. In addition to this, a more detailed study of the different data mining algorithms should be conducted to reach a complete working system.

Key Words

Big Data, Data Mining, NoSQL, Meteorology, Near East

CAPÍTULO 1: INTRODUCCIÓN

1.1. Ámbito del problema

La revolución informática, que empezó en los años 50 del siglo pasado, en las dos últimas décadas cuenta con un nuevo protagonista: los datos. Esta constatación puede parecer arriesgada, pues el ser humano tiene predilección por recolectar información desde los principios de la civilización. No obstante, gracias a la informática ha sido posible recoger, almacenar y analizar una cantidad de registros abrumadora.

Esto se debe, en primer lugar, a la automatización de la propia recolección de datos. Los sensores captan datos de magnitud física, biológica o química, pudiendo registrar una nueva medición en intervalos de tiempo muy cortos. Por otro lado, los sistemas de reconocimiento de lenguaje natural e imágenes pueden captar y clasificar cualquier tipo de comunicación humana. En segundo lugar, el aumento de las capacidades de memoria permite almacenar grandes cantidades de datos a un precio relativamente bajo.

La complejidad, hoy en día, reside en el manejo de estos volúmenes de datos, dado que las bases de datos tradicionales no ofrecen las herramientas de consulta suficientemente eficientes. Además, los enfoques tradicionales de análisis, basados en estadística y econometría, con frecuencia fallan a la hora de proponer respuestas. El nacimiento de la ciencia de los datos y *Big data* se debe precisamente a esta problemática. Al tratarse de disciplinas muy jóvenes, desde el mundo académico y profesional se siguen ofreciendo nuevos puntos de vista y soluciones prácticas.

1.2. Motivación

La importancia de poder prever la cantidad de agua proveniente de las precipitaciones es innegable en cualquier región del mundo. No obstante, en países con escasez de recursos hídricos esta necesidad tiene una influencia especialmente significativa sobre el bienestar de la población y la economía del país.

Se eligen, como casos de estudio, los estados de Jordania e Israel, cuyos Gobierno están revisando la política tarifaria de agua para el riego, industria y uso doméstico. Se trata de una de las regiones más limitadas del mundo en cuanto al acceso al agua dulce y donde se necesita estimar con poco margen de error los recursos hídricos que deben ser obtenidos de tratamiento y desalinización. Además, en actualidad Jordania está sufriendo una crisis migratoria por la afluencia de los refugiados sirios, lo cual hace que la situación de escasez sea más dramática.

Un sistema de *Big data* que tenga en cuenta las precipitaciones de años anteriores, así como otros datos meteorológicos y geográficos, permitiría realizar las políticas de recursos hídricos del país de forma más acertada. El sistema analizará series temporales de los datos mencionados para prever la cantidad de lluvia por meses.

La estimación de las precipitaciones según mes y región del país permitirá:

- Evaluar la necesidad de creación de nuevos depósitos del agua y su idoneidad según la región.
- Mejorar la gestión de los depósitos del agua existentes.
- Calcular necesidades de tratamiento de agua y producción de agua proveniente de desalinización.
- Mejorar la política tarifaria.
- Facilitar la planificación de cultivos.
- Facilitar el cálculo de viabilidad de inversiones en tecnologías de riego eficiente (por ejemplo, riego por goteo).

1.3. Objetivos

Los principales objetivos del trabajo están comprendidos en la siguiente lista:

- Mejorar el front end de la aplicación:
 - Permitir selección de idioma (español/inglés).
 - Permitir selección de diferentes países, regiones, estaciones.
 - Permitir selección de directorio donde se guardan y leen los archivos.
 - Incluir una barra de progreso para la carga de datos en MongoDB.
 - Incluir ventanas emergentes para guiar al usuario en la realización de tareas.
 - Visualizar la localización de la estación elegida en Google Maps.
 - Informar sobre el porcentaje de datos disponibles para el periodo elegido.
 - Reagrupar las diferentes opciones de la aplicación para facilitar el uso.
 - Ordenar de forma automática los años cargados en MongoDB.
- Incorporar variables adicionales al sistema:
 - Permitir la visualización de otros datos meteorológicos:
 - Temperatura máxima
 - Temperatura mínima
 - Permitir la visualización de los datos de temperatura tanto en grados Fahrenheit como en grados Celsius.
 - Incluir en el modelo de *data mining* datos de características geográficas: altitud, latitud, longitud.
- Ampliar las tareas de *data mining*:
 - Permitir seleccionar la variable en análisis univariable de series temporales.
 - Realizar análisis multivariable.
 - Desarrollar algoritmos adicionales.
 - Mejorar los algoritmos de *data mining* para eliminar outputs negativos.

1.4. Organización del resto del documento

En la segunda parte de este documento se presentará el marco teórico, tanto desde el punto de vista técnico como en lo relacionado con la problemática específica del trabajo. En este sentido, se introducirán brevemente los términos de *Big data* y *KDD* y se realizará un resumen de las técnicas de análisis de series temporales. A continuación, se describirá la problemática de la escasez de recursos hídricos y se presentarán ejemplos de sistemas informáticos para la predicción de los datos meteorológicos.

El tercer capítulo viene dedicado a la planificación temporal y de costes. Se ofrecerá un diagrama con las principales tareas a realizar y un resumen de presupuesto que incluirá recursos humanos y técnicos.

En la siguiente parte se presentará la propuesta. En primer lugar, se describirán brevemente las tareas de diseño y, a continuación, se explicará en detalle el desarrollo realizado de los distintos casos de uso. Cabe destacar que el trabajo cuenta con dos partes diferenciadas: los procesos de manejo y visualización de los datos y las tareas de *data mining*, distinción que quedará patente en la citada parte.

El capítulo cinco ofrecerá la evaluación de los desarrollos realizados, centrándose en los modelos de análisis obtenidos. Por último, en el epígrafe de conclusiones, se resumirá el progreso logrado respecto al software de partida y se indicarán las ideas de desarrollos y ampliaciones futuras.

CAPÍTULO 2: TRABAJOS RELACIONADOS

2.1. *Big data* y KDD.

2.1.1. *El gran reto de Big data.*

El concepto de *Big data* es en actualidad tan presente en la realidad empresarial e investigación científica que resulta sorprendente que hace tan solo 20 años fuera prácticamente desconocido (Diebold, 2012). En primer lugar, sería conveniente definir cuando hablamos de una base de datos suficiente grande para poder afirmar que estamos tratando con *Big data*. En una interesante reflexión, Adam Jacobs constata que no hay ningún tamaño que determine cuando se debe utilizar este término, sino que esto depende del objetivo del tratamiento de los datos (Jacobs, 2009).

El tema de este trabajo, los datos meteorológicos, constituye un buen ejemplo para entender la esencia y la problemática de *Big data*. Recolectar y analizar los registros de precipitaciones o temperatura puede ser una tarea puramente estadística si nos ceñimos a unos sets estáticos de una región determinada y si nos limitamos a realizar unos simples cálculos de medias aritméticas para relatar el pasado. No obstante, si queremos ofrecer un acceso dinámico a los datos y, aparte de presentar los datos históricos, realizar también predicciones a futuro con modelos de *data mining*, podríamos decir que estamos resolviendo un problema de *Big data*.

El principal reto relacionado con gestionar grandes cantidades de información no reside en el almacenamiento, ya que las tareas transaccionales de las bases de datos tradicionales, basadas en SQL, son suficiente eficientes para añadir nuevos registros. El problema viene por la necesidad de realizar consultas sobre estos datos para obtener conocimiento (Jacobs, 2009).

La primera vía que se buscó para abordar este reto fue la creación de los *Data Warehouses*. A través del proceso de ETL (extraer, transformar, cargar), se pretende crear, partiendo de datos operacionales, unas colecciones adecuadas para realizar las tareas analíticas. No

obstante, esta solución resulta en ocasiones insuficiente para grandes cantidades de registros, especialmente cuando se trata de series temporales con mediciones frecuentes. Es un enfoque que sigue siendo interesante cuando se realizan modelos de clasificación, *clustering* o asociación sobre entidades con múltiples atributos, pero presenta más fallos en series largas de unos pocos atributos. Las bases de datos relacionales normalizadas se centran en el ahorro del espacio del almacenamiento y en la consistencia, pero exigen una cantidad de tiempo muy relevante para interpretar las consultas y acceder a la información. Se necesita un cambio de paradigma, utilizar bases de datos desnormalizadas y aplicar un acceso secuencial a los datos en vez de uno aleatorio.

El auge de los sistemas NoSQL se debe en parte al surgimiento de la problemática de *Big data*. Se trata de unas soluciones de almacenamiento de datos que se distancian del modelo tradicional y apuestan más por disponibilidad y por la tolerancia al particionamiento dándole menos importancia a la consistencia (Zhingri & Augusto, 2016). No obstante, también es posible diferenciar distintos tipos de sistemas NoSQL (Faraj., Rashid & Shareef, 2014):

- Bases de datos clave-valor: son muy rápidas y escalables, pero no permiten consultas complejas como agrupación o *joins*. Un ejemplo de este tipo de base de datos es el sistema Dynamo de Amazon.
- Bases de datos documentales: están codificadas con un formato estándar como XML o JSON, permiten uso de objetos anidados. Un ejemplo de este tipo de base de datos es MongoDB.
- Bases de datos orientadas a columnas: se parecen a bases de datos relacionales, con la excepción que cada fila puede tener otro número de columnas. Un ejemplo de este tipo de base de datos es Cassandra.
- Bases de datos orientadas a grafos: se basan en la estructura de un grafo, cada registro tiene un puntero a otros registros relacionados en vez de un identificador. Un ejemplo de este tipo de base de datos es VertexDB.

Para elegir una solución adecuada para un proyecto hay que considerar las necesidades de este, tanto las referentes al almacenamiento como las vinculadas al análisis. En este sentido, las bases de datos documentales pueden ofrecer unas prestaciones muy buenas de guardado y recuperación de registros, sin perjudicar demasiado las posibilidades de realizar unas consultas relativamente complejas (Zhingri & Augusto, 2016).

2.1.2. El proceso de KDD.

El proceso de *KDD* (*Knowledge Discovery in Databases*) tiene como objetivo el descubrimiento de conocimiento en bases de datos. En ocasiones se confunde con el término de *data mining*. No obstante, la minería de datos es solamente una de las tareas de *KDD*, si bien se la podría ver como su parte central. El proceso de *KDD* completo engloba:

- Recopilación de datos.
- Selección, limpieza y transformación de datos.
- Minería de datos.
- Interpretación y evaluación de modelos.

La recopilación de datos no consiste solamente en reunir diferentes fuentes, sino también en unificarlas a través de un proceso ETL y almacenarlas en un *Data Warehouse*. Durante la selección se lleva a cabo el filtrado de atributos, para descartar los que sean irrelevantes para la investigación, así como el filtrado de registros, con el objetivo de eliminar los segmentos excluidos del estudio. La limpieza tiene que abordar dos problemáticas principales: la existencia de valores erróneos y la ausencia de valores. Por otro lado, el objetivo de la transformación consiste en preparar los datos para facilitar las actividades de *data mining* a través de diferentes técnicas, tales como: la numerización, la discretización, la normalización o la reducción de dimensionalidad.

Una vez preparada la vista minable, se pueden llevar a cabo las labores de *data mining*. Los numerosos algoritmos y técnicas de esta fase se pueden clasificar en las siguientes tareas:

- Tareas predictivas, cuyo objetivo es predecir un dato desconocido partiendo de otros conocidos. Incluyen las siguientes sub-categorías:
 - La clasificación: consiste en asignar el valor de un atributo cualitativo desconocido en base a otros atributos conocidos.
 - La regresión: consiste en asignar un valor numérico a un atributo cuantitativo desconocido en base a otros atributos conocidos.
- Tareas descriptivas, que buscan agrupar los datos de una manera que demuestre las relaciones existentes entre ellos y les dé significado. Incluyen las siguientes sub-categorías:
 - *Clustering*: su objetivo es dividir la población de estudio en grupos con la máxima homogeneidad posible.
 - Asociación: explora las relaciones entre los atributos para encontrar los que aparezcan juntos con más frecuencia.
 - Detección de atípicos: busca los objetos que presenten valores alejados de valores medios de la muestra.

A continuación, se resume los principales algoritmos y técnicas de *data mining* según su clasificación:

Tabla 1. Tareas, algoritmos y técnicas de Data Mining.

Tipo de tarea	Algoritmo / Técnica	Descripción
Clasificación	Árboles de decisión	La ramificación especifica los distintos conjuntos de atributos con significado clasificador. Cuanto más cerca se encuentre una rama del nodo raíz, más entropía presenta el atributo. Las hojas son los atributos que predecir.

Tipo de tarea	Algoritmo / Técnica	Descripción
	Redes neuronales: <ul style="list-style-type: none"> • <i>Perceptron</i> multicapa 	Se trata de un sistema que parte de una función y que, al recibir entradas, la transforma para producir un resultado cercano a las salidas correspondientes.
	Técnicas Bayesianas	Se basan en el teorema de Bayes que describe la probabilidad condicionada de un evento futuro basándose en otros eventos pasados.
	Técnicas basadas en casos: <ul style="list-style-type: none"> • K vecinos más próximos 	La clasificación se realiza por comparación con registros anteriores más similares a la nueva ocurrencia.
Regresión	Técnicas de regresión: <ul style="list-style-type: none"> • Lineal • No lineal 	Se trata de predecir una variable (llamada variable respuesta) como función de otras variables (variables predictoras). La función que describe esta relación puede ser tanto lineal como un polinomio.
<i>Clustering</i>	<i>Clustering</i> particional: <ul style="list-style-type: none"> • K-medias 	Se asigna cada objeto a un <i>cluster</i> de acuerdo con la distancia que lo separa de un elemento representante de cada partición.

Tipo de tarea	Algoritmo / Técnica	Descripción
	<p>Técnicas jerárquicas</p> <ul style="list-style-type: none"> • AGNES • DIANA 	Se trata de hacer un árbol de <i>clusters</i> en el que en cada nivel inferior los elementos estén más homogéneos.
	<p>Técnicas basadas en densidad</p> <ul style="list-style-type: none"> • DBSCAN 	Se basan en el concepto de densidad de un punto, entendida como el número de objetos que se pueden considerar como “vecinos” al encontrarse a una distancia comprendida dentro de un radio.
	<p>Técnicas basadas en <i>grid</i></p> <ul style="list-style-type: none"> • STING • WaveCluster • Clique 	Son algoritmos que dividen el espacio en muchas celdas en forma de rejilla.
Asociación	<p>Técnicas de asociación:</p> <ul style="list-style-type: none"> • Apriori TID • AprioriHybrid • AIS • SETM 	Los algoritmos de asociación tienen dos partes principales. En primer lugar, se encuentran los grupos de atributos (<i>itemsets</i>) que aparezcan juntos con frecuencia que supere un umbral establecido, llamado soporte. El siguiente paso consiste en determinar si existe una regla de asociación entre un <i>itemset</i> más pequeño y otro más grande. Este nivel de confianza se calcula dividiendo el soporte del <i>itemset</i>

Tipo de tarea	Algoritmo / Técnica	Descripción
		más grande entre el del <i>itemset</i> más pequeño.
Detección de atípicos	Aproximaciones estadísticas	Hace uso de herramientas estadísticas tales como la media o la desviación típica en una distribución normal.
	Aproximaciones basadas en otras tares de <i>data mining</i> : <ul style="list-style-type: none"> • Proximidad (k vecinos más cercanos) • Densidad • <u>Clustering</u> 	Se trata de usar las técnicas que encuentran similitud entre los elementos de manera inversa, buscando objetos que se caractericen por mantener una distancia significativa con los demás elementos.

Para que un modelo de *data mining* posea cierta fiabilidad es necesario realizar una prueba de su comportamiento. La forma más sencilla consiste en separar desde el principio de la muestra una parte de registros y, una vez construido el modelo, realizar predicciones sobre ellos y comprobar si concuerdan con datos reales. Tampoco se puede restar importancia a una óptima expresividad del modelo. Se debe evitar subajuste, ya que descripciones muy generales no ofrecerían ningún conocimiento nuevo o no trivial. De igual manera, sería un error aplicar un sobreajuste, pues se caería en proveer una descripción de los datos sin ningún valor predictivo o agrupador.

2.2. Análisis de series temporales.

Las tareas tradicionales de *data mining* son muy útiles para analizar grandes cantidades de datos de objetos diferentes, pero muestran deficiencias cuando se trata de describir registros continuados del mismo elemento. Sin embargo, en multitud de entornos, tales como la medicina, el sector financiero o la observación meteorológica el objeto de interés son precisamente las series temporales.

Por esta razón, en los últimos años se ha abierto un debate para encontrar técnicas que permitiesen un análisis adecuado de este tipo de datos. La diferencia de enfoque empieza ya en las tareas previas de selección, limpieza y transformación de datos. En algunos casos puede resultar interesante guardar solamente las mediciones que impliquen cambio de dato. Esto reduce el uso de memoria, pero hace más difíciles las tareas de análisis. Además, puede ocurrir que diferentes datos no se midan en los mismos intervalos de tiempo. Por ejemplo, en una observación meteorológica es posible que se observe la temperatura cada hora y el nivel de precipitaciones de forma diaria. A la hora de analizar estos datos sería necesario unificarlos.

Asimismo, las series temporales traen también la dificultad de la estacionalidad de las observaciones. De ahí que los datos tienen un doble sentido. Volviendo al ejemplo de las precipitaciones, para predecir la cantidad de lluvia de mes de junio hay que tener en cuenta tanto lo que ocurrió en los meses de abril y mayo, como los datos de verano de años anteriores. Es más, no tiene la misma importancia la observación del último año que de hace dos décadas, pues es muy probable que exista una tendencia que haga que la precipitación sea cada vez menor o mayor.

El estudio de series temporales implica numerosas complicaciones. La problemática más importante incluye (Keogh, 1997):

- Ruido: ocurre cuando en una tendencia principal de crecimiento aparecen intervalos cortos de decrecimiento o *viceversa*.

- La magnitud de valores: ocurre cuando dos series tienen una secuencia parecida de aumentos y disminuciones de valores, pero estos valores son más altos en una de las series.
- La amplitud de cambio: ocurre cuando dos series tienen una secuencia parecida de aumentos y disminuciones de valores, pero en una de ellas la diferencia entre los máximos y los mínimos es más significativa.
- La duración del cambio: ocurre cuando dos series tienen una secuencia parecida de aumentos y disminuciones de valores, pero en una de ellas los lapsos de tiempo entre cambios de tendencia tienen más duración.

Las series temporales pueden ser discretas o continuas, aunque este segundo término es realmente teórico, pues las distintas mediciones siempre se realizan en diferentes lapsos de tiempo (Das, 1994). No obstante, si la distancia entre las mediciones es muy pequeña se puede considerar la serie como si fuera continua. Por otro lado, los valores de los atributos también pueden tener carácter discreto o continuo.

Existen al menos dos enfoques para abordar el análisis de la serie temporal. El primero, más tradicional, implica el uso de modelos estadísticos. El segundo busca respuestas en las tareas de *data mining* modificadas para abordar esta problemática.

2.2.1. Modelos estadísticos.

Se considera un modelo estadístico de un proceso estocástico un conjunto de hipótesis correctamente definidas sobre sus propiedades (Mauricio, 2007). Los modelos univariantes más usados incluyen:

- Ruido blanco
- Paseo Aleatorio
- Modelo MARCH

Los modelos multivariantes más usados incluyen:

- Modelo ADL

- Modelo VAR

2.2.2. Transformadas.

Este enfoque está basado en la representación de una serie temporal como una descomposición de funciones trigonométricas (Kleist, 2015). La transformada más usada en *data mining* es la de Fourier. El teorema de Fourier dice que cualquier función periódica se puede expresar como una suma infinita ponderada de senos y cosenos (Medina, 2010). La ecuación básica de la transformada de Fourier es la siguientes:

$$f(x) = \sum_{n=-\infty}^{\infty} a_n e(nx)$$

(Ecuación 1)

Los $e(nx)$ llevan el nombre de ármonicos de la descomposición y los a_n se conocen como coeficientes de Fourier. Los ármonicos indican en qué subfunciones se descompone la función origen y los coeficientes asignan una ponderación a cada una de ellas. Puede parecer que este teorema no resuelve los problemas de manejo de datos temporales para su análisis, pues propone una descomposición de la función en un número infinito de otras funciones. No obstante, algunos autores indican que, para detectar tendencias, solamente se necesita obtener los componentes principales. Sin embargo, no existe el consenso de qué número de componentes sería suficiente para obtener una respuesta significativa (Lara, 2011).

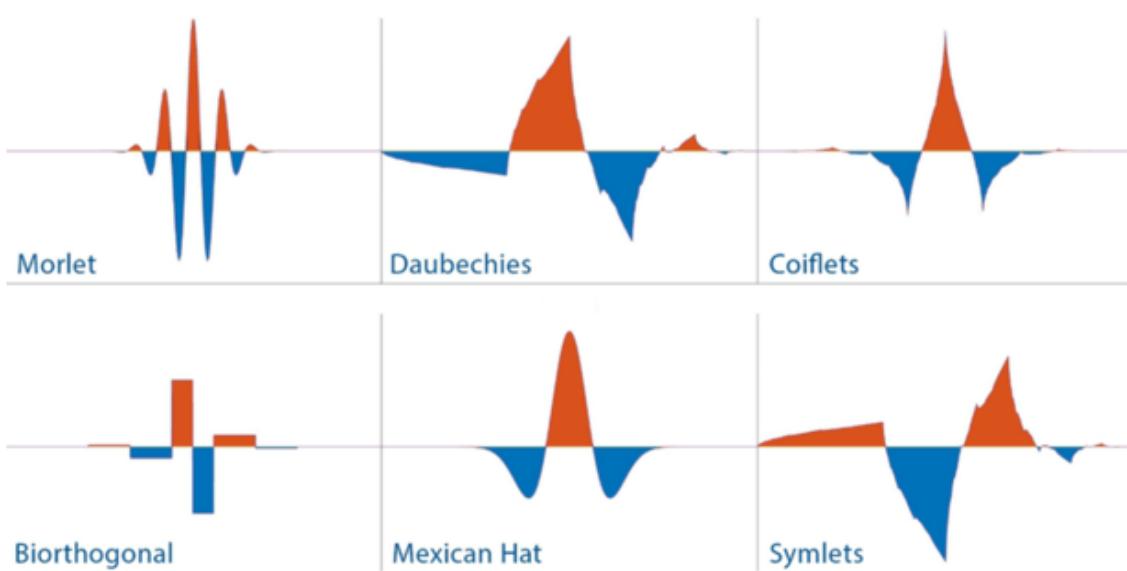
Otra alternativa ampliamente usada es la trasformada de Wavelet. El Wavelet es una función real o compleja que satisface las siguientes condiciones:

$$\int_{-\infty}^{\infty} \psi(u) du = 0 \text{ y } \int_{-\infty}^{\infty} |\psi^2(u)| du = 1$$

(Ecuación 2)

En las siguientes gráficas se pueden ver algunos ejemplos de funciones Wavelet:

Ilustración 1. Ejemplos de Wavelets.



Fuente: MathWorks

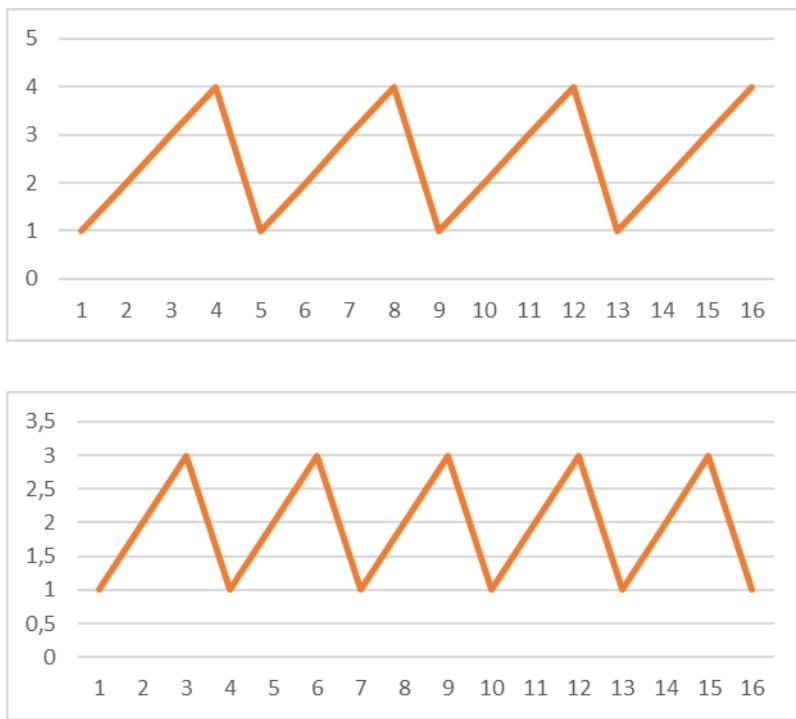
Esta técnica resulta especialmente interesante para datos no estacionales y no lineales (Minu, Lineesh & John, 2010). Su ventaja sobre la transformada de Fourier consiste en su facilidad de expresar cambios bruscos en tendencia. Se puede decir que, si la transformada de Fourier busca los ciclos dentro de una serie, la transformada de Wavelet hace lo contrario, su objetivo es encontrar los puntos que no se ajusten a una tendencia.

2.2.3. Comparación entre series.

La tarea de comparar dos series temporales se podría realizar calculando la distancia de valores de eje Y en cada punto de X. La deficiencia de este enfoque reside en que existen series con formas parecidas, pero en las que el cambio de tendencia de aumento a decrecimiento no se realiza en los mismos intervalos.

En el siguiente ejemplo se puede observar a simple vista que se trata de series que se comportan de una manera similar, siendo una alternancia de subidas y bajadas en la medición. No obstante, como en la primera el cambio de tendencia se registra después de cuatro mediciones y en la segunda esto sucede en la tercera ocurrencia la comparación de distancias entre los valores de eje Y y cada punto de X sugeriría que se trata de dos series muy distintas.

Ilustración 2. Ejemplo de dos series temporales.

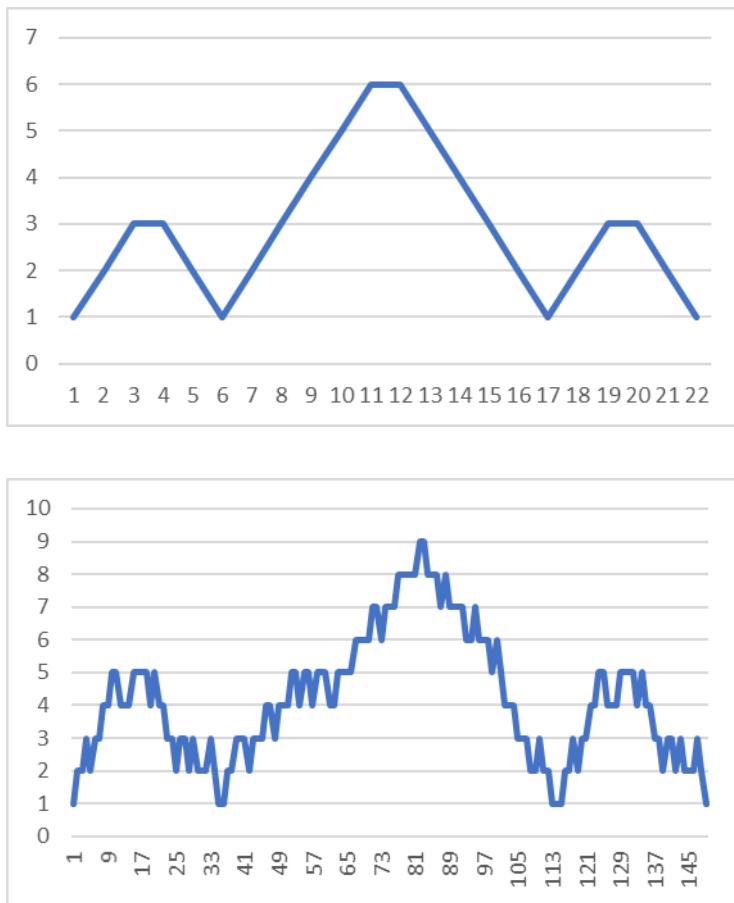


Fuente: Elaboración propia.

Para evitar llegar a estas conclusiones incorrectas, es recomendable hacer uso de una técnica conocida como *time wraping*, que consiste en estrechar los intervalos hasta llegar al máximo parecido entre las series. No obstante, puede que ni siquiera de esta manera podamos ver la similitud existente. Sirvan como explicación dos ejemplos del modelo de

“hombros y cabeza”. El parecido es fácilmente perceptible, pero las medidas de distancia no nos advertirían de él y la técnica de *time wraping* no aportaría mucho en este caso.

Ilustración 3. Ejemplos de modelo "hombros y cabeza".



Fuente: Elaboración propia.

Otro enfoque de detección de similitudes entre dos series, con especial utilidad cuando se trata de datos con mucho ruido, es la técnica de la subsecuencia común más larga, conocida como LCSS (Aghabozorgi, Saybani, & Wah, 2012). Este algoritmo permite omitir algunos registros de la serie, por lo que presenta más flexibilidad que *time wraping*.

2.2.4. Análisis de subsecuencias.

El objetivo de esta tarea consiste en descomponer una serie temporal en subsecuencias e intentar buscar similitudes entre estos fragmentos más pequeños. Una de las técnicas que se utiliza para este fin es MBR (Minimum Bounding Rectangles). Consiste en agrupar las distintas mediciones en un número reducido de rectángulos cuyos lados serían un intervalo de tiempo y un intervalo de los valores de Y. Un concepto más amplio sería el Bounding Box (BBOX) y utilizando esta técnica no sería necesario usar solamente rectángulos, pudiendo hacer las agrupaciones dentro de todo tipo de figuras bidimensionales.

Otra opción que considerar son los árboles de subsecuencia. Se trata de almacenar los pares del tipo de secuencia y la serie en la que había ocurrido. De esta manera se puede emparejar series aparentemente muy distintas si éstas poseen un número significativo de subsecuencias similares.

No se debe confundir este concepto con un árbol basado en MBR. En este caso se trataría de agrupar los rectángulos pequeños en rectángulos cada vez más grande, de forma que, en última instancia, la serie completa sería el rectángulo más grande (Keogh, 1997).

2.2.5. Análisis de eventos

En algunas disciplinas las mediciones no se realizan para estudiar la serie temporal en su totalidad, sino con el objetivo de detectar eventos de interés. El análisis debe basarse en los puntos de cambio, en los que la tendencia actual varía bruscamente (Lara, 2011). Este tipo de estudio sería interesante para detectar anomalías en datos médicos o para identificar los primeros indicios de una catástrofe natural.

2.3. Gestión de agua en países con escasez de recursos hídricos.

2.3.1. La escasez del agua como problema global.

La inmensa mayoría del agua disponible es salada y tan solo 3% son fuentes de agua dulce, aptos para industria, agricultura y consumo. Además, 70% de estos recursos se encuentran en glaciares e icebergs. El agua disponible viene de la lluvia, los ríos, lagos, manantiales y fuentes subterráneas (Roudi-Fahimi, Creel, & De Souza, 2002).

La escasez de recursos hídricos puede tener su origen tanto en una insuficiente disponibilidad de fuentes de agua dulce como en su gestión ineficiente. Además, no se trata de un término absoluto, sino de un desajuste entre la oferta y la demanda. No obstante, para simplificar la problemática se considera que un país sufre de escasez si se dispone de menos de 500 m^3 de agua per cápita por año, mientras que si se cuenta con entre 500 y 1000 m^3 hablaríamos de condiciones de estrés hídrico (Aquastat, F. A. O., 2011).

Otra forma de medir este fenómeno es la ratio WWR (Withdrawal to Water Resources):

$$WWR = W/Q$$

(Ecuación 3)

donde W es la captación anual del agua y Q es la cantidad de agua renovable disponible, que se calcula por aproximación usando los datos del caudal medio. Se considera que si esta tasa supera el 40% el país está bajo una escasez severa. No obstante, el cambio climático provoca que los caudales de muchos ríos crezcan en temporadas lluviosas de forma tan significativa que incluso provocan inundaciones, por lo que, a pesar del aumento del caudal medio, este crecimiento no es susceptible de aprovechamiento (Hanasaki, Fujimori, Yamamoto, Yoshikawa, Masaki, Hijioka, ... & Kanae, 2012).

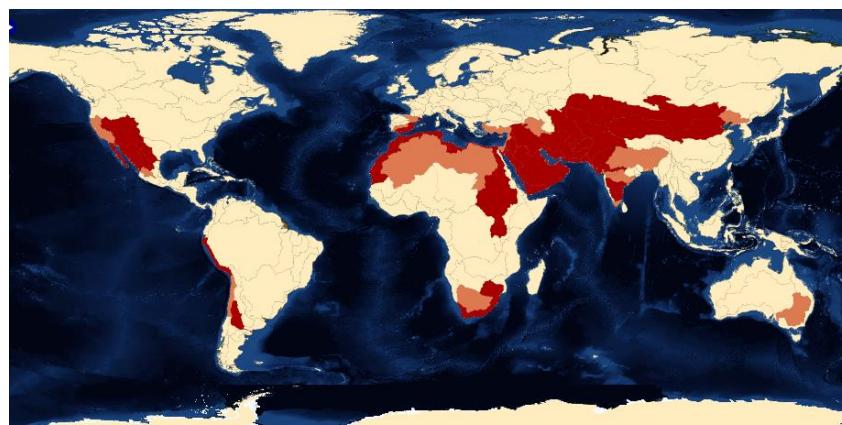
La escasez de agua se manifiesta por una demanda insatisfecha, alta competencia entre los diferentes usuarios (industria, agricultura, uso doméstico) que puede desembocar en

conflictos y el drenaje irreversible de aguas subterráneas que trae como consecuencia un impacto medioambiental negativo. Se estima que durante el siglo XX la demanda del agua creció dos veces más rápido que la población. El desarrollo económico, el cambio climático y la contaminación hacen que la necesidad de una gestión eficaz sea cada vez más patente (Steduto, Hoogeveen, Winpenny, & Burke, 2017).

2.3.2. MENA- la región impactada por la escasez de recursos hídricos.

Si bien se trata de un problema global, los países con clima árido o semi-árido son los que más sufren por la escasez de recursos hídricos. En el siguiente mapa se puede observar que las regiones más afectadas por este problema están concentrados en una región conocida como MENA (Oriente Medio y Norte de África). De 15 países con más escasez de agua, 12 se encuentran en MENA. La región está habitada por 5% de la población mundial y tiene acceso a solamente 1% de los recursos de agua dulce. Entre 2000 y 2009 la demanda insatisfecha sumó 42km³ y se estima que este número crecerá a 199 km³ entre 2040 y 2050 (Martens, 2017).

Ilustración 4. Regiones con escasez de agua.



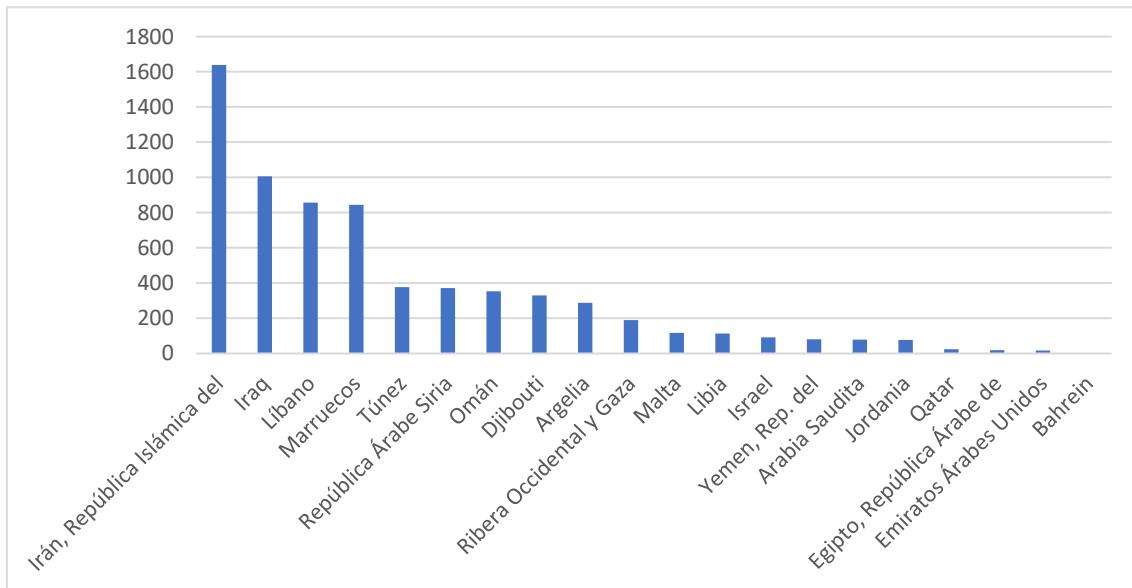
Fuente: FAO

Esta difícil realidad empeora por el aumento de la población, debido tanto al crecimiento natural como a los movimientos migratorios. Entre 1970 y 2001 la población de la región

se ha doblado, pasando de 173 a 386 millones (Roudi-Fahimi, Creel, & De Souza, 2002). Si bien desde 2001 la tasa de fertilidad bajó de 3.1 a 2.8, se sigue situando muy por encima del reemplazo generacional. Se espera que la demanda de agua aumente un 50 por ciento en 2050 en la región si las tasas actuales de crecimiento continúan y si se cumplen las predicciones sobre el calentamiento global (Mualla, 2018).

Por otro lado, en cuanto al aumento poblacional debido a la migración, un buen ejemplo es la situación de Jordania que, según el censo de 2015, había acogido a más de 1.2 millones de refugiados sirios en un periodo de cinco años (Krafft, Sieverding, Salemi & Keo, 2018). Hecho que es incluso más grave si se tiene en cuenta que Siria es uno de los países de la región que con más fuentes de agua renovable contaba, siendo Jordania uno de los países con menos agua renovable disponible. En la siguiente gráfica se puede observar que tan solo cuatro países de MENA superan el umbral de 500 metros cúbicos del agua por habitante, por encima del cual se considera que no existe el problema de escasez.

Ilustración 5. Recursos de agua dulce internos renovables per cápita (metros cúbicos) en 2014.

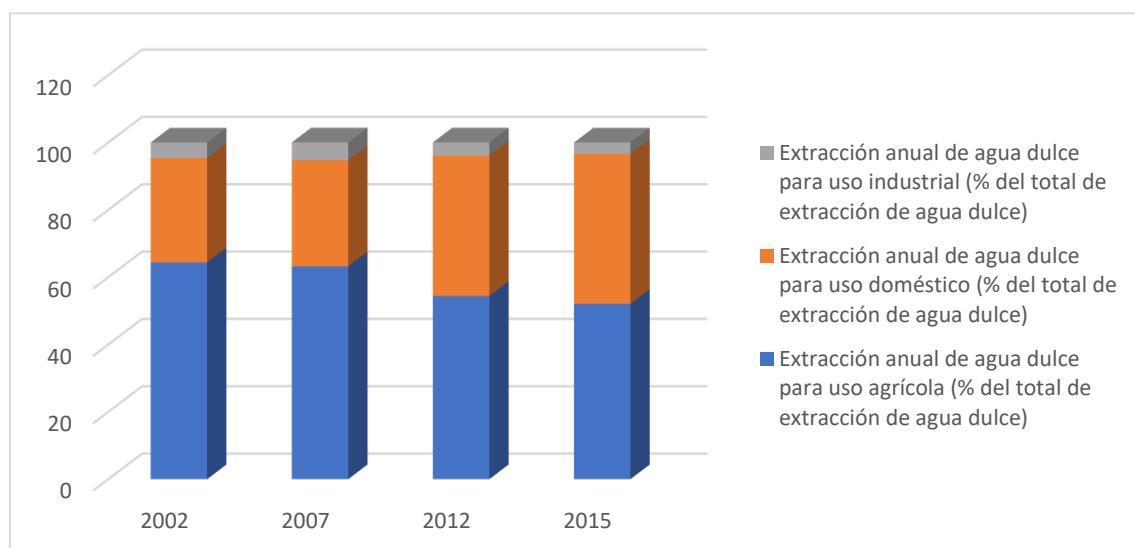


Fuente: Elaboración propia con datos de AQUASTAT de FAO.

2.3.3. La importancia de la gestión de recursos hídricos en países con escasez de agua.

El principal uso de agua sigue siendo la agricultura. No obstante, como se puede ver en el ejemplo de Jordania, el consumo doméstico va cobrando cada vez más importancia. Si la tendencia actual continua, en menos de 10 años la mayor parte de los recursos será destinada al uso doméstico. Por otro lado, el uso industrial constituye apenas 3,5% del total y va en decrecimiento.

Ilustración 6. Usos de agua dulce en Jordania (en %).



Fuente: Elaboración propia con datos de AQUASTAT de FAO.

Es reseñable que los recursos utilizados para la agricultura son muy difíciles de recuperar para su tratamiento y retorno. Si bien se pone mucho enfoque en el mal uso de agua en ámbito doméstico, cabe destacar que, aunque es muy importante reducir cualquier desperdicio, estos recursos son relativamente fáciles de recoger y tratar.

Como se ha indicado anteriormente, la escasez es un desajuste entre la demanda y la oferta y como tal se puede abordar por dos vías distintas, mejorando el almacenamiento y reutilización o aumentando la eficiencia en la gestión. La primera opción fue ampliamente

desarrollada en el pasado siglo, con la construcción de importantes presas en las principales cuencas, así como con la creación de infraestructuras más pequeñas para el riego. La tendencia actual se centra más en la reducción de las necesidades de uso de agua, disminución de sus pérdidas en los procesos industriales y un cambio en los modelos de agricultura para apostar por unos cultivos con más valor (Steduto, Hoogeveen, Winpenny, & Burke, 2017).

La eficiencia en el uso del agua puede ser lograda minimizando las pérdidas ocasionadas por las fugas, infiltraciones, evaporación no productiva o una tecnología obsoleta. Para conseguir este objetivo existe un abanico de alternativas que incluyen: revestimiento de canales de riego, mantenimiento de las tuberías y el alcantarillado, una mejor gestión del almacenamiento, uso de riego por goteo, instalación de dispositivos domésticos de conservación de agua, o desarrollo de sistemas para la reutilización.

Aunque los gobiernos de MENA han hecho importantes inversiones en estas tecnologías en los últimos años, todavía queda un gran margen de mejora. Por ejemplo, no se está aprovechando suficientemente el riego por goteo, siendo esta medida de gran utilidad en los países en los que la mayor parte de los recursos hídricos está destinada a la agricultura. Además, el desarrollo de las infraestructuras de recolección y tratamiento de aguas residuales está limitado por razones políticas y socioculturales (Jeuland, 2016).

El problema del desajuste entre la oferta y la demanda se puede resolver también con medidas económicas, es decir subiendo el precio de los recursos para forzar el desarrollo de tecnologías eficientes y una conciencia de la necesidad del ahorro. No obstante, esta vía es muchas veces vista como dudosa desde el punto de vista ético.

Las inversiones para el almacenamiento y el uso del agua, tales como plantas de tratamiento y desalinización, presas y redes de abastecimiento tienen un coste fijo muy alto. Por esta razón, su desarrollo se suele llevar a cabo desde el sector público. En los países en vías de desarrollo, como son la mayoría de los estados de MENA, la falta de unas instituciones sólidas hace difícil acometer estos proyectos. Históricamente, estas inversiones se realizaban con la ayuda del Banco Mundial y otros organismos de ayuda

al desarrollo, pero en los últimos años la disponibilidad de estos fondos es menor (Jeuland, 2016).

Cabe señalar que, a pesar de todas las dificultades, algunos países de la región han sido capaces de introducir políticas exitosas. Un ejemplo puede ser el progreso de Jordania, donde a pesar de que la precipitación haya disminuido en un 20% en las últimas ocho décadas, la proporción de agua dulce utilizada para la agricultura se redujo del 80% en la década de 1970 a alrededor del 60% en los últimos años. Asimismo, en 2014, se reutilizaron 125 m³ de aguas residuales tratadas, una cifra que se espera que alcance 240 m³ en 2025. Sin embargo, la buena planificación estratégica de Jordania resultó insuficiente debido a la afluencia masiva de refugiados, que ha aumentado la demanda total de agua en un 22% (Martens, 2017).

2.4. Sistemas informáticos para la predicción de datos meteorológicos.

El análisis de datos meteorológicos puede realizarse para hacer previsiones a corto plazo, sobre el tiempo en los próximos días o semanas, o para dar respuesta a cuestiones más generales sobre el clima y predecir las condiciones de temperatura y precipitaciones en un periodo más largo. En la literatura existen referencias sobre numerosos estudios acerca de este tema, basados tanto en modelos estadísticos como en *data mining*. A continuación, se citarán algunas de las propuestas realizadas.

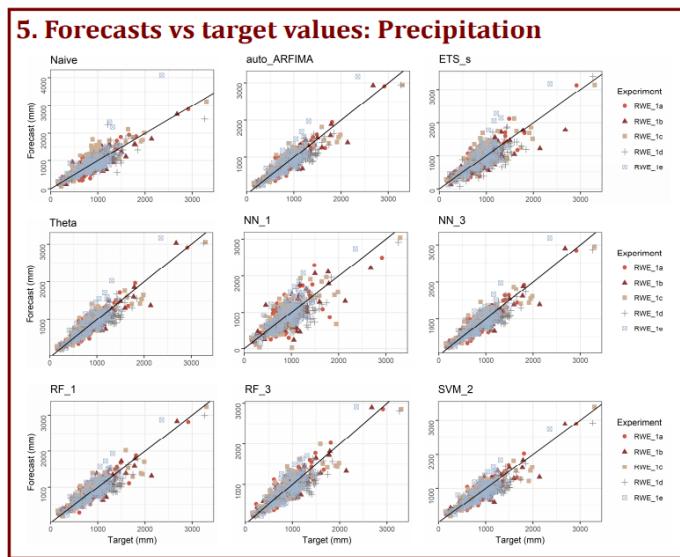
2.4.1. Análisis univariable de las precipitaciones y de la temperatura en el siguiente periodo anual- estudio comparativo de diferentes técnicas.

El año pasado se realizó en la Politécnica de Atenas un interesante trabajo, titulado “*One-step ahead forecasting of annual precipitation and temperature using univariate time series methods*”, en el que se estudió series históricas de precipitaciones y temperatura usando 16 métodos de predicción distintos (Papacharalampous, Tyralis, 2018). Los autores seleccionaron series de 50, 60, 70, 80 y 90 puntos de medición para hacer previsiones sobre el siguiente dato (respectivamente, el 51º, 61º, 71º, 81º y 91º).

El software usado en el proyecto se basaba en el lenguaje R. Las técnicas aplicadas se clasificaban en los siguientes grupos: simple, auto_ARFIMA, Exponential Smoothing, Redes neuronales, Bosque Aleatorio, Máquinas de Vectores Soporte. Los algoritmos utilizados se evaluaron haciendo uso de 7 medidas de exactitud: el error, el error absoluto, el error en porcentaje, el error absoluto en porcentaje, la mediana de errores absolutos, la mediana de errores absolutos en porcentaje y el coeficiente de la regresión lineal.

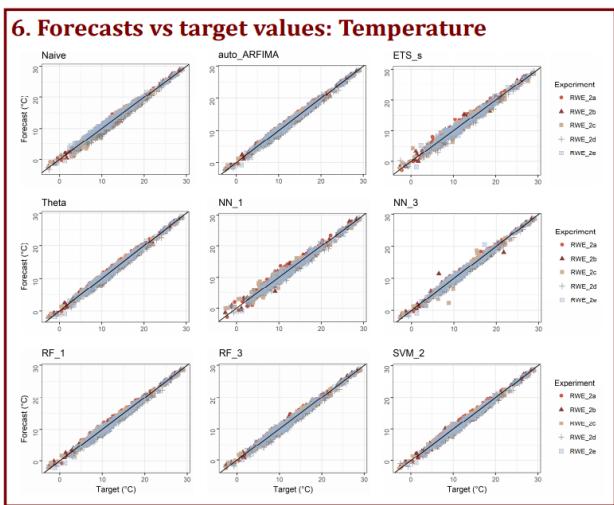
Los autores llegan a la conclusión que aunque algunos métodos de pronóstico funcionan generalmente mejor que otros, estos algoritmos también tienen fallos importantes en un número de casos demasiado grande para ser ignorado. Asimismo, como se puede observar en las siguientes gráficas, los errores de predicción de temperatura fueron menores que en el caso de las precipitaciones.

Ilustración 7. Comparación de previsión con datos reales de precipitaciones.



Fuente: Estudio “One-step ahead forecasting of annual precipitation and temperature using univariate time series methods”.

Ilustración 8. Comparación de previsión con datos reales de temperatura.



Fuente: Estudio “One-step ahead forecasting of annual precipitation and temperature using univariate time series methods”.

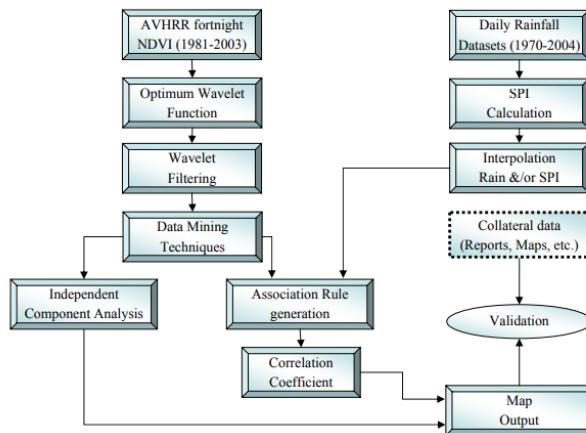
2.4.2. Data mining de datos espaciales y datos de precipitaciones para la observación de sequías.

Otro estudio reseñable sobre el tema fue el realizado por A. Sharma. En este trabajo se combinan datos de precipitaciones y datos geoespaciales para la detección de sequías (Sharma, 2006). El universo del proyecto es una región de India llamada Karnataka.

Los datos de precipitaciones cubren el periodo 1970-2004 y están extraídos de Drought Monitoring Cell. Por otro lado, el autor usa las series temporales de NOAA de índice de vegetación de diferencia normalizada (NDVI). Esta ratio, que toma valores de -1 a 1, describe el desarrollo de la vegetación basándose en la intensidad de la radiación del espectro electromagnético emitido por ella. Se ha demostrado que NDVI está relacionado con la capacidad fotosintética de las plantas (Sellers, 1985). Finalmente, en el proyecto se consideró también como variable la altitud del terreno de cada estación.

En la siguiente ilustración se puede observar la metodología del proyecto:

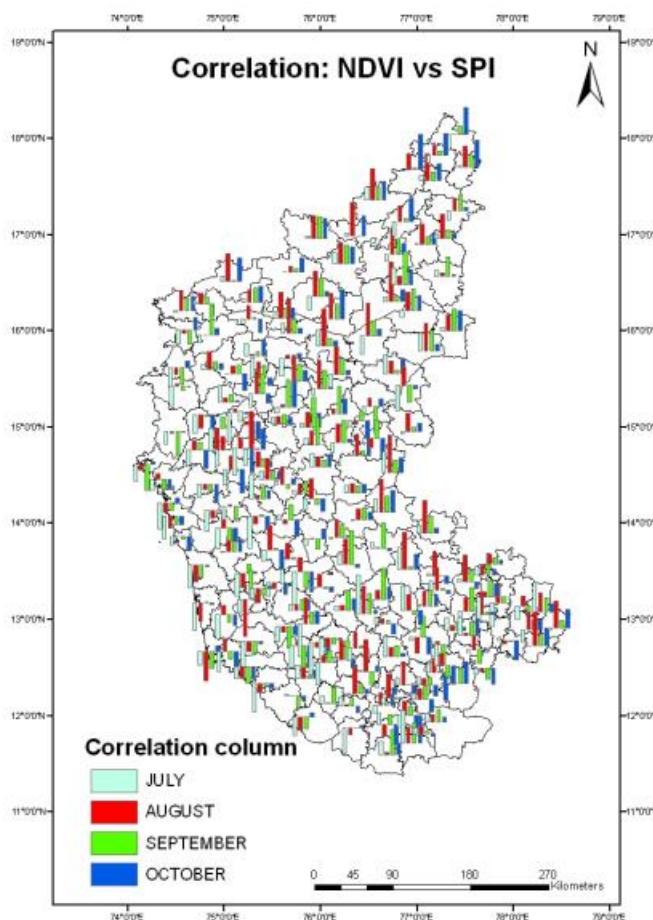
Ilustración 9. Metodología del proyecto “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”.



Fuente: Proyecto “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”

Los datos del índice NDVI se transforman en unos Wavelets y se filtran, para reducir el ruido y eliminar las redundancias. Luego, los datos de precipitaciones se discretizan usando la escala SPI. Se combina el estudio de ambas series con técnicas de asociación (algoritmo *a priori*) y, adicionalmente, se analiza los datos de NDVI a través del Análisis de Componentes Independientes. Se crea un mapa con los resultados obtenidos y se realiza la validación comparando con otras fuentes de información.

Ilustración 10. Resultados del estudio “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”.



Como se puede observar en la siguiente imagen, el autor del trabajo concluye que, aunque en general hay correlación positiva entre los dos tipos de índices, la regla de asociación no se cumple para la región costera. Eso se puede deber a que la vegetación en esta parte de Karnataka es muy densa y su color no presenta cambios tan repentinos como en el resto del estado, donde prevalecen los cultivos.

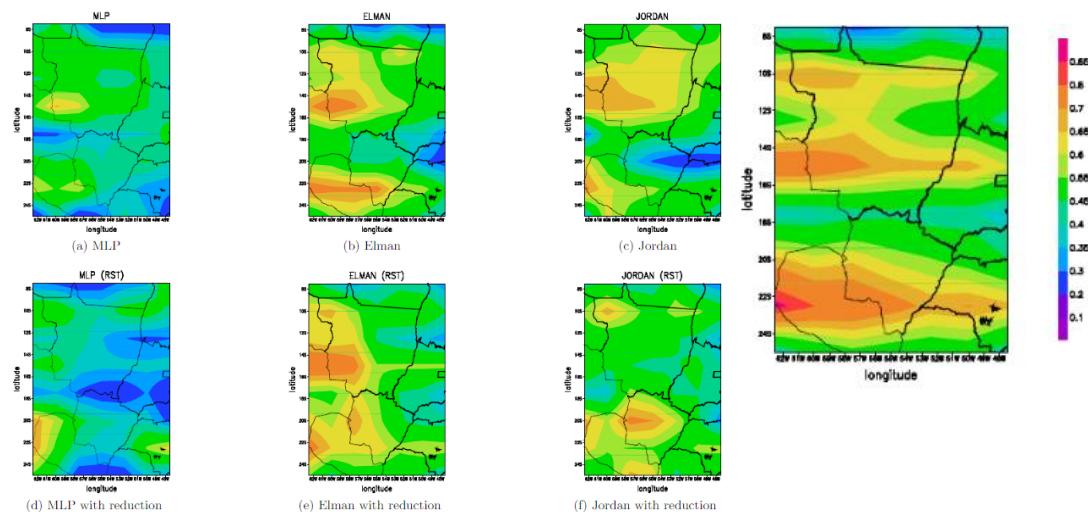
Fuente: Estudio “Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship”.

2.4.3. Data mining de datos meteorológicos con las técnicas de redes neuronales.

Otro estudio interesante fue conducido en el año 2015 en la región Centro-Oeste de Brasil. Los datos analizados fueron las medias mensuales de las precipitaciones comprendidas entre 1980 y 2009 (Anochia, 2015).

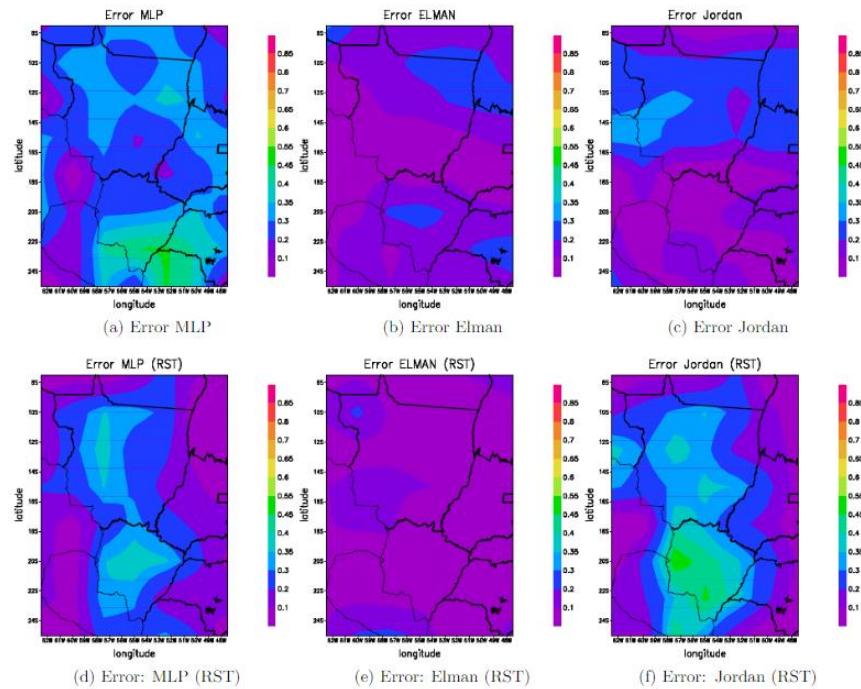
En este trabajo se estaba realizando predicciones con el uso de distintos algoritmos de redes neuronales: Perceptron Multicapa, la Red de Elman y la Red de Jordan. Además, los tres métodos se utilizaron en dos variantes: con los datos totales y con la aplicación de reducción de dimensionalidad. En ambas variantes el error cuadrático medio fue menor en la Red de Elman. Cabe destacar también que, si bien el Perceptron Multicapa se comportó mejor con la reducción de dimensionalidad, las redes de Elman y Jordan fueron más exactos con datos originales.

Ilustración 11. Predicciones realizadas sobre la precipitación en la región Centro-Oeste de Brasil en comparación con los datos originales.



Fuente: Estudio “*Meteorological data mining for climate precipitation prediction using neural networks*”.

Ilustración 12. Error cuadrático de las predicciones realizadas sobre la precipitación en la región Centro-Oeste de Brasil.



Fuente: Estudio “*Meteorological data mining for climate precipitation prediction using neural networks*”.

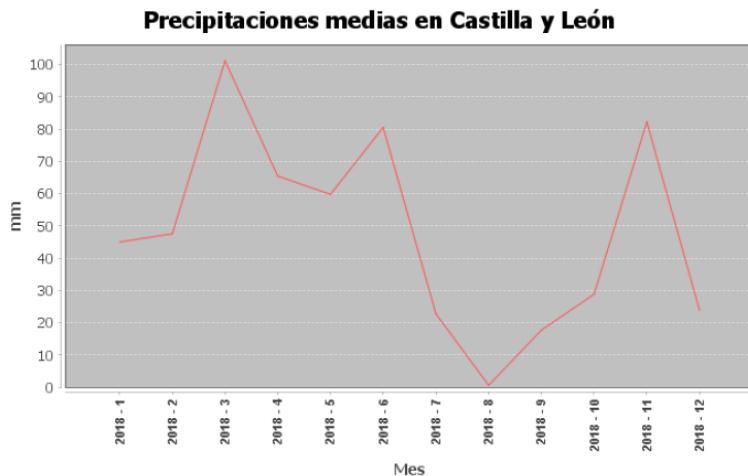
2.4.4. Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.

Finalmente se cita el proyecto que servirá de base de desarrollo del presente trabajo fin de grado. Se trata de un sistema que no solamente usa los modelos de *data mining*, sino que además facilita la carga, procesamiento y visualización de los datos de las precipitaciones en la región de Castilla y León de España (Hermosilla, 2019).

Para las tareas de manejo de esta gran cantidad de registros se ha utilizado el software gratuito MongoDB, junto con la aplicación Robo3T. El *front end* de la aplicación está desarrollado en el lenguaje Java y también se integran en él los algoritmos de *data mining* utilizados en un programa de software libre, WEKA.

Los datos de las precipitaciones se recogen desde la web de NOAA (National Oceanic and Atmospheric Administration de Estados Unidos). Se pueden descargar los registros del cualquier periodo disponible y filtrarlos para elegir los pertenecientes de las estaciones de Castilla y León. A continuación, el sistema permite calcular la media mensual de la región y visualizarla en un gráfico. Finalmente, se ofrecen cuatro algoritmos distintos (Gausian Processes, Linear Regression, ISMOreg y Multilayer Perceptron) para las tareas de análisis y predicción.

Ilustración 14. Ejemplo de visualización de los datos: serie temporal de las lluvias en Castilla y León el año 2018.



*Fuente: Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.
43*

Curso 2018-19 – Ingeniería Informática

Ilustración 13. Interfaz gráfica del proyecto “Sistema big data para mejorar los rendimientos



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

CAPÍTULO 3: PLANIFICACIÓN TEMPORAL Y COSTES

El trabajo se realiza en cuatro meses, siendo los primeros tres dedicados a la lectura sobre el tema, el diseño y el desarrollo de la aplicación, y la redacción del TFG, mientras que durante el último mes se preparará la presentación. Se estima que durante los primeros 90 días se dedicarán de media tres horas diarias a estas tareas, mientras que en el último mes se estipula una dedicación de 5 horas semanales como suficiente. De ahí que el tiempo total requerido es de 290 horas. En la página 43 se puede consultar la planificación temporal en detalle.

Por otro lado, el proyecto ha sido realizado por una estudiante de Ingeniería Informática, con experiencia en gestión de bases de datos y análisis de datos, con asesoramiento de su Tutor, Doctor en Ingeniería Informática. Además, se ha contado con apoyo puntual del autor de “Sistema big data para mejorar los rendimientos agrícolas en Castilla y León”, Francisco Javier Hermosilla. En un entorno real estas dedicaciones conllevarían los siguientes costes:

Tabla 2. Coste (Recursos Humanos).

	Coste horario	Dedicación	Total
Ingeniero Informático	20	290	5.800
Asesor: Doctor en Ingeniería Informática	50	30	1.500
Consultor: autor de software base	30	10	300
Total		330 (h)	7.600 (€)

Aparte de Recursos Humanos, el trabajo ha exigido también el uso de recursos tecnológicos que se detallan a continuación:

Tabla 3. Coste (Recursos técnicos)

Recurso	Coste
Ordenador	$1000*0,11=110$ € (se estima la amortización en 3 años)
Conexión a Internet	$40*4=160$ €
Microsoft Office	$10*4=40$ €
Disco duro externo	60 €
MongoDB	Software libre
Studio 3T	0 € (periodo de prueba)
NetBeans	Software libre
Star UML	0 € (periodo de prueba)
Total	370 €

En resumen, el coste total de proyecto sería el siguiente:

Tabla 4. Costes totales del proyecto.

Concepto	Precio
Recursos Humano	7.600 €
Recursos Técnicos	370 €
IVA	1673,7 €
Total	9643,7 €

Ilustración 15. Planificación temporal del TFG.

Tarea 1: Lectura de TFG de Francisco Javier Hermosilla.

Tarea 2: Instalación y familiarización con el software.

Tarea 3: Planificación temporal del proyecto.

Tarea 4: Lectura de fuentes y redacción de marco teórico: parte técnica.

Tarea 5: Lectura de fuentes y redacción de marco teórico: parte específica.

Tarea 6: Pruebas de la aplicación existente.

Tarea 7: Diseño de la ampliación.

Tarea 8: Desarrollo de aplicación: carga y visualización de los datos.

Tarea 9: Pruebas funcionales: carga y visualización de los datos.

Tarea 10: Redacción de TFG: planificación y propuesta (1^a parte).

Tarea 11: Desarrollo de aplicación: tareas de data mining.

Tarea 12: Pruebas funcionales de aplicación: tareas de data mining.

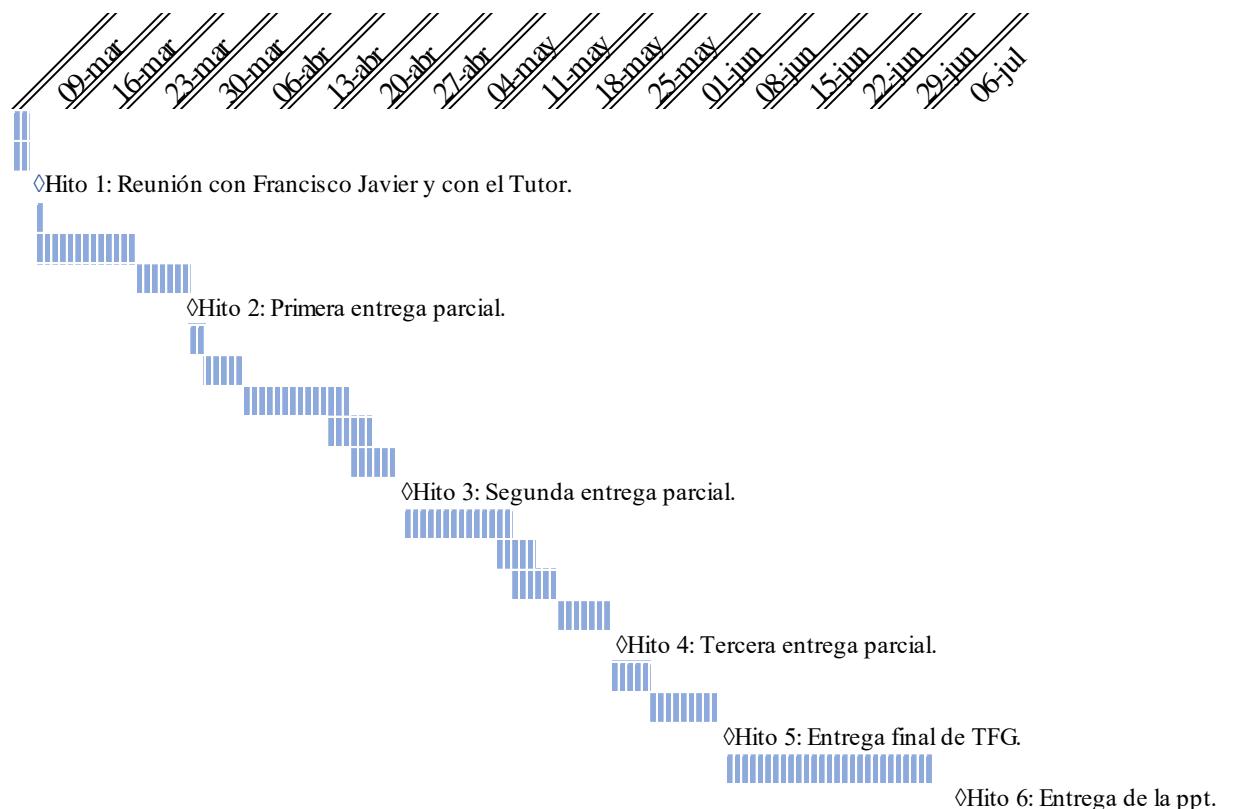
Tarea 10: Redacción de TFG: propuesta (segunda parte).

Tarea 11: Redacción de TFG: conclusiones y evaluación.

Tarea 12: Redacción de TFG: resumen, palabras claves.

Tarea 13: Revisión de TFG.

Tarea 14: Preparación de la presentación final.



CAPÍTULO 4: PROPUESTA

4.1. Diseño de la ampliación de la aplicación.

Las tareas de desarrollo se pueden clasificar en dos grupos: las que modifican los casos de uso existentes y los que crean casos de uso nuevos. En primer lugar, se presentarán en una tabla los casos de uso modificados.

Tabla 5. Modificaciones de casos de uso.

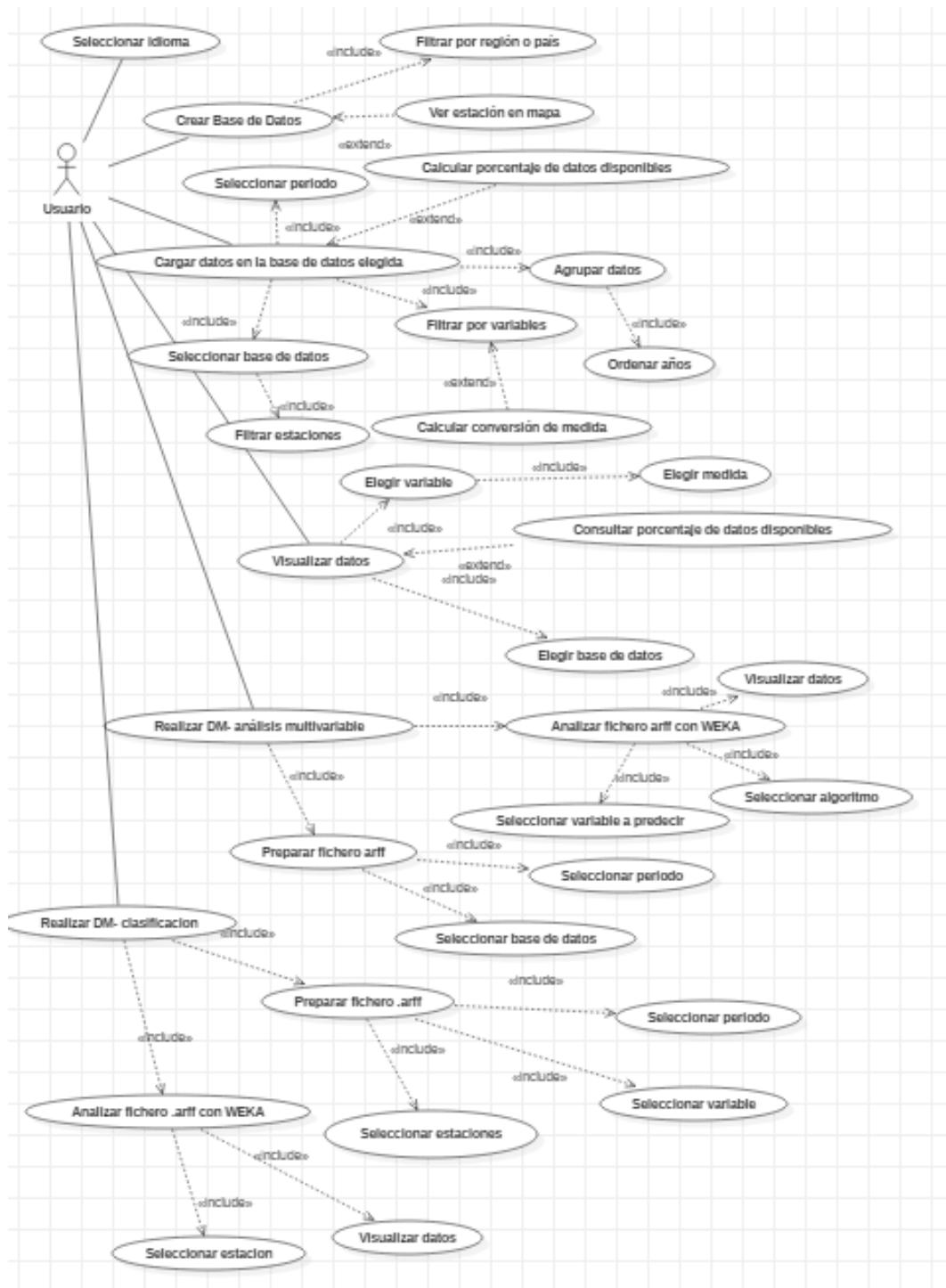
Caso de uso	Modificaciones
Descarga fichero datos aaaa.csv.gz	El sistema indicará a través de pantallas emergentes las tareas a realizar por el usuario.
Descomprimir aaaa.csv	Se permitirá elegir libremente el directorio para guardar el archivo descomprimido. El sistema indicará a través de pantallas emergentes las tareas a realizar por el usuario.
Import mongoDB	El sistema indicará a través de pantallas emergentes las tareas a realizar por el usuario. El sistema presentará el progreso de carga a través de una barra de progreso.
Ver evolución import	Se eliminará al permitir observar el progreso a través de la barra de progreso.
Refrescar lista colecciones	Se realizará automáticamente por el sistema sin necesidad de actuación del usuario.

Caso de uso	Modificaciones
Filtrar estaciones CyL	Se eliminará y se sustituirá por un caso de uso más amplio (“Filtrar estaciones”).
Filtrar precipitaciones mensuales	Se eliminará y se sustituirá por un caso de uso más amplio (“Filtrar por variables”).
Genera gráfica	Se eliminará y se sustituirá por un caso de uso más amplio (“Visualizar datos”).
Crear archivo arff	<p>Se permitirá crear archivo arff de diferentes bases de datos y de diferentes variables.</p> <p>Se permitirá elegir libremente el directorio para la descarga.</p> <p>El sistema indicará a través de pantallas emergentes las tareas a realizar por el usuario.</p>
Borrar tabla	Sin modificaciones
Data mining	Se permitirá elegir la variable a predecir. Se modificarán los algoritmos para eliminar valores negativos de predicciones de precipitaciones.

En conclusión, en las tareas iniciales (descargar, descomprimir e importar datos en MongoDB) se introducen modificaciones menores, mientras que en lo relativo al filtrado, visualización y análisis de datos, los cambios son muy significativos. De esta forma se cumple con el objetivo de ampliar la utilidad de la aplicación, aprovechando al mismo tiempo el código existente.

En la siguiente ilustración se pueden observar los nuevos casos de uso creados:

Ilustración 16. Nuevos casos de uso.

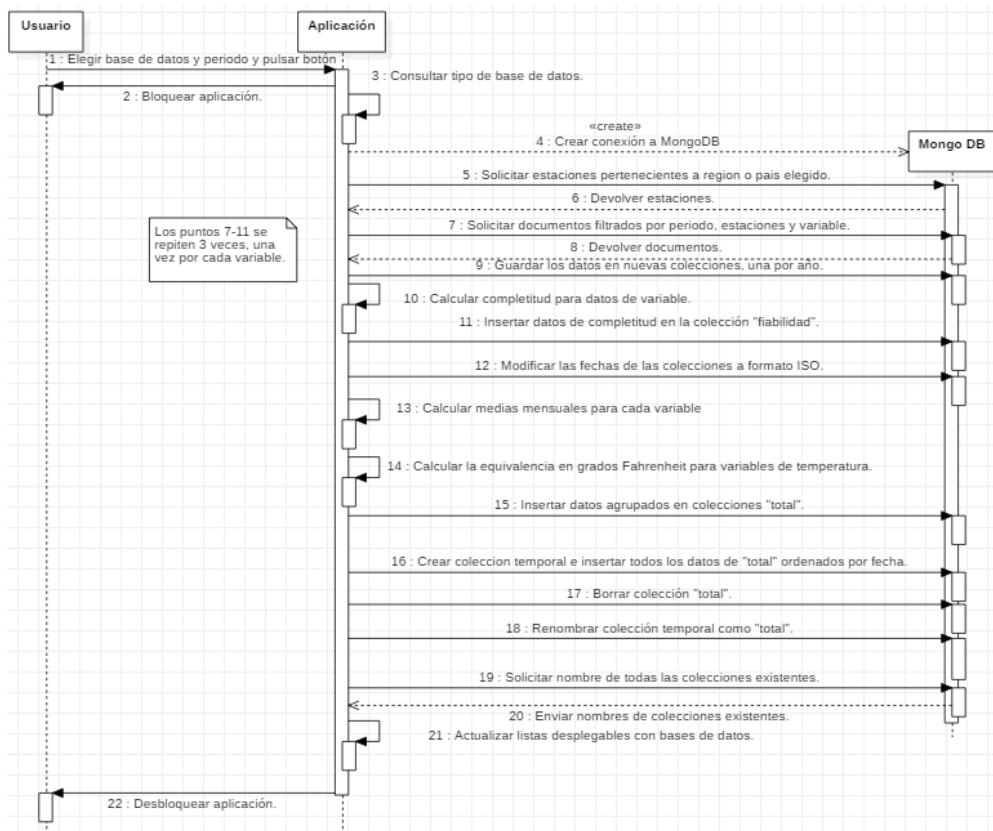


A continuación, se describen los pasos principales de los casos de uso añadidos. Entre paréntesis se identifica el agente que realiza la acción (usuario/aplicación/MongoDB). Adicionalmente, en casos de uso especialmente complejos, se ilustrará el proceso a través de un diagrama de secuencia:

- Seleccionar idioma:
 1. Iniciar aplicación (usuario).
 2. Iniciar el bucle que presenta al usuario el desplegable de selección de idioma siempre que el usuario pulse “cancelar” en vez de “aceptar” (aplicación).
 3. Seleccionar idioma en el desplegable (usuario).
 4. Aceptar (usuario).
 5. Leer selección del desplegable (aplicación).
 6. Seleccionar el archivo de propiedades que se va a utilizar (aplicación).
- Crear Base de Datos:
 1. Elegir región / Elegir país /Elegir Estación (usuario).
 2. Llamar a MongoDB para recuperar los países pertenecientes a la región /las estaciones pertenecientes al país (aplicación).
 3. Devolver los datos solicitados (Mongo DB).
 4. Filtrar las listas desplegables (aplicación).
 5. Pulsar botón para crear la base de datos de la región/ pulsar botón para crear la base de datos del país/ pulsar botón para crear la base de datos de la estación (usuario).
 6. Llamar MongoDB (aplicación).
 7. Crear base de datos con 3 colecciones vacías para almacenar los datos agrupados de precipitaciones/ temperatura mínima/ temperatura máxima (MongoDB).
 8. Filtrar listas desplegables que almacenan bases de datos creadas (aplicación).

- Ver estación en mapa:
 1. Elegir estación (usuario).
 2. Leer la selección de usuario y consultar MongoDB para devolver los datos de latitud y longitud (aplicación).
 3. Devolver los datos solicitados (MongoDB).
 4. Abrir el navegador en la aplicación Google Maps centrada en el punto según los datos de latitud y longitud devueltos (aplicación).

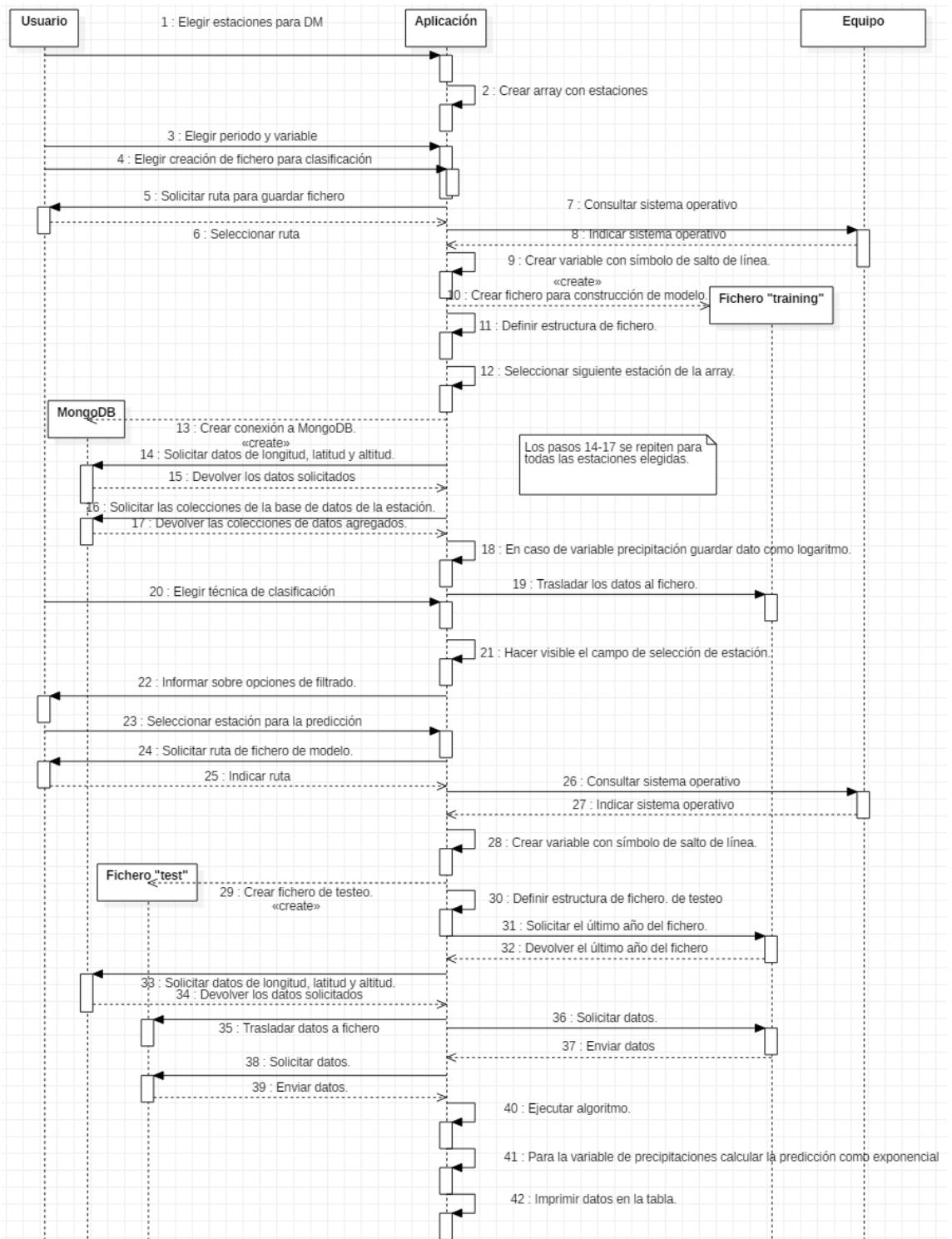
Ilustración 17. Diagrama de secuencia: caso de uso “Cargar los datos en la base de datos elegida”.



- Cargar los datos en la base de datos elegida:
 1. Seleccionar base de datos y año (usuario).
 2. Pulsar el botón de actualización (usuario).
 3. Bloquear la aplicación (aplicación).

4. Leer la selección de usuario y comprobar si la base de datos es de región, país o estación (aplicación).
5. Crear conexión a MongoDB (aplicación).
6. Solicitar las estaciones pertenecientes a región/país (aplicación).
7. Devolver las estaciones pertenecientes a región/país (MongoDB).
8. Solicitar documentos filtrados por periodo, estaciones y variable (aplicación).
9. Devolver documentos (MongoDB).
10. Guardar los documentos en nuevas colecciones, una por año (aplicación).
11. Calcular el número de documentos con datos completados para el periodo solicitado (aplicación).
12. Calcular el porcentaje de datos completados (aplicación).
13. Guardar el dato de completitud en la base de datos “fiabilidad” (aplicación).
14. Repetir los puntos 7-13 para la segunda variable.
15. Repetir los puntos 7-13 para la tercera variable.
16. Modificar las fechas de las colecciones a formato ISO (aplicación).
17. Calcular la media mensual (aplicación).
18. Convertir los grados Celsius a grados Fahrenheit (aplicación).
19. Guardar los datos en un array. En caso de no haber ningún dato para un mes guardar el símbolo “?” (aplicación).
20. Insertar los datos en la colección de datos agregados (aplicación).
21. Crear colección temporal (aplicación).
22. Guardar en la colección temporal la consulta de datos de colección de datos agregados ordenados por fecha de forma ascendente (aplicación).
23. Borrar la colección de datos agregados (aplicación).
24. Renombrar la colección ordenada (aplicación).
25. Solicitar a MongoDB nombre de todas las colecciones existentes (aplicación).
26. Enviar nombres de colecciones existentes (MongoDB).
27. Actualizar listas desplegables con bases de datos (aplicación).
28. Desbloquear aplicación (aplicación).

Ilustración 18. Diagrama de secuencia: caso de uso “Realizar DM- clasificación”.



- Realizar DM- clasificación:
 1. Elegir las estaciones para el análisis (usuario).
 2. Enviar solicitud (usuario).
 3. Leer solicitud y guardar las estaciones elegidas en un array (aplicación).
 4. Elegir periodo y variable (usuario).
 5. Marcar la opción de crear fichero .arff para clasificación (usuario).
 6. Enviar solicitud (usuario).
 7. Leer selección del usuario (aplicación).
 8. Comprobar si el usuario ha elegido estaciones previamente y de no ser así solicitar que el usuario vuelva al paso 4 (aplicación).
 9. Solicitar al usuario la ruta para guardar fichero (aplicación).
 10. Elegir ruta para guardar fichero (usuario).
 11. Leer selección del usuario (aplicación).
 12. Leer información sobre sistema operativo (aplicación).
 13. Guardar carácter de salto de línea adecuado para el sistema operativo: \r\n o \n (aplicación).
 14. Crear un nuevo archivo en la ruta indicada (aplicación).
 15. Definir estructura de fichero .arff (aplicación).
 16. Empezar bucle.
 - 16a. Seleccionar siguiente estación de array (aplicación).
 - 16b. Consultar MongoDB para recibir datos de latitud, longitud y altitud de estación (aplicación).
 - 16.c Enviar datos solicitados de la estación.
 - 16d. Solicitar acceso a base de datos de la estación correspondiente de MongoDB (aplicación).
 - 16e. Devolver las colecciones de datos agregados de la base de datos (MongoDB).
 - 16f. Para la variable de precipitaciones guardar el dato como logaritmo natural (aplicación).

- 16g. Trasladar los datos de latitud, longitud y altitud de estación y los datos agregados de variable elegida al fichero .arff (aplicación).
17. Finalizar bucle.
 18. Seleccionar la técnica de DM de clasificación (usuario).
 19. Enviar solicitud (usuario).
 20. Leer selección del usuario (aplicación).
 21. Hacer visible el campo específico de esta técnica: selección de estación para predicción (aplicación).
 22. Informar al usuario por medio de una ventana emergente que puede usar las listas desplegables de creación de bases de datos para filtrar la estación (aplicación).
 23. Aplicar filtros y seleccionar estación para la predicción (usuario).
 24. Enviar solicitud (usuario).
 25. Solicitar al usuario la ruta para abrir fichero (aplicación).
 26. Elegir ruta para abrir fichero (usuario).
 27. Leer selección del usuario (aplicación).
 28. Leer información sobre sistema operativo (aplicación).
 29. Guardar carácter de salto de línea adecuado para el sistema operativo: \r\n o \n (aplicación).
 30. Crear un nuevo archivo en la ruta indicada para el testeo (aplicación).
 31. Definir estructura de fichero .arff (aplicación).
 32. Leer el último año de fichero .arff de modelo (aplicación).
 33. Consultar MongoDB para recibir datos de latitud, longitud y altitud de estación (aplicación).
 34. Enviar datos solicitados de la estación.
 35. Trasladar los datos de latitud, longitud y altitud de estación, el año siguiente al último de modelo, los doce meses y el carácter "?" en lugar de variable al fichero .arff de testeo (aplicación).
 36. Ejecutar el algoritmo seleccionado sobre los dos ficheros de modelo y de prueba (aplicación).

- 37. Para la variable de precipitaciones calcular la predicción como exponencial (aplicación).
- 38. Guardar la predicción de los 12 meses siguientes en un array (aplicación).
- 39. Trasladar los datos a la tabla de resultados (aplicación).

- Realizar DM- análisis multivariable:
 1. Elegir periodo y base de datos (usuario).
 2. Marcar la opción de crear fichero .arff para series temporales (usuario).
 3. Enviar solicitud (usuario).
 4. Leer selección del usuario (aplicación).
 5. Solicitar al usuario la ruta para guardar fichero (aplicación).
 6. Elegir ruta para guardar fichero (usuario).
 7. Leer selección del usuario (aplicación).
 8. Leer información sobre sistema operativo (aplicación).
 9. Guardar carácter de salto de línea adecuado para el sistema operativo (aplicación).
 10. Crear un nuevo archivo en la ruta indicada (aplicación)
 11. Definir estructura del fichero .arff (aplicación).
 12. Solicitar acceso a BD correspondiente de MongoDB (aplicación).
 13. Devolver colecciones de datos agregados de la base de datos (MongoDB).
 14. Para la variable de precipitaciones guardar el dato como logaritmo natural (aplicación).
 15. Trasladar los datos de las colecciones al fichero .arff (aplicación).
 16. Seleccionar la técnica de *data mining* de análisis multivariable (usuario).
 17. Enviar solicitud (usuario).
 18. Leer selección del usuario (aplicación).
 19. Hacer visibles los campos específicos de esta técnica: selección de variable y de algoritmo (aplicación).
 20. Seleccionar algoritmo y variable (usuario).

21. Enviar solicitud (usuario).
 22. Solicitar al usuario la ruta para abrir fichero (aplicación).
 23. Elegir ruta para abrir fichero (usuario).
 24. Leer selección del usuario (aplicación).
 25. Ejecutar el algoritmo seleccionado sobre el fichero (aplicación).
 26. Para la variable de precipitaciones calcular la predicción como exponencial (aplicación).
 27. Guardar la predicción de los 12 meses siguientes en un array (aplicación).
 28. Trasladar los datos a la tabla de resultados (aplicación).
- Visualizar datos:
 1. Elegir periodo, base de datos y variable (usuario).
 2. Marcar la opción de consultar gráfica (usuario).
 3. Enviar solicitud (usuario).
 4. Leer selección del usuario (aplicación).
 5. Solicitar el porcentaje de completitud de dato (aplicación).
 6. Devolver el porcentaje de completitud de dato (MongoDB).
 7. Informar sobre la completitud del dato o sobre su falta de carga al usuario por medio de una ventana emergente (aplicación).
 8. Solicitar los datos agregados de periodo, variable y región/país/estación elegida (aplicación).
 9. Devolver los datos agregados (MongoDB).
 10. Presentar los datos en un gráfico (aplicación).

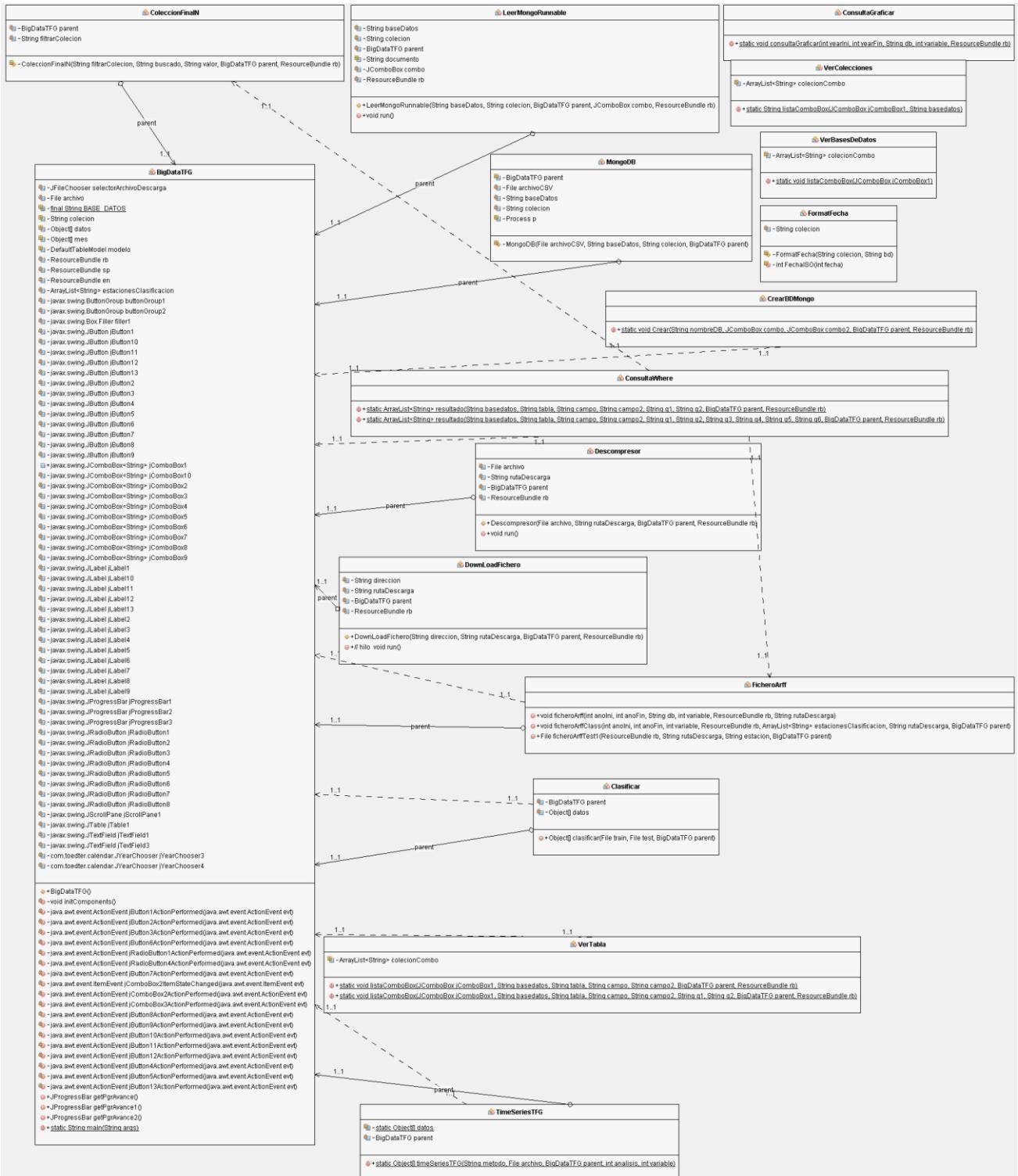
La clase principal de la aplicación, BigDataTFG, es la que contiene la interfaz gráfica de la aplicación y desde la que se llama a las demás clases y sus métodos para realizar las siguientes tareas:

- Descargar ficheros de la web de NOAA (clase DownloadFichero).
- Descomprimir ficheros descargados (clase Descompresor).
- Subir los datos a MongoDB (clase MongoDB).

- Ver progreso de la subida de datos a MongoDB (clase LeerMongoRunnable).
- Crear una base de datos en MongoDB de la región/ el país/ la estación elegida (clase CrearBDMongo).
- Subir los datos detallados y agregados a la base de datos de la región/ el país/ la estación elegida (clase ColeccionFinalN).
- Actualizar una lista desplegable con los nombres de todas las bases de datos creadas (clase VerBasesDeDatos).
- Actualizar una lista desplegable con los nombres de todas las colecciones de una base de datos (clase VerColecciones).
- Realizar una consulta a una base de datos y una colección determinada con uno o tres criterios (clase ConsultaWhere).
- Actualizar una lista desplegable con resultados de una consulta a una base de datos y una colección determinada (clase VerTabla).
- Cambiar formato de fecha a formato JSON (clase FormatFecha).
- Visualizar los datos (clase ConsultaGraficar).
- Crear los ficheros necesarios para las tareas de *data mining* (clase FicheroArff).
- Ejecutar los algoritmos de *data mining* de series temporales (clase TimeSeriesTFG).
- Ejecutar el algoritmo de *data mining* de clasificación (clase Clasificar).

En la siguiente ilustración se puede consultar el diagrama de las clases creadas en la aplicación:

Ilustración 19. Diagrama de clases de la aplicación.



4.2. Internacionalización de la aplicación.

4.2.1. Selección del idioma.

El primer objetivo de la ampliación de la aplicación consiste en permitir al usuario seleccionar el idioma en el que desea visualizarla. Para conseguirlo he utilizado la clase de java ResourceBundle y su método “getString”. El funcionamiento de esta clase consiste en la creación de ficheros de propiedades en las que se definen tuplas de clave-valor. El método “getString” accede a la clave de fichero indicado y recupera el valor. De esta forma, al ejecutar la aplicación se deja al usuario elegir el idioma con la siguiente pantalla emergente:

Ilustración 20. Ventana emergente para la selección del idioma.



Ilustración 21. Aplicación en español.

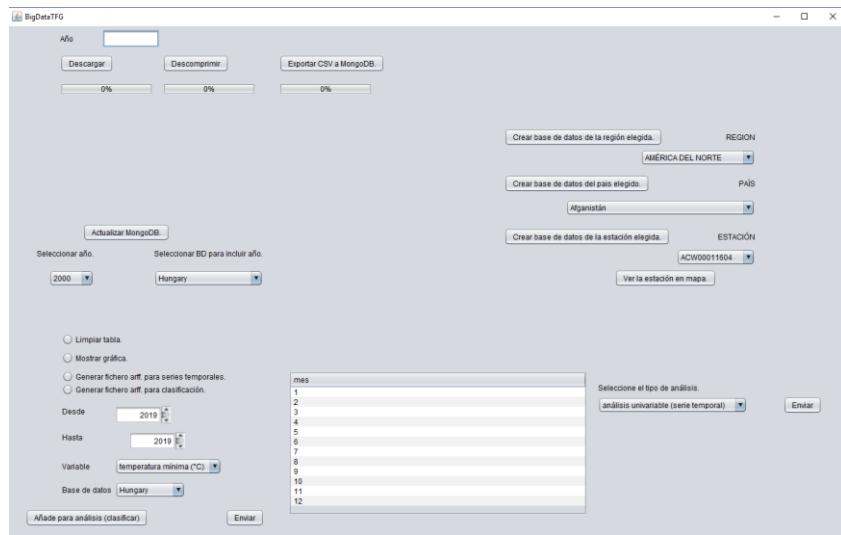
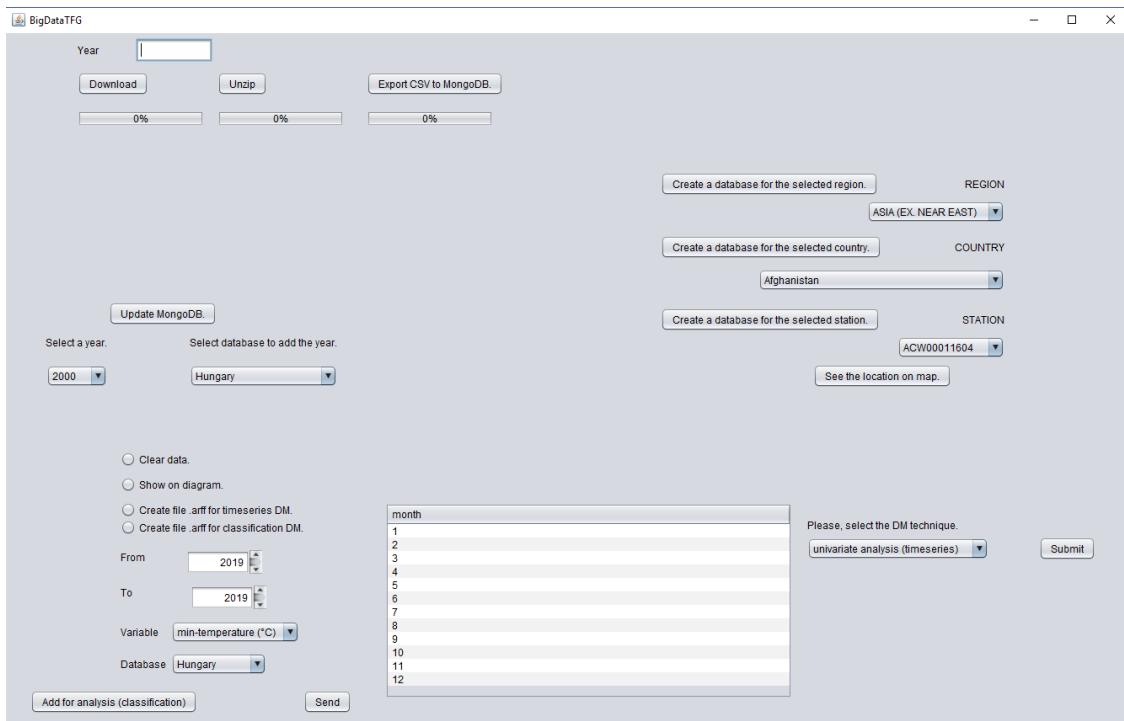


Ilustración 22. Aplicación en inglés.



La aplicación recupera la respuesta del usuario y accede al archivo de propiedades relacionado para traducir los elementos visuales y cualquier comunicación con el usuario. El código relacionado con esta funcionalidad es el siguiente:

```
String lang="en";
```

```
String country ="UK";
```

```
Locale e=new Locale (lang,country);
```

```
this.en= ResourceBundle.getBundle("bigdatatfg/english",e);
```

```
String lang2="sp";
```

```
String country2 ="SP";
```

```
Locale s=new Locale (lang2,country2);
```

```

this.sp=ResourceBundle.getBundle("bigdatatfg/spanish",s);

this.rb=en;

String[] values = {"Spanish", "English"};

boolean noRespondido=true;

String lenguajeElegido=new String();

while (noRespondido)

{Object selected = JOptionPane.showInputDialog(null, "Please select your
language", "Selection", JOptionPane.DEFAULT_OPTION, null, values, "0");

if (selected !=null)

{

lenguajeElegido = selected.toString();

if("Spanish".equals(lenguajeElegido))

{rb=sp;

noRespondido=false;}

else if ("English".equals(lenguajeElegido))

{rb=en;

noRespondido=false;}

}

}

```

Por otro lado, se incluyen algunos ejemplos del uso del método “getString”:

- En un visual: jLabel1.setText(rb.getString("año"));
- En otro método:

```
VerTabla.listaComboBox(jComboBox2,"opciones","regiones",rb.getString("region"),rb.getString("$region"));
```
- En una tabla: modelo.addColumn(rb.getString("mes"),mes);
- En una ventana emergente: JOptionPane.showMessageDialog(this, rb.getString("rutaDescarga"), "",JOptionPane.INFORMATION_MESSAGE);

Si bien, para la mayoría de las funcionalidades, la clase ResourceBundle ha sido una solución sencilla y eficiente, en el caso de las listas desplegables de selección de países y regiones he decidido optar por realizar llamadas a colecciones de MongoDB, dado que se trata de una cantidad de opciones considerable que haría difícil un mantenimiento ordenado de los ficheros de propiedades. Este punto será abordado en el párrafo dedicado al filtrado de los datos.

4.2.2. Inclusión de estaciones internacionales en la Base de Datos.

El proyecto de Francisco Javier Hermosilla recogía solamente los datos de las estaciones de Catilla y León (España). El objetivo de este proyecto es permitir que los usuarios creen las bases de datos de cualquier región y posteriormente visualicen y analicen la información relativa.

Como primer paso, he descargado los metadatos relativos (código, longitud, latitud, altitud) de todas las estaciones disponibles a través del siguiente enlace de la web de NOAA: <https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/ghcnd-stations.txt>

Los dos primeros dígitos de cada estación son el código FIPS de la región. Para agruparlos según el país al que pertenecen, se descarga la tabla de equivalencia entre códigos FIPS, y los códigos internacionales (ISO) de los países.

Se crea una tabla en MongoDB con el nombre de la estación, código internacional del país (ISO), código FIPS, nombre del país en español, nombre de país en inglés, nombre de la región en inglés, nombre de la región en español y los datos de longitud, latitud y altitud. Esta colección permitirá seleccionar al usuario qué tipo de base de datos quiere crear (a nivel regional, estatal o de estación) y servirá como fuente para realizar consultas y pasar parámetros de funciones en la aplicación.

Ilustración 23. Colección creada en MongoDB con los datos de las estaciones.

```

_id: ObjectId("5cab96ccb05d6d37d8b39bad")
estacion: "ACW00011604"
fips: "AC"
iso: "AG"
country: "Antigua and Barbuda"
pais: "Antigua y Barbuda"
latitud: 171167
longitud: -617833
altitud: 10
region: "LATIN AMER. & CARIB"
region_es: "AMÉRICA LATINA Y CARIBE"

_id: ObjectId("5cab96ccb05d6d37d8b39bae")
estacion: "ACW00011647"
fips: "AC"
iso: "AG"
country: "Antigua and Barbuda"
pais: "Antigua y Barbuda"
latitud: 171333
longitud: -617833
altitud: 19
region: "LATIN AMER. & CARIB"
region_es: "AMÉRICA LATINA Y CARIBE"

_id: ObjectId("5cab96ccb05d6d37d8b39baf")
estacion: "AE000041196"
fips: "AE"
iso: "AE"
country: "United Arab Emirates"
pais: "Emiratos Árabes Unidos"
latitud: 253330
longitud: 555170
altitud: 34
region: "NEAR EAST"
region_es: "PRÓXIMO ESTE"

```

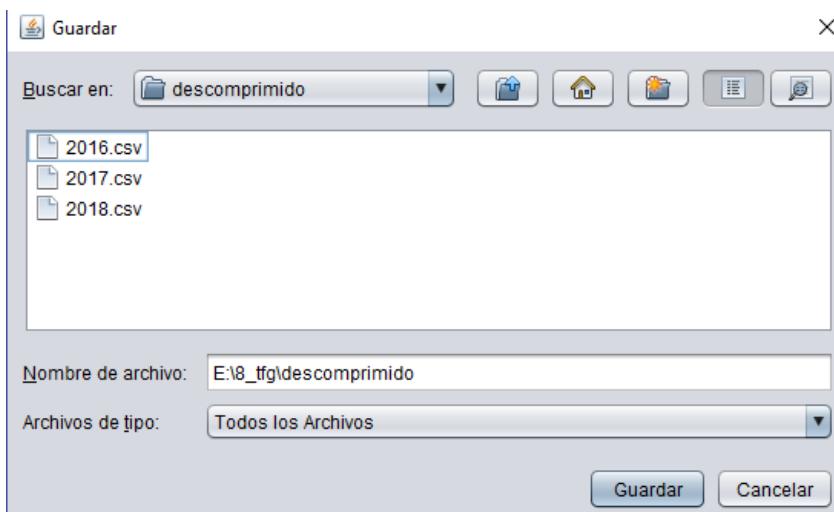
4.3. Mejoras de usabilidad en las tareas de descarga, descompresión e importación.

4.3.1. Portabilidad de la aplicación.

Para mejorar la portabilidad de la aplicación, se ha modificado la ruta absoluta (String rutaDescarga = "D:/tfg/tfg_descomprimido/"+nombreArchivoSinExtension;) para guardar el archivo descomprimido por el siguiente código:

```
JOptionPane.showMessageDialog(this, rb.getString("rutaDescomprimido"), "",  
JOptionPane.INFORMATION_MESSAGE);  
  
int seleccion = selectorArchivos2.showSaveDialog(this);  
  
if (seleccion == JFileChooser.APPROVE_OPTION)  
  
{  
  
    String fichero = selectorArchivos2.getSelectedFile().getPath();  
  
    String rutaDescarga = fichero+nombreArchivoSinExtension;  
  
    ...}
```

Ilustración 24. Ventana emergente para selección de directorio.



De esta forma no es necesario que los archivos se guarden siempre en el mismo directorio, se deja esta cuestión a elección del usuario. De hecho, en la anterior solución se corría el riesgo de tener que obligar al usuario a crear una nueva partición en caso de no tener un directorio en “D:\”. Una solución similar ha sido aplicada para permitir al usuario elegir las rutas de guardado y acceso de los ficheros .arff.

Por otro lado, en las tareas de escritura de fichero, se ha detectado un problema relacionado con la falta de universalidad del símbolo de salto de línea en distintos sistemas operativos. Este símbolo en Windows se expresa como “\r\n” mientras que en Linux e IOS es “\n”. Por lo tanto, la aplicación lee el sistema operativo y determina el carácter adecuado a usar. Esta operación se realiza a través del siguiente código:

```
String sSistemaOperativo = System.getProperty("os.name");

String so=sSistemaOperativo.substring(0,3);

String saltoLinea="\n";

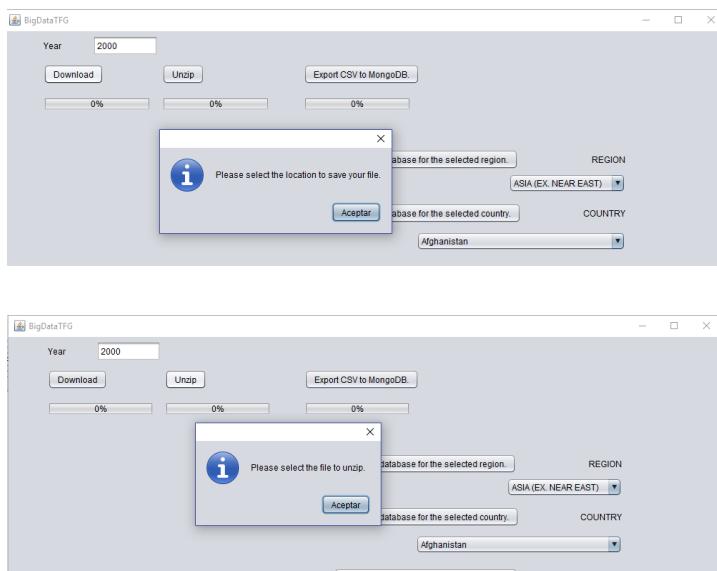
if(so.equals("Win"))

    saltoLinea="\r\n";
```

4.3.2. Elementos del apoyo para guiar al usuario en la realización de las tareas.

Para facilitar el uso de la aplicación por un usuario novel, se ha realizado unas modificaciones para guiarlo en la realización de las tareas. En primer lugar, cuando el sistema abre el directorio de archivos para elegir ruta de guardado o un archivo a utilizar en un proceso, se informa, a través de una ventana emergente, de la tarea que se espera que realice el usuario.

Ilustración 25. Ejemplos de ventanas emergentes.



En segundo lugar, en la versión inicial de la aplicación, para comprobar el estado de importación de datos en MongoDB, el usuario tenía que hacer consultas recursivas para conocer el número de documentos guardados. En el momento en el que este número dejaba de crecer el usuario podría suponer que la carga había finalizado.

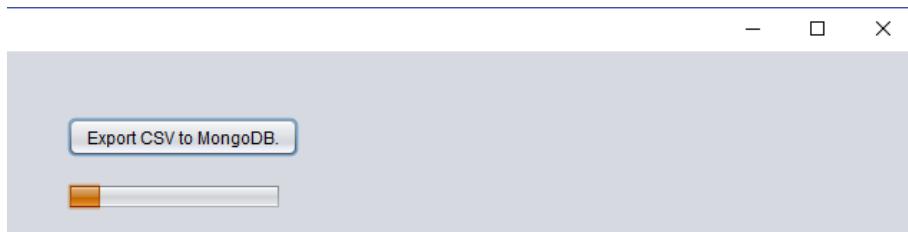
Ilustración 26. Documentos almacenados en la base de datos tfg (solución anterior).



Fuente: Hermosilla Moreno, F. J. (2019). *Sistema big data para mejorar los rendimientos agrícolas en Castilla y León*.

Si bien esta solución cumple con la funcionalidad prevista (controlar la finalización de la carga), parece poco intuitiva. Por esta razón, se ha decidido sustituir este mecanismo por una barra de progreso.

Ilustración 27. Barra de progreso para la importación de datos en MongoDB.



Con el objetivo de conseguir esta funcionalidad se ha partido de la clase existente LeerMongo para crear la clase LeerMongoRunnable. Es ahora el sistema que realiza consultas recurrentes cada 10 segundos y cuando dos consultas consecutivas tienen como resultado el mismo número la barra de progreso se llena y aparece mensaje de completitud de la carga. Además, dado que las colecciones tienen aproximadamente 35.000-45.000 documentos, se rellena la barra de progreso en función de la división entre el número de documentos devuelto por la consulta y el máximo de este intervalo.

La parte central de la clase responsable de ejecutar la tarea de actualizar la barra de progreso se encuentra en el siguiente bucle:

```
//se comparan las últimas dos consultas  
  
while (!(documento.equals(documentoTemp)))  
  
{ //se guarda en documentoTemp la consulta anterior  
  
documentoTemp=documento;  
  
wait(10000);  
  
contadorDocumento = collection.countDocuments();  
  
//se rellena la barra de progreso en función de número de documentos  
  
Long progreso=(contadorDocumento/1000);
```

```

String progresoS = Long.toString(progreso);

int progresoI=Integer.parseInt(progresoS);

//se guarda en documento la consulta actual

documento = Long.toString(contadorDocumento);

parent.getPgrAvance2().setValue(progresoI); }
```

Se puede consultar el código completo de la clase en los anexos.

Adicionalmente, se aprovecha esta clase para actualizar de manera automática el Combo Box con los años cargados. En la aplicación origen esta tarea tenía que ser realizada por el usuario a través del botón “Filtrado & Refresco” que cumplía diferentes funciones dependiendo del elemento elegido en la lista desplegable. He considerado que evitar las ambigüedades en el uso de este botón sería favorable para la facilidad de uso de la aplicación. Por lo tanto, por un lado, los refrescos de listados se realizan de forma automática y, por otro lado, las otras dos funciones se unifican en una sola actividad. Este punto se abordará en el apartado “4.4. Filtrado de datos”.

Ilustración 28. Opción refrescar lista (solución anterior).



4.4. Creación de bases de datos de usuario.

La principal novedad de la ampliación de la aplicación consiste en la posibilidad de crear bases de datos en los que el usuario pueda guardar colecciones de región, país o estación seleccionada. Para este fin se crean tres listas desplegables- una de las regiones, otra de los países y una tercera de las estaciones. Además, se deja disponibles tres botones para

crear la base de datos de la selección a nivel preferido. Cabe añadir que, filtrando en el primer *Combo Box*, el usuario puede tanto crear una base de datos a este nivel como filtrar la segunda lista desplegable. Lo mismo aplica al segundo y tercer *Combo Box*.

Ilustración 29. Los Combo Box para seleccionar base de datos a crear.

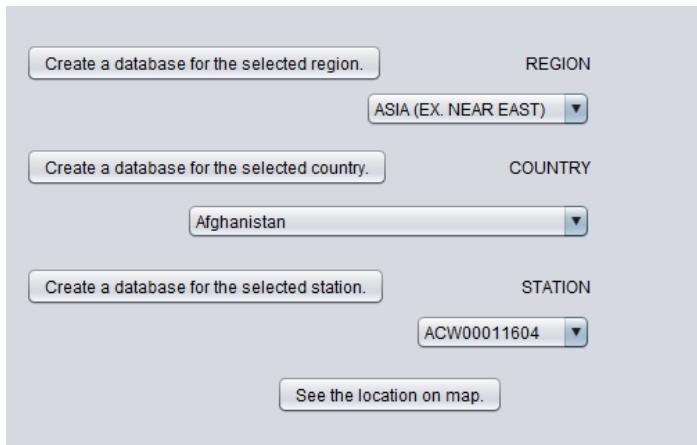
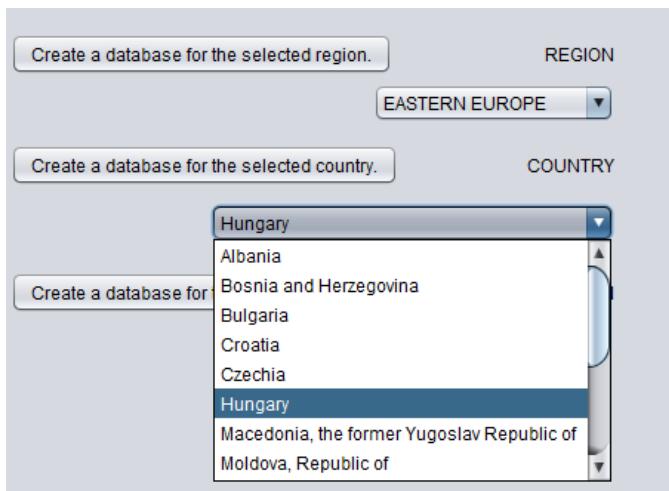


Ilustración 30. Efecto de filtrado en el segundo Combo Box.



El Código para la creación de los *Combo Box* está situado en la clase principal (“BigDataTFG”) y es el siguiente:

```

VerTabla.listaComboBox(jComboBox2,"opciones","regiones",rb.getString("region"),
                      rb.getString("$region"), this,rb);
VerTabla.listaComboBox(jComboBox3,"opciones","paises",rb.getString("pais"),
                      rb.getString("$pais"), this,rb);
VerTabla.listaComboBox(jComboBox5,"opciones","estaciones","estacion","$estacion",
                      this,rb);

```

El Código para el filtrado de los Combo Box está situado en la clase principal (“BigDataTFG”) y es el siguiente:

```

private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {

    JComboBox cb = (JComboBox)evt.getSource();

    String region = (String)cb.getSelectedItem();

    VerTabla.listaComboBox(jComboBox3,"opciones","paises",rb.getString("pais"),
                          rb.getString("$pais"),rb.getString("region"),region); // TODO add your handling
    code here:

}

private void jComboBox3ActionPerformed(java.awt.event.ActionEvent evt) {

    JComboBox cb = (JComboBox)evt.getSource();

    String pais = (String)cb.getSelectedItem();

    VerTabla.listaComboBox(jComboBox5,"opciones","estaciones","estacion","$estacion",
                          rb.getString("pais"),pais);

}

```

Ambas acciones utilizan la misma clase “VerTabla” y su método “listaComboBox”, aunque llaman a un número distinto de argumentos. En el primer caso se hace llamada a

la colección que reúne todos los regiones, países o estaciones, de acuerdo con la selección, y se devuelven los distintos valores del campo para utilizarlos en el Combo Box. Tanto la colección de países como la de regiones guardan los nombres en inglés y en español y, a través de ResorceBundle, se indica que campo utilizar.

Ilustración 31. Colección de los países en MongoDB.

```

_id: ObjectId("5cab9fffb985dd37dbb541f8")
_tips: "AC"
_iso: "AO"
country: "Angola and Cabo Verde"
pais: "Angola y Cabo Verde"
region: "LATIN AMER. & CARIB"
region_es: "AMÉRICA LATINA Y CARIBE"

_id: ObjectId("5cab9fffb985dd37dbb541f9")
_tips: "AE"
_iso: "AE"
country: "United Arab Emirates"
pais: "Emiratos Árabes Unidos"
region: "NEAR EAST"
region_es: "PRÓXIMO ESTE"

_id: ObjectId("5cab9fffb985dd37dbb541f9")
_tips: "AF"
_iso: "AF"
country: "Afghanistan"
pais: "Afganistán"
region: "ASIA (EX. NEAR EAST)"
region_es: "Asia"

```

Ilustración 32. Colección de las regiones en MongoDB.

```

_id: ObjectId("5ca8991fb985dd37dbb542dd")
region: "ASIA (EX. NEAR EAST)"
region_es: "ASIA"

_id: ObjectId("5ca8991fb985dd37dbb542de")
region: "BALTIOS"
region_es: "BALTIOS"

_id: ObjectId("5ca8991fb985dd37dbb542df")
region: "C.M. OF IND. STATES"
region_es: "C.M. DE ESTADOS UNIDOS"

```

En el segundo caso se hace llamada a la tabla de estaciones descrita en el punto 4.2.2. de este documento y el método pasa como parámetro los filtros que se deben aplicar. Se puede consultar el código completo de la clase VerTabla en los anexos.

Al crear la base de datos se abren automáticamente tres colecciones: total_p, total_tmin, total_tmax, que se usarán para almacenar, respectivamente, los datos de precipitaciones, temperaturas mínimas y temperaturas máximas. Para este fin se crea la clase “CrearBDMongo”, en la que se establece la conexión y se abre las bases de datos con el comando “getDatabase” (este método crea la base de datos si no existe) y, posteriormente, se abren las colecciones con el comando “createCollection”. El funcionamiento de esta clase es muy sencillo, la única operación necesaria es la sustitución de espacios por el símbolo “_” para obtener un nombre de base de datos válido.

Ilustración 33. MongoDB antes de la creación de la base de datos.

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays 'My Cluster' with 5 DBS and 12 COLLECTIONS. The main area shows a table of databases:

Database Name	Storage Size	Collections	Indexes
admin	16.0KB	0	1
config	24.0KB	0	2
local	40.0KB	1	1
opciones	7.0MB	5	5
tfg	8.6GB	6	6

Ilustración 34. MongoDB después de la creación de la base de datos.

The screenshot shows the MongoDB Compass interface after database creation. The sidebar now shows 'My Cluster' with 6 DBS and 15 COLLECTIONS. A new database named 'Hungary' has been added to the list. The main area shows the same table of databases as in Illustration 33, plus the newly created 'Hungary' database:

Database Name	Storage Size	Collections	Indexes
Hungary	12.0KB	3	3
admin	16.0KB	0	1
config	24.0KB	0	2
local	40.0KB	1	1
opciones	7.0MB	5	5
tfg	8.6GB	6	6

El Código para la creación de las bases de datos está situado en la clase principal (BigDataTFG) y es el siguiente:

```
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    CrearBDMongo.Crear((String)jComboBox2.getSelectedItem(),jComboBox4,  
    jComboBox6, this, rb);  
  
}  
  
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    CrearBDMongo.Crear((String)jComboBox3.getSelectedItem(),jComboBox4,  
    jComboBox6, this, rb);  
  
}  
  
private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    CrearBDMongo.Crear((String)jComboBox5.getSelectedItem(),jComboBox4,  
    jComboBox6, this, rb);  
  
}
```

Se puede consultar el código completo de las clases “BigDataTFG” y “CrearBDMongo” en los anexos.

Si bien la localización de las regiones y los países debe ser conocida por el usuario, se considera necesario indicarle la localización de las estaciones. Para este fin se crea el botón “See the location on map” que abre el navegador con Google Maps centrado en la estación seleccionada.

Ilustración 35. Botón para consultar la localización de las estaciones en el mapa.

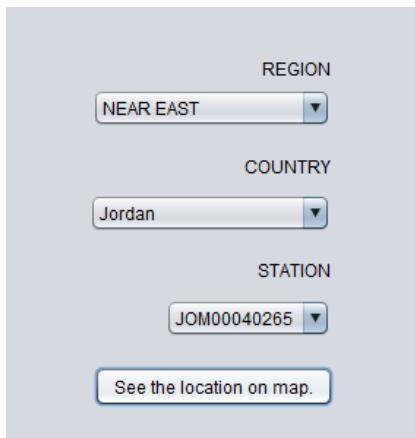
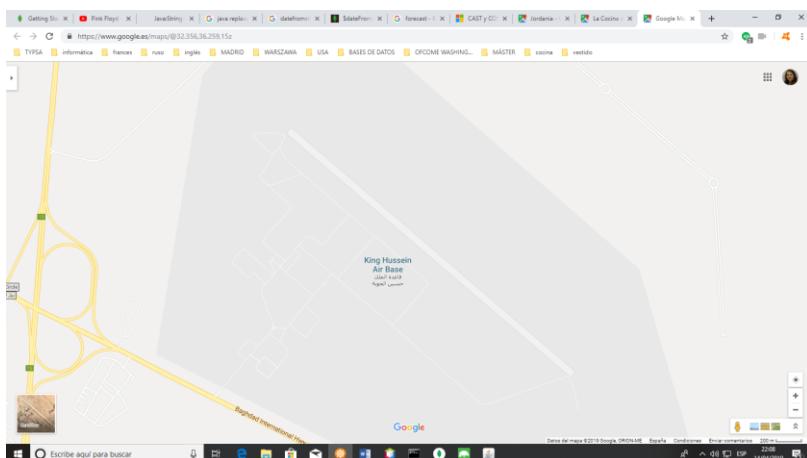


Ilustración 36. El mapa abierto por el botón.



El Código de esta funcionalidad está situado en la clase principal (BigDataTFG) y es el siguiente:

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String estacion = (String)jComboBox5.getSelectedItem();  
  
    ArrayList<String> resultado=null;  
  
    resultado=ConsultaWhere.resultado("opciones", "estaciones", "latitud",  
    "$latitud","estacion",estacion);
```

```

String latitud=resultado.get(0);

int largo=latitud.length();

String lat1=latitud.substring(0, largo-4);

String lat2=latitud.substring(largo-4);

resultado=ConsultaWhere.resultado("opciones", "estaciones", "longitud",
"$longitud","estacion",estacion);

String longitud=resultado.get(0);

largo=longitud.length();

String lon1=longitud.substring(0, largo-4);

String lon2=longitud.substring(largo-4);

try {

java.awt.Desktop.getDesktop().browse(new
URI("https://www.google.es/maps/place/@"+lat1+"."+lat2+","+lon1+"."+lon2+
,15z/"));

} catch (URISyntaxException ex) {

Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

}

}

```

Se recuperan los datos de longitud y latitud de las estaciones haciendo llamada a la colección comentada en el párrafo 4.2.2., se cambian de formato y se crea un enlace a Google Maps con estos datos. Para facilitar la selección de datos de esta colección se crea la clase ConsultaWhere, cuya funcionalidad es parecida a una consulta de tipo “where” de SQL. Se pasan como parámetros la base de datos, la colección para la búsqueda y los criterios. El método principal está sobrecargado y permite 8 o 12 parámetros, dependiendo de si se quiere establecer 1 o 3 filtros. Se puede consultar el código completo de la clase “ConsultaWhere” en los anexos.

4.5. Filtrado de datos.

Una vez descargados los datos anuales y creada la base de datos del usuario, se procede a filtrar los datos totales para guardar en la base de datos particular solamente aquellos que se correspondan con la región/el país/ la estación seleccionada. Para este fin, el usuario debe elegir un año y una de las bases de datos creadas y pulsar el botón de refresco (“Upload MongoDB”).

Ilustración 37. Botón de filtrado.



Para cargar los datos en las bases de datos he juntado las clases existentes: “ColeccionFinal” y “Consulta” en una clase nueva “ColeccionFinalN”. El objetivo de esta modificación fue la eliminación de la ambigüedad de comportamiento de botón de “Filtrar y refrescar” que en el proyecto de Francisco Javier Hermosilla tenía distintas funciones dependiendo de campo seleccionado en el Combo Box. En el primer paso se trasladaba los datos de un año a la colección AAAA_CyL (siendo AAAA el año en cuestión) y luego se consolidaban todas las colecciones de tipo AAAA_CyL en una colección CyL. En la versión actualizada ambos pasos se realizan al pulsar el botón “Update MongoDB”:

- Primero se filtran los registros de país/region/estación elegida y se crea una nueva colección de datos de precipitaciones con el nombre de año (por ejemplo “2018”) en la base de datos elegida. De igual manera se crean colecciones de temperaturas mínimas (por ejemplo “2018TMIN”) y temperaturas máximas (por ejemplo “2018TMAX”).

- Se hacen los cálculos y se llevan los datos agregados a las colecciones total_p, total_tmin, total_tmax en la base de datos.

Ilustración 38. Resultado de tareas de filtrado.

Las principales modificaciones del código incluyen:

- Se sustituye un listado estático de estaciones (de Castilla y León) por una selección dinámica a través del siguiente código:

```
estaciones=ConsultaWhere.resultado("opciones", "estaciones", "estacion",
"$estacion",buscado, valor.replace("_", " "), parent, rb);
```
- Se recogen los datos de temperaturas mínimas y máximas aparte de las precipitaciones.
- Se ordenan los datos por año en las colecciones de datos agrupados (total_p, total_tmin, total_tmax). En la versión inicial de la aplicación, al no existir esta ordenación, era necesario importar los datos en orden cronológico para una visualización posterior correcta. El código de esta funcionalidad es el siguiente (ejemplo para la colección tmin):

```
//ordenación de collection
Document project4 =new Document("Anho", true).append("Mes",
true).append("Temperatura",true).append("TemperaturaF",true);
//projection
Document project5 = new Document("$project", project4);
List<Document> pipeline3 = Arrays.asList(project5,sort2);
MongoCursor <Document> docsTF =
total.aggregate(pipeline3).iterator();
databaseDestino.createCollection("total_tminT");
MongoCollection<Document> totalTT =
databaseDestino.getCollection("total_tminT");
while (docsTF.hasNext()) {
    Document itTF=docsTF.next();
    totalTT.insertOne(itTF);
}
total.drop();
MongoNamespace newName2 = new
MongoNamespace(valor , "total_tmin");
```

```
totalTT.renameCollection(newName2);
```

- Se guardan los datos de temperatura tanto en grados Celsius como en grados Fahrenheit a través del siguiente código:

```
cur = col2.aggregate(pipeline).iterator();
// colección donde irán los nuevos documentos
total = databaseDestino.getCollection("total_tmin");
// bucle que calcula para cada documento la media mensual de temperatura
while (cur.hasNext()) {
    Document doc = cur.next();
    List list = new ArrayList(doc.values());
    double temp= (double) list.get(3);
    double tempF;
    temp=(double) list.get(3);
    temp=temp/10;
    tempF=(temp*9/5)+32;
    // Crea los documentos que se van a insertar
    Document document = new
    Document("Anho",Integer.parseInt(filtrarColecion)).
        append("Mes",list.get(0)).
        append("Temperatura",temp).
        append("TemperaturaF",tempF);
    total.insertOne(document);
}
```

- En el caso de que no haya ningún dato para algún mes en la colección de datos agrupados se marca la variable con el símbolo “?”. De esta manera se facilita la interpretación de datos por los algoritmos de WEKA durante las tareas de *data mining*.
- Al observar que para determinados países o estaciones la completitud de los datos era muy baja he decidido calcular el porcentaje de los días para los que se tienen los registros para poder informar del mismo al usuario durante las tareas de

visualización. El código de esta funcionalidad es el siguiente (ejemplo para los datos de precipitaciones):

```
MongoDatabase databaseFiabilidad = mongoClient.getDatabase("opciones");
MongoCollection<Document> collectionFiabilidad = databaseFiabilidad.getCollection("fiabilidad");
(...)

long numDocumentos=col.countDocuments();
//cálculo de porcentaje de datos disponibles
double fiabilidad=numDocumentos/(numEstaciones*365)*100;
if(fiabilidad>100)
{fiabilidad=100;}

Document document1 = new Document("bd",valor).
append("year",filtrarColeccion).
append("variable","rainfall").
append("fiabilidad",fiabilidad);
collectionFiabilidad.insertOne(document1);
```

Ilustración 39. Porcentaje de datos completos.

_id	bd	year	variable	fiabilidad
5cb99fe6b05d6d3544d1b2d8	Jordan	2013	rainfall	8

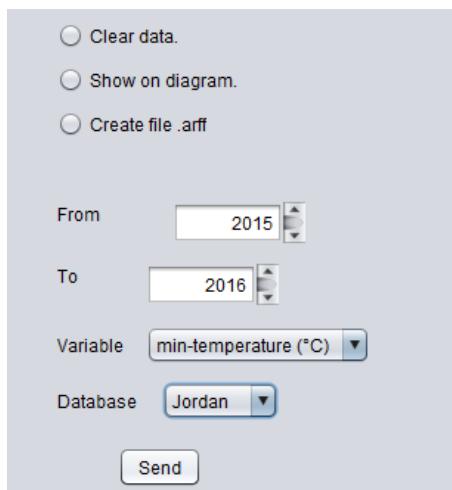
Se puede consultar el código completo de la clase “ColeccionFinalN” en los anexos.

4.6. Visualización de datos.

El siguiente paso, después de guardar los documentos filtrados y agrupados en las bases de datos correspondientes, consiste en visualizar la información en las gráficas. En la aplicación origen el usuario podía elegir el periodo para el diagrama, pero los datos siempre mostraban las precipitaciones de Castilla y León.

En esta ampliación el usuario puede elegir una de las bases de datos creadas, el periodo, la variable- precipitaciones o temperatura- y, para esta segunda también la medida (en grados Celsius o en grados Fahrenheit). En consecuencia, la clase ConsultaGraficar pasa de tener dos argumentos a cinco (yearIni, yearFin, bd, variable, rb), siendo el tercero la base de datos elegida, el cuarto la variable con su medida y el quinto el ResourceBundle para poder traducir los títulos y los ejes de las gráficas.

Ilustración 40. Opciones para la visualización.



Al pulsar el botón “send” se informa al usuario sobre los datos de fiabilidad de los distintos años o, en su caso, de no existencia de un año en la base de datos elegida. A continuación, se le presenta la gráfica solicitada.

Ilustración 41. Información sobre porcentaje de completitud del dato.

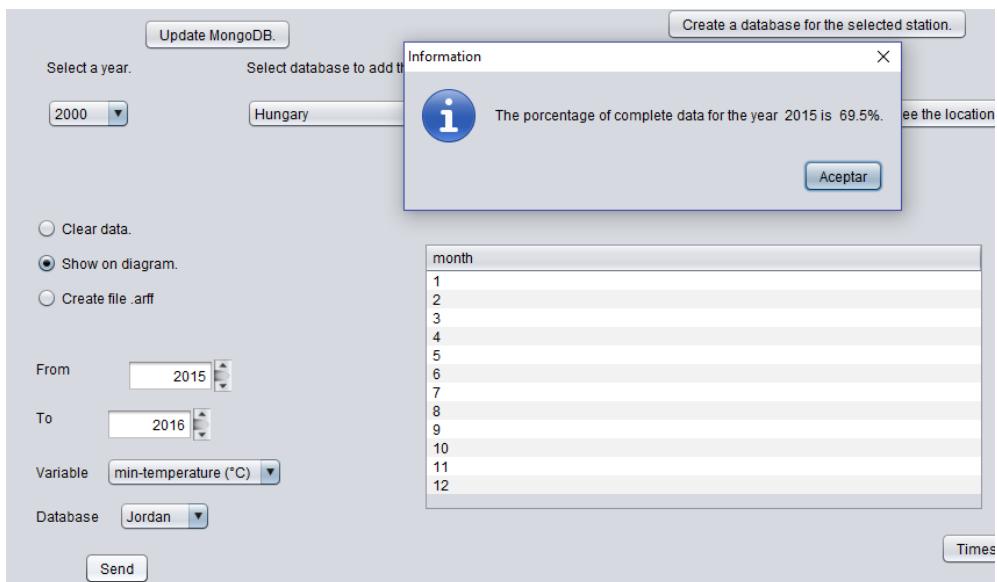
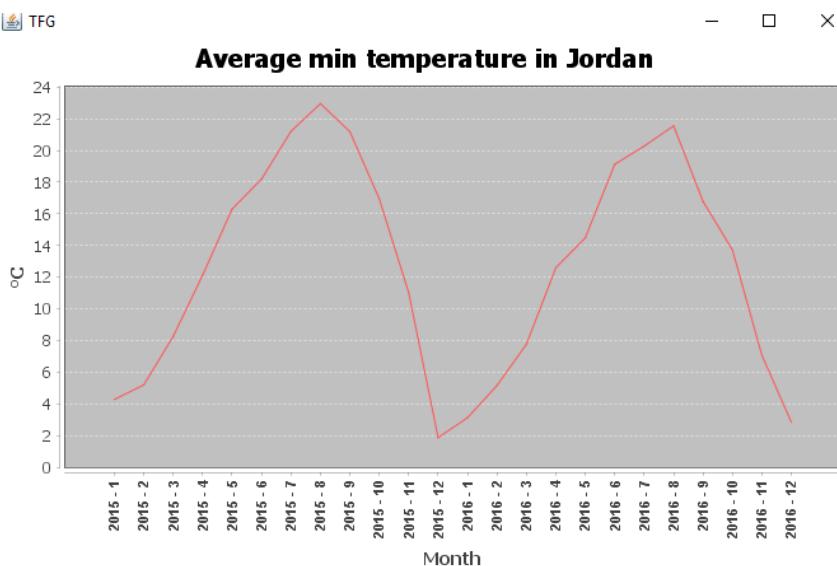


Ilustración 42. Gráfica con los datos seleccionados.



El código de la clase principal (BigDataTFG) responsable de esta parte es el siguiente:

```
if (yearIni<=yearFin) // comprueba que la fecha inicial es menor que la final
```

```

{ ArrayList<String> resultadoF=null;

for (int anho=yearIni; anho<=yearFin; anho=anho + 1 )

{

resultadoF=ConsultaWhere.resultado("opciones",
"fiabilidad","fiabilidad","$fiabilidad","bd",bd,"variable",variableS,"year",Integer
.toString(anho), this,rb);

String resultadoFiabilidad=null;

try{resultadoFiabilidad=substring(resultadoF.get(0),0,4);

JOptionPane.showMessageDialog(this, rb.getString("infoFiabilidadp1") + anho +
" " + rb.getString("infoFiabilidadp2") + resultadoFiabilidad
+"%",rb.getString("info"),      JOptionPane.INFORMATION_MESSAGE);}

catch(Exception ex) {

    JOptionPane.showMessageDialog(this, rb.getString("error1")+" "+anho+
" "+rb.getString("error2"));

; } }

ConsultaGraficar.consultaGraficar(yearIni,yearFin, bd, variable,rb);}

```

Se puede consultar el código completo de la clase “ConsultaGraficar” en los anexos.

4.7. Tareas de data mining.

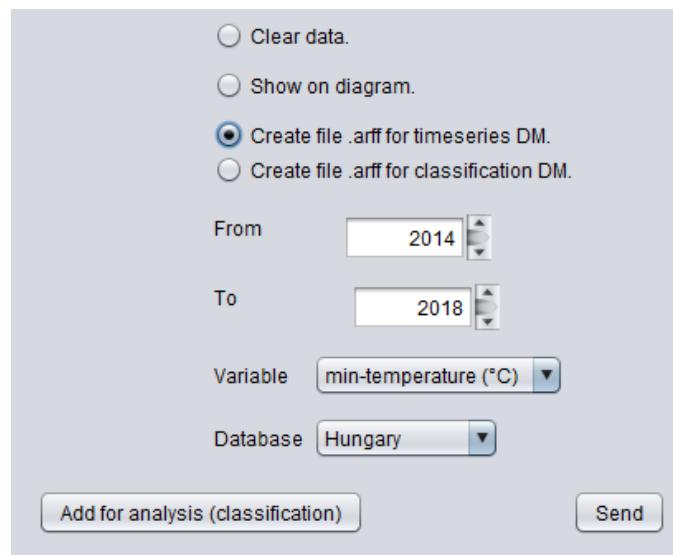
4.7.1. Análisis de series temporales.

En la aplicación original la única tarea de *data mining* propuesta era el análisis univariable (de precipitaciones) de series temporales. Además, dado que la aplicación estaba dedicada a la región de Castilla la Mancha (España), solamente se podía analizar los datos de esta región. Asimismo, a pesar de buscar predicciones de precipitaciones, el algoritmo podía devolver datos negativos.

En la ampliación propuesta se han desarrollado los siguientes aspectos:

- Se permite seleccionar la base de datos de estudio para guardar el fichero .arff. Cabe destacar que para fichero .arff de series temporales solamente se tiene que elegir base de datos y periodo (la selección de variable se aplica en la ejecución del algoritmo).

Ilustración 43. Opciones de selección para creación del fichero .arff.



- Se permite elegir al usuario la ruta en la que se guarda el fichero.

- Se eliminan las predicciones negativas de las precipitaciones. Este punto se consigue transformando los datos de precipitaciones a logaritmo natural y, a posteriori, transformando las predicciones a exponencial.

Ilustración 44. Resultados de análisis (predicción de precipitaciones sin números negativos).

The screenshot shows a user interface for a Data Mining application. On the left, there is a table titled "GaussianProcesses" with columns "month" and "GaussianProcesses". The data is as follows:

month	GaussianProcesses
1	3.1829137455127965
2	18.530056265286557
3	30.07445580152595
4	177.93540912331392
5	81.17633626108687
6	13.12848550416848
7	0.6949883341400839
8	0
9	0.1894907765049182
10	0.6950140487652516
11	23.796187134977416
12	77.15629087474943

On the right, there are two sections: "Please, select the DM technique." and "Please, select the algorithm and the variable.". The "DM technique" dropdown is set to "multivariate analysis (timeseries)". The "variable" dropdown is set to "rainfall". There are also three radio buttons for "GaussianProcesses", "LinearRegression", "SMOreg", and "MultilayerPerceptron", with "GaussianProcesses" being selected. Two "Submit" buttons are present at the bottom.

- Se permite al usuario seleccionar la técnica de Data Mining.

Ilustración 45. Selección de técnica de Data Mining.

The screenshot shows a dropdown menu for selecting a Data Mining technique. The options listed are "multivariate analysis (timeseries)", "univariate analysis (timeseries)", "multivariate analysis (timeseries)", and "classification". The first option, "multivariate analysis (timeseries)", is highlighted with a blue background. A "Submit" button is located to the right of the dropdown.

- Se permite al usuario seleccionar la variable a predecir (precipitaciones, temperatura mínima en grados Celsius, temperatura mínima en grados Fahrenheit, temperatura máxima en grados Celsius, temperatura máxima en grados Fahrenheit). Aunque el algoritmo hace predicción para todas las variables, solamente se muestra por pantalla la predicción de la variable elegida. En el fichero se guardan datos de precipitaciones y los datos de temperatura en grados Celsius. La transformación a grados Fahrenheit se realiza para visualizar resultados.

En esta parte de código se puede observar que, si se elige análisis multivariable, se añaden las tres variables al modelo mientras que, si se elige análisis univariable, la aplicación comprueba la variable elegida para crear el modelo solamente en esta base:

```
if(analisis==1)

{forecaster.setFieldsToForecast("precipitacion,tmin,tmax");}

else if (variable==0)

{{forecaster.setFieldsToForecast("tmin");} }

else if (variable==1)

{{forecaster.setFieldsToForecast("tmin");} }

else if (variable==2)

{{forecaster.setFieldsToForecast("tmax");} }

else if (variable==3)

{{forecaster.setFieldsToForecast("tmax");} }

else if (variable==4)

{{forecaster.setFieldsToForecast("precipitacion");} }
```

En la siguiente parte de código se accede a la predicción de la variable elegida y se guarda el resultado en un array que, posteriormente, se usa para visualizar los datos:

```
for (int i = 0; i < 12; i++) {

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

if(analisis==0 || variable==4 )
```

```

//si análisis univariable o variable=precipitacion

{

//coger predicción de primera variable (o única si univariable)

NumericPrediction predForTarget = predsAtStep.get(0);

//si variable es temperatura en Fahrenheit hacer transformación

if(variable==3 || variable==1)

datos [i] =((predForTarget.predicted())*9/5)+32;

//si variable es precipitación calcular exponencial

else if(variable==4)

{

double dato=exp(predForTarget.predicted());

if( dato<0.1)

datos [i]=0;

else

datos [i] =dato;

}

else

datos [i] =predForTarget.predicted();

}

```

```

//si variable es temperatura mínima

else if (variable==0 || variable==1)

{

    NumericPrediction predForTarget = predsAtStep.get(1);

    if(variable==1)

        datos [i] =((predForTarget.predicted())*9/5)+32;

    else

        datos [i] =predForTarget.predicted();

}

//si variable es temperatura máxima

else if (variable==2 || variable==3)

{

    NumericPrediction predForTarget = predsAtStep.get(2);

    if(variable==3)

        datos [i] =((predForTarget.predicted())*9/5)+32;

    else

        datos [i] =predForTarget.predicted();

}

```

Se puede consultar el código completo de las clases “FicheroArff” y “TimeSeriesTFG” en los anexos.

4.7.2. Clasificación.

Si bien la tarea de análisis de series temporales ya existía en la versión original de la aplicación, la tarea de clasificación es un desarrollo nuevo de este proyecto. La idea detrás de utilizar otras técnicas que las series temporales nace de la detección del problema de la baja completitud de datos para determinadas estaciones. En otras palabras, las predicciones basadas en un conjunto de prueba con múltiples valores desconocidos son poco fiables. Se busca, por lo tanto, predecir una variable de una estación en un mes en base a los valores que toma esta variable en estaciones parecidas (por criterios de latitud, longitud y altitud) en el mismo mes en otros años.

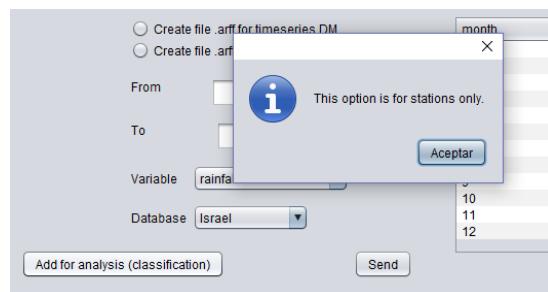
He probado distintos algoritmos utilizando la aplicación de WEKA y he elegido para la implementación el algoritmo de *Bagging*, por ser el que menos error expresaba. Cabe resaltar que *Bagging* no es un algoritmo tradicional. Se trata de una técnica que divide el conjunto de prueba en varios subconjuntos y aplica un algoritmo diferente a cada uno de ellos.

Antes de entrar en detalle de la aplicación de esta técnica, querría destacar que otras partes del programa pueden servir para que el usuario realice tareas previas necesarias para conducir el análisis. En primer lugar, se puede utilizar la funcionalidad de ver estaciones en el mapa para encontrar las estaciones cercanas a la estación objetivo. En segundo lugar, se puede utilizar la funcionalidad de visualizar datos para ver el porcentaje de completitud de datos, con el objetivo de seleccionar las estaciones que dispongan de un porcentaje alto de datos completos. Una vez elegidas las estaciones, el usuario debe añadirlas a un listado, a través de una lista desplegable. Esta selección se guarda en un array.

Ilustración 46. Selección de estaciones para el modelo de clasificación.

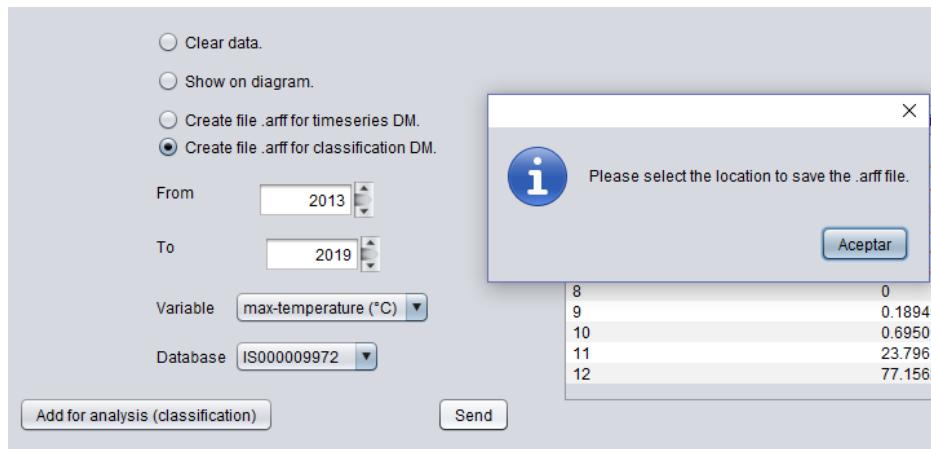


Ilustración 47. Información sobre el error al intentar elegir una base de datos que no es estación.



Cuando se hayan elegido todas las estaciones, se debe seleccionar la variable y el periodo, marcar la opción de creación de fichero .arff para clasificación y enviar la solicitud.

Ilustración 48. Solicitud de creación de fichero .arff.



La aplicación lee las selecciones del usuario y crea un archivo .arff en la ruta indicada. Aquí también se realiza la comprobación del sistema operativo mencionada en el apartado

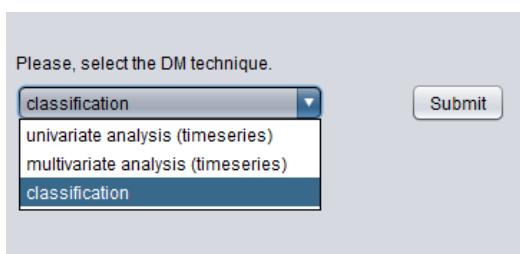
4.3.1. A continuación, se define la estructura de fichero. Luego, la aplicación accede a MongoDB para recoger los datos de longitud, latitud y altitud de las estaciones elegidas y, posteriormente, accede a la base de datos de la estación para recuperar los datos agregados de la variable y periodo seleccionados. Finalmente, se trasladan todos estos datos al fichero .arff.

Ilustración 49. Fichero .arff para la creación de modelo (fichero de entrenamiento).

```
DM_classify: Bloc de notas
Archivo Edición Formato Ver Ayuda
@relation tfg
@attribute anho numeric
@attribute mes numeric
@attribute t_max numeric
@attribute latitud numeric
@attribute longitud numeric
@attribute altitud numeric
@data
2013,1,22.596153846153847,295500,349500,12
2013,2,25.714999999999996,295500,349500,12
2013,3,29.163999999999998,295500,349500,12
2013,4,31.456521739130437,295500,349500,12
2013,5,38.27368421052632,295500,349500,12
2013,6,40.26086956521739,295500,349500,12
2013,7,39.525,295500,349500,12
2013,8,40.45652173913044,295500,349500,12
2013,9,37.532,295500,349500,12
2013,10,32.434615384615384,295500,349500,12
2013,11,29.3375,295500,349500,12
2013,12,21.944444444444446,295500,349500,12
2014,1,22.529166666666665,295500,349500,12
2014,2,24.9181818181818,295500,349500,12
```

La segunda parte de la tarea de clasificación consiste en la creación del archivo de testeo y la ejecución del algoritmo. En primer lugar, el usuario debe elegir la técnica de clasificación en la lista desplegable.

Ilustración 50. Selección de técnica de Data Mining.



El siguiente paso consiste en seleccionar la estación para preparar la predicción. Cabe destacar que no es necesario que la estación forme parte del modelo creado. Es posible elegir unas estaciones para producir el modelo y predecir los valores de la variable elegida de otra estación. Dado que en la lista desplegable están todas las estaciones disponibles (más de 100.000) se facilita al usuario filtrarlas con las listas desplegables de creación de bases de datos (descritas en el apartado 4.4). La aplicación informa sobre esta posibilidad a través de una ventana emergente.

Ilustración 51. Selección de estación para la predicción.

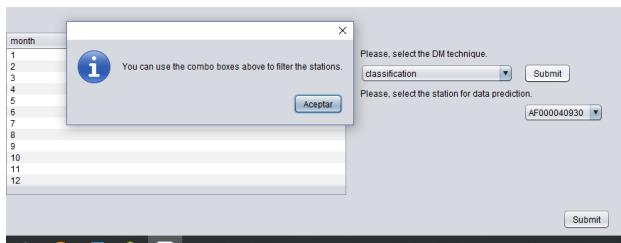
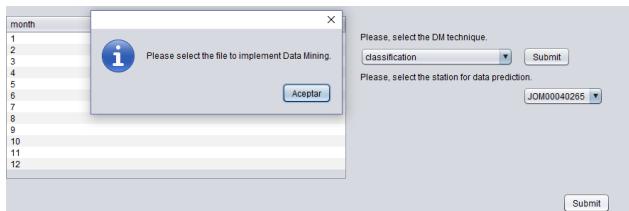
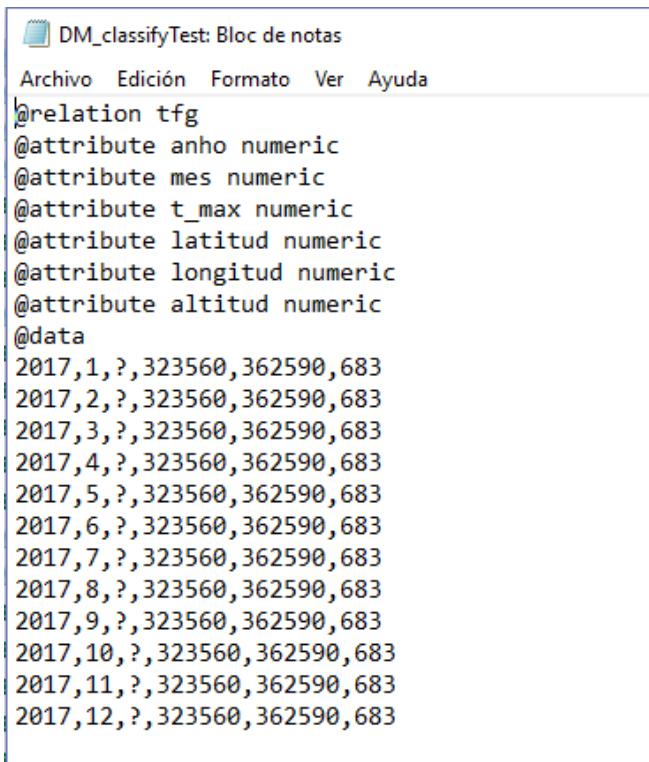


Ilustración 52. Selección de ruta de archivo de entrenamiento.



A continuación, la aplicación pide al usuario que elija el archivo de entrenamiento para crear el modelo (el archivo presentado en la ilustración 49). La aplicación accede a este archivo para recuperar la variable a predecir y el último año con datos. Luego, consulta a MongoDB para obtener los datos de latitud, longitud y altitud de la estación seleccionada. Todos estos valores se utilizan para crear el archivo de testeo, en el que el atributo a predecir se marca con el símbolo “?”.

Ilustración 53. Archivo de testeo.



```
DM_classifyTest: Bloc de notas
Archivo Edición Formato Ver Ayuda
@relation tfg
@attribute anho numeric
@attribute mes numeric
@attribute t_max numeric
@attribute latitud numeric
@attribute longitud numeric
@attribute altitud numeric
@data
2017,1,?,323560,362590,683
2017,2,?,323560,362590,683
2017,3,?,323560,362590,683
2017,4,?,323560,362590,683
2017,5,?,323560,362590,683
2017,6,?,323560,362590,683
2017,7,?,323560,362590,683
2017,8,?,323560,362590,683
2017,9,?,323560,362590,683
2017,10,?,323560,362590,683
2017,11,?,323560,362590,683
2017,12,?,323560,362590,683
```

Una vez creados los archivos de entrenamiento y testeо, la aplicación llama a la clase “Clasificar” para ejecutar el algoritmo. Finalmente, la aplicación guarda la predicción de los 12 meses siguientes en un array y traslada los datos a la tabla de resultados. Esta parte de código ilustra las principales tareas de creación de modelo y predicción:

```
//archivo de conjunto de entrenamiento
File archivoTrain = train;
//archivo de conjunto de prueba
File archivoTest = test;
Instances iTrain = new Instances(new BufferedReader(new FileReader(archivoTrain)));
Instances iTest = new Instances(new BufferedReader(new FileReader(archivoTest)));
```

```

//atributo a predecir

String atributo=iTrain.attribute(2).name();

//selección de algoritmo

Classifier cls = new Bagging();

//establecer atributo de clase

iTrain.setClassIndex(iTrain.numAttributes() - 4);

iTest.setClassIndex(iTest.numAttributes() - 4);

//ejecución de algoritmo

cls.buildClassifier(iTrain);

Evaluation eval = new Evaluation(iTrain);

//predicción

eval.evaluateModel(cls, iTest);

```

Ilustración 54. Resultados de Data Mining- clasificación.

The screenshot shows a user interface for a data mining application. On the left, there is a table with two columns: 'month' and 'classification'. The 'month' column lists integers from 1 to 12. The 'classification' column lists various numerical values. On the right, there are two dropdown menus and a submit button. The top dropdown is labeled 'Please, select the DM technique.' and has 'classification' selected. The bottom dropdown is labeled 'Please, select the station for data prediction.' and has 'JOM00040265' selected. Below these dropdowns is another submit button.

month	classification
1	14.84458432702051
2	15.691399890337555
3	21.720393897838
4	27.054450477510528
5	31.79581180318445
6	34.845600356011346
7	37.45221777473567
8	37.61386259376536
9	34.758358239140264
10	29.715636795598414
11	19.149928867044373
12	16.303154884788512

Se puede consultar el código completo de las clases “FicheroArff” y “Clasificar” en los anexos.

CAPÍTULO 5: EVALUACIÓN

El proyecto de la ampliación de la aplicación de Big data para mejorar los rendimientos agrícolas contaba con dos objetivos principales y uno complementario. Se buscaba principalmente universalizar el sistema y profundizar en las tareas de *data mining* así como realizar los ajustes necesarios para garantizar una mejor usabilidad. Cabe destacar, no obstante, que este tercer punto debe ser visto como adicional y que un desarrollo pleno de *front end* podría ser objetivo de un proyecto de mayor alcance.

En lo que se refiere a la universalización, considero que se ha cumplido plenamente con este objetivo. Se ha implementado la interfaz en dos idiomas, español e inglés, y, gracias a la solución elegida, sería relativamente sencillo añadir otras opciones de idioma. Tan solo habría que modificar una pequeña porción del código de la clase principal y crear un nuevo archivo de propiedades.

Asimismo, de un sistema específico para la región de Castilla y León se evoluciona o una aplicación que permite visualizar y analizar los datos de todas las estaciones meteorológicas. Además, las estaciones están categorizadas por país y región para facilitar su búsqueda al usuario. De igual manera, se añade dos nuevas variables: temperatura mínima y temperatura máxima. Teniendo en cuenta el uso de distintas medidas en diferentes regiones del mundo, se permite visualizar estos datos tanto en grados Celsius como en grados Fahrenheit.

En cuanto a las tareas de *data mining*, se ofrece al usuario la posibilidad de conducir este estudio para las tres variables. Además, se pasa de un solo tipo de tarea (análisis univariable de series temporales) a tres opciones: análisis univariable, análisis multivariable y clasificación. De esta forma se permite no solamente contrastar la predicción con diferentes métodos, sino además elegir la mejor técnica para el tipo de variable y su grado de completitud. Además, en la fase previa, durante la visualización de los datos del pasado se informa sobre el porcentaje de datos llenos, por lo que el usuario cuenta con información adicional para crear el modelo.

Para evaluar correctamente la ampliación de la aplicación es importante revisar si los nuevos algoritmos de *data mining* implementados han mejorada la precisión de la previsión. Para este fin se ejecutan 8 casos distintos:

- Previsión de precipitación para Jordania.
- Previsión de temperatura mínima para Jordania.
- Previsión de precipitación para Israel.
- Previsión de temperatura mínima para Israel.
- Previsión de precipitación para estación JOM00040265 en Jordania.
- Previsión de temperatura mínima para estación JOM00040265 en Jordania.
- Previsión de precipitación para estación ISE00105694 en Israel.
- Previsión de temperatura mínima para estación ISE00105694 en Israel.

En todos los casos se solicita la previsión para los primeros 6 meses de 2018 y se calcula el error absoluto medio y error cuadrático medio contrastando con datos reales. Los algoritmos que se prueban son los siguientes:

- Análisis de series temporales con Gaussian Process¹ de acuerdo con la lógica de la aplicación anterior (mostrando valores negativos de precipitaciones).
- Análisis de series temporales con Gaussian Process con la lógica de ejecución de algoritmo con valores logarítmicos (aplica solamente a precipitaciones).
- Análisis multivariable de series temporales con Gaussian Process.
- Clasificación con modelo basado en estaciones próximas sin descartes por poca completitud (aplica solamente a datos de estaciones).
- Clasificación con modelo basado en estaciones próximas que tengan alta completitud de dato (aplica solamente a datos de estaciones).

¹ Aunque existen cuatro algoritmos de series temporales se realizan las pruebas con Gaussian Process ya que se ha observado que es el que mejores resultados proporciona.

Los ocho casos cuentan con un porcentaje de datos completados muy dispar, lo cual impacta de forma significativa en las predicciones. A continuación, se presentan los resultados obtenidos.

Tabla 6. Caso1: precipitaciones en Jordania. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	2,73%	3,67%	2,52%	8,00%	3,12%	5,26%	5,58%	2,73%	4,20%

Tabla 7. Caso1: precipitaciones en Jordania. Previsiones.

mes	dato real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi- variable)	Clasificación
ene-18	51,62	-49,39	128,04	491,45	NA
feb-18	63,82	-93,07	49,68	265,09	NA
mar-18	20,00	-10,92	61,71	368,66	NA
abr-18	115,65	-40,82	6,74	67,76	NA
may-18	42,27	-13,44	1,08	7,16	NA
jun-18	0,00	22,31	0,53	1,51	NA

Tabla 8. Caso1: precipitaciones en Jordania. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multivariable)	Clasificación
ene-18	101,01	76,42	439,83	NA
feb-18	156,89	14,14	201,27	NA
mar-18	30,92	41,71	348,66	NA
abr-18	156,47	108,91	47,89	NA
may-18	55,71	41,20	35,11	NA
jun-18	22,31	0,53	1,51	NA
error absoluto medio	87,22	47,15	179,04	NA
error cuadrático medio	103,17	59,64	244,63	NA

Como se puede observar, se trata de un caso de una completitud del dato muy baja. El algoritmo que mejor se comporta es el análisis univariable modificado de series temporales. Cabe subrayar, no obstante, que todas las previsiones están lejos del dato real.

Tabla 9. Caso 2: precipitaciones en la estación JOM00040265. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	5,47%	9,04%	6,30%	13,90%	0,00%	9,31%	11,70%	4,38%	7,51%

Tabla 10. Caso 2: precipitaciones en la estación JOM00040265. Previsiones.

mes	real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi- variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	82,93	141,48	165,70	255,56	41,73	57,93
feb-18	73,75	58,74	65,66	95,39	46,51	54,44
mar-18	30,00	19,33	34,33	68,24	32,83	7,24
abr-18	51,00	19,22	18,27	41,74	16,21	5,66
may-18	60,00	6,20	6,13	13,74	0,91	1,66
jun-18	0,00	39,39	5,62	12,82	0,00	0,00

Tabla 11. Caso 2: precipitaciones la estación JOM00040265. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	58,55	82,77	172,64	41,20	25,00
feb-18	15,01	8,09	21,64	27,24	19,31
mar-18	10,67	4,33	38,24	2,83	22,76
abr-18	31,78	32,73	9,26	34,79	45,34
may-18	53,80	53,87	46,26	59,09	58,34
jun-18	39,39	5,62	12,82	0,00	0,00
error absoluto medio	34,87	31,24	50,14	27,53	28,46
error cuadrático medio	39,21	42,70	75,42	34,52	34,10

De nuevo se trata de un caso de una completitud del dato muy baja. El algoritmo que mejor se comporta es la clasificación. La diferencia entre utilizar para el modelo todas las estaciones o solamente las que contengan alto nivel de datos disponibles es mínima, de hecho, el error absoluto es menor para la primera opción y el error cuadrático para la segunda. No debe sorprender que, como se verá a continuación, sea el único caso en el

que usar el algoritmo de clasificación sería recomendable. Este tipo de análisis es menos exacto que una serie temporal, pero en una situación en la que los valores ausentes suponen más de 90% de todos los registros, utilizar datos de estaciones próximas aporta mucha más fiabilidad a la previsión.

Tabla 12. Caso 3: precipitaciones en Israel. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	58,2%	58,2%	56,6%	50,5%	48,4%	48,1%	43,4%	39,5%	50,3%

Tabla 13. Caso 3: precipitaciones en Israel. Previsiones.

mes	dato real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multivariable)	Clasificación
ene-18	174,41	80,25	50,63	71,99	NA
feb-18	77,24	43,15	37,28	47,04	NA
mar-18	7,80	13,15	14,55	13,31	NA
abr-18	29,36	2,73	5,23	5,18	NA
may-18	9,17	-5,47	3,15	3,00	NA
jun-18	3,15	-11,56	0,58	0,54	NA

Tabla 14. Caso 3: precipitaciones la Israel. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación
ene-18	94,16	123,78	102,41	NA
feb-18	34,09	39,96	30,20	NA
mar-18	5,35	6,75	5,51	NA
abr-18	26,64	24,14	24,18	NA
may-18	14,65	6,02	6,18	NA
jun-18	14,71	2,57	2,60	NA
error absoluto medio	31,60	33,87	28,52	NA
error cuadrático medio	43,20	54,14	44,83	NA

En este caso la disponibilidad del dato tiene un nivel medio, de 50,3%. La mejor previsión evaluada en error absoluto la ofrece el análisis multivariable, mientras que, de acuerdo con el error cuadrático, sería más fiable el análisis univariable de la versión anterior del software. Este caso, así como otros ejemplos en los que la solución para eliminar outputs negativos ha empeorado la calidad de la predicción, muestran la necesidad de reconsiderar la solución aplicada. Aunque parezca menos correcto cambiar simplemente todas las predicciones menores de 0 a 0, es posible que este simple arreglo tuviese mejores efectos que la ejecución del algoritmo sobre logaritmos.

Tabla 15. Caso 4: precipitaciones en la estación ISE00105694. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	99,4%	99,1%	96,4%	98,6%	99,1%	93,4%	98,9%	98,6%	97,9%

Tabla 16. Caso 4: precipitaciones en la estación ISE00105694. Previsiones.

mes	datp real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi- variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	200,57	109,21	75,93	137,76	66,28	71,71
feb-18	91,18	92,32	43,84	57,13	52,61	65,87
mar-18	5,79	5,11	20,48	22,45	32,04	25,23
abr-18	30,72	16,50	3,61	3,70	15,37	12,65
may-18	3,19	-4,21	3,71	4,02	22,35	4,10
jun-18	9,40	-17,84	0,89	0,82	0,00	0,00

Tabla 17. Caso 4: precipitaciones en la estación ISE00105694. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	91,36	124,64	62,81	134,29	128,86
feb-18	1,15	47,34	34,05	38,57	25,31
mar-18	0,68	14,69	16,66	26,25	19,43
abr-18	14,23	27,11	27,03	15,36	18,07
may-18	7,41	0,52	0,83	19,16	0,91
jun-18	27,24	8,51	8,58	9,40	9,40
error absoluto medio	23,68	37,13	24,99	40,50	33,66
error cuadrático medio	39,47	55,98	32,11	59,02	54,83

En el caso de la estación ISE0010594 los datos están muy completos. En cuanto al comportamiento de los algoritmos, la situación es parecida al caso anterior, los más recomendables serían el análisis univariable de la antigua versión o el análisis multivariable. Es importante señalar que el análisis multivariable viene afectado por el cálculo de valores en logaritmos, por lo que cabe esperar que al mejorar esta solución daría mejores resultados que el análisis univariable. Finalmente, es reseñable una mala

predicción del algoritmo de clasificación. Como he comentado antes, este resultado es el esperado, pues la clasificación es muy útil cuando el nivel de valores ausentes es alto. Si los datos de la estación son disponibles, el uso de registros de estaciones próximas contamina la predicción en vez de mejorarla.

Tabla 18. Caso 5. Temperatura mínima en Israel. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	44,2%	45,5%	44,2%	43,4%	44,4%	41,2%	37,1%	29,3%	41,2%

Tabla 19. Caso 5. Temperatura mínima en Israel. Previsiones.

mes	dato real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multivariable)	Clasificación
ene-18	8,18	6,65	NA	6,86	NA
feb-18	10,04	7,76	NA	8,09	NA
mar-18	11,66	10,37	NA	10,37	NA
abr-18	14,19	13,60	NA	13,53	NA
may-18	20,04	17,59	NA	17,22	NA
jun-18	20,94	20,68	NA	20,54	NA

Tabla 20. Caso 5. Temperatura mínima en Israel. Medición de errores

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación
ene-18	1,52	NA	1,31	NA
feb-18	2,28	NA	1,94	NA
mar-18	1,29	NA	1,29	NA
abr-18	0,59	NA	0,66	NA
may-18	2,44	NA	2,81	NA
jun-18	0,26	NA	0,40	NA
error absoluto medio	1,40	NA	1,40	NA
error cuadrático medio	1,61	NA	1,62	NA

Se trata de un caso de nivel bajo de disponibilidad del dato. Ambos algoritmos, análisis univariable y multivariable arrojan resultados muy parecidos. Como punto a subrayar, al contrario de lo esperado, es el caso con más precisión de predicción a pesar de tener completitud de datos más baja de los cuatro casos con variable de temperatura mínima.

Tabla 21. Caso 6. Temperatura mínima en Jordania. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	69,3%	59,2%	63,6%	72,6%	61,1%	69,5%	59,1%	66,5%	65,1%

Tabla 22. Caso 6. Temperatura mínima en Jordania. Previsiones.

mes	dato real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multivariable)	Clasificación
ene-18	5,21	2,23	NA	4,96	NA
feb-18	8,04	3,32	NA	4,53	NA
mar-18	10,11	6,52	NA	7,07	NA
abr-18	13,08	10,73	NA	10,15	NA
may-18	18,32	14,63	NA	13,37	NA
jun-18	18,63	18,04	NA	16,70	NA

Tabla 23. Caso 6. Temperatura mínima en Jordania. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación
ene-18	2,97	NA	0,25	NA
feb-18	4,72	NA	3,51	NA
mar-18	3,59	NA	3,03	NA
abr-18	2,35	NA	2,93	NA
may-18	3,69	NA	4,95	NA
jun-18	0,59	NA	1,93	NA
error absoluto medio	2,99	NA	2,77	NA
error cuadrático medio	3,25	NA	3,12	NA

Se trata de un caso de nivel medio de disponibilidad del dato. El algoritmo con mejor precisión es el análisis multivariable, tanto según el error absoluto como de acuerdo con el error cuadrático.

Tabla 24. Caso 7. Temperatura mínima en la estación JOM00040265. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	75,6%	76,9%	74,7%	81,6%	0,0%	82,4%	85,7%	87,9%	70,6%

Tabla 25. Caso 7. Temperatura mínima en la estación JOM00040265. Previsiones.

mes	datp real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi- variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	3,67	2,56	NA	3,69	1,64	2,13
feb-18	6,17	3,60	NA	5,00	1,76	2,75
mar-18	7,93	5,74	NA	7,34	6,23	6,80
abr-18	9,56	9,45	NA	9,37	9,79	9,28
may-18	15,84	12,88	NA	10,98	12,71	12,29
jun-18	15,98	16,17	NA	13,31	16,24	16,19

Tabla 26. Caso 7. Temperatura mínima en la estación JOM00040265. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	1,11	NA	0,03	2,03	1,53
feb-18	2,57	NA	1,17	4,41	3,42
mar-18	2,19	NA	0,59	1,70	1,12
abr-18	0,11	NA	0,19	0,23	0,28
may-18	2,96	NA	4,86	3,14	3,55
jun-18	0,19	NA	2,67	0,26	0,21
error absoluto medio	1,52	NA	1,59	1,96	1,69
error cuadrático medio	1,89	NA	2,33	2,46	2,16

En el caso de la estación JOM00040265 se tiene una disponibilidad del dato media-alta. El algoritmo con más precisión es el análisis univariable. Es destacable que el descarte de

estaciones con poca información disponible para la creación del modelo de clasificación ha mejorado sustancialmente el comportamiento de esta técnica. Ahora bien, el avance no es suficiente para adelantar en precisión al análisis univariable de la serie temporal.

Tabla 27. Caso 8. Temperatura mínima en la estación ISE00105694. Completitud del dato.

año	2010	2011	2012	2013	2014	2015	2016	2017	media
% completitud	100%	100%	97,8%	98,6%	99,7%	100%	97,8%	90,9%	98,1%

Tabla 28. Caso 8. Temperatura mínima en la estación ISE00105694. Previsiones.

mes	datp real	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi- variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	9,11	7,07	NA	7,23	4,91	6,52
feb-18	10,03	7,70	NA	7,88	5,38	6,64
mar-18	11,10	10,22	NA	9,62	7,91	10,95
abr-18	14,25	13,26	NA	12,90	11,41	13,19
may-18	19,64	17,20	NA	16,89	14,54	17,22
jun-18	21,31	20,70	NA	20,43	17,67	20,28

Tabla 29. Caso 8. Temperatura mínima en la estación ISE00105694. Medición de errores.

mes	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
ene-18	2,05	NA	1,89	4,21	2,60
feb-18	2,33	NA	2,14	4,65	3,39
mar-18	0,88	NA	1,48	3,19	0,15
abr-18	0,99	NA	1,34	2,84	1,06
may-18	2,44	NA	2,75	5,10	2,42
jun-18	0,62	NA	0,88	3,64	1,04
error absoluto medio	1,55	NA	1,75	3,94	1,78
error cuadrático medio	1,72	NA	1,85	4,02	2,09

Se trata del caso con mayor porcentaje de datos completos de los ocho ejemplos presentados. Los resultados son muy parecidos al caso anterior, el mejor algoritmo es el análisis univariable y se percibe una mejora importante en el algoritmo de clasificación al descartar estaciones con poca disponibilidad del dato.

En conclusión, el algoritmo de clasificación solamente es recomendable con baja disponibilidad del dato. En casos de disponibilidad media el análisis multivariable da resultados ligeramente superiores que análisis univariable, aunque también hay ejemplos en sentido contrario. En la siguiente tabla se resumen las mediciones de exactitud de las predicciones expresadas en error cuadrático. Como se puede observar, en cinco casos sería preferible el análisis univariable, en dos el análisis multivariable y en uno la clasificación.

Tabla 30. Resumen de mediciones de error cuadrático.

	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
Precipitación en Jordania	103,17	59,64	244,63	NA	NA
Precipitación en Israel	43,20	54,14	44,83	NA	NA
Precipitación en estación JOM00040265	39,21	42,70	75,42	34,52	34,10
Precipitación en estación ISE00105694	39,47	55,98	32,11	59,02	54,83
Temperatura mínima en Jordania	3,25	NA	3,12	NA	NA

	Gaussian Process (antigua versión)	Gaussian Process (nueva versión)	Gaussian Process (multi-variable)	Clasificación	Clasificación (solo completitud alta)
Temperatura mínima en Israel	1,61	NA	1,62	NA	NA
Temperatura mínima en estación JOM00040265	1,89	NA	2,33	2,46	2,16
Temperatura mínima en estación ISE00105694	1,72	NA	1,85	4,02	2,09

Por otro lado, sería necesario buscar otra solución para eliminación de valores negativos de precipitaciones, ya que en algunos casos el nuevo cálculo ofrece peores resultados que la versión antigua. Además, si se consiguiese mejorar este procedimiento, probablemente el análisis multivariable daría de forma más clara predicciones más fiables que el univariable en casos de disponibilidad media de información. Por último, si la información está rellena para la estación/ país/ región de interés en un nivel cercano a 100% la técnica recomendada es el análisis univariable. No obstante, cabe señalar que las pruebas se realizaron con una predicción basada en ocho años anteriores. Para ofrecer conclusiones más fundadas sería necesario realizar experimentos con más estaciones y distintos períodos.

Finalmente, cabe señalar, que aparte de desarrollos relacionados con la universalización del alcance y de la ampliación de tareas de *data mining*, se han llevado a cabo pequeñas mejoras de usabilidad de la aplicación. Se ha permitido elegir al usuario todos los directorios de guardado de ficheros y se han añadido ventanas emergentes para guiar al usuario a través de los distintos pasos. Además, se han automatizado algunas tareas, tales como el seguimiento de progreso de la subida de datos a MongoDB o la actualización de las listas desplegables. Asimismo, los datos se reordenan cronológicamente con cada actualización, de forma que el usuario no tiene que realizar las cargas de forma secuencial. Se han añadido también pequeñas ayudas visuales como la posibilidad de consultar la localización de las estaciones en el Google Maps.

No obstante, como he indicado anteriormente, estas mejoras pueden ser consideradas como lo mínimo indispensable para conseguir una aplicación sencilla, útil y portable. Para lograr una usabilidad real sería necesario acometer desarrollos con un alcance mucho mayor.

CAPÍTULO 6: CONCLUSIONES Y LÍNEAS FUTURAS

El proyecto “Sistema big data para mejorar los rendimientos agrícolas en Castilla y León” de F. J. Hermosilla Moreno que formó base para este trabajo se puede considerar como un primer prototipo para un sistema de mayor envergadura. F. J. Hermosilla Moreno consiguió de manera exitosa integrar una interfaz en java con una base de datos NoSQL MongoDB y con algoritmos de *open source* de WEKA.

El mayor logro del presente proyecto de ampliación es el aumento del alcance del análisis a todas las estaciones meteorológicas disponibles. Se ha conseguido evolucionar de una aplicación con un objetivo muy concreto a un sistema flexible y que abarca todas las regiones y distintas variables. No obstante, esta universalización del alcance ha provocado ralentización de algunos procesos. Por esta razón, se debería marcar como línea de trabajo futuro una mejora de rapidez de carga y lectura de datos. Sería también interesante que el usuario pudiese lanzar procesos de descargas masivas en segundo plano.

Para facilitar el trabajo a los analistas se debería ofrecer la opción de crear las bases de datos del nivel inferior junto con una de nivel superior. Por ejemplo, si el usuario desea crear la base de datos de Europa se le preguntaría si quiere de forma simultánea abrir bases de datos de los países pertenecientes a la región. Lo mismo aplicaría a los procesos de carga. Por último, el lenguaje NoSQL de MongoDB en un proyecto de mayor envergadura exige de un estudio más pormenorizado lo cual permitiría una mayor eficiencia de las consultas creadas.

Es también destacable que, al aumentar el espectro del análisis a todas las estaciones, he tenido que abordar nuevos retos de *data mining* y hallar soluciones a los problemas encontrados. En primer lugar, en muchas estaciones los valores ausentes superan el 50% y, en algunos casos, pueden incluso suponer más de 90%. Esta casuística provoca errores que no estaban detectados en el proyecto de F. J. Hermosilla al dedicarse únicamente a

una región con porcentajes de completitud de dato muy altos. La primera tarea para afrontar esta situación consistió en hacer las modificaciones necesarias para que el sistema no rechace años enteros en caso de faltar un mes de medición.

Un segundo problema causado por esta realidad es de mayor gravedad y complejidad. La baja completitud del dato impide realizar tareas de *data mining* con resultados fiables. Ya en la agrupación por meses el cálculo de la media se realiza con los días en los que han venido los registros. En otras palabras, se presupone que los días con datos y sin ellos son igual de representativos. Aunque sería necesario realizar un estudio exhaustivo sobre este tema, se podría arriesgar a concluir que, si bien para los datos de temperatura media unos pocos días del mes pueden considerarse suficientemente representativos, esto no es cierto para el caso de precipitaciones.

Está problemática queda especialmente patente en la región que ha sido objeto de un estudio más detallado en este proyecto, el Próximo Este (destacando los países de Jordania e Israel). Se trata de los países con el clima árido, en los que la lluvia es más un evento de poca frecuencia y variante intensidad, que una tendencia a detectar. La falta de datos en los meses más secos y su relativa abundancia en meses más lluviosos hace sospechar que los valores ausentes no son aleatorios.

Este problema se ha intentado abordar realizando análisis multivariable y algoritmos de clasificación, con el objetivo de aprovechar datos de variables más estables y las mediciones más fiables de estaciones cercanas. Como ha quedado demostrado en el apartado de evaluación, el uso de clasificación mejora las predicciones para las estaciones con pocos datos disponibles. En cuanto al análisis multivariable, sorprendentemente esta solución ha resultado más adecuada para modelos de *data mining* de precipitaciones con completitud de dato más alta. Sin duda se trata de un avance, no obstante, en un futuro desarrollo sería recomendable para casos de poca disponibilidad de registros no agrupar los datos por meses, sino intentar conducir *data mining* directamente con datos diarios.

Adicionalmente, se ha ofrecido al usuario herramientas para construir un modelo de minería de datos a medida. Es el analista que tiene la opción de consultar la ubicación de

las estaciones más cercanas, pudiendo además elegir las que tengan condiciones más parecidas a la región de interés (por ejemplo, por la cercanía o lejanía del mar). Asimismo, se pueden comprobar los porcentajes de completitud de los datos para descartar las localidades con niveles bajos de información. Quedaría para un futuro desarrollo hacer estas tareas más sencillas. Se propone permitir al usuario elegir estaciones de un radio determinado desde un punto de interés y seleccionar automáticamente los datos de las localidades y períodos con una completitud por encima de un umbral definido.

Por otro lado, en el apartado de evaluación ha quedado demostrado que la selección de la mejor técnica y algoritmo de *data mining* depende del porcentaje de datos disponibles y la variable elegida. Dicho de otra forma, no existe ningún algoritmo que sea el mejor para todos los casos. En este sentido, sería muy útil informar al usuario sobre los errores de las mediciones, basándose, por ejemplo, en la *10 fold cross validation*. En un desarrollo ambicioso podría ser el propio sistema que probase diferentes algoritmos y recomendase el más adecuado. Resultaría también interesante añadir otros algoritmos de clasificación, puesto que esta ampliación exigiría cambios mínimos en el código y podría suponer una mejora significativa en la exactitud de las predicciones.

Finalmente, como ya se mencionó en el apartado anterior, se llevaron a cabo labores para incrementar la usabilidad del sistema. No obstante, quedan muchas mejoras por lograr en este campo. Las tareas pendientes se pueden clasificar en tres grandes grupos: una interfaz más simple y atractiva, la portabilidad y la automatización. En lo que se refiere al primer punto, dado que el proceso completo se compone ya de cerca de diez pasos, varios de ellos con ramificaciones, lo más sensato sería abandonar la idea de una pantalla única y crear diferentes pestañas para preparación de datos, visualización y *data mining*.

Asimismo, el usuario no debería tener visibilidad de archivos que utiliza el sistema ni debería ser su responsabilidad elegir las rutas para guardar y leer los archivos ya que esto puede llevar a errores. La existencia de ficheros tendría que ser transparente y los usuarios

seleccionarían los años, variables y localidades a procesar sin necesidad de saber qué archivos son necesarios para realizar estas tareas.

En cuanto a la portabilidad, se ha mejorado mucho en este punto gracias a la eliminación de los directorios estáticos y a la solución de un problema de diferencias entre los sistemas operativos. Ahora bien, en el estado actual es necesario importar a MongoDB unas colecciones auxiliares con los datos estáticos de las estaciones para el correcto funcionamiento de la aplicación. La primera línea de trabajo futura debería consistir en guardar estos ficheros en un paquete inicial y ejecutar su importación a MongoDB en el proceso de la instalación de la aplicación.

No obstante, considero que el objetivo final debería ser transformar el sistema en una aplicación web. En primer lugar, esto permitiría aprovechar los servicios de MongoDB en la nube, ahorrando de esta forma los recursos de los equipos de los usuarios. En segundo lugar, las colecciones auxiliares no tendrían que ser instaladas por separado. Finalmente, esta solución permitiría incluso disponer de todos los datos subidos a MongoDB y el usuario se dedicaría solamente a las tareas de visualización y construcción de modelos de *data mining*.

Este salto cualitativo permitiría también centrarse más en las tareas analíticas y considerar la utilidad de inclusión de otros datos, tales como el aprovechamiento de recursos hídricos o los distintos usos y su peso en la demanda total. En futuro la aplicación presentada en este trabajo podría servir de base para construir un sistema experto para la toma de decisiones relacionadas con las políticas de la gestión de los recursos hídricos.

Bibliografía

- Aghabozorgi, S., Saybani, M. R., & Wah, T. Y. (2012). Incremental clustering of time-series by fuzzy clustering. *Journal of Information Science and Engineering*, 28(4), 671-688.
- Anochia, J. A., & de Campos Velho, H. F. (2015). Meteorological data mining for climate precipitation prediction using neural networks. *J. Comp. Int. Sci.*, 6(2), 71-78.
- Aquastat, F. A. O. (2011). FAO's information system on water and agriculture.
- Das, S. (1994). Time series analysis. Princeton University Press, Princeton, NJ.
- Diebold, F. X. (2012). On the Origin (s) and Development of the Term 'Big Data'.
- Faraj, A., Rashid, B., & Shareef, T. (2014). Comparative study of relational and non-relations database performances using Oracle and MongoDB systems. *Journal Impact Factor*, 5(11), 11-22.
- Hanasaki, N., Fujimori, S., Yamamoto, T., Yoshikawa, S., Masaki, Y., Hijioka, Y., ... & Kanae, S. (2012). A global water scarcity assessment under shared socio-economic pathways--Part 2: Water availability and scarcity. *Hydrology & Earth System Sciences Discussions*, 9(12).
- Hermosilla Moreno, F. J. (2019). Sistema big data para mejorar los rendimientos agrícolas en Castilla y León.
- Jacobs, A. (2009). The pathologies of big data. *Communications of the ACM*, 52(8), 36-44.
- Jeuland, M. (2016). The Case for Improving Water Efficiency in MENA Countries. En *Policy Perspective* (No. 19).

- Keogh, E. (1997). A fast and robust method for pattern matching in time series databases. *Proceedings of WUSS*, 97(1), 99.
- Keogh, E. J., & Pazzani, M. J. (1998). An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. En *Kdd* (Vol. 98, pp. 239-243).
- Kleist, C. (2015). Time series data mining methods: A review (Tesis Doctoral, Humboldt-Univ.).
- Krafft, C., Sieverding, M., Salemi, C., & Keo, C. (2018). Syrian Refugees in Jordan: Demographics, Livelihoods, Education, and Health. En *Economic Research Forum Working Paper Series* (No. 1184).
- Lara, A. T. (2011). Marco de descubrimiento de conocimiento para datos estructuralmente complejos con énfasis en el análisis de eventos en series temporales (Tesis Doctoral, Universidad Politécnica de Madrid).
- Martens, M. (2017). Food and Water Security in the Middle East and North Africa. NATO Parliamentary Assembly.
- Mauricio, J. A. (2007). Análisis de series temporales. Universidad Complutense de Madrid.
- Medina, J. (2010). Análisis de Fourier para el tratamiento de señales. Proc. XII Encuentro de Matemáticas y sus Aplicaciones, EPN-Quito, Ecuador.
- Minu, K. K., Lineesh, M. C., & John, C. J. (2010). Wavelet neural networks for nonlinear time series analysis. *Applied Mathematical Sciences*, 4(50), 2485-2495.
- Mualla, W. (2018). Water Demand Management Is a Must in MENA Countries... But Is It Enough?. *Journal of Geological Resource and Engineering*, 6, 59-64.

- Papacharalampous, G., & Tyralis, H. (2018). One-step ahead forecasting of annual precipitation and temperature using univariate time series methods. En EGU General Assembly Conference Abstracts (Vol. 20, p. 2298).
- Ralanamahatana, C. A., Lin, J., Gunopoulos, D., Keogh, E., Vlachos, M., & Das, G. (2005). Mining time series data. En Data mining and knowledge discovery handbook (pp. 1069-1103). Springer, Boston, MA.
- Roudi-Fahimi, F., Creel, L., & De Souza, R. M. (2002). Finding the balance: Population and water scarcity in the Middle East and North Africa. Population Reference Bureau Policy Brief, 1-8.
- Sellers, P. J. (1985). Canopy reflectance, photosynthesis, and transpiration. International Journal of Remote Sensing, 6, 1335-1372.
- Sharma, A. (2006). Spatial data mining for drought monitoring: an approach using temporal NDVI and rainfall relationship. ITC.
- Steduto, P., Hoogeveen, J., Winpenny, J., & Burke, J. (2017). Coping with water scarcity: an action framework for agriculture and food security. Food and Agriculture Organization of the United Nations Roma, Italia.
- Zhingri, V., & Augusto, C. (2016). Análisis de rendimiento entre la base de datos relacional: MySQL y una base de datos no relacional: MongoDB (Trabajo fin de carrera, Universidad del Azuay).

Anexos

Anexo 1: Las clases de la aplicación y su autoría.

Clase	Autoría	Anexo
BigDataTFG	Francisco Javier Hermosilla con modificaciones y desarrollos relevantes de Paulina Pyzel	2
LeerMongoRunnable	Paulina Pyzel con aprovechamiento de código de la clase LeerMongo	3
CrearBDMongo	Paulina Pyzel	4
VerTabla	Paulina Pyzel	5
ConsultaWhere	Paulina Pyzel	6
VerBasesDeDatos	Paulina Pyzel	7
ColeccionFinalN	Paulina Pyzel con aprovechamiento de código de las clases Consulta y ColeccionFinal	8
ConsultaGraficar	Francisco Javier Hermosilla con modificaciones y desarrollos relevantes de Paulina Pyzel	9
VerColecciones	Francisco Javier Hermosilla con modificaciones menores de Paulina Pyzel	10
Descompresor	Francisco Javier Hermosilla con modificaciones menores de Paulina Pyzel	11
DownloadFichero	Francisco Javier Hermosilla con modificaciones menores de Paulina Pyzel	12

FormatFecha	Francisco Javier Hermosilla	13
MongoDB	Francisco Javier Hermosilla	14
FicheroArff	Francisco Javier Hermosilla con modificaciones y desarrollos relevantes de Paulina Pyzel	15
TimeSeriesTFG	Francisco Javier Hermosilla con modificaciones y desarrollos relevantes de Paulina Pyzel	16
Clasificar	Paulina Pyzel	17

Anexo 2: Código de la clase BigDataTFG.

```
package bigdatatfg;

import java.awt.HeadlessException;
import java.io.File;
import java.io.IOException;
import java.text.ParseException;
import java.util.Locale;
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;
import java.util.ResourceBundle;
import static java.lang.Character.isDigit;
import java.net.URI;
```

```

import java.net.URISyntaxException;
import java.util.ArrayList;
import javax.swing.JComboBox;
import static jdk.nashorn.internal.objects.NativeString.substring;

// 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

private ResourceBundle rb;

private ResourceBundle sp;

private ResourceBundle en;

private ArrayList<String> estacionesClasificacion=new ArrayList<String>();

public BigDataTFG() {

    //seleccion del idioma de la aplicación

    String lang="en";

    String country ="UK";

    Locale e=new Locale (lang,country);

    this.en=ResourceBundle.getBundle("bigdatatfg/english",e);

    String lang2="sp";

    String country2 ="SP";

    Locale s=new Locale (lang2,country2);

    this.sp=ResourceBundle.getBundle("bigdatatfg/spanish",s);
}

```

```

this.rb=en;

String[] values = {"Spanish", "English"};

boolean noRespondido=true;

String lenguajeElegido=new String();

while (noRespondido)

{Object selected = JOptionPane.showInputDialog(null, "Please select your
language", "Selection", JOptionPane.DEFAULT_OPTION, null, values, "0");

if (selected !=null)

{

lenguajeElegido = selected.toString();

if("Spanish".equals(lenguajeElegido))

{rb=sp;

noRespondido=false;}

else if ("English".equals(lenguajeElegido))

{rb=en;

noRespondido=false;}

}

initComponents();

```

```
//se pone como invisibles las partes particulares de diferentes técnicas de DM  
//al elegir la técnica de DM se hacen visibles los elementos correspondientes  
  
jLabel12.setVisible(false);  
  
jRadioButton4.setVisible(false);  
  
jRadioButton5.setVisible(false);  
  
jRadioButton6.setVisible(false);  
  
jRadioButton7.setVisible(false);  
  
jButton7.setVisible(false);  
  
jLabel13.setVisible(false);  
  
jComboBox10.setVisible(false);  
  
jComboBox9.setVisible(false);  
  
jButton13.setVisible(false);  
  
this.setLocationRelativeTo(null);  
  
//ver los años descargados en una lista desplegables  
  
VerColecciones.listaComboBox(jComboBox1, "tfg");  
  
//ver las bases de datos creadas en desplegables  
  
VerBasesDeDatos.listaComboBox(jComboBox4);  
  
VerBasesDeDatos.listaComboBox(jComboBox6);  
  
//ver estaciones/paises/regiones en listas desplegables correspondientes
```

```
VerTabla.listaComboBox(jComboBox10,"opciones","estaciones","estacion","$estacion"  
, this,rb);
```

```
VerTabla.listaComboBox(jComboBox2,"opciones","regiones",rb.getString("region"),rb.  
getString("$region"), this,rb);
```

```
VerTabla.listaComboBox(jComboBox3,"opciones","paises",rb.getString("pais"),rb.getStri  
ng("$pais"), this,rb);
```

```
VerTabla.listaComboBox(jComboBox5,"opciones","estaciones","estacion","$estacion",  
this,rb);
```

```
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);
```

```
buttonGroup1.add(jRadioButton8);
```

```

// añade los jRadioButton a grupo de botones para solo poder seleccionar 1
simultaneamente

// grupo de método de predicción

buttonGroup2.add(jRadioButton4);

buttonGroup2.add(jRadioButton5);

buttonGroup2.add(jRadioButton6);

buttonGroup2.add(jRadioButton7);

// tabla que presenta predicciones

modelo=new DefaultTableModel(); // creación

modelo.addColumn(rb.getString("mes"),mes); // columna mes

this.jTable1.setModel(modelo); // se hace visible

}// fin de método constructor

@SuppressWarnings("unchecked")

// 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();

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();  
  
jYearChooser3 = new com.toedter.calendar.JYearChooser();  
  
jYearChooser4 = new com.toedter.calendar.JYearChooser();  
  
jComboBox2 = new javax.swing.JComboBox<>();  
  
jComboBox3 = new javax.swing.JComboBox<>();  
  
jComboBox5 = new javax.swing.JComboBox<>();  
  
jButton8 = new javax.swing.JButton();  
  
jButton9 = new javax.swing.JButton();  
  
jButton10 = new javax.swing.JButton();  
  
jButton11 = new javax.swing.JButton();  
  
jComboBox4 = new javax.swing.JComboBox<>();  
  
jLabel6 = new javax.swing.JLabel();  
  
jButton12 = new javax.swing.JButton();  
  
jLabel5 = new javax.swing.JLabel();  
  
jLabel8 = new javax.swing.JLabel();
```

```
jLabel9 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jComboBox6 = new javax.swing.JComboBox<>();

jLabel10 = new javax.swing.JLabel();

jComboBox7 = new javax.swing.JComboBox<>();

filler1 = new javax.swing.Box.Filler(new java.awt.Dimension(0, 5), new
java.awt.Dimension(0, 5), new java.awt.Dimension(32767, 5));

jProgressBar3 = new javax.swing.JProgressBar();

jComboBox8 = new javax.swing.JComboBox<>();

jLabel11 = new javax.swing.JLabel();

jComboBox9 = new javax.swing.JComboBox<>();

jLabel12 = new javax.swing.JLabel();

jButton4 = new javax.swing.JButton();

jRadioButton8 = new javax.swing.JRadioButton();

jButton5 = new javax.swing.JButton();

jComboBox10 = new javax.swing.JComboBox<>();

jLabel13 = new javax.swing.JLabel();

jButton13 = new javax.swing.JButton();
```

```
jTextField3.setText("jTextField3");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setTitle("BigDataTFG");

jLabel1.setText(rb.getString("año"));

jButton1.setText(rb.getString("descargar"));

jButton1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton1ActionPerformed(evt);

    }

});

jProgressBar1.setStringPainted(true);

jButton2.setText(rb.getString("descomprimir"));

jButton2.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
jButton2ActionPerformed(evt);

}

});

jProgressBar2.setStringPainted(true);

jButton3.setText(rb.getString("csvamongo"));

jButton3.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton3ActionPerformed(evt);

    }

});

jLabel2.setText(rb.getString("elegirAño"));

jRadioButton1.setText(rb.getString("limpiar"));

jRadioButton2.setText(rb.getString("grafica"));
```

```
jRadioButton3.setText(rb.getString("arff"));

jButton6.setText(rb.getString("Enviar"));

jButton6.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton6ActionPerformed(evt);

    }

});}

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

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

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        new String [] {
            "", "", "", "", ""
        }
    },
    new String [] {
        "Código", "Nombre", "Apellido", "DNI", "Teléfono"
    }
));

new String [] {
    "Código", "Nombre", "Apellido", "DNI", "Teléfono"
});
```

```
rb.getString("Mes"), rb.getString("Predicción")

}

) {

Class[] types = 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");

jRadioButton5.setText("LinearRegression");

jRadioButton6.setText("SMOreg");

jRadioButton7.setText("MultilayerPerceptron");

jButton7.setText(rb.getString("PredecirST"));

jButton7.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton7ActionPerformed(evt);

    }

});;

jComboBox2.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]

{ }));
```

```
jComboBox2.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
        jComboBox2ActionPerformed(evt);  
  
    }  
  
});  
  
  
  
jComboBox3.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]  
{}));  
  
jComboBox3.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
        jComboBox3ActionPerformed(evt);  
  
    }  
  
});  
  
  
  
jComboBox5.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]  
{}));  
  
  
  
jButton8.setText(rb.getString("ver"));  
  
jButton8.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
        jButton8ActionPerformed(evt);  
  
    }  
  
});
```

```
public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
    jButton8ActionPerformed(evt);  
  
}  
  
});  
  
  
  
jButton9.setText(rb.getString("crearRegion"));  
  
jButton9.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
        jButton9ActionPerformed(evt);  
  
    }  
  
});  
  
  
  
jButton10.setText(rb.getString("crearPais"));  
  
jButton10.addActionListener(new java.awt.event.ActionListener() {  
  
    public void actionPerformed(java.awt.event.ActionEvent evt) {  
  
        jButton10ActionPerformed(evt);  
  
    }  
  
});
```

```
jButton11.setText(rb.getString("crearEstacion"));

jButton11.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton11ActionPerformed(evt);

    }

});

jComboBox4.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]

{ }));

jLabel6.setText(rb.getString("elegirBD"));

jButton12.setText(rb.getString("ActualizarMongoDB"));

jButton12.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton12ActionPerformed(evt);

    }

});

});
```

```
jLabel5.setText("REGION");
```

```
jLabel8.setText(rb.getString("ESTACION"));
```

```
jLabel9.setText(rb.getString("PAIS"));
```

```
jLabel7.setText(rb.getString("bd"));
```

```
jComboBox6.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] {}));
```

```
jLabel10.setText("Variable");
```

```
jComboBox7.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { rb.getString("tempmin"), rb.getString("tempminf"),rb.getString("tempmax"),rb.getString("tempmaxf"),rb.getString("rain") }));
```

```
jProgressBar3.setStringPainted(true);
```

```
jComboBox8.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { rb.getString("1v"),rb.getString("mv"),rb.getString("clas") }));
```

```
jLabel11.setText(rb.getString("seleccionar"));

jComboBox9.setModel(new javax.swing.DefaultComboBoxModel<>(new String[]
{ rb.getString("tempmin"), rb.getString("tempminf"),rb.getString("tempmax"),
rb.getString("tempmaxf"),rb.getString("rain") }));

jLabel12.setText(rb.getString("seleccionar2"));

jButton4.setText(rb.getString("PredecirST"));

jButton4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton4ActionPerformed(evt);

    }

});

jRadioButton8.setText(rb.getString("arff2"));

jButton5.setText(rb.getString("paraClasificacion"));
```

```
jButton5.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton5ActionPerformed(evt);

    }

});;

jComboBox10.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] {}));

jLabel13.setText(rb.getString("seleccionar3"));

jButton13.setText(rb.getString("PredecirST"));

jButton13.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton13ActionPerformed(evt);

    }

});;

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
```

```
getContentPane().setLayout(layout);

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(30, 30, 30)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    )
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                )
            .addGroup(layout.createSequentialGroup()
                .addGap(56, 56, 56)

        .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)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
G)

    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE))

        .addComponent(jButton1,
javax.swing.GroupLayout.Alignment.LEADING)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

    .addComponent(jButton2)

    .addComponent(jProgressBar2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
```

```
.addGroup(layout.createSequentialGroup()  
  
    .addGap(18, 18, 18)  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING  
G)  
  
    .addComponent(jLabel2)  
  
    .addComponent(jComboBox1,  
javax.swing.GroupLayout.PREFERRED_SIZE, 72,  
javax.swing.GroupLayout.PREFERRED_SIZE))  
  
    .addGap(98, 98, 98)  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
, false)  
  
    .addComponent(jLabel6,  
javax.swing.GroupLayout.DEFAULT_SIZE,  
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
  
    .addComponent(jComboBox4, 0,  
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))  
  
.addGroup(layout.createSequentialGroup()  
  
    .addGap(93, 93, 93)  
  
.addComponent(jButton12)))
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

.addGroup(layout.createSequentialGroup()

.addGap(27, 27, 27)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

.addComponent(jButton3)

.addComponent(jProgressBar3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(0, 0, Short.MAX_VALUE))

.addGroup(layout.createSequentialGroup()

.addComponent(filler1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))  
  
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createSequentialGroup()  
  
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,  
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
)  
  
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createSequentialGroup()  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
)  
  
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  
  
        .addGroup(layout.createSequentialGroup()  
  
        .addComponent(jButton11)  
  
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,  
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
        .addComponent(jLabel8))

        .addGroup(layout.createSequentialGroup()

            .addComponent(jButton10)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
                javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(jLabel9))

        .addGroup(layout.createSequentialGroup()

            .addComponent(jButton9)

            .addGap(103, 103, 103)

            .addComponent(jLabel5)))

        .addComponent(jComboBox3,
                javax.swing.GroupLayout.Alignment.TRAILING,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jComboBox5,
                javax.swing.GroupLayout.Alignment.TRAILING,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jComboBox2,
                javax.swing.GroupLayout.Alignment.TRAILING,
                javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(149, 149, 149))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addComponent(jButton8)

.addGap(212, 212, 212)))))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
G)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)

.addComponent(jRadioButton2)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel4)

.addGap(49, 49, 49)
```

```
        .addComponent(jYearChooser3,  
javax.swing.GroupLayout.PREFERRED_SIZE,  
javax.swing.GroupLayout.DEFAULT_SIZE,  
javax.swing.GroupLayout.PREFERRED_SIZE))  
  
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createSequentialGroup()  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
)  
  
        .addComponent(jLabel3)  
  
        .addComponent(jLabel7)  
  
        .addComponent(jLabel10))  
  
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,  
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
)  
  
        .addComponent(jComboBox7,  
javax.swing.GroupLayout.PREFERRED_SIZE,  
javax.swing.GroupLayout.DEFAULT_SIZE,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
  
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING  
)  
)
```

```
        .addComponent(jYearChooser4,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jComboBox6,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addComponent(jButton1))

.addComponent(jRadioButton3)

.addComponent(jRadioButton8))

.addGroup(layout.createSequentialGroup()

.addComponent(jButton5)

.addGap(126, 126, 126))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

.addGroup(layout.createSequentialGroup()

.addComponent(jButton6)

.addGap(40, 40, 40)
```

```
.addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 483,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
G)

.addGroup(layout.createSequentialGroup()

.addComponent(jRadioButton4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 64,
Short.MAX_VALUE)

.addComponent(jComboBox9,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(0, 0, Short.MAX_VALUE)

.addComponent(jButton13)))
```

```
        .addGap(31, 31, 31))

        .addGroup(layout.createSequentialGroup()

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
            .addComponent(jRadioButton5)

            .addComponent(jRadioButton6)

            .addComponent(jRadioButton7)

            .addComponent(jLabel11)

            .addComponent(jLabel12)

            .addComponent(jLabel13))

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

        .addComponent(jComboBox8,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
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)
)

    .addComponent(jComboBox10,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

    .addComponent(jButton4))

    .addGap(41, 41, 41)))

    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

    .addComponent(jButton7)

    .addGap(179, 179, 179)))))

);

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(layout.createSequentialGroup()

        .addContainerGap()

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)
```

```
.addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
.addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)
```

```
.addComponent(jButton1)
```

```
.addComponent(jButton2)
```

```
.addComponent(jButton3))
```

```
.addGap(18, 18, 18)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)
```

```
.addComponent(jProgressBar3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
        .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)
)

.addGroup(layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 214,
Short.MAX_VALUE)

.addComponent(filler1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addComponent(jButton12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jLabel2)

.addComponent(jLabel6)

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jComboBox4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jComboBox1)

.addComponent(jButton8))

.addGap(76, 76, 76))

.addGroup(layout.createSequentialGroup()

.addGap(54, 54, 54)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jLabel5)
```

```
.addComponent(jButton9))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jComboBox2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(15, 15, 15)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jButton10)

.addComponent(jLabel9))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jComboBox3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(20, 20, 20)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jButton11)
```

```
.addComponent(jLabel8))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jComboBox5,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
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)
)

.addComponent(layout.createSequentialGroup()
.addGroup(layout.createParallelGroup())
.addComponent(jRadioButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jRadioButton2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jRadioButton3)

.addGap(3, 3, 3)
```

```
.addComponent(jRadioButton8)

.addGap(18, 18, 18)

.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(14, 14, 14)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
)

.addComponent(jLabel10)
```

```
        .addComponent(jComboBox7,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jLabel7)

.addComponent(jComboBox6,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jButton6)

.addComponent(jButton5))

.addGroup(layout.createSequentialGroup()
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
)

    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

        .addGap(79, 79, 79)

        .addComponent(jLabel11)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

        .addComponent(jComboBox8,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(jButton4))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addComponent(jLabel13)

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addComponent(jComboBox10,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addComponent(jLabel12)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jButton4)

.addComponent(jComboBox9,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(18, 18, 18)

.addComponent(jButton5)

.addGap(18, 18, 18)

.addComponent(jButton6)

.addGap(18, 18, 18)

.addComponent(jButton7))
```

```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

        .addGap(56, 56, 56)

        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 232,
javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
E)

.addComponent(jButton7)

.addComponent(jButton13)))

.addContainerGap(89, Short.MAX_VALUE))

);

pack();

}// </editor-fold>

// Acciones al pulsar botón descargar ...

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// Obtenemos el posible nombre del archivo a descargar del jTextField1

```

```

String ficheroAno= jTextField1.getText(); // cadena con el año del fichero que queremos descargar

if ("all_programador".equals(ficheroAno)) {// si ponemos all_programador descarga todos o 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";

        JFileChooser selectorArchivos = new JFileChooser();

        JOptionPane.showMessageDialog(this,
rb.getString("rutaDescarga"), "", JOptionPane.INFORMATION_MESSAGE);

        int seleccionD = selectorArchivos.showSaveDialog(this);

        if (seleccionD == JFileChooser.APPROVE_OPTION)

        {

            File ficheroD = selectorArchivos.getSelectedFile();

            String ruta = ficheroD+enteroString+".csv.gz";

            DownLoadFichero descarga = new DownLoadFichero(urlDatos, ruta, this, rb);

            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 pedimos

// 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() + ".cs
v.gz";

String nombreArchivo = jTextField1.getText() + ".csv.gz"; // renombramos 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

JOptionPane.showMessageDialog(this,
rb.getString("rutaDescarga"), "", JOptionPane.INFORMATION_MESSAGE);

```

```

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,rb); //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 {

JOptionPane.showMessageDialog(this,

```

```

rb.getString("rutasDescomprimir"), "", JOptionPane.INFORMATION_MESSAGE);

JFileChooser selectorArchivos2 = new JFileChooser();

// muestra el cuadro de diálogo de archivos, para que el usuario pueda elegir el archivo a
abrir

selectorArchivos2.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

// indica cual fue la accion de usuario sobre el jfilechooser

int resultado = selectorArchivos2.showOpenDialog(this);

archivo = selectorArchivos2.getSelectedFile(); // obtiene el archivo seleccionado

// muestra error si es inválido

if ((archivo == null) || (archivo.getName().equals("")))) {

    JOptionPane.showMessageDialog(this, rb.getString("nombreInvalido"),
    rb.getString("nombreInvalido"), 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];

```

```

JOptionPane.showMessageDialog(this, rb.getString("rutaDescomprimido"),
"", JOptionPane.INFORMATION_MESSAGE);

int seleccion = selectorArchivos2.showSaveDialog(this);

if (seleccion == JFileChooser.APPROVE_OPTION)

{

String fichero = selectorArchivos2.getSelectedFile().getPath();

String rutaDescarga = fichero+"/"+nombreArchivoSinExtension;

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,rb); // 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

    JOptionPane.showMessageDialog(this,
        rb.getString("rutasGuardar"), "", JOptionPane.INFORMATION_MESSAGE);

    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 accion 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,rb.getString("debeSerCsv"));// 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);

```

```

LeerMongoRunnable progresoMongo = new
LeerMongoRunnable(BASE_DATOS,colecion,this,jComboBox1,rb);

Thread thread2 = new Thread(progresoMongo);

thread2.start();

} else // si no tiene extensión .csv saca emergente de error

JOptionPane.showMessageDialog(this,rb.getString("debeSerCsv")); // saca
emergente con fallo

} // fin else

}

//boton enviar de los 4 radio botons paramongo graficar, limpiar tabla y crear archivos
arff

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    int yearIni;

    int yearFin;

    String mensaje=rb.getString("revisarAnhos"); // mensaje para emergente

    yearIni=jYearChooser3.getValue(); //recoge año seleccionado por jYearChooser1

    yearFin=jYearChooser4.getValue(); //recoge año seleccionado por jYearChooser2

    String bd=(String) jComboBox6.getSelectedItem();

```

```
int variable=jComboBox7.getSelectedIndex();

String variableS="total_p";

switch (variable) {

    case 0:

        variableS="tmin";

        break;

    case 1:

        variableS="tmin";

        break;

    case 2:

        variableS="tmax";

        break;

    case 3:

        variableS="tmax";

        break;

    case 4:

        variableS="rainfall";

        break;
}
```

```

}

if (jRadioButton1.isSelected()) { // jRadioButton1 de borrar columnas tabla (limpiar
tabla)

    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

    {

ArrayList<String> resultadoF=null;

// imprimir resultados de completitud de datos

for (int anho=yearIni; anho<=yearFin;anho=anho + 1 )

{

resultadoF=ConsultaWhere.resultado("opciones",
"fiabilidad","fiabilidad","$fiabilidad","bd",bd,"variable",variableS,"year",Integer.toStrin
g(anho), this,rb);

String resultadoFiabilidad=null;

try{resultadoFiabilidad=substring(resultadoF.get(0),0,4);

JOptionPane.showMessageDialog(this, rb.getString("infoFiabilidadp1")+anho+
" "+rb.getString("infoFiabilidadp2")+resultadoFiabilidad+"%.",rb.getString("info"),
JOptionPane.INFORMATION_MESSAGE);

```

```

    }

    catch(Exception ex) {

        JOptionPane.showMessageDialog(this, rb.getString("error1")+" "+anho+
"+rb.getString("error2"));

    }

}

ConsultaGraficar.consultaGraficar(yearIni,yearFin,bd,variable,rb);}

else JOptionPane.showMessageDialog(this, mensaje);// saca emergente para
revisarfechas

// boton para generar fichero arff para timeseries con los datos de los años que
seleccionemos

}else

if (jRadioButton3.isSelected()){//jRadioButton3 genera fichero arff

    JOptionPane.showMessageDialog(this, rb.getString("rutaGuardrararff"),
"",JOptionPane.INFORMATION_MESSAGE);

    JFileChooser selectorArchivos3 = new JFileChooser();

selectorArchivos3.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

    int seleccion = selectorArchivos3.showSaveDialog(this);

    String rutaDescarga = "E:/8_tfg/clima.arff";
}

```

```

if (seleccion == JFileChooser.APPROVE_OPTION)

{
    String fichero = selectorArchivos3.getSelectedFile().getPath();

    String punto= fichero.substring(fichero.length()-4,fichero.length()-3);

    String punto2= fichero.substring(fichero.length()-5,fichero.length()-4);

    System.out.println(punto);

    System.out.println(punto2);

    if(punto.equals(".")||punto2.equals("."))

        rutaDescarga = fichero;

    else

        rutaDescarga = fichero+"/"+"DM_timeseries.arff"; }           try {

}

if (yearIni<=yearFin)

FicheroArff.ficheroArff(yearIni,yearFin,bd,variable,rb,rutaDescarga);

else JOptionPane.showMessageDialog(this, mensaje);

}catch (ParseException ex) {

Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null,
ex);

}

```

```

    }

    // boton para generar fichero arff para clasificación con los datos de los años que
    seleccionemos

    else

        if (jRadioButton8.isSelected()){

            //decir al usuario que primero tiene que elegir estaciones

            if(estacionesClasificacion.isEmpty())

                {JOptionPane.showMessageDialog(this, rb.getString("noHayEstaciones"),
                "",JOptionPane.INFORMATION_MESSAGE); }

            //pedir al usuario elegir ruta para guardar

            else

                {JOptionPane.showMessageDialog(this, rb.getString("rutaGuardrarff"),
                "",JOptionPane.INFORMATION_MESSAGE);

                JFileChooser selectorArchivos3 = new JFileChooser();

                selectorArchivos3.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

                int seleccion = selectorArchivos3.showSaveDialog(this);

                String rutaDescarga = "E:/8_tfg/clima.arff";

                if (seleccion == JFileChooser.APPROVE_OPTION)

                {

```

```

String fichero = selectorArchivos3.getSelectedFile().getPath();

String punto= fichero.substring(fichero.length()-4,fichero.length()-3);

String punto2= fichero.substring(fichero.length()-5,fichero.length()-4);

System.out.println(punto);

System.out.println(punto2);

if(punto.equals(".")||punto2.equals("."))

rutaDescarga = fichero;

else

rutaDescarga = fichero+"/"+"DM_classify.arff"; }

rutaDescarga = fichero+"/"+"DM_classify.arff"; }

try {

if (yearIni<=yearFin)

FicheroArff.ficheroArffClass(yearIni,yearFin,variable,rb,estacionesClasificacion,rutaDe
scarga,this);

else JOptionPane.showMessageDialog(this, mensaje);

} catch (ParseException ex) {

Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null,
ex);

}

```

```

        }

    }

// botón para predecir por uno de los cuatro métodos, controla los cuatro radiobuttons

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

// 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 accion de usuario sobre el jfilechooser

JOptionPane.showMessageDialog(this, rb.getString("rutaCoger"),
"", JOptionPane.INFORMATION_MESSAGE);

selectorArchivosArff.showOpenDialog(this);

archivo = selectorArchivosArff.getSelectedFile(); // obtiene el archivo seleccionado

String ruta = archivo.getName(); // path del archivo

// compruebo si es arrf el archivo

String[] token = ruta.split("\\."); // separa archivo

int cantidadToken = token.length;

String extension = token[cantidadToken - 1];

```

```

if("arff".equals(extension)){
    String metodo;
    int analisis=jComboBox8.getSelectedIndex();
    int variable=jComboBox9.getSelectedIndex();
    // GaussianProcesses
    if (jRadioButton4.isSelected()) {
        metodo="GaussianProcesses";
        datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this,analisis,variable);
        // 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,analisis,variable);
            // modelo.addColumn("mes",mes);
            modelo.addColumn(metodo, datos);
        }else // SMOreg

```

```

if (jRadioButton6.isSelected()){

    metodo="SMOreg";

    datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this,analisis,variable);

    // modelo.addColumn("mes",mes);

    modelo.addColumn(metodo, datos);

}else //MultilayerPerceptron

{metodo="MultilayerPerceptron";

    datos=TimeSeriesTFG.timeSeriesTFG(metodo,archivo,this,analisis,variable);

    // modelo.addColumn("mes",mes);

    modelo.addColumn(metodo, datos);

}

}else

JOptionPane.showMessageDialog(this,"El archivo no es -arff");// saca emergente
con el error

}

//filtrar las listas desplegables de paises y estaciones

private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {


```

```
JComboBox cb = (JComboBox)evt.getSource();

String region = (String)cb.getSelectedItem();

VerTabla.listaComboBox(jComboBox3,"opciones","paises",rb.getString("pais"),rb.getString("$pais"),rb.getString("region"),region,this,rb); // TODO add your handling code here:

}
```

```
//filtrar las listas desplegables de estaciones

private void jComboBox3ActionPerformed(java.awt.event.ActionEvent evt) {

JComboBox cb = (JComboBox)evt.getSource();

String pais = (String)cb.getSelectedItem();

VerTabla.listaComboBox(jComboBox5,"opciones","estaciones","estacion","$estacion",rb.getString("pais"),pais, this,rb);

VerTabla.listaComboBox(jComboBox10,"opciones","estaciones","estacion","$estacion"
,rb.getString("pais"),pais, this,rb);

}
```

//boton para ver la posicion de la estacion

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {

String estacion = (String)jComboBox5.getSelectedItem();

ArrayList<String> resultado=null;
```

```

resultado=ConsultaWhere.resultado("opciones", "estaciones", "latitud",
"$latitud","estacion",estacion, this, rb);

String latitud=resultado.get(0);

int largo=latitud.length();

String lat1=latitud.substring(0, largo-4);

String lat2=latitud.substring(largo-4);

resultado=ConsultaWhere.resultado("opciones", "estaciones", "longitud",
"$longitud","estacion",estacion, this, rb);

String longitud=resultado.get(0);

largo=longitud.length();

String lon1=longitud.substring(0, largo-4);

String lon2=longitud.substring(largo-4);

try {

    java.awt.Desktop.getDesktop().browse(new
URI("https://www.google.es/maps/place/@"+lat1+"."+lat2+","+lon1+"."+lon2+",15z"))

};

} catch (URISyntaxException ex) {

    Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

    Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);
}

```

```

}

}

//creacion de base de datos de region/pais/estacion

private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {

    CrearBDMongo.Crear((String)jComboBox2.getSelectedItem(),jComboBox4,
jComboBox6, this, rb);

}

private void jButton10ActionPerformed(java.awt.event.ActionEvent evt) {

    CrearBDMongo.Crear((String)jComboBox3.getSelectedItem(),jComboBox4,
jComboBox6, this, rb);

}

private void jButton11ActionPerformed(java.awt.event.ActionEvent evt) {

    CrearBDMongo.Crear((String)jComboBox5.getSelectedItem(),jComboBox4,
jComboBox6, this, rb);

}

//botón para pasar datos anuales a bases de datos particulares de region/pais/estacion

```

```
private void jButton12ActionPerformed(java.awt.event.ActionEvent evt) {  
  
String anhoElegido=jComboBox1.getSelectedItem().toString();  
  
String bdElegida= jComboBox4.getSelectedItem().toString();  
  
String bdElegidaEspacios=bdElegida.replace("_"," ");  
  
String tipo=new String();  
  
if(isDigit(bdElegidaEspacios.charAt(5)))  
  
{tipo="estacion";}  
  
else  
  
{  
  
tipo=ConsultaWhere.resultado("opciones", "tipo", "tipo", "$tipo", "bd",  
bdElegidaEspacios, this, rb).get(0);  
  
}  
  
//metodo que traslada, filtra datos y crea datos agrupados  
  
ColeccionFinalN nuevaCol=new ColeccionFinalN(anhoElegido,tipo, bdElegida,  
this,rb);  
  
}  
  
//se hacen visibles los campos específicos al elegir técnica de DM  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
int analisis=jComboBox8.getSelectedIndex();

if (analisis !=2)

{

jLabel12.setVisible(true);

jRadioButton4.setVisible(true);

jRadioButton5.setVisible(true);

jRadioButton6.setVisible(true);

jRadioButton7.setVisible(true);

jButton7.setVisible(true);

jComboBox9.setVisible(true);

jLabel13.setVisible(false);

jComboBox10.setVisible(false);

jButton13.setVisible(false);

}

else

{

jLabel12.setVisible(false);

jRadioButton4.setVisible(false);

jRadioButton5.setVisible(false);
```

```

jRadioButton6.setVisible(false);

jRadioButton7.setVisible(false);

jButton7.setVisible(false);

jComboBox9.setVisible(false);

jLabel13.setVisible(true);

jComboBox10.setVisible(true);

jButton13.setVisible(true);

JOptionPane.showMessageDialog(this, rb.getString("usaFiltros"),
"", JOptionPane.INFORMATION_MESSAGE);

};

}

//se guardan las estaciones elegidas para el modelo de clasificacion (DM)

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

String estacionElegida=(String)jComboBox6.getSelectedItem();

if(isDigit(estacionElegida.charAt(5)))

estacionesClasificacion.add(estacionElegida);

else

```

```

JOptionPane.showMessageDialog(this, rb.getString("soloEstacion"),
"", JOptionPane.INFORMATION_MESSAGE);

}

//tareas de DM- clasificacion

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {

JFileChooser selectorArchivosArff = new JFileChooser();

selectorArchivosArff.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);

// indica cual fue la accion de usuario sobre el jfilechooser

JOptionPane.showMessageDialog(this, rb.getString("rutaCoger"),
"", JOptionPane.INFORMATION_MESSAGE);

selectorArchivosArff.showOpenDialog(this);

String rutaDescarga = selectorArchivosArff.getSelectedFile().getPath();

archivo = selectorArchivosArff.getSelectedFile(); // obtiene el archivo seleccionado

String ruta =archivo.getName();           // path del archivo

// compruebo si es arrf el archivo

String[] token = ruta.split("\\."); // separa archivo

int cantidadToken = token.length;

String extension = token[cantidadToken - 1];

```

```

if("arff".equals(extension))

{

String estacion=(String)jComboBox10.getSelectedItem();

System.out.println(estacion);

File test;

try {

//se crea archivo de testeo

test = FicheroArff.ficheroArffTest(rb, rutaDescarga, estacion, this);

try {

//ejecución de algoritmo

datos=Clasificar.clasificar(archivo,test,this);

String metodo=rb.getString("clas");



//visualización de datos

modelo.addColumn(metodo, datos);

} catch (Exception ex) {

Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

}

} catch (ParseException ex) {

```

```

        Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

    } catch (IOException ex) {

        Logger.getLogger(BigDataTFG.class.getName()).log(Level.SEVERE, null, ex);

    }

}

else

{ JOptionPane.showMessageDialog(this,"El archivo no es -arff");}

}

// 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;

}

```

```

// método get retorna barra de progreso para la clase LeerMongoRunnable

public JProgressBar getPgrAvance2() {

    return jProgressBar3;

}

/***
 * @param args the command line arguments
 * Método main
 */

public static void main(String args[]) {

    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.UnsupportedLookAndFeel ex) {
    java.util.logging.Logger.getLogger(BigDataTFG.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
}

// crea y visualiza Jpannel 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.Box.Filler filler1;

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton10;

private javax.swing.JButton jButton11;

private javax.swing.JButton jButton12;

private javax.swing.JButton jButton13;

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;

private javax.swing.JButton jButton8;

private javax.swing.JButton jButton9;

public javax.swing.JComboBox<String> jComboBox1;

private javax.swing.JComboBox<String> jComboBox10;

private javax.swing.JComboBox<String> jComboBox2;

private javax.swing.JComboBox<String> jComboBox3;
```

```
private javax.swing.JComboBox<String> jComboBox4;  
  
private javax.swing.JComboBox<String> jComboBox5;  
  
private javax.swing.JComboBox<String> jComboBox6;  
  
private javax.swing.JComboBox<String> jComboBox7;  
  
private javax.swing.JComboBox<String> jComboBox8;  
  
private javax.swing.JComboBox<String> jComboBox9;  
  
private javax.swing.JLabel jLabel1;  
  
private javax.swing.JLabel jLabel10;  
  
private javax.swing.JLabel jLabel11;  
  
private javax.swing.JLabel jLabel12;  
  
private javax.swing.JLabel jLabel13;  
  
private javax.swing.JLabel jLabel2;  
  
private javax.swing.JLabel jLabel3;  
  
private javax.swing.JLabel jLabel4;  
  
private javax.swing.JLabel jLabel5;  
  
private javax.swing.JLabel jLabel6;  
  
private javax.swing.JLabel jLabel7;  
  
private javax.swing.JLabel jLabel8;  
  
private javax.swing.JLabel jLabel9;
```

```
private javax.swing.JProgressBar jProgressBar1;

private javax.swing.JProgressBar jProgressBar2;

private javax.swing.JProgressBar jProgressBar3;

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.JRadioButton jRadioButton8;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable jTable1;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField3;

private com.toedter.calendar.JYearChooser jYearChooser3;

private com.toedter.calendar.JYearChooser jYearChooser4;

// End of variables declaration      }//fin clase
```

Anexo 3: Código de la clase LeerMongoRunnable.

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.util.ResourceBundle;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import org.bson.Document;

public class LeerMongoRunnable implements Runnable{
    private final String baseDatos;
    private final String colecion;
    private final BigDataTFG parent;
    String documento;
    private final JComboBox combo;
    private final ResourceBundle rb;
```

```

public LeerMongoRunnable (String baseDatos,String colecion,BigDataTFG parent,
JComboBox combo, ResourceBundle rb){

this.parent = parent;

this.baseDatos = baseDatos;

this.colecion = colecion;

this.combo=combo;

this.rb=rb;};

@Override

public void run(){

synchronized(this) {

//Conexión a la base de datos

try {// PASO 1: Conexión al Server de MongoDB Pasandole el host y el puerto

String documentoTemp="0";

parent.getPgrAvance2().setString("");

parent.getPgrAvance2().setMinimum(0);

parent.getPgrAvance2().setMaximum(45000);

parent.getPgrAvance2().setValue(0);

MongoClient mongoClient = new MongoClient("localhost", 27017);

```

```

// PASO 2: Conexión a la base de datos

MongoDatabase db = mongoClient.getDatabase(baseDatos);

// PASO 3: Obtenemos una colección para trabajar con ella

MongoCollection<Document> collection = db.getCollection(coleccion);

long contadorDocumento;

String documento = "1";

//se comparan las últimas dos consultas

while (!(documento.equals(documentoTemp)))

{

//se guarda en documentoTemp la consulta anterior

documentoTemp=documento;

wait(10000);

contadorDocumento = collection.countDocuments();

//se rellena la barra de progreso en función de número de documentos

Long progreso=(contadorDocumento/1000);

String progresoS = Long.toString(progreso);

int progresoI=Integer.parseInt(progresoS);

//se guarda en documento la consulta actual

```

```

documento = Long.toString(contadorDocumento);

parent.getPgrAvance2().setValue(progresoI);

}

combo.removeAllItems();// elimina todos item para no repetir las colecciones

// se actualiza el Combo Box con las colecciones de los años

VerColecciones.listaComboBox(combo,"tfg");

parent.getPgrAvance2().setValue(45000);

parent.getPgrAvance2().setString(rb.getString("CargaCompleta"));

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(),

rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);

}

}

```

Anexo 4: Código de la clase CrearBDMongo.

```
package bigdatatfg;

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.client.MongoDatabase;

import java.util.ResourceBundle;

import javax.swing.JComboBox;

import javax.swing.JOptionPane;

public class CrearBDMongo {

    public static void Crear (String nombreDB, JComboBox combo, JComboBox
    combo2,BigDataTFG parent, ResourceBundle rb)

    {

        try (MongoClient client = new MongoClient("localhost", 27017)) {

            String nombreValido=nombreDB.replace(" ", "_");

            System.out.println(nombreValido);

            MongoDatabase database2 = client.getDatabase(nombreValido);

            database2.createCollection("total_p");

            database2.createCollection("total_tmin");

        }

    }

}
```

```

database2.createCollection("total_tmax");

combo.removeAllItems();// elimina todos item para no repetir las colecciones

VerBasesDeDatos.listaComboBox(combo);

combo2.removeAllItems();// elimina todos item para no repetir las colecciones

VerBasesDeDatos.listaComboBox(combo2);

}

catch (MongoException exp)

{JOptionPane.showMessageDialog(parent, exp.getMessage(),

rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);}

}// fin método }//fin clase

```

Anexo 5: Código de la clase VerTabla.

```

package bigdatatfg;

import com.mongodb.Block;

import com.mongodb.MongoClient;

import com.mongodb.MongoException;

import com.mongodb.client.MongoCollection;

import com.mongodb.client.MongoDatabase;

```

```

import java.util.ArrayList;
import java.util.ResourceBundle;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import org.bson.Document;

public class VerTabla {
    ArrayList<String> colecionCombo;

    public static void listaComboBox(JComboBox jComboBox1, String basedatos, String tabla, String campo, String campo2, BigDataTFG parent, ResourceBundle rb ) {
        ArrayList<String> colecionCombo = new ArrayList<>();
        jComboBox1.removeAllItems();//se elimina para no repetir
        try (MongoClient client = new MongoClient("localhost", 27017)) {
            MongoDB database = client.getDatabase(basedatos);
            MongoCollection<Document> collection = database.getCollection(tabla);
            Document query = new Document();//creación de consulta
            Document projection = new Document();
            projection.append(campo, campo2);
        }
    }
}

```

```

projection.append("_id", 0);

Document sort = new Document();

sort.append(campo, 1);      //ordenación

Block<Document> processBlock = new Block<Document>() {

    @Override

    public void apply(final Document document) {

        colecionCombo.add(document.get(campo).toString());

    }

};

collection.find(query).projection(projection).sort(sort).forEach(processBlock);

} catch (MongoException exp) {

    JOptionPane.showMessageDialog(parent, exp.getMessage(),

        rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);

}

int x=colecionCombo.size();

//va añadiendo todas las colecciones en jComboBox

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

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

}

```

```
}

}// fin método
```

```
public static void listaComboBox(JComboBox jComboBox1, String basedatos, String
tabla, String campo, String campo2, String q1, String q2, BigDataTFG parent,
ResourceBundle rb ) {

ArrayList<String> colecionCombo = new ArrayList<>();

jComboBox1.removeAllItems();

try (MongoClient client = new MongoClient("localhost", 27017)) {

    MongoDatabase database = client.getDatabase(basedatos);

    MongoCollection<Document> collection = database.getCollection(tabla);

    Document query = new Document();

    query.append(q1, q2);//las condiciones del filtro, el campo por el que se filtra y su
valor

    Document projection = new Document();

    projection.append(campo, campo2);

    projection.append("_id", 0);

    Document sort = new Document();

    sort.append(campo, 1);

    Block<Document> processBlock = new Block<Document>() {
```

```

@Override

public void apply(final Document document) {

    colecionCombo.add(document.get(campo).toString());

}

};

collection.find(query).projection(projection).sort(sort).forEach(processBlock);

} catch (MongoException exp) {

    JOptionPane.showMessageDialog(parent, exp.getMessage(),

        rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);

}

int x=colecionCombo.size();

//va añadiendo todas las colecciones en jComboBox

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

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

}// fin método

}//fin clase

```

Anexo 6: Código de la clase ConsultaWhere

```
package bigdatatfg;

import com.mongodb.Block;
import com.mongodb.MongoClient;
import com.mongodb.MongoException;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import java.util.ArrayList;
import java.util.ResourceBundle;
import javax.swing.JOptionPane;
import org.bson.Document;

public class ConsultaWhere {

    public static ArrayList<String> resultado(String basedatos, String tabla, String campo,
        String campo2, String q1, String q2, BigDataTFG parent, ResourceBundle rb) {
        ArrayList<String> resultado = new ArrayList<>();
    }
}
```

```
try (MongoClient client = new MongoClient("localhost", 27017)) {  
  
    MongoDatabase database = client.getDatabase(basedatos);  
  
    MongoCollection<Document> collection = database.getCollection(tabla);  
  
    Document query = new Document();  
  
    query.append(q1, q2);  
  
    Document projection = new Document();  
  
    projection.append(campo, campo2);  
  
    projection.append("_id", 0);  
  
    Document sort = new Document();  
  
    sort.append(campo, 1);  
  
  
  
    Block<Document> processBlock = new Block<Document>() {  
  
        @Override  
  
        public void apply(final Document document) {  
  
            resultado.add(document.get(campo).toString());  
  
            System.out.println(document.get(campo));  
  
        }  
    };  
}
```

```

};

collection.find(query).projection(projection).sort(sort).forEach(processBlock);

}

} catch (MongoException exp) {

JOptionPane.showMessageDialog(parent, exp.getMessage(),

rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);

}

return resultado;

}// fin método

public static ArrayList<String> resultado(String basedatos, String tabla, String campo,
String campo2, String q1, String q2, String q3, String q4, String q5, String q6,
BigDataTFG parent, ResourceBundle rb) {

ArrayList<String> resultado = new ArrayList<>();

try (MongoClient client = new MongoClient("localhost", 27017)) {

MongoDatabase database = client.getDatabase(basedatos);

MongoCollection<Document> collection = database.getCollection(tabla);

```

```

Document query = new Document();

query.append(q1, q2);

query.append(q3, q4);

query.append(q5, q6);

Document projection = new Document();

projection.append(campo, campo2);

projection.append("_id", 0);

Document sort = new Document();

sort.append(campo, 1);

Block<Document> processBlock = new Block<Document>() {

    @Override

    public void apply(final Document document) {

        resultado.add(document.get(campo).toString());

        System.out.println(document.get(campo));

    }

};

collection.find(query).projection(projection).sort(sort).forEach(processBlock);

```

```
        } catch (MongoException exp) {  
  
            JOptionPane.showMessageDialog(parent, exp.getMessage(),  
  
                rb.getString("errorConexion"), JOptionPane.ERROR_MESSAGE);  
  
        }  
  
    return resultado;  
  
}// fin método  
  
}//fin clase
```

Anexo 7: Código de la clase VerBasesDeDatos.

```
package bigdatatfg;

import com.mongodb.MongoClient;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JComboBox;

public class VerBasesDeDatos {
    ArrayList<String> colecionCombo;
    public static void listaComboBox(JComboBox jComboBox1) {
        ArrayList<String> colecionCombo = new ArrayList<>();
        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 'tf'
            List<String> dbs = mongoClient.getDatabaseNames();
```

```
int x=dbs.size();

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

if (((!(dbs.get(y).equals("opciones")))) && ((!(dbs.get(y).equals("tfg"))))&&
((!(dbs.get(y).equals("admin"))))&& ((!(dbs.get(y).equals("config"))))&&
((!(dbs.get(y).equals("local")))))

{jComboBox1.addItem(dbs.get(y));}

} } }// fin método

}
```

Anexo 8: Código de la clase ColeccionFinalN.

```
package bigdatatfg;

import com.mongodb.MongoClient;
import com.mongodb.MongoNamespace;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
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 java.util.Arrays;
import java.util.List;
```

```

import java.util.ResourceBundle;

import javax.swing.JOptionPane;

import org.bson.Document;

public class ColeccionFinalN {

    private final BigDataTFG parent;

    private final String filtrarColecion;

    ColeccionFinalN(String filtrarColecion, String buscado, String valor, BigDataTFG parent, ResourceBundle rb)

    {

        //se consulta mongoDB para extraer las estaciones de la region/país/estacion elegida por el usuario

        ArrayList<String> estaciones = new ArrayList<>();

        int mes=1;

        //se busca las estaciones pertenecientes a pais/región

        estaciones=ConsultaWhere.resultado("opciones", "estaciones", "estacion",
        "$estacion", buscado, valor.replace("_", " "), parent, rb);

        float numEstaciones=estaciones.size();

        String estacionCod[] = new String[estaciones.size()];

        for (int j = 0; j < estaciones.size(); j++)

```

```

{estacionCod[j] = estaciones.get(j); }

//el año seleccionado

this.filtrarColeccion = filtrarColeccion;

this.parent = parent;

try

{

MongoClient mongoClient = new MongoClient();

//acceso a base de datos con los años descargados

MongoDatabase database = mongoClient.getDatabase("tfg");

//acceso a base de datos seleccionada

MongoDatabase databaseDestino = mongoClient.getDatabase(valor);

//acceso a base de datos "opciones", a colección "fiabilidad"

//ahí se guardan los porcentajes de completitud del dato

MongoDatabase databaseFiabilidad = mongoClient.getDatabase("opciones");

MongoCollection<Document> collectionFiabilidad =
databaseFiabilidad.getCollection("fiabilidad");

// Acceso a colección aaaa

MongoCollection<Document> collection = database.getCollection(filtrarColeccion);

//guardar los datos de precipitaciones

```

```

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;

it.into(docs);

MongoCollection<Document> col = databaseDestino.getCollection(colecionFinal);

docs.stream().map((doc) -> {

// inserta los documentos en la colección aaaa

col.insertOne((org.bson.Document) (Document) doc);

return doc;

}).forEachOrdered((doc) -> {

});

long numDocumentos=col.countDocuments();

//cálculo de porcentaje de datos disponibles

double fiabilidad=numDocumentos/(numEstaciones*365)*100;

if(fiabilidad>100)

{fiabilidad=100;}

```

```

Document document1 = new Document("bd",valor).

append("year",filtrarColecion).

append("variable","rainfall").

append("fiabilidad",fiabilidad);

collectionFiabilidad.insertOne(document1);

//guardar los datos de temperatura mínima

FindIterable it2 = collection.find(and(in("ID_estacion",
estacionCod),eq("elemento","TMIN"))).

// elemento que se guardan son lo projection

projection(fields(include("fecha","ID_estacion", "valor_dato"), excludeId()));

ArrayList<Document> docs2 = new ArrayList();

String colecionFinal2=filtrarColecion+"TMIN";

it2.into(docs2);

MongoCollection<Document> col2 = databaseDestino.getCollection(colecionFinal2);

docs2.stream().map((doc) -> {

// inserta los documentos en la colección aaaa

col2.insertOne((org.bson.Document) (Document) doc);

return doc;
}

```

```

}).forEachOrdered((doc) -> {

});

numDocumentos=col2.countDocuments();

fiabilidad=numDocumentos/(numEstaciones*365)*100;

if(fiabilidad>100)

{fiabilidad=100;}

Document document2 = new Document("bd",valor).

append("year",filtrarColecion).

append("variable","tmin").

append("fiabilidad",fiabilidad);

collectionFiabilidad.insertOne(document2);

//guardar los datos de temperatura máxima

FindIterable it3 = collection.find(and(in("ID_estacion",
estacionCod),eq("elemento","TMAX"))).

// elemento que se guardan son lo projection

projection(fields(include("fecha","ID_estacion", "valor_dato"), excludeId()));

ArrayList<Document> docs3 = new ArrayList();

String colecionFinal3=filtrarColecion+"TMAX";

```

```

it3.into(docs3);

MongoCollection<Document> col3 = databaseDestino.getCollection(coleccionFinal3);

docs3.stream().map((doc) -> {

    // inserta los documentos en la colección aaaa

    col3.insertOne((org.bson.Document) (Document) doc);

    return doc;
}).forEachOrdered((doc) -> {

});

numDocumentos=col3.countDocuments();

fiabilidad=numDocumentos/(numEstaciones*365)*100;

if(fiabilidad>100)

{fiabilidad=100;}

Document document3 = new Document("bd",valor).

append("year",filtrarColecion).

append("variable","tmax").

append("fiabilidad",fiabilidad);

collectionFiabilidad.insertOne(document3);

```

```

//cambio de formato de fecha

// primero definimos la projection con los campos que interesan

try (MongoCursor<Document> cur = col.find().iterator()) {

    while (cur.hasNext()) {

        Document doc = cur.next();

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

        int x=(int) list.get(2);

        // va guardando todos los documentos con la fecha en formato ISODate

    }
}

FormatFecha b =new FormatFecha(filtrarColecion,valor);

try (MongoCursor<Document> cur2 = col2.find().iterator()) {

    while (cur2.hasNext()) {

        Document doc = cur2.next();

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

        int x=(int) list.get(2);

        // va guardando todos los documentos con la fecha en formato ISODate

    }
}

FormatFecha c =new FormatFecha(filtrarColecion+"TMIN",valor);

```

```

//cambio de formato de fecha

// primero definimos la projection con los campos que interesan

try (MongoCursor<Document> cur3 = col3.find().iterator()) {

    while (cur3.hasNext()) {

        Document doc = cur3.next();

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

        int x=(int) list.get(2);

        // va guardando todos los documentos con la fecha en formato ISODATE

    }
}

FormatFecha d =new FormatFecha(filtrarColecion+"TMAX",valor);

//guardar los datos en total_p

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"));

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

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

MongoCollection<Document> total = databaseDestino.getCollection("total_p");

// 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

int seguir=0;

Document doc = cur.next();

List list= new ArrayList();

```

```
while (mes<13)

{
    if (seguir==1)

        {try{doc = cur.next();}

    catch (Exception e1) {}}

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

    int mesCol=(int) list.get(0);

    //compruebo si hay datos para el mes

    if(mes==mesCol)

    {

        seguir=1;

        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("Anho",Integer.parseInt(filtrarColecion)).append("Mes",list.get(0)).append("Litros",litros);total.insertOne(document); }

//si no hay datos en un mes pongo símbolo "?"

else

{

//indico de no avanzar cogiendo documentos de la colección seguir=0;

Document document = new Document("Anho",Integer.parseInt(filtrarColecion)).append("Mes",mes).append("Litros","?");total.insertOne(document);

}

mes=mes+1;

}

//ordenacion de collection

Document sort2 = new Document("$sort", new Document("Anho",1));

```

```

Document project2 = new Document("Anho", true).append("Mes", true).append("Litros",true);

//projection

Document project3 = new Document("$project", project2);

List<Document> pipeline2 = Arrays.asList(project3,sort2);

MongoCursor <Document> docsF = total.aggregate(pipeline2).iterator();

databaseDestino.createCollection("total_pT");

MongoCollection<Document> totalT = databaseDestino.getCollection("total_pT");

while (docsF.hasNext()) {

    Document itF=docsF.next();

    totalT.insertOne(itF);

}

total.drop();

MongoNamespace newName = new MongoNamespace(valor , "total_p");

totalT.renameCollection(newName);

//guardar los datos en total_tmin

cur = col2.aggregate(pipeline).iterator();

// colección donde irán los nuevos documentos

```

```

total = databaseDestino.getCollection("total_tmin");

// bucle que calcula para cada documento la media mensual de temperatura

mes=1;

seguir=0;

doc = cur.next();

while (mes<13) {

    if (seguir==1)

    {try{doc = cur.next();}

    catch (Exception e1) {}}

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

int mesCol=(int) list.get(0);

if(mes==mesCol)

{

seguir=1;

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

double tempF;

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

temp=temp/10;

tempF=(temp*9/5)+32;

```

```

// Crea los documentos que se van a insertar

Document document = new Document("Anho",Integer.parseInt(filtrarColecion)).append("Mes",list.get(0)).append("Temperatura",temp).append("TemperaturaF",tempF);total.insertOne(document);

}

else

{

seguir=0;

Document document = new Document("Anho",Integer.parseInt(filtrarColecion)).append("Mes",mes).append("Temperatura","?").append("TemperaturaF","?");total.insertOne(document);

}

mes=mes+1;

}

```

```

//ordenacion de collection

Document project4 =new Document("Anho", true).append("Mes", true).append("Temperatura",true).append("TemperaturaF",true);

//projection

Document project5 = new Document("$project", project4);

List<Document> pipeline3 = Arrays.asList(project5,sort2);

MongoCursor <Document> docsTF = total.aggregate(pipeline3).iterator();

databaseDestino.createCollection("total_tminT");

MongoCollection<Document> totalTT = databaseDestino.getCollection("total_tminT");

while (docsTF.hasNext()) {

    Document itTF=docsTF.next();

    totalTT.insertOne(itTF);

}

total.drop();

MongoNamespace newName2 = new MongoNamespace(valor , "total_tmin");

totalTT.renameCollection(newName2);

//guardar los datos en total_tmax

cur = col3.aggregate(pipeline).iterator();

```

```

// colección donde irán los nuevos documentos

total = databaseDestino.getCollection("total_tmax");

// bucle que calcula para cada documento la media mensual de temperatura

mes=1;

seguir=0;

doc = cur.next();

while (mes<13) {

    if (seguir==1)

    {try{doc = cur.next();}

    catch (Exception e1) {}}

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

int mesCol=(int) list.get(0);

if(mes==mesCol)

{

seguir=1;

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

double tempF;

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

```

```

temp=temp/10;

tempF=(temp*9/5)+32;

// Crea los documentos que se van a insertar

Document document = new Document("Anho",Integer.parseInt(filtrarColecion)).append("Mes",list.get(0)).append("Temperatura",temp).append("TemperaturaF",tempF);total.insertOne(document);

}

else

{

seguir=0;

Document document = new Document("Anho",Integer.parseInt(filtrarColecion)).append("Mes",mes).append("Temperatura","?").append("TemperaturaF","?");total.insertOne(document);

}

mes=mes+1;

```

```

}

//ordenacion de collection

docsTF = total.aggregate(pipeline3).iterator();

databaseDestino.createCollection("total_tmaxT");

totalTT = databaseDestino.getCollection("total_tmaxT");

while (docsTF.hasNext()) {

    Document itTF=docsTF.next();

    totalTT.insertOne(itTF);

}

total.drop();

MongoNamespace newName3 = new MongoNamespace(valor , "total_tmax");

totalTT.renameCollection(newName3);

mongoClient.close();//cerrar conexión

} catch (Exception ex) { // error

    JOptionPane.showMessageDialog(parent, ex.getMessage(),
    rb.getString("ErrormongoDB"), JOptionPane.ERROR_MESSAGE); } } }

```

Anexo 9: Código de la 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 java.util.ResourceBundle;
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;

// clase para dibujar la gráfica de los años seleccionados

public class ConsultaGraficar{

    // método constructor recibe año inicial y año final para dibujar gráfica

    public static void consultaGraficar(int yearIni,int yearFin, String db, int variable,
    ResourceBundle rb) {

        Bson filtro = new Document("$gte", yearIni).append("$lte", yearFin); //filtro de busqueda

        try ( //para intentar actualizar a iSOJSON la fecha y no en Date

            MongoClient mongoClient = new MongoClient("localhost", 27017)) {

            MongoDB database = mongoClient.getDatabase(db);

            Stringleccion="total_p";

            String medida="mm";

            int dato=3;

            String titulo="titulo";

            switch (variable) {

```

case 0:

colección="total_tmin";

medida="°C";

dato=3;

titulo="titulo2";

break;

case 1:

colección="total_tmin";

medida="°F";

dato=4;

titulo="titulo2";

break;

case 2:

colección="total_tmax";

medida="°C";

dato=3;

titulo="titulo3";

break;

case 3:

```
colección="total_tmax";  
  
medida="°F";  
  
dato=4;  
  
título="título3";  
  
break;  
  
case 4:  
  
colección="total_p";  
  
medida="mm";  
  
dato=3;  
  
título="título";  
  
break;  
  
}  

```

```
MongoCollection<Document> col = database.getCollection(colección);
```

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

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

```
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));

try

{

    Double parametro=(Double) list.get(dato);

    dataset.addValue(parametro,medida,ano+" - "+(Comparable) list.get(2));

}

//obviar los datos que vengan como "?"

catch (Exception err1)

{



}

// crear gráfica...

JFreeChart chart = ChartFactory.createLineChart( // gráfica de lineas

rb.getString(titulo)+db, // Titulo

rb.getString("Mes"), // Etiqueta de datos

medida, // 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

```

Anexo 10: Código de la 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;

// clase para presentarnos en desplegable del combobox las colecciones en la base de
// datos
// tfg y seleccionar la que queremos 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, String basedatos) {
```

```

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(basedatos);

//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));

    System.out.println(colecionCombo.get(y));

}

}

}// fin método

}//fin clase
```

Anexo 11: Código de la clase Descompresor.

```
package bigdatatfg;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.GZIPInputStream;
import javax.swing.JOptionPane;

// 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
```

```

private final ResourceBundle rb;

// constructor recibe el archivo a descomprimir, la ruta de descarga y objeto
BigDataTFG

public Descompresor(File archivo,String rutaDescarga,BigDataTFG parent,
ResourceBundle rb){

this.archivo=archivo;

this.rutaDescarga = rutaDescarga;

this.parent = parent;

this.rb=rb; }

// 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 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(rb.getString("archivoDescomprimido"));

} //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(),  
rb.getString("Errorgz"), JOptionPane.ERROR_MESSAGE);  
  
}  
  
} // hilo descomprimir  
  
}// fin clase
```

Anexo 12: Código de la 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 java.util.ResourceBundle;
import javax.swing.JOptionPane;

//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/yyyy

public class DownLoadFichero implements Runnable {

    private final String direccion; // url
    private final String rutaDescarga; // ruta para descargar fichero
    private final BigDataTFG parent; // objeto
```

```

private final ResourceBundle rb;

// método constructor recibe url, path descarga y objeto BigDataTFG

public DownLoadFichero(String direccion, String rutaDescarga, BigDataTFG parent,
ResourceBundle rb) {//constructor

this.direccion = direccion;

this.rutaDescarga = rutaDescarga;

this.parent = parent;

this.rb=rb;

}

@Override//hilo de descarga

// hilo

public void run() {

try {

URL url = new URL(direccion);// url de descarga

// establecemos conexion

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(rb.getString("descargaCompleta"));

```

```
// cierre de conexion y fichero.  
  
} fichero.close(); // cerrar fichero  
  
} catch (IOException ex) { // error devuelve url que ha fallado  
  
    JOptionPane.showMessageDialog(parent, ex.getMessage(),  
        rb.getString("errorNoEncontrado"), JOptionPane.ERROR_MESSAGE);  
  
}  
  
}// fin hilo  
  
}// fin clase
```

Anexo 13: Código de la 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, String bd) throws ParseException{
    this.colecion = colecion;

    try (MongoClient mongoClient = new MongoClient("localhost", 27017)) {

        MongoDatabase database = mongoClient.getDatabase(bd);

        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());

                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))));

            }
        }
    }
}

//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
```

Anexo 14: Código de la clase MongoDB.

```
package bigdatatfg;

import java.io.File;
import java.io.IOException;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;

// 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 fichero no
cargado", 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

```

Anexo 15: Código de la 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.io.BufferedReader;
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.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import static java.lang.Math.log;
import java.util.Iterator;
```

```
import java.util.ResourceBundle;
import java.util.logging.Level;
import java.util.logging.Logger;
import weka.core.Instances;

public class FicheroArff{

    public static void     ficheroArff(int  anoIni,int  anoFin,String  db,  int  variable,
ResourceBundle rb, String rutaDescarga) throws ParseException{
        String colección="total_p";
        String colección2="total_tmin";
        String colección3="total_tmax";
        FileWriter escribir=null;
        String sSistemaOperativo = System.getProperty("os.name");
        String so=sSistemaOperativo.substring(0,3);
        String saltoLinea="\n";
        if(so.equals("Win"))
            saltoLinea="\r\n";
        System.out.println(sSistemaOperativo);
```

```

try {

    //cabecera del fichero arff

    String saludo="@relation tfg"+saltoLinea;

    String atributo1="@attribute Date date \'yyyy-MM-dd\'"+saltoLinea;

    String atributo2="@attribute precipitacion numeric"+saltoLinea;

    String atributo3="@attribute tmin numeric"+saltoLinea;

    String atributo4="@attribute tmax numeric"+saltoLinea;

    String dato="@data"+saltoLinea;

    try // conexión a mongoDB, base de datos y colección

        (MongoClient mongoClient = new MongoClient("localhost", 27017)) {

            MongoDB database = mongoClient.getDatabase(db);

            MongoCollection<Document> col = database.getCollection(coleccion);

            MongoCollection<Document> col2 = database.getCollection(coleccion2);

            MongoCollection<Document> col3 = database.getCollection(coleccion3);

            //Crear un objeto File clima.arff

            File archivo=new File(rutaDescarga);

            //Crear objeto FileWriter que sera el que nos ayude a escribir sobre archivo

            escribir = new FileWriter(archivo,false);// false sobreescribe y true añade

            //Escribimos en el archivo con el metodo write

```

```

escribir.write(saludo+atributo1+atributo2+atributo3+atributo4+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("Anho", filtro);

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

MongoCursor<Document> cur2 = col2.find(findDocument).iterator();

MongoCursor<Document> cur3 = col3.find(findDocument).iterator();

{

while (cur.hasNext()) {

    Document doc = cur.next();

    Document doc2 = cur2.next();

    Document doc3 = cur3.next();

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

    List list2 = new ArrayList(doc2.values());

    List list3 = new ArrayList(doc3.values());

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

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

```

```

escribir.write("01,");

try{

if((double)list.get(3)==0)

{escribir.write("-4"+",");}

else

{escribir.write(log((double)list.get(3))+",");}

catch (Exception e1)

{escribir.write(list.get(3)+",");

escribir.write(list2.get(3)+",");

escribir.write(list3.get(3)+saltoLinea);

}// fin bucle

escribir.close(); //Cerramos fichero

}mongoClient.close(); //Cerramos la conexion

}

} catch (IOException ex) {

Logger.getLogger(FicheroArff.class.getName()).log(Level.SEVERE, null, ex);

}

// fin método

```

```

public static void ficheroArffClass(int anoIni,int anoFin,int variable, ResourceBundle rb,
ArrayList<String> estacionesClasificacion, String rutaDescarga, BigDataTFG parent)
throws ParseException

{

//comprobacion de SO para definir carácter de salto de línea

String sSistemaOperativo = System.getProperty("os.name");

String so=sSistemaOperativo.substring(0,3);

String saltoLinea="\n";

if(so.equals("Win"))

saltoLinea="\r\n";

String colección="total_p";

String atributo="precipitacion";

//comprobación de variable elegida

if (variable==4)

{colección="total_p";

atributo="precipitacion";}

else if (variable==0 || variable==1)

{colección="total_tmin";

atributo="t_min";}

```

```

else

{colección="total_tmax";

atributo="t_max";}

FileWriter escribir=null;

try {

//cabecera del fichero arff

String saludo="@relation tfg"+saltoLinea;

String atributo1="@attribute anho numeric"+saltoLinea;

String atributo2="@attribute mes numeric"+saltoLinea;

String atributo3="@attribute "+atributo+" numeric"+saltoLinea;

String atributo4="@attribute latitud numeric"+saltoLinea;

String atributo5="@attribute longitud numeric"+saltoLinea;

String atributo6="@attribute altitud numeric"+saltoLinea;

String dato="@data"+saltoLinea;

//iterator de la array de estaciones elegidas

Iterator i = estacionesClasificación.iterator();

String db;

File archivo;

//filtro para recoger datos del periodo indicado

```

```
Bson filtro = new Document("$gte", anoIni).append("$lte", anoFin);

Document findDocument = new Document("Anho", filtro);

//creación del archivo

archivo=new File(rutaDescarga);

escribir = new FileWriter(archivo,false);

escribir.write(saludo+atributo1+atributo2+atributo3+atributo4+atributo5+atributo6+dato);

//bucle de estaciones elegidas

while (i.hasNext())

{

    db=(String)i.next();

    try // conexión a mongoDB

        (MongoClient mongoClient = new MongoClient("localhost", 27017))

    {

        ArrayList<String> resultado=null;

        String latitud;

        String longitud;

        String altitud;
```

```

MongoDatabase database;

MongoCollection<Document> col ;

//recoger dato de latitud, longitud y altitud

resultado=ConsultaWhere.resultado("opciones",      "estaciones",      "latitud",
"$latitud","estacion",db, parent, rb);

latitud=resultado.get(0);

resultado=ConsultaWhere.resultado("opciones",      "estaciones",      "longitud",
"$longitud","estacion",db, parent, rb);

longitud=resultado.get(0);

resultado=ConsultaWhere.resultado("opciones",      "estaciones",      "altitud",
"$altitud","estacion",db, parent, rb);

altitud=resultado.get(0);

//acceso a base de datos de la estacion

database = mongoClient.getDatabase(db);

//acceso a la colección de la variable elegida

col = database.getCollection(coleccion);

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

//se guardan los datos

while (cur.hasNext())

{

```

```

Document doc = cur.next();

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

escribir.write(list.get(1)+",");

escribir.write((list.get(2))+",");

//si es precipitacion se calcula ln

if(variable==4)

{

try

{

if((double)list.get(3)==0)

{escribir.write("-4"+",");}

else

{escribir.write(log((double)list.get(3))+",");}

}

catch (Exception e1)

{escribir.write(list.get(3)+",");}

}

else

```

```

        {escribir.write(list.get(3)+",");}

        escribir.write(latitud+",");

        escribir.write(longitud+",");

        escribir.write(altitud+saltoLinea);

    }// fin bucle

    mongoClient.close(); //Cerramos la conexion

}

}

escribir.close(); //Cerramos fichero

}

catch (IOException ex)

{Logger.getLogger(FicheroArff.class.getName()).log(Level.SEVERE, null, ex);}

}// fin método

```

```

public static File ficheroArffTest(ResourceBundle rb, String rutaDescarga, String
estacion, BigDataTFG parent) throws ParseException, IOException {

String sSistemaOperativo = System.getProperty("os.name");

String so=sSistemaOperativo.substring(0,3);

String saltoLinea="\n";

```

```

if(so.equals("Win"))

saltoLinea="\r\n";

FileWriter escribir;

File archivo;

int largo=rutaDescarga.length();

//acceso al archivo de entrenamiento

archivo=new File(rutaDescarga);

Instances iTrain = new Instances(new BufferedReader(new FileReader(archivo)));

int numInst=iTrain.numInstances();

//acceso a la última instancia, a dato del año

int año=(int)iTrain.instance(numInst-1).value(0);

//establecimiento de año para predicción como +1 de año de archivo de entrenamiento

int anho=año+1;

//creación del archivo de testeo

rutaDescarga=rutaDescarga.substring(0, largo-5)+"Test.arff";

archivo=new File(rutaDescarga);

escribir = new FileWriter(archivo,false);

//acceso a atributo de archivo de entrenamiento

```

```

String atributo=iTrain.attribute(2).name();

//definición de encabezados

String saludo="@relation tfg"+saltoLinea;

String atributo1="@attribute anho numeric"+saltoLinea;

String atributo2="@attribute mes numeric"+saltoLinea;

String atributo3="@attribute "+atributo+" numeric"+saltoLinea;

String atributo4="@attribute latitud numeric"+saltoLinea;

String atributo5="@attribute longitud numeric"+saltoLinea;

String atributo6="@attribute altitud numeric"+saltoLinea;

String dato="@data"+saltoLinea;

escribir.write(saludo+atributo1+atributo2+atributo3+atributo4+atributo5+atributo6+dato);

ArrayList<String> resultado;

String latitud;

String longitud;

String altitud;

//recuperación de datos de latitud, longitud y altitud

resultado=ConsultaWhere.resultado("opciones", "estaciones", "latitud",
"$latitud","estacion",estacion, parent, rb);

latitud=resultado.get(0);

```

```

resultado=ConsultaWhere.resultado("opciones",           "estaciones",           "longitud",
"$longitud", "estacion", estacion, parent, rb);

longitud=resultado.get(0);

resultado=ConsultaWhere.resultado("opciones",           "estaciones",           "altitud",
"$altitud", "estacion", estacion, parent, rb);

altitud=resultado.get(0);

int mes=1;

while (mes<13)

{

escribir.write(anho+",");

escribir.write(mes+",");

escribir.write("?,,");

escribir.write(latitud+",");

escribir.write(longitud+",");

escribir.write(altitud+saltoLinea);

mes=mes+1;

}// fin bucle

escribir.close();

return archivo;

```

}// fin método }// fin clase

Anexo 16: Código de la clase TimeSeriesTFG.

```
package bigdatatfg;

import java.io.*;
import static java.lang.Math.exp;
import java.util.List;
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;

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, int
analisis, int variable) {

    //

    try {

        // path a clima.arffto

        File pathToWineData = archivo;

        // carga datosa

        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

        System.out.println(analisis);

        System.out.println(variable);

        if(analisis==1)

            {forecaster.setFieldsToForecast("precipitacion,tmin,tmax");}

        else if (variable==0)

            {{forecaster.setFieldsToForecast("tmin");} }

        else if (variable==1)

```

```

{{forecaster.setFieldsToForecast("tmin");} }

else if (variable==2)

{{forecaster.setFieldsToForecast("tmax");} }

else if (variable==3)

{{forecaster.setFieldsToForecast("tmax");} }

else if (variable==4)

{{forecaster.setFieldsToForecast("precipitacion");} }

// 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);

    if(analisis==0 || variable==4 )

        //si análisis univariable o variable=precipitacion

    {

        //coger predicción de primera variable (o única si univariable)

        NumericPrediction predForTarget = predsAtStep.get(0);

        //si variable es temperatura en fahrenheit hacer transformacion

        if(variable==3 || variable==1)

            datos [i] =((predForTarget.predicted())*9/5)+32;

        //si variable es precipitacion calcular exponencial

        else if(variable==4)

    {

        double dato=exp(predForTarget.predicted());

        if( dato<0.1)

            datos [i]=0;

```

```

else
    datos [i] =dato;
}

else
    datos [i] =predForTarget.predicted();

}

//si variable es temperatura mínima

else if (variable==0 || variable==1)

{
    NumericPrediction predForTarget = predsAtStep.get(1);

    if(variable==1)

        datos [i] =((predForTarget.predicted())*9/5)+32;

    else

        datos [i] =predForTarget.predicted();

}

//si variable es temperatura máxima

else if (variable==2 || variable==3)

{
    NumericPrediction predForTarget = predsAtStep.get(2);
}

```

```
if(variable==3)

    datos [i] =((predForTarget.predicted())*9/5)+32;

else

    datos [i] =predForTarget.predicted();

}

}

// fin bucle

} catch (Exception ex) {

    for (int i = 0; i < 12; i++) { // pone error en la tabla

        datos [i] ="error";

    }

    return datos;

} // fin método

}// fin clase
```

Anexo 17: Código de la clase Clasificar.

```
package bigdatatfg;

import java.io.*;
import static java.lang.Math.exp;
import weka.classifiers.Classifier;
import weka.classifiers.evaluation.Evaluation;
import weka.core Instances;
import weka.classifiers.meta.Bagging;

public class Clasificar

{
    static Object [] datos = new Object[12];// guarda prediccion

    private BigDataTFG parent; // objeto

    // método constructor recibe el método de algoritmo a aplicar y el fichero arff

    public static Object[] clasificar(File train, File test, BigDataTFG parent) throws Exception

    {
        try
        {
```

```

//archivo de conjunto de entrenamiento

File archivoTrain = train;

//archivo de conjunto de prueba

File archivoTest = test;

Instances iTrain = new Instances(new BufferedReader(new
FileReader(archivoTrain)));

Instances iTest = new Instances(new BufferedReader(new FileReader(archivoTest)));

//atributo a predecir

String atributo=iTrain.attribute(2).name();

//selección de algoritmo

Classifier cls = new Bagging();

//establecer atributo de clase

iTrain.setClassIndex(iTrain.numAttributes() - 4);

iTest.setClassIndex(iTest.numAttributes() - 4);

//ejecución de algoritmo

cls.buildClassifier(iTrain);

Evaluation eval = new Evaluation(iTrain);

//predicción

eval.evaluateModel(cls, iTest);

```

```

//resultados de predicción

for (int i = 0; i < iTest.numInstances(); i++)

{

    double pred = cls.classifyInstance(iTest.instance(i));

    if (atributo.equals("precipitacion"))

    {

        double dato=exp(pred);

        if( dato<0.1)

            datos [i]=0;

        else

            datos [i] =exp(pred);

    }

    else

    {datos [i] =pred; }

}

}

catch (Exception ex)

{

```

```
{  
    datos [i] ="error";  
}  
  
}  
  
return datos;  
  
} // fin método  
}// fin clase
```