Grado en Matemática Computacional

Proyecto Final de Grado

Técnicas estadísticas basadas en la teoría de la decisión aplicadas a la optimización de procesos de control en tecnologías de la información

Autor:
Carla Hernando Fuster

Supervisor:
Ernesto Artola Ferrer
Tutor académico:
José Antonio López Ortí

Fecha de entrega: 14 de Julio de 2015
Curso académico 2014/2015

Resumen

Este documento contiene la descripción, el proceso y el resultado del Trabajo Final de Grado que ha sido desarrollado durante las prácticas externas en la asignatura MT1030.
Esta estancia ha constado de 290 horas presenciales que se han realizado en el edificio Walhalla de la empresa TISSAT, Tecnología e Ingeniería de Sistemas y Servicios Avanzados de Telecomunicaciones, S.A., situado en la Universidad Jaume I (Castellón de la Plana) y otras 160 horas no presenciales dedicadas a la redacción de una propuesta técnica, seis informes quincenales, varias reuniones con el tutor José Antonio López Ortí y la redacción de este proyecto.

El trabajo realizado en la empresa es un programa que calcule el umbral óptimo de un equipo característico de la empresa. Este programa hace uso de un algoritmo matemático basado en el criterio Minimax. Además, realizamos comparaciones con el programa que se hizo hace años basado en el criterio de Bayes, otro criterio característico de la Teoría de la Decisión.

Palabras clave

Teoría de la Decisión, Criterio de Bayes, Criterio Minimax, Estimación vía Kernel, Umbral Óptimo.

Keywords

Decision Theory, Bayesian Problems, Minimax Procedures, Estimation via Kernel, Optimal Threshold.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto presentado en esta memoria se desarrolló durante las 290 horas de estancia en prácticas de la asignatura MT1030 en el edificio Walhalla de la empresa TISSAT, Tecnología e Ingeniería de Sistemas y Servicios Avanzados de Telecomunicaciones, S.A., situado en la Universidad Jaume I, Castellón de la Plana.

En este primer capítulo, se describirá brevemente el funcionamiento de la empresa en la que se ha realizado la estancia en prácticas.
En el segundo, se detallarán los objetivos, la metodología y definición de las tareas realizadas en la empresa, la planificación temporal de esas tareas y la estimación de los recursos utilizados para realizar este proyecto.
El tercer capítulo se centrará en la exposición de los fundamentos de la teoría de la decisión haciendo especial hincapié en los métodos minimax y los métodos Bayesianos.
En el siguiente, se realizará un estudio de los métodos no paramétricos de estimación de funciones de densidad centrándose en los estimadores vía Kernel.
En el quinto capítulo, se explicará de forma esquemática el código en Java y se estudiarán los resultados ayudándose de las gráficas resultantes de la ejecución de los programas.
Y, por último, se escribirán las conclusiones a las que se han llegado con el estudio de los dos modelos.

Haciendo referencia a la empresa, el área de Gestión de Activos Tecnológicos de TISSAT cubre un gran número de servicios, entre ellos uno de sus cometidos principales es el de conseguir no solamente mantener al máximo la operatividad y disponibilidad de los distintos sistemas en producción, sino también su constante vigilancia y estudio para lograr una utilización óptima y obtener el máximo rendimiento de los recursos en producción.

El sistema de monitorización de TISSAT cuenta actualmente con cerca de 4000 monitores implantados para 600 servidores ubicados en sus CPDs (*Centros de Procesos de Datos*) de Valencia, Castellón y Madrid, y aproximadamente 450 remotos en instalaciones de clientes, para plataformas Windows, Linux y SPARC (*Scalable Processor ARChitecture*), así como numerosos

sistemas de gestión de base de datos (comerciales y opensource).

Tissat dispone de un sistema que permite centralizar la monitorización, supervisión y gestión de todos los elementos críticos de su Data Center, tanto los referentes a elementos TIC (*Tecnologías de la Información y la Comunicación*), como pueden ser servidores físicos, firewalls, cabinas de almacenamiento, electrónica de red, etc. como de los elementos no-TIC correspondientes a las infraestructuras tecnológicas. Este sistema se encarga de:

- Generar alertas, si los valores monitorizados superan los umbrales definidos en el sistema de monitorización Nagios. Estas alertas se centralizan en el Sistema de Gestión de Incidencias.
- Registrar dichos datos, de forma que permita conocer la disponibilidad de cada sistema o elemento monitorizado.

Además, Tissat también dispone del Sistema de Gestión de Incidencias. Éste se encarga de la gestión de los servicios que se ofrecen a los clientes, según la norma de calidad ISO (*International Organization for Standardization*) 20000, y recoge tanto las incidencias que se generan de forma automática, a partir de las alertas recibidas desde Nagios, como las incidencias y peticiones reportadas directamente por el cliente o usuario.

Este Sistema de Gestión de Incidencias es una herramienta software desarrollada por la empresa y recoge la experiencia de más de 12 años en Gestión de Servicios TI (*Tecnologías de la Información*). Permite:

- Cumplir con las buenas prácticas recomendadas por ITIL (*Information Technology Infrastructure Library*) y cumplir con las norma ISO 20000.
- Gestionar de forma centralizada todos los flujos de información del Centro de Soporte, a través de los distintos procesos que implementa y módulos que lo componen.
- Registrar, gestionar y explotar las incidencias, problemas y cambios relacionados con la gestión de servicios TIC, pero también con los elementos no-TIC o infraestructuras tecnológicas.
- Dispone una base de datos de la configuración que centraliza la información de todos los elementos y que forma parte del núcleo de la herramienta.
- Medir la calidad del servicio ofrecido mediante el establecimiento de SLAs (*Service Level Agreement-Acuerdo de nivel de servicio*).

El esquema general y simplificado de los dos sistemas es el que se muestra en la Figura 1.1.

El servicio de monitorización tiene como objetivo la detección, diagnóstico y, en su caso, escalado de incidencias de la plataforma puesta a disposición del cliente (tanto de la disponibilidad de la infraestructura como de los sistemas y aplicativos que la integran). Además, se presta en horario 24x7, es decir, 24 horas al día, los 365 días al año. Este servicio incluye las siguientes tareas:

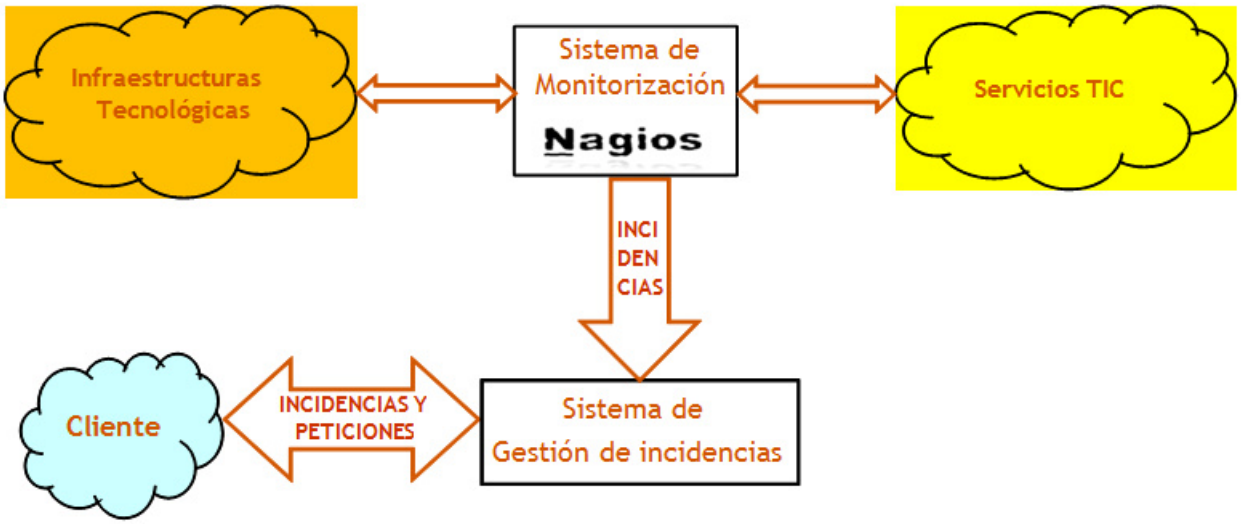

Figura 1.1: Descripción del sistema de monitorización y del sistema de gestión de incidencias. Figura modificada de un documento privado de la empresa TISSAT

- Revisión, de forma continua, de las variables a monitorizar en los sistemas y aplicaciones y procedimientos de resolución de incidencias para los nuevos servidores en producción.
- Monitorización de los sistemas en horario 24x7. Se comprueba que los servidores y procesos se encuentran en funcionamiento y no se producen caídas o fallos en los mismos, de acuerdo a las variables de monitorización definidas.

En este apartado se han explicado los dos sistemas más importantes utilizados en esta empresa, el sistema de gestión de incidencias y el sistema de monitorización.
Como se ha visto, estos dos sistemas son los pilares de esta empresa tecnológica.

Capítulo 2

Descripción del proyecto

El trabajo que se ha realizado en la empresa ha tenido funciones tanto informáticas (como, por ejemplo, la realización del programa para pasar los datos de ficheros a una tabla de la base de datos Oracle o la implementación del algoritmo matemático) como matemáticas (como, por ejemplo, el estudio del criterio Minimax para la realización del algoritmo matemático o el estudio de los resultados de los dos criterios, Minimax y de Bayes, para luego realizar una comparación entre ambos).

Para la estancia en prácticas de la carrera de Matemática Computacional es considerado un trabajo completo e interesante, ya que se centra en las dos áreas importantes que rige la carrera: el área de las matemáticas y el de la informática.

2.1. Objetivos del proyecto

El objetivo principal en la empresa es desarrollar procedimientos automatizados que ayuden a la toma de decisión en la Gestión de Servicios de IT que presta Tissat mediante una herramienta que da servicio a los procesos de resolución de incidencias y problemas, y a los procesos de control en cambios y configuraciones, de acuerdo a lo establecido en la norma ISO 20000 de gestión de servicio IT.

Se decidió que el tema en el que se basaría el Trabajo Final de Grado sería *la Teoría de la Decisión*[1], ya que el algoritmo matemático que se aplica para calcular los umbrales óptimos se basa en el criterio Minimax[1].

Además, en años anteriores, se realizó un estudio similar en la misma empresa utilizando algoritmos matemáticos basados en el criterio de Bayes[1], otro criterio característico de la Teoría de la Decisión. Este criterio implica un conocimiento inicial de cómo se distribuyen los estados de la naturaleza. Los dos problemas más importantes que pueden surgir al implementar este criterio es que el conocimiento no sea todo lo bueno debido o que directamente se desconozca por completo.
La meta propuesta será comparar los resultados de los dos criterios y estudiar cuál se adapta

mejor teniendo en cuenta estas circunstancias que se presentan.

El objetivo principal varía dependiendo del área que se va a abordar.
Si nos referimos al ámbito matemático, el objetivo es la realización del algoritmo matemático utilizando técnicas matemáticas como, por ejemplo, estimación de funciones de densidad vía Kernel, en particular, Kernel Gaussiano, métodos de integración numérica como el método de Simpson y Gaussiano, criterios de decisión como el criterio Minimax y de Bayes, etc. Por otro lado, si nos referimos al ámbito informático, éste es la realización del código en Java, la conexión a la base de datos Oracle, la muestra de las figuras resultantes del cálculo de probabilidades, etc.

2.2. Metodología y definición de tareas

Este proyecto consiste en generar unas clases en Java, compatibles con los métodos de control de la empresa, siendo estas clases necesarias para resolver, mediante métodos estadísticos y numéricos, el cálculo del umbral óptimo a partir de unos datos recogidos de un equipo y monitor propio de la empresa.
Se debe procurar que la programación de dichas clases sea los más eficiente posible para evitar un sobrecoste innecesario.

En el apartado *Implementación y Resultados* se describen cada una de estas clases implementadas para comprender mejor el objetivo de la estancia en prácticas en la empresa.

Las tareas que se han llevado a cabo para este trabajo son:

- Realizar un programa para pasar ficheros a base de datos Oracle para tener un mejor acceso a los datos.
- Realizar un algoritmo matemático para calcular el umbral óptimo de cada equipo-monitor siguiendo el criterio Minimax.
- Comparar los resultados del algoritmo que sigue el criterio Minimax y el que sigue el criterio de Bayes, éste último estudiado hace tres años.

2.3. Planificación temporal de las tareas

La planificación de la estancia en prácticas se divide quincenalmente, ya que los informes que se han realizado en la empresa han sido redactados cada dos semanas.

- Primera quincena:

  El trabajo realizado para esta quincena ha sido estudiar el funcionamiento de la empresa y leer documentación del estudio matemático que se realizó hace tres años.

En la primera semana, lo primero que se hizo fue saber el funcionamiento de la empresa tanto a nivel físico (hardware) como a nivel de servidores (software). Cómo se almacenaban los datos, en qué consistía el trabajo de cada técnico de la empresa, qué servicios ofrecían a los clientes y cuáles de ellos monitorizaban, qué niveles de aviso existían, en qué programas observaban todas las incidencias que se producían y cómo las solucionaban, etc. fueron algunos de los temas vistos.
Fueron facilitados tres documentos privados de la empresa para que fueran leídos detenidamente.
Estos documentos eran memorias de un trabajo que se realizó hace tres años con la ayuda del departamento de matemáticas de la Universidad Jaume I.
De los tres documentos mencionados, el primero resume el análisis del problema y de los datos disponibles con Nagios (sistema de monitorización de redes, de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado). Es el que utiliza la empresa.
El segundo anexo resume el estudio matemático y sus resultados (resolviendo el problema mediante técnicas de investigación operativa, es decir, utilizando la Teoría de la decisión).
Y el tercer anexo resume el desarrollo del algoritmo en la herramienta propia de Tissat y los códigos Java que se propusieron.

A principios de la segunda semana, se dio acceso como usuario en la base de datos mencionada anteriormente, pero con sólo permisos para acceder y consultar las tablas, no con opción de modificar ningún dato de ella.
Fueron enviados otros códigos que, en vez de acceder a la base de datos, recogían los datos a partir de un fichero generado con diferente información acerca de los datos.
Como el funcionamiento del programa que accedía a la base de datos no era el correcto, se probó a utilizar el otro programa (el que hacía uso de ficheros) con los datos de la base de datos colocados en un nuevo fichero y arreglar el programa que accede a la base de datos para que se mostraran por pantalla los mismos datos.

Al finalizar la semana, el programa que accede a la base de datos logró funcionar correctamente.

- Segunda quincena:

  La función planteada para esta quincena fue realizar un programa para leer ficheros y pasarlos a una base de datos Oracle, para que luego esos datos fueran más accesibles para trabajar con ellos.

  Primero, fueron recogidos los datos de ficheros .txt que habían sido generados directamente de la página web de Nagios pero observamos que habían otros ficheros, los ficheros .log, que eran generados cada día por Nagios y contenían más información que los ficheros .txt. Entonces, a partir de los ficheros .log, se realizó un programa para recoger la información necesaria de cada (equipo, servicio) e insertar esos datos en la nueva tabla que fue creada expresamente para esta tarea (I2TM_PROD.I2TM_UMBRALES_DETALLE_DATOS).

  Al finalizar las dos semanas, la nueva tabla contenía los datos de los 7 equipos seleccionados con sus respectivos servicios.

- Tercera quincena:

  La tarea propuesta fue comprender a la perfección el estudio de estimación de funciones por métodos no paramétricos e, inclusive, el estudio de la teoría de la decisión basada en varios criterios como, por ejemplo, el criterio Minimax y el criterio de Bayes.

En la primera semana, fue leído el documento proporcionado por el tutor sobre la construcción de kernels y funciones de densidad de probabilidad, el artículo [14].
Este documento se centraba en estudiar los diferentes modelos de kernels, aunque sólo nos centraremos en el Kernel Gaussiano (también llamado Kernel normal).
Otro tema que se explicaba en el documento era el cálculo del ancho de banda óptimo, parámetro importante para la correcta construcción del Kernel.

En la segunda semana, fue leído el libro de La Teoría de la Decisión ([1]) que fue proporcionado también por el tutor.
En este documento estaban explicados ciertos conceptos como, por ejemplo, la función de riesgo junto con los problemas que conlleva y las soluciones propuestas, es decir, los criterios adicionales planteados.
Se centra en dos criterios: el criterio Minimax, desarrollado más adelante en el apartado 3.2, que es el criterio en el que nos basaremos para realizar el algoritmo matemático, y el criterio de Bayes, desarrollado más adelante en el apartado 3.3, que es el criterio en el que se basaron hace años cuando realizaron el estudio.

Al finalizar las dos semanas, fueron aclarados los conceptos de Kernel y los dos criterios, así que la semana próxima se comenzó con la programación del algoritmo matemático basado en el criterio Minimax.

- Cuarta quincena:

  El trabajo asignado para hacer en estas dos semanas era el de realizar un programa en código Java cuya función fuera calcular el umbral óptimo de un equipo-monitor. Este programa seguiría el modelo Minimax, aunque también incluiría el criterio de Bayes (ya realizado en otro estudio anterior) para que luego las comparaciones entre los dos fueran más sencillas de hacer.
  Como el algoritmo matemático se basa en dos políticas, el trabajo se dividió para esas dos semanas centrándose en la Política1 la primera semana y en la Política2 (que es más complicada) en la segunda semana.

  En la primera semana, la tarea encomendada fue entender, de nuevo, todo el código que se realizó hace años para el cálculo del umbral óptimo basado en el criterio de Bayes. Una vez estuvieran los conceptos fortalecidos, se podría empezar con la programación del algoritmo basado en el criterio Minimax.
  Primero, se hizo un programa para la Política1, el cuál mostraba por pantalla el umbral óptimo junto con el riesgo óptimo de ambos criterios, para que luego fuera más sencillo compararlos.
  Además, se acordó mostrar una gráfica para cada criterio con todos los riesgos recogidos para que, de manera visual, también se pudiera observar como variaban los riesgos respectivamente.
  Para hacer la gráfica, lo primero que se hizo fue mostrar por pantalla un vector con todos los riesgos y umbrales de cada criterio (el vector de umbrales de los dos criterios era el mismo: valores desde 0 a 1 con intervalos de 0.01).
  Se realizó un programa en Matlab que recogiera esos vectores con todos los riesgos de Bayes y Minimax (Eje de las Y) junto con los umbrales (Eje de las X) y dibujara las dos gráficas correspondientes.
  No era sencillo copiar un vector de 100 elementos del programa en Java a Matlab. Para poder realizarlo, se intentó que, en vez de crear un programa aparte en Matlab, se programara todo en el código Java haciendo uso de interfaces gráficas.
  El resultado final de la ejecución del programa Política1 es el riesgo y umbral óptimos

para cada criterio respectivamente y una gráfica que muestra como van evolucionando los riesgos de ambos criterios para cada valor del umbral (Figura 5.4).

En la segunda semana, se empezó a programar el algoritmo para la Política2. Este algoritmo es más complicado que el primero ya que hay más cálculos y el programa tarda más en mostrar la solución, por eso fue hecho en un programa aparte para que no hubieran complicaciones.
También se programó la Política2 para que mostrara, cuando se ejecutara el programa, el riesgo y umbral óptimos para cada criterio respectivamente y una gráfica que muestra como van evolucionando los riesgos de ambos criterios para cada valor del umbral (Figura 5.5).

Al finalizar las dos semanas, se obtuvieron dos programas, uno para cada política, que calculaban los riesgos óptimos de los dos criterios y mostraban una gráfica con la evolución de esos riesgos para cada valor tomado del umbral.

- Quinta quincena:

  Para esta quincena, se debían estudiar los riesgos óptimos y la gráfica de evolución.

  Primero, se observó que el umbral óptimo del criterio Minimax fuera mayor que el umbral óptimo del criterio de Bayes porque el riesgo de Bayes ($l(d;\theta_1)$) es menor o igual que el riesgo Minimax ($max\{l(d;\theta)/\theta \in [0,1]\}$). Y así era, ya que en el programa Politica1, el riesgo óptimo del criterio de Bayes valía 0.78 y el del criterio Minimax valía 0.82; y en el programa Politica2, el riesgo óptimo del criterio de Bayes valía 0.78 y el del criterio Minimax valía 0.8.
  Con lo que respecta a las gráficas de los dos programas, los riesgos recogidos tanto del programa Politica1 como del programa Politica2 son prácticamente iguales, no varían mucho, y tienen la misma forma. Por tanto, se confirma lo comentado en el apartado anterior.
  Estas gráficas parten del riesgo cuando el umbral es 0 y van descendiendo hasta que alcanzan el umbral óptimo y mantienen ese valor hasta cuando el umbral es 1.

  En el apartado *Implementación y Resultados* son mostradas las gráficas resultantes con los resultados de los umbrales y riesgos óptimos para cada Política (Figura 5.4 y Figura 5.5).

  Al finalizar las dos semanas, se tuvo un estudio sobre las gráficas y los riesgos óptimos.

- Sexta quincena:

  El trabajo asignado para hacer esta quincena era el de realizar un programa que recogiera los datos correspondientes para calcular los umbrales y los introdujera en la base de datos Oracle para tener un acceso más fácil a la hora de calcular los umbrales óptimos.

  En la primera semana se implementó un nuevo programa, llamado FicheroGere2ABBDD, cuya función es leer el fichero y pasar a base de datos los datos que nos interesan (es decir, el equipo, el monitor y el umbral correspondiente).
  Para recoger esos datos en la base de datos, se creó una nueva tabla, llamada I2TM_PROD.I2TM_UMBRALES_PRACTICA, para no utilizar las otras ya creadas anteriormente, ya que para el proyecto solo se utilizarán los datos de esta tabla.

  Cuando el programa FicheroGere2ABBDD funcionó de acuerdo a lo esperado (es decir, insertaba correctamente los datos en la tabla correspondiente en la base de datos Oracle), la otra tarea pendiente para esta quincena era implementar un nuevo programa principal,

que en vez de leer del fichero directamente (como hacía anteriormente) leyera de la tabla de la base de datos y realizara los mismos cálculos matemáticos vistos en quincenas anteriores.

Este programa fue sencillo de implementar ya que en la segunda quincena se realizó también un programa para insertar datos en base de datos y leerlos a continuación directamente de la tabla de la base de datos en vez del propio fichero.

Una vez programado correctamente el nuevo Programa Principal y el programa Fichero-Gere2ABBDD, como aún faltaban por completar unas horas de las 290 que debía realizar en la empresa, se utilizaron para perfeccionar el código de los programas, añadiendo comentarios para facilitar la comprensión del mismo.

Después de estas seis quincenas, finalizó el trabajo a realizar en la empresa Tissat, cumpliendo con las tareas previstas con éxito.

2.4. Estimación de recursos del proyecto

En este apartado se detallan todos los recursos que se han utilizado para el desarrollo del proyecto, tanto hardware como software así como los recursos humanos.

2.4.1. Recursos hardware

El ordenador de sobremesa utilizado para el desarrollo del proyecto ha sido tanto el proporcionado por la empresa como el mío propio.
Las características del ordenador personal son las siguientes:

- *Procesador*: Intel Core i5-2410M CPU 2.3 GHz
- *Memoria RAM*: 4 GB
- *Disco duro*: 544 GB
- *Sistema operativo*: Windows 7 Home Premium

Las características del ordenador de la empresa son las siguientes:

- *Procesador*: Intel Core i5-2400 CPU 3.10 GHz
- *Memoria RAM*: 4 GB (3.88 GB utilizable)
- *Disco duro*: 465 GB
- *Sistema operativo*: Windows 7 Professional

2.4.2. Recursos software

A continuación se describen brevemente los programas y herramientas que se han utilizado para el desarrollo del trabajo.

- *Eclipse*: Entorno de desarrollo para la implementación de las clases en Java.
  Programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama 'Aplicaciones de Cliente Enriquecido'.[17]

- *Oracle*: Sistema de gestión de base de datos objeto-relacional.
  Se considera a Oracle Database como uno de los sistemas de bases de datos más completos, destacando el soporte de transacciones, la estabilidad, la escalabilidad y el soporte multiplataforma.[12]

- *SQL*: Lenguaje declarativo de acceso a bases de datos relacionales.
  Permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de bases de datos, así como hacer cambios en ellas.[21]

- *MyOraSQL*: Herramienta avanzada de monitorización de Oracle.
  Esta herramienta es simple, rápida y fácil de usar. Supervisa el rendimiento de la base de datos en tiempo real.

- *Google Chrome*: Navegador web.

- *Bloc de notas*: Editor de texto.

- Creately: Programa para crear diagramas de clase.

Para la parte referente a la documentación y los informes quincenales se han utilizado las siguientes tecnologías:

- *Microsoft Word*: Redacción de los informes quincenales.
  Software destinado al procesamiento de textos.[10]

- *Latex*: Redacción del proyecto.
  Sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas.[9]

- *Picasa*: Edición de imágenes.
  Herramienta web para visualizar, organizar y editar fotografías digitales.[19]

2.4.3. Recursos humanos

El trabajo realizado en la empresa junto con la redacción de este proyecto han sido desarrollados por la autora de esta memoria, bajo la dirección del tutor y del supervisor de la empresa.

Capítulo 3

Teoría de la Decisión

Para el desarrollo de la Teoría de la Decisión se han utilizado las siguientes fuentes documentales: Arnaiz (1978), Ríos (1971) y Dixon, Massey (1965).

Cuando observamos fenómenos físicos, biológicos y sociales, vemos que muchos de ellos están gobernados por leyes que son de naturaleza estadística, entonces la especificación de la ley, es decir, como realmente ocurre y se comporta el fenómeno, es lo que se conoce como *estado de la naturaleza.*

En estadística, con objeto de descubrir el estado en que se encuentra la naturaleza, se realizan experimentaciones obteniéndose datos, que consisten en la obtención de muestras aleatorias. Debemos de señalar que la experimentación o la toma de muestras es siempre más sencilla en las ciencias físicas y naturales que en las ciencias sociales. Los motivos principales son la mayor cantidad de variables no controlables que existen en las ciencias sociales y la dificultad de repetir la experimentación en condiciones análogas, ya que el transcurso del tiempo hace cambiar gustos y actitudes.

El objeto de la estadística es, tomando como base esta experimentación, dar criterios, tanto para la toma de decisiones como para realizar predicciones.

Para estudiarla, se dará primero una estructura matemática de los problemas de toma de decisiones.
Observamos una variable o vector aleatorio x que se supone que tiene una distribución de probabilidad $f(x;\theta)$ (el valor del parámetro θ se desconoce, pero se sabe que pertenece a un cierto conjunto Θ. Es decir, se sabe que x tiene una distribución de probabilidad $f(x|\theta)$ que pertenece a una cierta familia, el valor real que tenga θ es lo que llamaremos estado de la naturaleza).
El problema que se nos presenta es el de tomar una decisión sobre el valor de θ, tomando como base los valores observados de x al hacer la experimentación. Éste es el problema que resuelve la *inferencia estadística.*

Cuando se hace una inferencia estadística por medio de los valores observados $x_1; x_2; ...; x_n$ de una variable aleatoria x, debemos de introducir una función $d(x_1; x_2; ...; x_n)$ que llamaremos

función de decisión y que aplica el espacio muestra sobre el espacio de las acciones o decisiones. Hay diferentes problemas que se pueden aplicar dependiendo de la función de decisión. A continuación, se nombran dos ejemplos:

- Si la función de decisión nos separa el espacio muestra en dos partes (R_1 y R_2, tales que si el punto muestra $x_1; x_2; ...; x_n \in R_1$ se toma la acción a_1 y si $x_1; x_2; ...; x_n \in R_2$ se toma la acción a_2), los problemas de este tipo reciben el nombre de *problemas de contrastación de hipótesis.*

- Si el espacio de las acciones o decisiones tiene cinco elementos, una función de decisión que resuelva este problema deberá dividir el espacio muestra E en cinco regiones $R_1; R_2; ...; R_5$ tal que $R_i \cap R_j = \Phi \ i \neq j$ con $i, j : 1, ..., 5$ y $\bigcup_{1}^{5} R_i = E$ tal que si $(x_1; x_2; ...; x_n) \in R_i$, entonces $d(x_1; x_2; ...; x_n) = a_i$. Estos problemas reciben el nombre de *problemas de decisión múltiple.*

3.1. Función de Riesgo

El éxito de una función de decisión en el cumplimiento de su misión es necesario medirlo de alguna forma. Si el experimentador es capaz de ponderar las pérdidas que se derivan de hacer decisiones incorrectas, estas ponderaciones pueden definir lo que llamaremos *función de pérdida* $L(\theta; a)$, que nos mide la pérdida en que incurrimos cuando tomamos la acción a y el estado de la naturaleza θ.

Por ejemplo, supongamos que tomamos una función de decisión $d(x_1; x_2; ...; x_n)$. Esta función asigna a cada punto del espacio muestra una acción $a \in A$. El problema que tenemos que resolver es, dada una función de pérdida $L(\theta; a) = l[\theta; d(x_1; x_2; ...; x_n)]$, elegir $d(x_1; x_2; ...; x_n)$ tal que la pérdida sea mínima. La función $l[\theta; d(x_1; x_2; ...; x_n)]$ debe ser de tal tipo que decrezca cuando la magnitud del error decrece y crezca cuando crece la magnitud del error.

Supongamos que en todos los problemas que vamos a tratar es deseable introducir una función de pérdida y que $l[\theta; d(x_1; x_2; ...; x_n)]$ es una función de este tipo mayor o igual a cero.

Para la toma de decisiones, cuando medimos la efectividad de cualquier función de decisión, no debemos de considerar nunca lo que ocurre en un solo experimento, sino el comportamiento medio de la función de decisión. Entonces, la función de decisión $d(x_1; x_2; ...; x_n)$ será una variable aleatoria y $l[\theta; d(x_1; x_2; ...; x_n)]$ también lo será, entonces definiremos una esperanza matemática de esta pérdida que llamaremos *función de riesgo.*

3.1.1. Definición de función de riesgo

La función de riesgo $R[\theta; d] = E\{l[\theta; d(x_1; x_2; ...; x_n)]\}$ en donde la esperanza matemática se toma respecto a la distribución aleatoria $[x_1; x_2; ...; x_n]$ con θ fijo.

En el caso de muestreo aleatorio con reemplazamiento $R[\theta; d] = E\{l[\theta; d(x)]\} = \int l[\theta; d(x)] f(x|\theta)\, dx$ en donde la integral es múltiple y $x = [x_1; x_2; ...; x_n]$ [1], [13], [5].

Consideremos el caso de dos funciones de decisión $d_1; d_2$, cada una de estas funciones nos dará lugar a dos riesgos distintos $R[\theta; d_1], R[\theta; d_2]$.

Entonces el problema sería representar los riesgos que corresponden a cada función de decisión y considerar aquella que para todo valor de θ nos dé el menor riesgo.

Esto sucedería si los riesgos correspondientes a las dos funciones de decisión fueran como se muestra en la Figura 3.1, en donde el riesgo correspondiente a la decisión d_1 es siempre mayor que el correspondiente a la decisión d_2, cualquiera que sea el valor de θ.

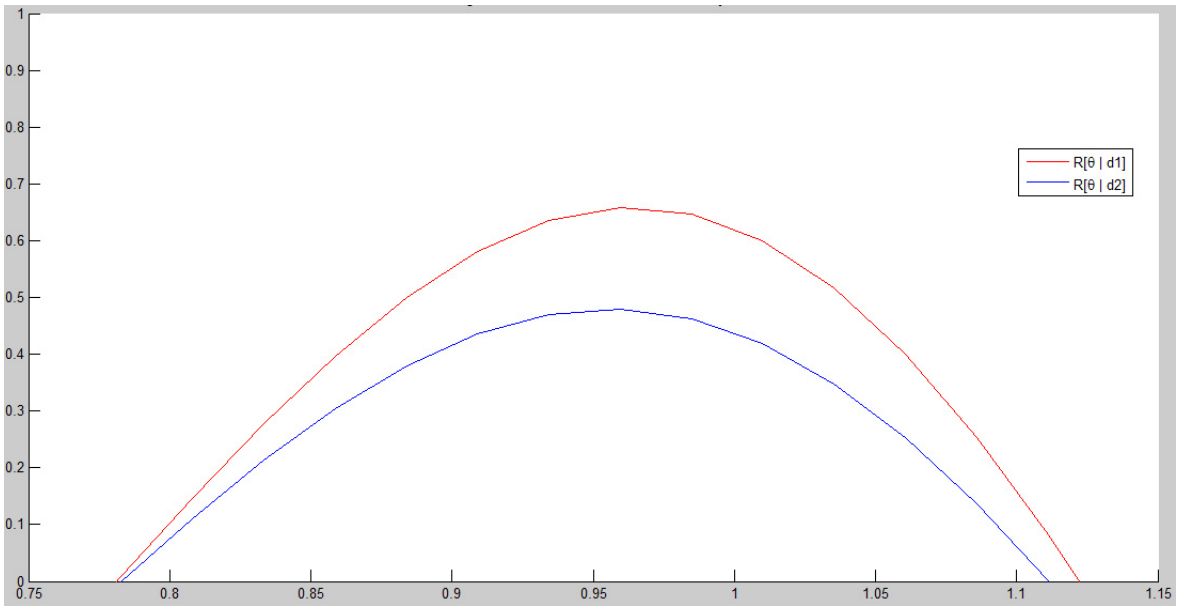

Figura 3.1: Riesgos de las dos funciones de decisión d_1 y d_2. Elaboración propia.

Pero las cosas no suelen suceder como se muestra en la Figura 3.1, en donde el riesgo $R[\theta; d_2] < R[\theta; d_1]$ para cualquier valor de θ.
En general ocurre que, para un cierto valor de θ, el riesgo será menor para la función de decisión d_1, y para otros valores de θ, el riesgo será menor para la función de decisión d_2 (como se muestra en la Figura 3.2).

Entonces, hay que introducir unos criterios adicionales con objeto de llegar a la elección de una función de decisión.

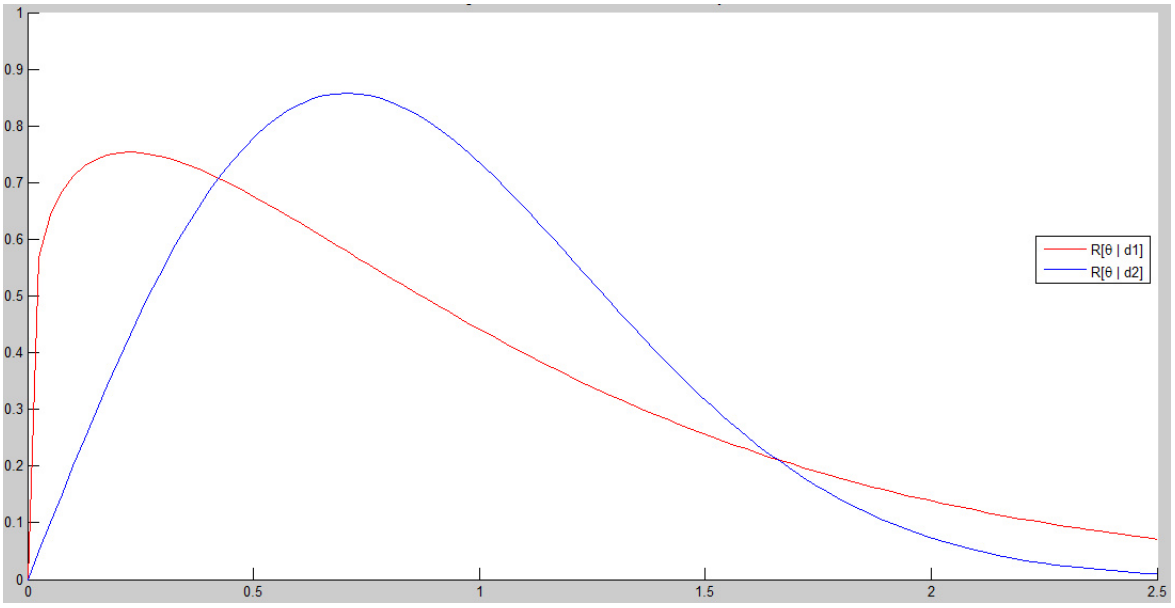

Figura 3.2: Riesgos de las dos funciones de decisión d_1 y d_2. Elaboración propia.

3.2. Criterio Minimax

Consideremos una clase D^c de las funciones de decisión y vamos a establecer el criterio minimax.

Una función $d_0 \in D$ se llama *función de decisión minimax* si satisface la siguiente igualdad: $\max_{\theta} R[\theta; d_0] = \min_{k \in D} \max_{\theta} R[\theta; d]$ [1].
Este principio está basado en la siguiente idea: Para cada función de decisión $d \in D$ calculamos $R[\theta; d]$ y en cada una de estas funciones obtenemos el valor de θ que lo hace máximo para cada d, y elegimos aquella función que hace mínimo estos máximos.

Por ejemplo, suponemos que tenemos tres funciones de decisión con sus respectivos riesgos. Como se muestra en la Figura 3.3, $R[\theta; d_3]$ tiene el mínimo riesgo máximo. Luego d_3 será la decisión minimax.

3.3. Criterio de Bayes

Consideramos ahora que, además de una muestra $[x_1; x_2; ...; x_n]$, conocemos una distribución a priori de θ, es decir, que ahora tenemos dos variables aleatorias $x = [x_1; x_2; ...; x_n]$ y θ.
Entonces $R[\theta; d]$ será una variable aleatoria que dependerá de θ y deberemos calcular el valor medio de este riesgo, también llamado *riesgo esperado* o *riesgo de Bayes*.

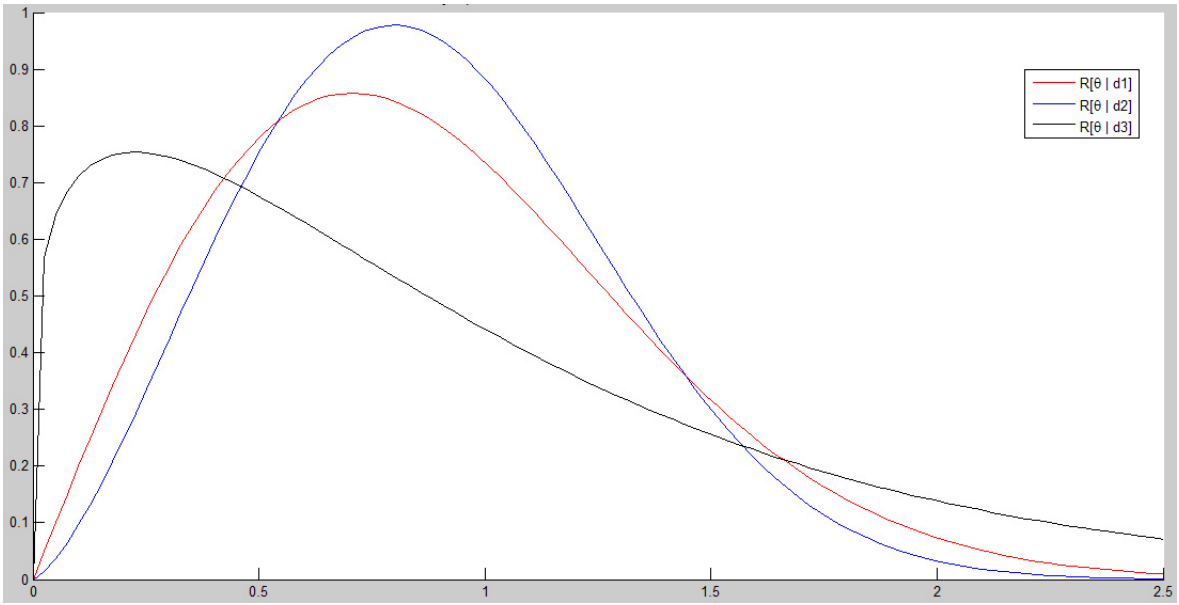

Figura 3.3: Ejemplo. Elección de la función de decisión minimax. Elaboración propia.

3.3.1. Definición del riesgo de Bayes

Consideremos una distribución a priori del parámetro θ, que designaremos por $\pi(\theta)$ y que sea una función de densidad o de cuantía, según que θ sea continuo o discreto.

El riesgo de Bayes viene definido, si la distribución es continua, de la siguiente forma $r[d] = \int_{-\infty}^{\infty} R[\theta; d]\pi(\theta)\, d\theta$ y, si la distribución es discreta, $r[d] = \sum R[\theta_i; d]\pi(\theta_i)$ [1].

3.3.2. Decisión de Bayes

Una función de decisión d_0 recibe el nombre de *decisión de Bayes* respecto una distribución a priori $\pi(\theta)$ y una clase D de funciones de decisión si $r(d_0) = \min_{d \in D} r(d)$ [1].

3.4. Ejemplo ilustrativo de los dos criterios

A continuación, mediante un caso discreto, vamos a analizar todo lo comentado anteriormente.
Supongamos una empresa que tiene cuatro acciones a tomar: $A = \{a_1; a_2; a_3; a_4\}$. Los estados de la naturaleza son: $\Theta = \{\theta_1; \theta_2; \theta_3\}$.
La función de pérdida según las acciones que se toman y el estado en que se encuentra la naturaleza tiene los siguientes valores (Cuadro 3.1):

	a_1	a_2	a_3	a_4
θ_1	$l(\theta_1; a_1) = 3$	$l(\theta_1; a_2) = 6$	$l(\theta_1; a_3) = 0$	$l(\theta_1; a_4) = 2$
θ_2	$l(\theta_2; a_1) = 4$	$l(\theta_2; a_2) = 0$	$l(\theta_2; a_3) = 3$	$l(\theta_2; a_4) = 2$
θ_3	$l(\theta_3; a_1) = 2$	$l(\theta_3; a_2) = 1$	$l(\theta_3; a_3) = 4$	$l(\theta_3; a_4) = 0$

Cuadro 3.1: Ejemplo. Valores de la función de pérdida dependiendo las acciones que se tomen y el estado de la naturaleza. Elaboración propia.

Antes de tomar una decisión se observará una variable Z que pueda tomar los valores 0 y 1 con las siguientes probabilidades:

$$\begin{cases} P[z=0|\theta=\theta_1] = 0{,}9 \\ P[z=1|\theta=\theta_1] = 0{,}1 \end{cases} \quad \begin{cases} P[z=0|\theta=\theta_2] = 0{,}3 \\ P[z=1|\theta=\theta_2] = 0{,}7 \end{cases} \quad \begin{cases} P[z=0|\theta=\theta_3] = 0{,}6 \\ P[z=1|\theta=\theta_3] = 0{,}4 \end{cases}$$

Consideremos en este caso las nueve posibles funciones de decisión:

Cualquiera que sea el valor de z que se obtenga, se toma la decisión a_1: $\begin{cases} d_1(0) = a_1 \\ d_1(1) = a_1 \end{cases}$

Si $z = 0$ se toma la decisión a_1 y si $z = 1$ se toma la decisión a_2: $\begin{cases} d_2(0) = a_1 \\ d_2(1) = a_2 \end{cases}$

Si $z = 0$ se toma la decisión a_1 y si $z = 1$ se toma la decisión a_3: $\begin{cases} d_3(0) = a_1 \\ d_3(1) = a_3 \end{cases}$

Si $z = 0$ se toma la decisión a_1 y si $z = 1$ se toma la decisión a_4: $\begin{cases} d_4(0) = a_1 \\ d_4(1) = a_4 \end{cases}$

Si $z = 0$ se toma la decisión a_2 y si $z = 1$ se toma la decisión a_1: $\begin{cases} d_5(0) = a_2 \\ d_5(1) = a_1 \end{cases}$

Cualquiera que sea el valor de z que se obtenga, se toma la decisión a_2: $\begin{cases} d_6(0) = a_2 \\ d_6(1) = a_2 \end{cases}$

Si $z = 0$ se toma la decisión a_2 y si $z = 1$ se toma la decisión a_3: $\begin{cases} d_7(0) = a_2 \\ d_7(1) = a_3 \end{cases}$

Si $z = 0$ se toma la decisión a_2 y si $z = 1$ se toma la decisión a_4: $\begin{cases} d_8(0) = a_2 \\ d_8(1) = a_4 \end{cases}$

Si $z = 0$ se toma la decisión a_3 y si $z = 1$ se toma la decisión a_1: $\begin{cases} d_9(0) = a_3 \\ d_9(1) = a_1 \end{cases}$

Si $z = 0$ se toma la decisión a_3 y si $z = 1$ se toma la decisión a_2: $\begin{cases} d_{10}(0) = a_3 \\ d_{10}(1) = a_2 \end{cases}$

Cualquiera que sea el valor de z que se obtenga, se toma la decisión a_3: $\begin{cases} d_{11}(0) = a_3 \\ d_{11}(1) = a_3 \end{cases}$

Si $z = 0$ se toma la decisión a_3 y si $z = 1$ se toma la decisión a_4: $\begin{cases} d_{12}(0) = a_3 \\ d_{12}(1) = a_4 \end{cases}$

Si $z = 0$ se toma la decisión a_4 y si $z = 1$ se toma la decisión a_1: $\begin{cases} d_{13}(0) = a_4 \\ d_{13}(1) = a_1 \end{cases}$

Si $z = 0$ se toma la decisión a_4 y si $z = 1$ se toma la decisión a_2: $\begin{cases} d_{14}(0) = a_4 \\ d_{14}(1) = a_2 \end{cases}$

Si $z = 0$ se toma la decisión a_4 y si $z = 1$ se toma la decisión a_3: $\begin{cases} d_{15}(0) = a_4 \\ d_{15}(1) = a_3 \end{cases}$

Cualquiera que sea el valor de z que se obtenga, se toma la decisión a_4: $\begin{cases} d_{16}(0) = a_4 \\ d_{16}(1) = a_4 \end{cases}$

Cada una de estas funciones da lugar a un riesgo según sea el valor de θ.

$R[\theta_1; d_1] = E\{l[\theta_1; d_1]\} = l[\theta_1; d_1(0)]P[z = 0|\theta = \theta_1] + l[\theta_1; d_1(1)]P[z = 1|\theta = \theta_1] = l[\theta_1; a_1]P[z = 0|\theta = \theta_1] + l[\theta_1; a_1]P[z = 1|\theta = \theta_1] = 3 * 0{,}9 + 3 * 0{,}1 = 3$

$R[\theta_2; d_1] = E\{l[\theta_2; d_1]\} = l[\theta_2; d_1(0)]P[z = 0|\theta = \theta_2] + l[\theta_2; d_1(1)]P[z = 1|\theta = \theta_2] = l[\theta_2; a_1]P[z = 0|\theta = \theta_2] + l[\theta_2; a_1]P[z = 1|\theta = \theta_2] = 4 * 0{,}3 + 4 * 0{,}7 = 4$

$R[\theta_3; d_1] = E\{l[\theta_3; d_1]\} = l[\theta_3; d_1(0)]P[z = 0|\theta = \theta_3] + l[\theta_3; d_1(1)]P[z = 1|\theta = \theta_3] = l[\theta_3; a_1]P[z = 0|\theta = \theta_3] + l[\theta_3; a_1]P[z = 1|\theta = \theta_3] = 2 * 0{,}6 + 2 * 0{,}4 = 2$

Entonces, realizando estas operaciones para todas las funciones, tenemos que:

$$\begin{cases} R[\theta_1; d_1] = 3 * 0{,}9 + 3 * 0{,}1 = 3 \\ R[\theta_2; d_1] = 4 * 0{,}3 + 4 * 0{,}7 = 4 \\ R[\theta_3; d_1] = 2 * 0{,}6 + 2 * 0{,}4 = 2 \end{cases} \quad \begin{cases} R[\theta_1; d_2] = 3 * 0{,}9 + 6 * 0{,}1 = 3{,}3 \\ R[\theta_2; d_2] = 4 * 0{,}3 + 0 * 0{,}7 = 1{,}2 \\ R[\theta_3; d_2] = 2 * 0{,}6 + 1 * 0{,}4 = 1{,}6 \end{cases}$$

$$\begin{cases} R[\theta_1; d_3] = 3 * 0{,}9 + 0 * 0{,}1 = 2{,}7 \\ R[\theta_2; d_3] = 4 * 0{,}3 + 3 * 0{,}7 = 3{,}3 \\ R[\theta_3; d_3] = 2 * 0{,}6 + 4 * 0{,}4 = 2{,}8 \end{cases} \quad \begin{cases} R[\theta_1; d_4] = 3 * 0{,}9 + 2 * 0{,}1 = 2{,}9 \\ R[\theta_2; d_4] = 4 * 0{,}3 + 2 * 0{,}7 = 2{,}6 \\ R[\theta_3; d_4] = 2 * 0{,}6 + 0 * 0{,}4 = 1{,}2 \end{cases}$$

$$\begin{cases} R[\theta_1; d_5] = 6 * 0{,}9 + 3 * 0{,}1 = 5{,}7 \\ R[\theta_2; d_5] = 0 * 0{,}3 + 4 * 0{,}7 = 2{,}8 \\ R[\theta_3; d_5] = 1 * 0{,}6 + 2 * 0{,}4 = 1{,}4 \end{cases} \quad \begin{cases} R[\theta_1; d_6] = 6 * 0{,}9 + 6 * 0{,}1 = 6 \\ R[\theta_2; d_6] = 0 * 0{,}3 + 0 * 0{,}3 = 0 \\ R[\theta_3; d_6] = 1 * 0{,}6 + 1 * 0{,}4 = 1 \end{cases}$$

$$\begin{cases} R[\theta_1; d_7] = 6 * 0{,}9 + 0 * 0{,}1 = 5{,}4 \\ R[\theta_2; d_7] = 0 * 0{,}3 + 3 * 0{,}7 = 2{,}1 \\ R[\theta_3; d_7] = 1 * 0{,}6 + 4 * 0{,}4 = 2{,}2 \end{cases} \quad \begin{cases} R[\theta_1; d_8] = 6 * 0{,}9 + 2 * 0{,}1 = 5{,}6 \\ R[\theta_2; d_8] = 0 * 0{,}3 + 2 * 0{,}7 = 1{,}4 \\ R[\theta_3; d_8] = 1 * 0{,}6 + 0 * 0{,}4 = 0{,}6 \end{cases}$$

$$\begin{cases} R[\theta_1; d_9] = 0 * 0{,}9 + 3 * 0{,}1 = 0{,}3 \\ R[\theta_2; d_9] = 3 * 0{,}3 + 4 * 0{,}7 = 3{,}7 \\ R[\theta_3; d_9] = 4 * 0{,}6 + 2 * 0{,}4 = 3{,}2 \end{cases} \quad \begin{cases} R[\theta_1; d_{10}] = 0 * 0{,}9 + 6 * 0{,}1 = 0{,}6 \\ R[\theta_2; d_{10}] = 3 * 0{,}3 + 0 * 0{,}7 = 0{,}3 \\ R[\theta_3; d_{10}] = 4 * 0{,}6 + 1 * 0{,}4 = 2{,}8 \end{cases}$$

$$\begin{cases} R[\theta_1; d_{11}] = 0 * 0{,}9 + 0 * 0{,}1 = 0 \\ R[\theta_2; d_{11}] = 3 * 0{,}3 + 3 * 0{,}7 = 3 \\ R[\theta_3; d_{11}] = 4 * 0{,}6 + 4 * 0{,}4 = 4 \end{cases} \quad \begin{cases} R[\theta_1; d_{12}] = 0 * 0{,}9 + 2 * 0{,}1 = 0{,}2 \\ R[\theta_2; d_{12}] = 3 * 0{,}3 + 2 * 0{,}7 = 2{,}3 \\ R[\theta_3; d_{12}] = 4 * 0{,}6 + 0 * 0{,}4 = 2{,}4 \end{cases}$$

$$\begin{cases} R[\theta_1; d_{13}] = 2 * 0{,}9 + 3 * 0{,}1 = 2{,}1 \\ R[\theta_2; d_{13}] = 2 * 0{,}3 + 4 * 0{,}7 = 3{,}4 \\ R[\theta_3; d_{13}] = 0 * 0{,}6 + 2 * 0{,}4 = 0{,}8 \end{cases} \quad \begin{cases} R[\theta_1; d_{14}] = 2 * 0{,}9 + 6 * 0{,}1 = 2{,}4 \\ R[\theta_2; d_{14}] = 2 * 0{,}3 + 0 * 0{,}7 = 0{,}6 \\ R[\theta_3; d_{14}] = 0 * 0{,}6 + 1 * 0{,}4 = 0{,}4 \end{cases}$$

$$\begin{cases} R[\theta_1; d_{15}] = 2 * 0{,}9 + 0 * 0{,}1 = 1{,}8 \\ R[\theta_2; d_{15}] = 2 * 0{,}3 + 3 * 0{,}7 = 2{,}7 \\ R[\theta_3; d_{15}] = 0 * 0{,}6 + 4 * 0{,}4 = 1{,}6 \end{cases} \quad \begin{cases} R[\theta_1; d_{16}] = 2 * 0{,}9 + 2 * 0{,}1 = 2 \\ R[\theta_2; d_{16}] = 2 * 0{,}3 + 2 * 0{,}7 = 2 \\ R[\theta_3; d_{16}] = 0 * 0{,}6 + 0 * 0{,}4 = 0 \end{cases}$$

Reproduzcamos todo lo anterior en la siguiente tabla (Cuadro 3.2).

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	d_{16}
$R[\theta_1; d_i]$	3	3,3	2,7	2,9	5,7	6	5,4	5,6	0,3	0,6	0	0,2	2,1	2,4	1,8	2
$R[\theta_2; d_i]$	4	1,2	3,3	2,6	2,8	0	2,1	1,4	3,7	0,3	3	2,3	3,4	0,6	2,7	2
$R[\theta_3; d_i]$	2	1,6	2,8	1,2	1,4	1	2,2	0,6	3,2	2,8	4	2,4	0,8	0,4	1,6	0
$\max_{\theta} R[\theta_i; d_i]$	4	3,3	3,3	2,9	5,7	6	5,4	5,6	3,7	2,8	4	2,4	3,4	2,4	2,7	2

Cuadro 3.2: Ejemplo. Valores de los riesgos de cada función de decisión para los dos estados de la naturaleza θ_1, θ_2 y θ_3. Elaboración propia.

La decisión minimax en este ejemplo es la d_{16} porque el mínimo de todos los máximos (la última fila del Cuadro 3.2) es 2, que equivale a la función de decisón 16.

Vamos ahora a considerar el criterio de Bayes, siguiendo con el mismo ejemplo, pero partimos de que θ tiene una distribución de probabilidad:

$$\begin{cases} P[\theta = \theta_1] = 0{,}5 \\ P[\theta = \theta_2] = 0{,}3 \\ P[\theta = \theta_3] = 0{,}2 \end{cases}$$

Entonces, para cada función de decisión, tendremos un riesgo de Bayes:

$r(d_1) = E\{[R[\theta_i; d_1]]\} = R[\theta_1; d_1]P[\theta = \theta_1] + R[\theta_2; d_1]P[\theta = \theta_2] + R[\theta_3; d_1]P[\theta = \theta_3] = 3 * 0{,}5 + 4 * 0{,}3 + 2 * 0{,}2 = 3{,}1$

$r(d_2) = 3{,}3 * 0{,}5 + 1{,}2 * 0{,}3 + 1{,}6 * 0{,}2 = 2{,}33$

$r(d_3) = 2{,}7 * 0{,}5 + 3{,}3 * 0{,}3 + 2{,}8 * 0{,}2 = 2{,}9$

$r(d_4) = 2{,}9 * 0{,}5 + 2{,}6 * 0{,}3 + 1{,}2 * 0{,}2 = 2{,}47$

$r(d_5) = 5{,}7 * 0{,}5 + 2{,}8 * 0{,}3 + 1{,}4 * 0{,}2 = 3{,}97$

$r(d_6) = 6 * 0{,}5 + 0 * 0{,}3 + 1 * 0{,}2 = 3{,}2$

$r(d_7) = 5{,}4 * 0{,}5 + 2{,}1 * 0{,}3 + 2{,}2 * 0{,}2 = 3{,}77$

$r(d_8) = 5{,}6 * 0{,}5 + 1{,}4 * 0{,}3 + 0{,}6 * 0{,}2 = 3{,}34$

$r(d_9) = 0{,}3 * 0{,}5 + 3{,}7 * 0{,}3 + 3{,}2 * 0{,}2 = 1{,}9$

$r(d_{10}) = 0{,}6 * 0{,}5 + 0{,}3 * 0{,}3 + 2{,}8 * 0{,}2 = 0{,}95$

$r(d_{11}) = 0 * 0{,}5 + 3 * 0{,}3 + 4 * 0{,}2 = 1{,}7$

$r(d_{12}) = 0{,}2 * 0{,}5 + 2{,}3 * 0{,}3 + 2{,}4 * 0{,}2 = 1{,}27$

$r(d_{13}) = 2{,}1 * 0{,}5 + 3{,}4 * 0{,}3 + 0{,}8 * 0{,}2 = 2{,}23$

$r(d_{14}) = 2{,}4 * 0{,}5 + 0{,}6 * 0{,}3 + 0{,}4 * 0{,}2 = 1{,}46$

$r(d_{15}) = 1{,}8 * 0{,}5 + 2{,}7 * 0{,}3 + 1{,}6 * 0{,}2 = 2{,}03$

$r(d_{16}) = 2 * 0{,}5 + 2 * 0{,}3 + 0 * 0{,}2 = 1{,}6$

$\text{mín}\{r(d_1), r(d_2), r(d_3), r(d_4), r(d_5), r(d_6), r(d_7), r(d_8), r(d_9), r(d_{10}), r(d_{11}), r(d_{12}), r(d_{13}), r(d_{14}), r(d_{15}), r(d_{16})\} = r(d_{10}) = 0{,}95$

Vemos, en este caso, que la decisión de Bayes es d_{10}.

3.5. Elección de la Función de Pérdida

La elección de la función de pérdida $L(\theta; a)$, dependerá de la naturaleza del problema y de la importancia que tenga el tomar decisiones incorrectas.
Si para nosotros tiene importancia cometer grandes errores, podríamos considerar como función de pérdida $L[\theta; a] = c[a - \theta]^2$, en donde $c > 0$, es decir, la pérdida es proporcional al cuadrado de la diferencia respecto al verdadero valor.
Si la pérdida la consideramos como proporcional al tamaño del error tomaríamos como función de pérdida $c[a - \theta]$.

Una función de pérdida comúnmente utilizada y que se adapta a distintas situaciones muy convenientemente es $L[\theta; a] = c[(\theta)(|a - \theta|)^h]$, en donde $b > 0$, $c(\theta) \geq 0 \ \forall \ \theta$.

Capítulo 4

Métodos no paramétricos de estimación de funciones de densidad

4.1. Métodos paramétricos y no paramétricos

Los procedimientos inferenciales que presentan estimaciones con respecto a los parámetros de la población de interés se llaman *métodos paramétricos* y los que no se encuentran sujetos a la forma de la distribución de la población y no requieren que las observaciones estén dadas en escala de intervalo se llaman *métodos no paramétricos.*[4],[8].
Los métodos paramétricos tienen las siguientes características:

- Precisan especificación de una distribución para la población de interés.
- Las observaciones de la mayoría de métodos tienen carácter cuantitativo.

Los métodos no paramétricos tienen las siguientes características:

- No necesitan que se especifique la forma de la distribución de la población de interés.
- Pueden aplicarse cuando la variable respuesta es cualitativa.

4.2. Kernels

La distribución de probabilidad de una variable aleatoria continua X se describe mediante una función denominada *función de densidad* $f(x)$ con la cual se pueden determinar valores de probabilidad con la definición:

$$P(a \leq X \leq b) = \int_a^b f(x)\,dx$$

Si $f(x)$ no es un modelo conocido, es de interés para la investigación poder estimar $f(x)$ a partir de una muestra de observaciones $x_1, x_2, ..., x_n$ que suponemos que son resultados independientes y tienen la misma distribución de probabilidad.

El método Kernel utiliza un conjunto de datos que provienen de una distribución continua, univariada y desconocida para aproximar esta función.
Los kernels son funciones que se asocian a cada uno de los datos. Entonces, la suma ponderada de estas funciones es un estimador para aproximar la función de densidad desconocida.

4.2.1. Propiedades de los Kernels

Definición

Un kernel es una función de variable real ($K : R \to R$) con las siguientes propiedades:

a) $K(x) \in [0, \infty)$, $x \in [-1, 1]$

b) $K(x) = 0$, $x \notin [-1, 1]$

c) $K(x) = K(-x)$

d) $\int_{-1}^{1} K(x)\, dx = 1$

e) $\int_{-1}^{1} xK(x)\, dx = 0$

f) $\int_{-1}^{1} x^2 K(x)\, dx \in R^+$

De aquí en adelante, nos referiremos solamente al intervalo en el cual K no es negativo, es decir, el intervalo $[-1, 1]$. Además, es deseable que K sea diferenciable [14].

Parametrización

Sea $h \in R^+$, el kernel parametrizado en h es $K_h(x) = \frac{1}{h}K(\frac{x}{h})$, $x \in [-h, h]$.

Esta modificación mantiene las propiedades anteriores, pero referidas al intervalo $[-h, h]$.
A h se la llama *ancho de banda de* K y es la semi-amplitud del kernel en el intervalo de interés.

Si h se incrementa ($h > 1$), la amplitud de K aumenta, pero el factor $\frac{1}{h}$ reduce el rango de K para mantener el área igual a 1.
Si h se reduce ($h < 1$), la amplitud de K se reduce, pero el factor $\frac{1}{h}$ incrementa el rango de K para mantener el área igual a 1 [14].

Traslación

El kernel se puede centrar en cualquier punto $x_i \in R$. El kernel parametrizado en h y centrado en x_i es $K_h(x) = \frac{1}{h}K(\frac{x-x_i}{h})$, $x \in [x_i - h, x_i + h]$.

Esta modificación mantiene las propiedades anteriores en el nuevo intervalo $[x_i - h, x_i + h]$. Por tanto, $K_h(x)$ es una función de densidad de probabilidad centrada en el punto x_i [14].

4.2.2. Modelos de Kernels

A continuación se describen algunos de los kernels más conocidos [14].

Kernel Rectangular o Uniforme

Es un rectángulo que se coloca sobre cada punto. Al interactuar con los kernels de los otros puntos, el efecto en la suma es un cambio abrupto.

$K(x) = 0{,}5,\ x \in [-1, 1]$

Kernel Triangular

Es un triángulo que se coloca sobre cada punto. Al interactuar con los otros kernels el efecto combinado es lineal pero más liso que los rectángulos.

$K(x) = 1 - |x|,\ x \in [-1, 1]$

Kernel de Epanechnikov

Es un segmento del perfil de un arco de parábola que se coloca sobre cada punto.

$K(x) = \frac{3}{4}(1 - x^2),\ x \in [-1, 1]$

Kernel Gaussiano o Normal

Este kernel es un caso especial. Para este kernel se define como intervalo el conjunto R, por lo que cada kernel influye en todos los otros kernels colocados en los puntos de la muestra. La suma resultante es continua y suave.

$K(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}},\ x \in (-\infty, \infty)$

4.3. Estimación de una función de densidad vía Kernel

Sea X una variable aleatoria con distribución de probabilidad continua, univariada y desconocida $f(x)$ de la cual se dispone una muestra aleatoria de n observaciones independientes:

$x_1, x_2, ..., x_n$. El objetivo es usar estos datos para obtener un estimador $\hat{f}(x)$ de la función de densidad de probabilidad $f(x)$.

4.3.1. Estimador Kernel

El estimador Kernel es el siguiente:
$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h}K(\frac{x - x_i}{h})$, $x \in [x_i - h, x_i + h]$ en cada Kernel i.
Intervalo de $\hat{f}(x)$: $[x_1 - h, x_n + h]$, suponiendo que $x_1 \leq x_2 \leq ... \leq x_n$.
El ancho de banda h es el parámetro de ajuste o suavizado de $\hat{f}(x)$ su elección es crítica para el modelo.
Mientras más pequeño es h, más concentrada está la contribución del kernel en cada punto x_i y mientras más grande es h, mayor es la influencia e interacción del kernel hacia los puntos vecinos.
En el límite, cuando $h \to 0$, la contribución de cada kernel está concentrada en cada punto x_i, así el estimador $\hat{f}(x)$ tendrá una distribución puntual concentrada en cada dato.
Por otra parte, cuando $h \to \infty$, la distribución de $\hat{f}(x)$ se aplanará, teniendo mayor dispersión.
Es necesario buscar un ancho de banda adecuado para construir el estimador [16],[14].

4.3.2. Criterios para elegir el Kernel

Varianza: El primer criterio para elegir al kernel más eficiente es seleccionar el de menor varianza. Aumentará su dispersión.

Enlace: Definimos el coeficiente $c = e^{-|d|}$, en donde d es el valor de la tangente en el borde. El valor más alto es 1, como en el modelo normal o gaussiano, que se conecta con suavidad a los otros kernels y el menor valor es 0, como en el kernel rectangular.

Efecto del factor enlace: Se debe seleccionar el kernel y el ancho de banda que permitan detectar detalles y una apariencia aceptable.

Concluimos diciendo que es mejor el kernel cuya varianza sea menor, adicionalmente el coeficiente de enlace debe ser alto para que el perfil del estimador sea liso.

4.3.3. Ancho de banda óptimo para el Kernel Gaussiano

Para obtenerlo se elige el Kernel Gaussiano, entonces $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, $x \in (-\infty, \infty)$.

Sabemos que $\int_R K^2(z)\, dz = 0{,}2821$ y que ${\sigma_K}^2 = 1$.

Sustituyendo en h^*, tenemos que $h^*_G = n^{-\frac{1}{5}}(0{,}2115\sigma^{-5})^{-\frac{1}{5}}(\frac{0{,}2821}{1^2})^{\frac{1}{5}} = 1{,}0592\sigma n^{-\frac{1}{5}}$

Pero, se han desarrollado modificaciones a esta fórmula ya que ésta es adecuada sólo si $f(x)$ se parece a la distribución normal y la siguiente funciona bien para diferentes tipos de densidades. Se debe a Silverman [3].

$h_G^* = 0{,}9n^{-\frac{1}{5}}$ mín$(\sigma, \frac{rango-intercuartil}{1{,}349})$, donde σ puede sustituirse con una estimación tomada de la muestra.

En nuestro caso, siempre supondremos que $min(\sigma, \frac{rango-intercuartil}{1{,}349}) = \sigma$.

Capítulo 5

Implementación y Resultados

El código de este proyecto se ha implementado en *Java2*, versión 7 [18].
Otra manera de haber implementado este código es utilizando el programa *R* [20]. Este lenguaje y entorno de programación es el más usado para el análisis estadístico y gráfico, además, es un proyecto de software libre, pero se eligió programar las clases en *Java* ya que estábamos en la empresa tecnólogica TISSAT y los técnicos no utilizan el lenguaje matemático *R*.

Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Como características principales están las siguientes:

- Orientado a objetos (diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones).
- Independencia de la plataforma (programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware).
- Soporte para trabajar en red.
- Ejecutar código en sistemas remotos de forma segura.
- Fácil de usar y la sintaxis deriva en gran medida de *C++*.

Aunque la última versión de Java, lanzada en marzo de 2014, es *Java SE 8*, en la empresa se ha utilizado la anterior (*Java SE 7*).

Van a ser explicadas cada variable y método utilizados en las respectivas clases implementadas para la realización de este proyecto.

Primero, empezamos por las dos clases más importantes: Política1 y Política2. (Figura 5.1). Estas dos clases implementan todos los cálculos matemáticos cuyos resultados serán vistos al final de este apartado.

Se comenzará comentando las variables de la clase Política1:

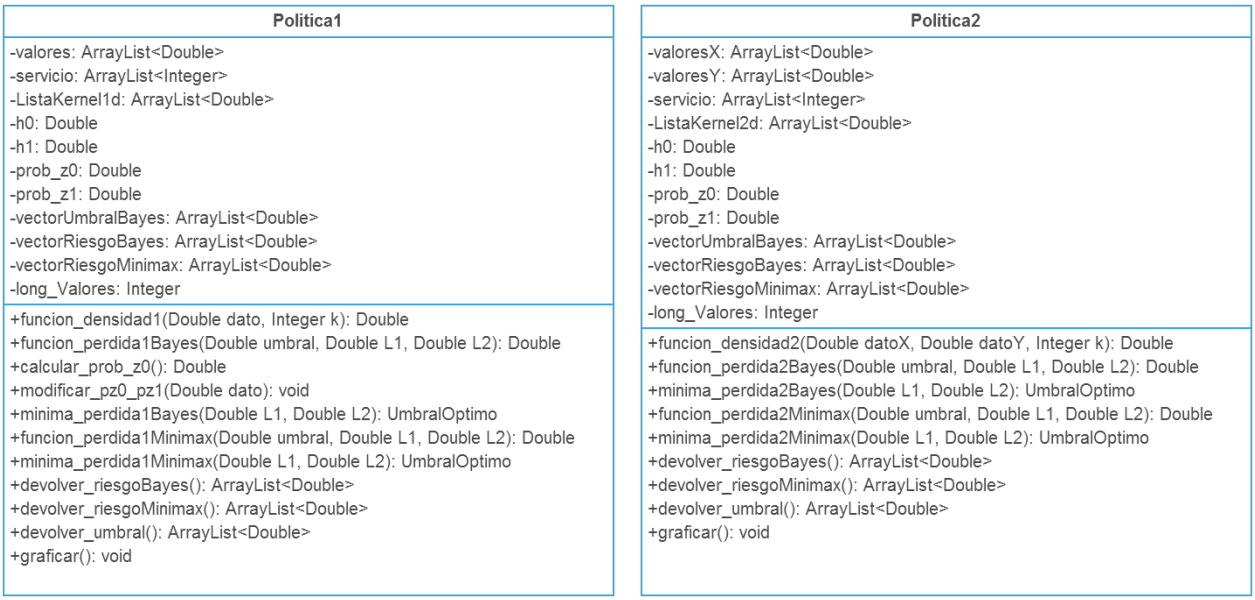

Figura 5.1: Clase Política1 y Política2. Elaboración propia.

- Valores: lista donde se almacenarán todos los valores recogidos de la base de datos y se generarán algunos más para generar fallos. Esta lista comprende valores entre 0 y 1.
- Servicio: lista que contiene si el servicio se mantiene o no. Necesario para estimar la probabilidad de funcionamiento o no.
  Se evaluará como 0 (no caída de servicio) y se alargará la lista Valores añadiendo como fallos (Servicio = 1) los valores de la lista Valores que superen el 80 % para operar como en el caso real.
- ListaKernel1d: lista de constantes que aparecen al evaluar la función de densidad en una dimensión mediante un Kernel Guassiano o Normal.
  Se calcula mediante integración de Simpson en 50 subintervalos.
- h0: ancho de banda del Kernel para la muestra de valores con servicio.
- h1: ancho de banda del Kernel para la muestra de valores sin servicio.
- prob_z0: probabilidad de que funcione el servicio (no requiere intervención necesaria).
- prob_z1: probabilidad de que no funcione el servicio (requiere intervención necesaria).
- vectorRiesgoBayes: vector con los riesgos para el criterio de Bayes.
- vectorUmbralBayes: vector con los umbrales para el criterio de Bayes.
- vectorRiesgoMinimax: vector con los riesgos para el criterio Minimax. El vector umbral es igual que el de Bayes.
- long_Valores: número de valores que tiene la lista Valores antes de hacer simulaciones.

Las variables de la clase Política2 son prácticamente las mismas y, por tanto, explicaremos sólo aquellas que difieren de la clase anterior:

- ValoresX y ValoresY: dos listas donde se almacenarán todos los valores recogidos de la base de datos y se generarán algunos más para generar fallos. Estas listas comprenden valores entre 0 y 1.

- ListaKernel2d: lista de constantes que aparecen al evaluar la función de densidad en dos dimensiones mediante un Kernel Gaussiano o Normal.
  Se calcula mediante integración de Simpson en 50×50 subintervalos.

A continuación, se explicará cada método de la clase Política1:

- En el constructor de la clase se calcula la nueva lista de Valores (generando los fallos) y se rellena la lista ListaKernel1d, evaluando la función de densidad en una dimensión mediante un Kernel Guassiano para cada valor de la lista Valores haciendo uso de la integración de Simpson para calcular dicha función de densidad.
  La fórmula matemática de la intergración de Simpson [6] es

$$\int_a^b f(x)\,dx = \frac{b-a}{6}[f(a) + 4f(\frac{a+b}{2}) + f(b)$$

  siendo en este caso $f(x) = K_h(x) = K(\frac{x-x_i}{h}) = \frac{1}{\sqrt{2\pi}}e^{(-\frac{1}{2}(\frac{x-x_i}{h})^2)}$ (Kernel Gaussiano), $a = 0$ y $b = 1$.
  En el programa, $\frac{1}{\sqrt{2\pi}}$ no lo hemos escrito ya que multiplicar por un escalar todos los Kernels de la lista ListaKernel1d no afecta a los máximos y ni a los mínimos a la hora de ser elegidos.

- funcion_densidad1: dados como parámetros un dato x y la variable k, que nos dirá si usamos el kernel cuando cae el servicio ($k = 0$) o cuando no cae ($k = 1$), calculamos la función de densidad de la siguiente forma

$$\hat{f}(u) = \frac{1}{n}\sum_{i=0}^{n} \bar{K}(\frac{x-x_i}{h})$$

  siendo $\bar{K}(\frac{x-x_i}{h}) = \dfrac{K(\frac{x-x_i}{h})}{\displaystyle\int_0^1 K(\frac{x-x_i}{h})\,dx}$
  $\bar{K}(\frac{x-x_i}{h})$ recibe el nombre de *función de distribución truncada*. Este cálculo se realiza porque hay que normalizar el Kernel Guassiano, ya que éste toma valores de $x \in (-\infty, \infty)$ y nosotros queremos que $x \in (0, 1)$.

- funcion_perdida1Bayes: dados como parámetros un umbral u, $L1$ (pérdida cuando se realiza una intervención que no es realmente necesaria) y $L2$ (pérdida cuando no se realiza una intervención siendo ésta necesaria), calculamos la función de riesgo para el criterio de Bayes de esta forma

riesgoBayes = L1*prob_1_0*prob_z0 + L2*prob_0_1*prob_z1

siendo prob_1_0 la probabilidad de emitir ticket si el sistema no falla, es decir, $P(Y_i \geq umbral|Servicio = 0) = 1 - P(Y_i < umbral|Servicio = 0)$ y siendo prob_0_1 la probabilidad de no emitir ticket si el sistema falla, es decir, $P(Y_i < umbral|Servicio = 1)$.
Estas dos probabilidades se calculan utilizando la integración compuesta de Simpson [6], siendo ésta

$$\frac{b-a}{6n}\sum_{i=0}^{n-1}\int_{x_{2i}}^{x_{2i+1}} f(x)\,dx = \frac{b-a}{6n}[f(x_0) + f(x_{2n}) + 4\sum_{i=0}^{n-1} f(x_{2i+1}) + 2\sum_{i=1}^{n-1} f(x_{2i})]$$

- calcular_prob_z0: calculamos la probabilidad de que la intervención no sea necesaria, es decir, la proporción entre los servicios que valen 0 (Servicio = 0) y todos los servicios (longitud de la lista Servicio).
- modificar_pz0_pz1: cambiar la probabilidad prob_z0 y, por tanto, también de prob_z1 = 1 - prob_z1.
- minima_perdida1Bayes: en este método rellenamos los vectores vectorUmbralBayes (valores de 0 a 1 con un intervalo de 0.01 entre valor y valor) y vectorRiesgoBayes (riesgos calculados para cada umbral del vectorUmbralBayes).
  Este método muestra el umbral cuyo riesgo es mínimo.
- funcion_perdida1Minimax: se realiza el mismo cálculo que para el método funcion_perdida1Bayes pero, en este caso, la función de riesgo para el criterio Minimax es de esta forma

  riesgoMinimax = Math.max(L1*prob_1_0, L2*prob_0_1)

- minima_perdida1Minimax: se realiza el mismo cálculo que para el método minima_perdida1Bayes. Este método muestra el umbral cuyo riesgo es mínimo.
- devolver_riesgoBayes: devuelve el vector vectorRiesgoBayes.
- devolver_riesgoMinimax: devuelve el vector vectorRiesgoMinimax.
- devolver_umbral: devuelve el vector vectorUmbralBayes, que es el mismo vector para ambos criterios.
- graficar: muestra dos gráficas en la misma pantalla.
  En la primera muestra los riesgos de Bayes (Eje Y) para cada umbral (Eje X) y en la segunda, los riesgos Minimax para cada umbral.

Los métodos de la clase Política2 son prácticamente los mismos y son los siguientes:

- En el constructor de la clase se calcula la nueva lista de Valores (generando los fallos) y se rellena la lista ListaKernel1d, evaluando la función de densidad en dos dimensiones mediante un Kernel Gaussiano para cada valor de la lista Valores haciendo uso de la integración de Simpson [6] para calcular dicha función de densidad.
- funcion_densidad2: dados como parámetros dos datos x, y y la variable k, calculamos la función de densidad de la siguiente forma

$$\hat{f}(u) = \frac{1}{n}\sum_{i=0}^{n} \bar{K}(\frac{x-x_i}{h})\bar{K}(\frac{y-y_i}{h})$$

siendo $\bar{K}(\frac{x-x_i}{h}) = \dfrac{K(\frac{x-x_i}{h})}{\int_0^1 K(\frac{x-x_i}{h})\,dx}$ y $\bar{K}(\frac{y-y_i}{h}) = \dfrac{K(\frac{y-y_i}{h})}{\int_0^1 K(\frac{y-y_i}{h})\,dx}$

- funcion_perdida2Bayes: dados como parámetros un umbral u, $L1$ y $L2$, calculamos la función de riesgo para el criterio de Bayes de la misma forma que en la clase anterior (riesgoBayes = L1*prob_1_0*prob_z0 + L2*prob_0_1*prob_z1), siendo ahora prob_1_0 la probabilidad de emitir ticket si el sistema no falla, es decir, $P(X_i \geq umbral, Y_i \geq umbral|Servicio = 0)$ y siendo prob_0_1 la probabilidad de no emitir ticket si el sistema falla, es decir, $P(X_i < umbral, Y_i < umbral|Servicio = 1)$.
  Estas dos probabilidades se calculan utilizando una regla gaussiana bidimensional de 20×20 nodos.
- calcular_prob_z0: mismo procedimiento que el usado para Política1.
- minima_perdida2Bayes: mismo procedimiento que el usado para Política1.
- funcion_perdida2Minimax: se realiza el mismo cálculo que para el método funcion_perdida2Bayes pero, en este caso, la función de riesgo para el criterio Minimax es riesgoMinimax = Math.max(L1*prob_1_0, L2*prob_0_1).
- minima_perdida2Minimax: mismo procedimiento que el usado para Política1.
- devolver_riesgoBayes: devuelve el vector vectorRiesgoBayes.
- devolver_riesgoMinimax: devuelve el vector vectorRiesgoMinimax.
- devolver_umbral: devuelve el vector vectorUmbralBayes, que es el mismo vector para ambos criterios.
- graficar: muestra dos gráficas en la misma pantalla.
  En la primera muestra los riesgos de Bayes (Eje Y) para cada umbral (Eje X) y en la segunda, los riesgos Minimax para cada umbral.

Una vez explicadas las clases que realizan las operaciones matemáticas, se explicarán las dos clases (ConexionBBDD y FicheroGere2ABBDD) que realizan las operaciones computacionales en este proyecto presentado. (Figura 5.2).

El objetivo de la clase FicheroGere2ABBDD es leer de un fichero de datos, llamado equipoGere2, los umbrales e insertar en una tabla de la base de datos Oracle todos los datos necesarios. Se va a explicar con detalle la variable y el método de esta clase:

- Y: es una lista donde se almacenarán todos los valores del fichero. Esta lista comprende valores entre 0 y 1.
- Cuerpo principal (*main*): realiza una conexión a base de datos e inserta los datos equipo-monitor-umbral.
  Se conecta a la base de datos de Oracle con la ayuda de la clase ConexionBBDD. Crea

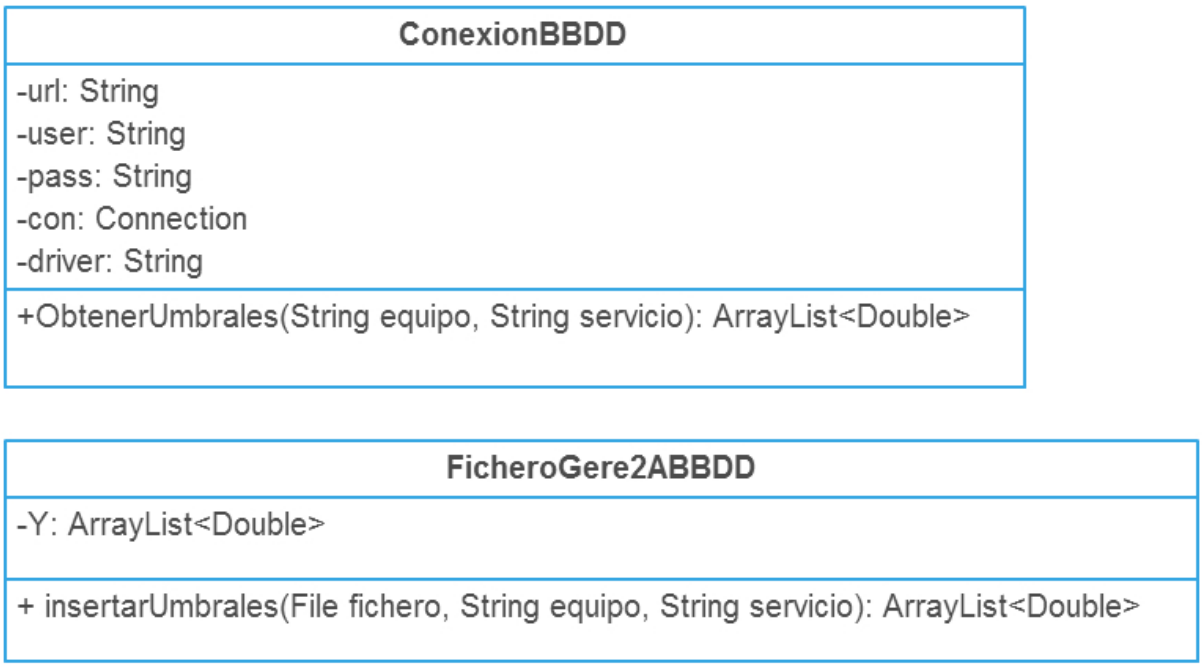

Figura 5.2: Clase ConexionBBDD y FicheroGere2ABBDD. Elaboración propia.

una lista de umbrales (método insertarUmbrales) que insertará uno a uno en la tabla I2TM_PROD.I2TM_UMBRALES_PRACTICA junto con el equipo (Gere2) y el servicio (CPU_SERVIDOR_LINUX).

- insertarUmbrales: método que lee del fichero e inserta los umbrales para el correspondiente equipo-servicio (en este caso, es siempre el mismo) en la tabla de la base de datos Oracle.

El objetivo de la clase ConexionBBDD es implementar el código necesario para conectarse con la base de datos para facilitar la programación y comprensión de la clase FicheroGere2ABBDD.
Las variables y métodos de la clase son los siguientes:

- url: la dirección de la base de datos.
- user: usuario para acceder a la base de datos.
- pass: contraseña para el usuario correspondiente.
- con: variable de la clase Connection necesaria para que se realiza la conexión.
- driver: el driver utilizado para la conexión, en este caso, es el driver *OracleDriver*. Es necesario que éste en la misma carpeta donde está almacenada esta clase Java.
- En el constructor de la clase se inicializan las variables comentadas anteriormente a los variables correctos para que la conexión se realice sin ningún inconveniente.

- ObtenerUmbrales: dados los parámetros equipo y servicio, se realiza una consulta en la tabla I2TM_PROD.I2TM_UMBRALES_PRACTICA de la base de datos Oracle que añade a una lista vacía todos los umbrales para ese equipo-servicio.

Por último, falta por explicar una clase, llamada UmbralOptimo. (Figura 5.3).

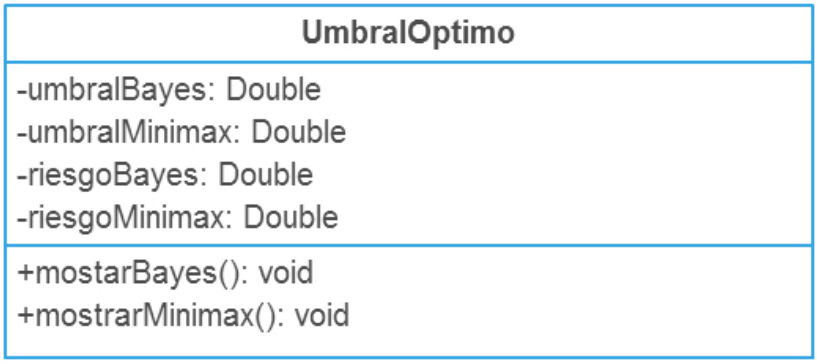

Figura 5.3: Clase UmbralOptimo. Elaboración propia.

El objetivo de esta clase es el mismo que el de la clase ConexionBBDD. Implementa el código necesario para mostrar el umbral y riesgo óptimos del riesgo Bayesiano y del riesgo Minimax para facilitar la programación y comprensión de las clases Política1 y Política2.
Las variables y métodos de la clase son los siguientes:

- umbralBayes: umbral para el criterio de Bayes.
- umbralMinimax: umbral para el criterio Minimax.
- riesgoBayes: riesgo para el criterio de Bayes.
- riesgoMinimax: riesgo para el criterio Minimax.
- mostrarBayes: muestra el umbral y riesgo óptimos para el criterio de Bayes.
- mostrarMinimax: muestra el umbral y riesgo óptimos para el criterio Minimax.

Una vez se han visto con detalle las clases Java, se estudiarán los resultados obtenidos.

El *Programa Principal*, aplicando la *Política 1*, nos muestra por pantalla todos los siguientes datos:

```
UMBRALES OPTIMOS -> Comprobando....
UMBRALES OPTIMOS -> Iniciando proceso
UMBRALES OPTIMOS -> Aplicamos la politica 1
UMBRALES OPTIMOS -> Conectando a la Base de Datos
UMBRALES OPTIMOS -> Obteniendo valores de (gere2, CPU SERVIDOR LINUX)
Numero total de valores en la tabla (gere2, CPU SERVIDOR LINUX): 1907
UMBRALES OPTIMOS -> Calculando optimo de (gere2, CPU SERVIDOR LINUX)
UMBRALES OPTIMOS -> Se aplica Politica 1
CRITERIO DE BAYES:
```

```
riesgo Bayes para el umbral 0 es 0.5479885057471264
Politica 1, utilizando el criterio de Bayes: el umbral optimo resultante es 0.78
umbral optimo: 0.78, riesgo de Bayes optimo: 0.4637428651737154
CRITERIO MINIMAX:
riesgo Minimax para el umbral 0 es 1.0
Politica 1, utilizando el criterio Minimax: el umbral optimo resultante es
0.8200000000000001
umbral optimo: 0.8200000000000001, riesgo Minimax optimo: 0.800274320447347
UMBRALES OPTIMOS -> Grafica
UMBRALES OPTIMOS -> Finalizacion proceso
```

Además de calcular el riesgo y umbral óptimos de cada criterio, calcula las dos gráficas de la Figura 5.4.
La gráfica superior corresponde a la evolución del riesgo de Bayes desde el umbral 0 hasta el umbral 1, tomando éste valores con un intervalo de 0.01 entre cada uno. Cada riesgo es dibujado en la gráfica y así podemos apreciar visualmente donde se encuentra el riesgo óptimo de este criterio (que, en este caso, el riesgo óptimo es 0.46 y ocurre cuando el umbral tiene un valor de 0.78).
Por consecuente, la gráfica inferior corresponde a la evolución del riesgo minimax. En este caso, el riesgo óptimo es 0.8 y el umbral óptimo es 0.82.

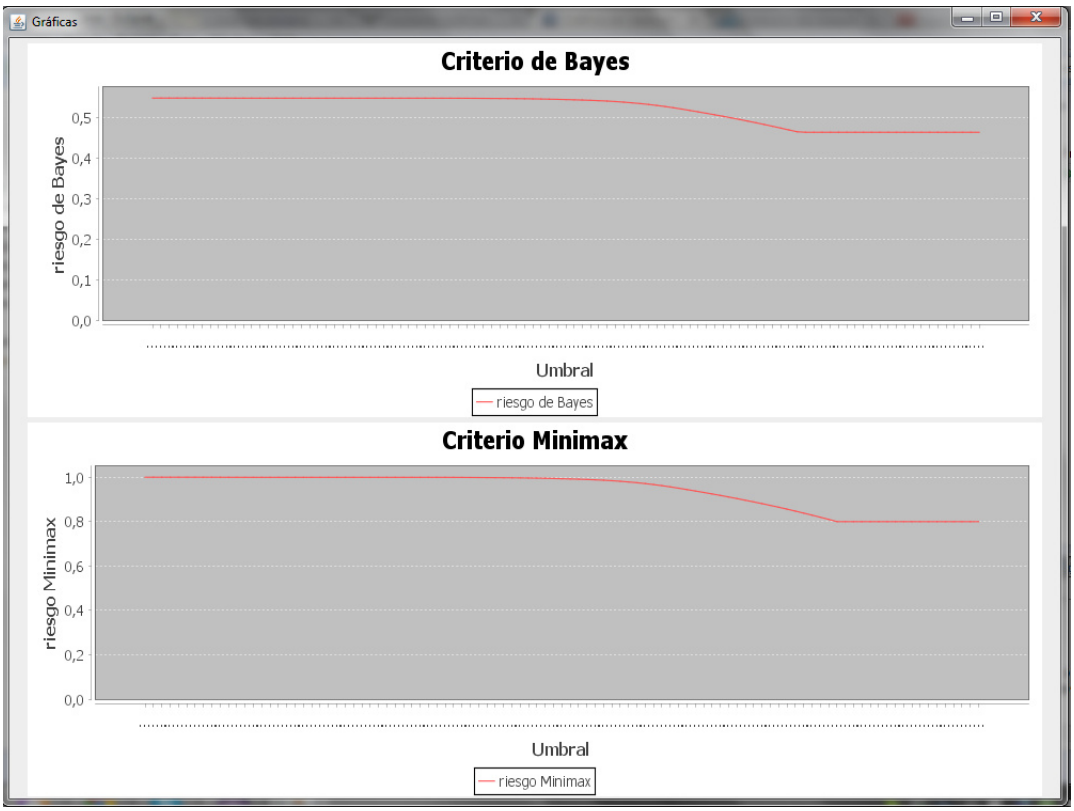

Figura 5.4: Gráfica resultante para la Política1. Elaboración propia.

El *Programa Principal*, aplicando la *Política 2*, nos muestra por pantalla todos los siguientes datos:

```
UMBRALES OPTIMOS -> Comprobando ....
```

```
UMBRALES OPTIMOS -> Iniciando proceso
UMBRALES OPTIMOS -> Aplicamos la politica 2
UMBRALES OPTIMOS -> Conectando a la Base de Datos
UMBRALES OPTIMOS -> Obteniendo valores de (gere2, CPU SERVIDOR LINUX)
Numero total de valores en la tabla (gere2, CPU SERVIDOR LINUX): 1907
UMBRALES OPTIMOS -> Calculando optimo de (gere2, CPU SERVIDOR LINUX)
UMBRALES OPTIMOS -> Se aplica Politica 2
CRITERIO DE BAYES:
riesgo Bayes para el umbral 0 es 0.6001259445843828
Politica 2, utilizando el criterio de Bayes: el umbral optimo resultante es 0.78
umbral optimo: 0.78, riesgo de Bayes optimo: 0.41887261438657747
CRITERIO MINIMAX:
riesgo Minimax para el umbral 0 es 1.0
Pol tica 2, utilizando el criterio Minimax: el umbral optimo resultante es 0.8
umbral optimo: 0.8, riesgo Minimax optimo: 0.6594587804220037
UMBRALES OPTIMOS -> Grafica
UMBRALES OPTIMOS -> Finalizacion proceso
```

Además de calcular el riesgo y umbral óptimos de cada criterio, calcula las dos gráficas de la Figura 5.5.
La gráfica superior corresponde a la evolución del riesgo de Bayes. En este caso, el riesgo óptimo es 0.418 y ocurre cuando el umbral es 0.78 (mismo valor que en la Política 1). Y, la gráfica inferior corresponde a la evolución del riesgo minimax que, en este caso, el riesgo óptimo es 0.65 y el umbral óptimo es 0.82.

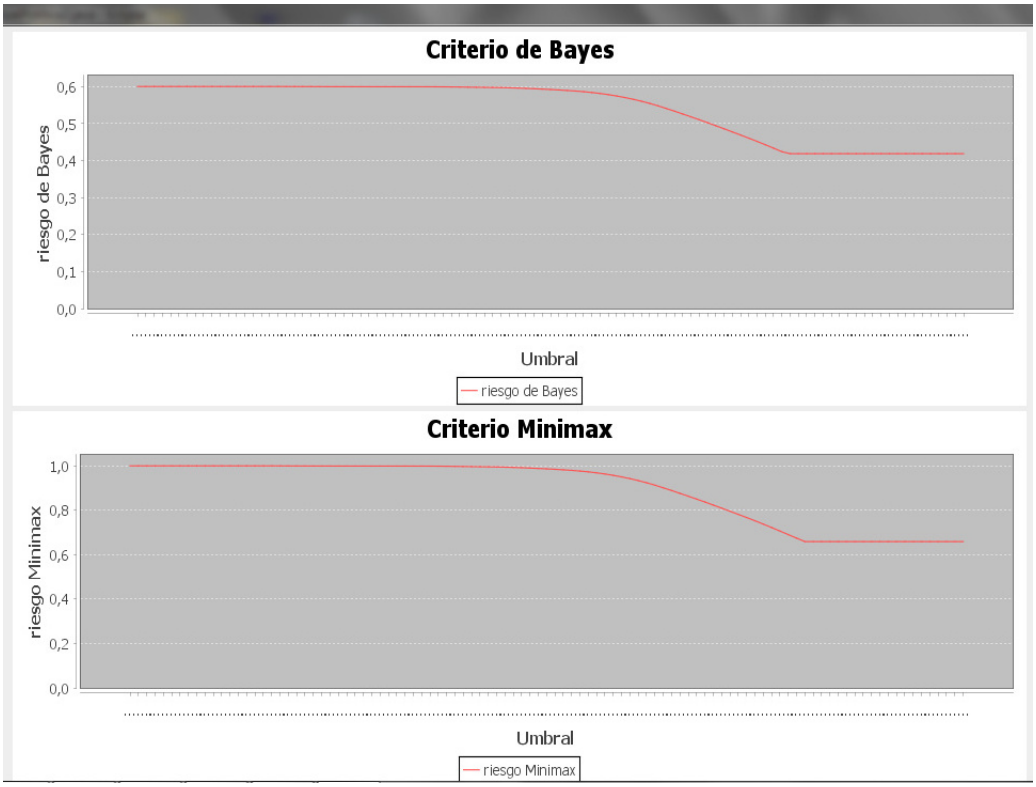

Figura 5.5: Gráfica resultante para la Política2. Elaboración propia.

Capítulo 6

Conclusiones

El problema de determinación de métodos estadísticos basados en la teoría de la decisión aplicados a la optimización de procesos de control en tecnologías de la información ha sido realizado en la empresa Tissat.

Se ha estudiado la Teoría de la Decisión, junto con los dos criterios más característicos (el criterio de Bayes y el criterio minimax) siendo aplicable al proyecto el criterio minimax.

También ha sido estudiada la estimación de funciones de densidad vía Kernel, necesaria para la realización de los cálculos matemáticos que abordaban este problema.

Se han completado satisfactoriamente todas las tareas que se propusieron en la empresa para la estancia en prácticas.

Para concluir, respecto a lo comentado en el apartado *Objetivos del proyecto*, la diferencia entre el criterio de Bayes y el minimax es que el primero es necesario un conocimiento inicial de cómo se distribuyen los estados de la naturaleza y el segundo no necesita ninguna información a priori de los estados.
Por tanto, será mejor usar el criterio minimax cuando no sepamos nada de la distribución inicial del estado de la naturaleza.

Bibliografía

[1] Arnaiz Vellando, G. 1978. *Introducción a la estadística teórica / Gonzalo Arnaiz Vellando*, Ed Lex Nova, Valladolid. 3ª Edición.

[2] *American Mathematical Society. 2010 Mathematics Subject Classification: http://www.ams.org/msc/msc2010.html [Consulta: 30 de Junio de 2015].*

[3] Brufman, J., Urbisaia, H. 2006. *Distribución del ingreso según género: un enfoque no paramétrico.* Cuadernos del Cimbage nº 8, pág.9-16.

[4] Canavos, F., George, C. 1988. *Probabilidad y Estadística. Aplicaciones y Métodos.* Ed Mc Graw Hill.

[5] Dixon, W.J., Massey F.J. 1965. *Introducción al análisis estadístico / Wilfrid J. Dixon, Frank J. Massey.* Revisado por el Dr. D. Sixto Ríos y traducido por José Pérez Vilaplana, Ediciones del Castillo, Madrid.

[6] *Integración Numérica. Regla de Simpson simple y compuesta: http://campus.fi.uba.ar/pluginfile.php/49688/mod_resource/content/0/INTEGRACION_NUMERICA.pdf [Consulta: 26 de Mayo de 2015].*

[7] Juárez, F., Villatoro, J.A., López, E.K. 2002. *Apuntes de Estadística Inferencial*, D.F.: Instituto Nacional de Psiquiatría Ramón de la Fuente, México.

[8] Lara Porras, A.M. 2001. *Diseño estadístico de experimentos, análisis de la varianza y temas relacionados: tratamiento informático mediante SPSS.* Ed Proyecto Sur.

[9] *LaTeX: http://www.latex-project.org/. [Consulta: 26 de Mayo de 2015].*

[10] *Microsoft Office. Word: https://products.office.com/es-es/word [Consulta: 26 de Mayo de 2015].*

[11] *MyOraSQL: http://www.myorasql.com/index.html [Consulta: 26 de Mayo de 2015].*

[12] *Oracle Database: https://www.oracle.com/database/index.html [Consulta: 26 de Mayo de 2015].*

[13] Ríos, S. 1971. *Métodos estadísticos / Sixto Ríos*, Ediciones del Castillo, Madrid. 6ª Edición.

[14] Rodríguez Ojeda, L. 2014. *Construcción de Kernels y funciones de densidad de probabilidad: https://www.dspace.espol.edu.ec/bitstream/123456789/25019/1/CONSTRUCCION_DE_KERNELS_Y_FUNCIONES_DE_DENSIDAD_DE_PROBABILIDAD.pdf [Consulta: 26 de Mayo de 2015].*

[15] Sidney, S. 1979. *Estadística no paramétrica aplicada a las ciencias de la conducta*, Ed Trillas, México. 5ª Edición.

[16] Victor 2014. *Estimación de la densidad del Kernel de la función de densidad de probabilidad desconocida. Artículo sobre programación en el lenguaje MLQ5 - Estadística y Análisis: https://www.mql5.com/es/articles/396 [Consulta: 26 de Mayo de 2015].*

[17] *Wikipedia. Eclipse (software): http://es.wikipedia.org/wiki/Eclipse_(software) [Consulta: 26 de Mayo de 2015].*

[18] *Wikipedia. Java (lenguaje de programación): http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29 [Consulta: 26 de Mayo de 2015].*

[19] *Wikipedia. Picasa: http://es.wikipedia.org/wiki/Picasa [Consulta: 26 de Mayo de 2015].*

[20] *Wikipedia. R (lenguaje de programación): http://es.wikipedia.org/wiki/R_%28lenguaje_de_programaci%C3%B3n%29 [Consulta: 26 de Mayo de 2015].*

[21] *Wikipedia. SQL: http://es.wikipedia.org/wiki/SQL [Consulta: 26 de Mayo de 2015].*

Anexo A

Códigos en Java

A.1. Programa Principal

```
import java.util.ArrayList;

public class Principal {

        public static void main(String[] args){

                System.out.println("UMBRALES_OPTIMOS_->_Comprobando....");
                System.out.println();

                try{
                        if (!(args.length == 1)){
                                System.out.println("ERROR:_Parametros_erroneos,_
                                    pasar_como_argumento_1_o_2_dependiendo_de_la_
                                    politica_que_use");
                        }

                        else{
                                System.out.println("UMBRALES_OPTIMOS_->_Iniciando
                                    _proceso");
                                System.out.println();

                                String politica = args[0].trim();
                                System.out.println("UMBRALES_OPTIMOS_->_Aplicamos
                                    _la_politica_" + politica);
                                System.out.println();

                                ConexionBBDD con = new ConexionBBDD();
                                System.out.println("UMBRALES_OPTIMOS_->_
                                    Conectando_a_la_Base_de_Datos");
                                System.out.println();

                                ArrayList<Double> valores = new ArrayList<Double
                                    >();

                                String equipo = "gere2";
                                String servicio = "CPU_SERVIDOR_LINUX";
```

```
valores = con.ObtenerUmbrales(equipo, servicio);
System.out.println("UMBRALES OPTIMOS -> 
    Obteniendo valores de (" + equipo + ", " +
    servicio + ")");
System.out.println();

System.out.println("Numero total de valores en la
    tabla ("+equipo+", " +servicio+"): " +
    valores.size());
System.out.println();

System.out.println("Los valores son: " + valores)
    ;
System.out.println();

UmbralOptimo umbral_optimo1 = new UmbralOptimo();
UmbralOptimo umbral_optimo2 = new UmbralOptimo();

double L1 = 1.;
double L2 = 20.;

System.out.println("UMBRALES OPTIMOS -> 
    Calculando optimo de (" + equipo + ", " +
    servicio + ")");
System.out.println();

if (politica.equals("1")){
        System.out.println("UMBRALES OPTIMOS -> 
            Se aplica Politica 1");
        System.out.println();
        System.out.println();

        Politica1 calculo1 = new Politica1(
            valores);

        System.out.println("CRITERIO DE BAYES:");
        System.out.println();
        double riesgo1Bayes = calculo1.
            funcion_perdida1Bayes(0, L1, L2);
        System.out.println("riesgo Bayes para el
            umbral 0 es " + riesgo1Bayes);
        System.out.println();

        umbral_optimo1 = calculo1.
            minima_perdida1Bayes(L1, L2);

        System.out.println("Politica 1, 
            utilizando el criterio de Bayes: el
            umbral optimo resultante es " +
            umbral_optimo1.get_umbralBayes());
        System.out.println();
        umbral_optimo1.mostrarBayes();
        System.out.println();
        System.out.println("vectorUmbralBayes = "
             + calculo1.devolver_umbral());
        System.out.println();
        System.out.println("vectorRiesgoBayes = "
             + calculo1.devolver_riesgoBayes());
        System.out.println();
        System.out.println();
```

```
                System.out.println("CRITERIO MINIMAX:");
                System.out.println();
                double riesgo1Minimax = calculo1.
                    funcion_perdida1Minimax(0, L1, L2);
                System.out.println("riesgo Minimax para
                    el umbral 0 es " + riesgo1Minimax);
                System.out.println();

                umbral_optimo1 = calculo1.
                    minima_perdida1Minimax(L1, L2);

                System.out.println("Politica 1,
                    utilizando el criterio Minimax: el
                    umbral optimo resultante es " +
                    umbral_optimo1.get_umbralMinimax());
                System.out.println();
                umbral_optimo1.mostrarMinimax();
                System.out.println();
                System.out.println("vectorUmbralMinimax =
                    " + calculo1.devolver_umbral());
                System.out.println();
                System.out.println("vectorRiesgoMinimax =
                    " + calculo1.devolver_riesgoMinimax()
                    );

                calculo1.graficar();
            }

            else{
                System.out.println("UMBRALES OPTIMOS -> Se
                    aplica Politica 2");
                System.out.println();
                System.out.println();

                Politica2 calculo2 = new Politica2(
                    valores,2);

                System.out.println("CRITERIO DE BAYES:");
                System.out.println();
                double riesgo2Bayes = calculo2.
                    funcion_perdida2Bayes(0, L1, L2);
                System.out.println("riesgo Bayes para el
                    umbral 0 es " + riesgo2Bayes);
                System.out.println();

                umbral_optimo2 = calculo2.
                    minima_perdida2Bayes(L1, L2);

                System.out.println("Politica 2,
                    utilizando el criterio de Bayes: el
                    umbral optimo resultante es " +
                    umbral_optimo2.get_umbralBayes());
                System.out.println();
                umbral_optimo2.mostrarBayes();
                System.out.println();
                System.out.println("vectorUmbralBayes =" +
                     calculo2.devolver_umbral());
                System.out.println();
```

```
                                        System.out.println("vectorRiesgoBayes = "
                                            + calculo2.devolver_riesgoBayes());
                                        System.out.println();
                                        System.out.println();


                                        System.out.println("CRITERIO MINIMAX:");
                                        System.out.println();
                                        double riesgo2Minimax = calculo2.
                                            funcion_perdida2Minimax(0, L1, L2);
                                        System.out.println("riesgo Minimax para
                                            el umbral 0 es " + riesgo2Minimax);
                                        System.out.println();

                                        umbral_optimo2 = calculo2.
                                            minima_perdida2Minimax(L1, L2);

                                        System.out.println("Politica 2,
                                            utilizando el criterio Minimax: el
                                            umbral optimo resultante es " +
                                            umbral_optimo2.get_umbralMinimax());
                                        System.out.println();
                                        umbral_optimo2.mostrarMinimax();
                                        System.out.println();
                                        System.out.println("vectorUmbralMinimax =
                                            " + calculo2.devolver_umbral());
                                        System.out.println();
                                        System.out.println("vectorRiesgoMinimax =
                                            " + calculo2.devolver_riesgoMinimax()
                                            );

                                        calculo2.graficar();
                                }

                                con.closeConnection();
                                System.out.println();
                                System.out.println("UMBRALES OPTIMOS ->
                                    Finalizacion proceso");
                        }
                }

                catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

A.2. Politica1

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;

import javax.swing.JFrame;
import javax.swing.JPanel;
```

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;


public class Politica1 extends JFrame{

        // Lista donde se almacenaran todos los valores recogidos de la BBDD y se
            generaran algunos mas para generar fallos. Valores entre [0,1].
        private ArrayList<Double> Valores = new ArrayList<Double>();


        // Lista que contiene si el servicio se mantiene o no. Necesario para
            estimar la probabilidad de funcionamiento o no.
        /* Se evaluara como 0 (no caida de servicio) y se alargara la lista
            Valores anyadiendo como fallos (Servicio=1) los
        valores de la lista Valores que superen el 80% para operar como en un
            caso real. */
        private ArrayList<Integer> Servicio = new ArrayList<Integer>();


        // Lista de constantes que aparecen al evaluar la funcion de densidad en
            una dimensi n mediante un Kernel Gaussiano o Normal.
        // Se calcula mediante integracion de Simpson en 50 subintervalos.
        private ArrayList<Double> ListaKernel1d = new ArrayList<Double>();


        // Ancho de banda del Kernel1d para la muestra de valores con servicio.
        private double h0;


        // Ancho de banda del Kernel1d para la muestra de valores sin servicio.
    private double h1;


    // Probabilidad de que funcione el servicio (no requiere intervencion
        necesaria).
    private double prob_z0;


    // Probabilidad de que no funcione el servicio (requiere intervencion
        necesaria).
    private double prob_z1;


    // Vector con los riesgos y umbrales para el criterio de Bayes.
    private ArrayList<Double> vectorUmbralBayes = new ArrayList<Double>();
        private ArrayList<Double> vectorRiesgoBayes = new ArrayList<Double>();


    // Vectores con los riesgos y umbrales para el criterio Minimax. El vector
        umbral es igual que el de Bayes.
        private ArrayList<Double> vectorRiesgoMinimax = new ArrayList<Double>();


        // Numero de valores que tiene la lista Valores antes de hacer
            simulaciones.
```

```
    private int long_Valores;


public Politica1(){
}


public Politica1(ArrayList<Double> valores){

    // lista valores tiene todos los umbrales recogidos de la BBDD
    for (int i=0; i < valores.size(); i++){
            Valores.add(valores.get(i));
            Servicio.add(0);
    }


    // Longitud de la lista Valores, antes de realizar ninguna simulacion
    long_Valores = Valores.size();


    // Contador de la cantidad de datos que anyadimos en la lista Valores, es
         decir, el numero de datos nuevos que anyadimos en la lista.
    int diferencia = 0;


    // Recorrer la lista Valores, y para cada valor que supere el 80% se
        anyade en la lista Valores y se anyade 1 en la lista Servicio (caida
        de servicio).
    for (int i = 0; i < long_Valores; i++){
            if (Valores.get(i) >= 0.8){
                    Valores.add(Valores.get(i));
                    Servicio.add(1);
                    diferencia++;
            }
    }


    // Inicializar las probabilidades de que funcione o no el servicio (
        prob_z0 calculado en un metodo aparte).
    prob_z0 = calcular_prob_z0();
    prob_z1 = 1 - prob_z0;


    // Contador de la suma de los valores con servicio 0 o 1 con exponente 1
        o 2 (elevados al cuadrado) respectivamente.
    double Valores_Serv0_Exp1=0, Valores_Serv0_Exp2=0;
    double Valores_Serv1_Exp1=0, Valores_Serv1_Exp2=0;


    // Contador del numero de valores que hay con servicio 0 o 1
        respectivamente.
    int Cont_Serv0=0,Cont_Serv1=0;


    // Recorrer la lista valores para realizar el sumatorio de los valores y
        de sus cuadrados cuando el servicio es 0 o 1.
    for (int i = 0; i < Valores.size(); i++){
            if (Servicio.get(i) == 0){
                    Valores_Serv0_Exp1 += Valores.get(i);
                    Valores_Serv0_Exp2 += Math.pow(Valores.get(i), 2);
```

```
                Cont_Serv0++;
        }

        else if (Servicio.get(i) == 1){
                Valores_Serv1_Exp1 += Valores.get(i);
                Valores_Serv1_Exp2 += Math.pow(Valores.get(i), 2);
                Cont_Serv1++;
        }
}


// Calcular la media de todos los valores cuando el servicio es 0 y el
    exponente es 1 o 2.
double Media_Serv0_Exp1 = Valores_Serv0_Exp1/Cont_Serv0;
double Media_Serv0_Exp2 = Valores_Serv0_Exp2/Cont_Serv0;


// Varianza para los valores cuando el servicio es 0.
double Varianza_Serv0 = Math.sqrt((Media_Serv0_Exp2 - Math.pow(
    Media_Serv0_Exp1, 2)));


// Calcular el ancho de banda del Kernel1d para la muestra de valores.
h0 = 0.9*Varianza_Serv0/Math.pow(Cont_Serv0, 0.2);


// Calcular la media de todos los valores cuando el servicio es 1 y el
    exponente es 1 o 2.
double Media_Serv1_Exp1 = Valores_Serv1_Exp1/Cont_Serv1;
double Media_Serv1_Exp2 = Valores_Serv1_Exp2/Cont_Serv1;


// Varianza para los valores cuando el servicio es 1.
double Varianza_Serv1 = Math.sqrt((Media_Serv1_Exp2 - Math.pow(
    Media_Serv1_Exp1, 2)));


// Calcular el ancho de banda del Kernel1d para la muestra de valores.
h1 = 0.9*Varianza_Serv1/Math.pow(Cont_Serv1, 0.2);


// el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/6.
double paso = 0.01;


// Integracion de Simpson en 50 subintervalos. Integracion entre [0,1].
double Kernel1d, a, b;
for (int i = 0; i < Valores.size(); i++){
        Kernel1d = 0;

        if (Servicio.get(i) == 0){

                for (int j = 0; j <= 50; j++){ // intervalo de [0,1] =
                    [0, 0.02, 0.04, ..., 0.98, 1]
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
                        Kernel1d += Math.exp(-0.5*Math.pow((a - Valores.
                            get(i))/h0,2));
                        Kernel1d += 4*Math.exp(-0.5*Math.pow(((a+b)/2 -
                            Valores.get(i))/h0,2));
```

```
                        Kernel1d += Math.exp(-0.5*Math.pow((b - Valores.
                            get(i))/h0,2));
                }
            }

            else if (Servicio.get(i) == 1){

                for (int j = 0; j <= 50; j++){
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
                        Kernel1d += Math.exp(-0.5*Math.pow((a - Valores.
                            get(i))/h1,2));
                        Kernel1d += 4*Math.exp(-0.5*Math.pow(((a+b)/2 -
                            Valores.get(i))/h1,2));
                        Kernel1d += Math.exp(-0.5*Math.pow((b - Valores.
                            get(i))/h1,2));
                }
            }

            // Resultado: Lista de constantes que aparecen al evaluar la
                funcion de densidad en una dimension mediante un Kernel
                Gaussiano o Normal.
            Kernel1d = Kernel1d*paso/3; // (b-a)/6 = paso/3
            ListaKernel1d.add(Kernel1d);
    }
}


public double funcion_densidad1 (double dato, int k){  // f(u) = sumatorio[K
    ((u - Y_i)/h)/integral_0_1(K((u - Y_i)/h))]/n.

    // sumatorio de todas las funciones de densidad.
    double f_densidad = 0;


    // numero total de elementos.
    int n = 0;


    // k nos dira si usamos el kernel cuando cae el servicio (k==0) o cuando
        no cae (k==1).
    double h;
    if (k==0){
            h = h0;
    }
    else{
            h = h1;
    }


    // calcular la funcion de densidad.
    for (int i = 0; i < Valores.size(); i++){
            if (k == Servicio.get(i)){
                    f_densidad += Math.exp(-0.5*Math.pow((dato - Valores.get(
                        i))/h, 2))/ListaKernel1d.get(i);
                    n++;
            }
    }

    return f_densidad/n;
```

```
}

public double funcion_perdida1Bayes (double umbral, double L1, double L2){
    /* parametros:  L1 (perdida cuando se realiza una intervencion que no es
        realmente necesaria)
                                    L2 (perdida cuando no se realiza una
                                        intervencion siendo esta necesaria) */


    // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/(6n).
    double paso = umbral*0.01;


    // Calculo de la probabilidad de emitir ticket si el sistema no falla. P(
        Yi >= Umbral | Serv = 0)
    double prob_1_0;

    // Integracion compuesta de Simpson.
    double densidad1 = this.funcion_densidad1(0, 0) + this.funcion_densidad1(
        umbral, 0);
    for (int i = 0; i < 50; i++){
            densidad1 += 4*this.funcion_densidad1((2*i+1)*paso, 0);
    }
    for (int i = 1; i < 50; i++){
            densidad1 += 2*this.funcion_densidad1(2*i*paso, 0);
    }

    prob_1_0 = 1 - densidad1*paso/3;


    // Calculo de la probabilidad de no emitir ticket si el sistema falla. P(
        Yi < Umbral | Serv = 1)
    double prob_0_1;

    // Integracion compuesta de Simpson.
    double densidad2 = this.funcion_densidad1(0, 1) + this.funcion_densidad1(
        umbral, 1);
    for (int i = 0; i < 50; i++){
            densidad2 += 4*this.funcion_densidad1((2*i+1)*paso, 1);
    }
    for (int i = 1; i < 50; i++){
            densidad2 += 2*this.funcion_densidad1(2*i*paso, 1);
    }

    prob_0_1 = densidad2*paso/3;


    // Formula para calcular el riesgo de Bayes.
    double riesgoBayes = L1*prob_1_0*prob_z0 + L2*prob_0_1*prob_z1;

    return riesgoBayes;
}

public double calcular_prob_z0(){ // probabilidad de que la intervencion no
    sea necesaria.

    double no_nec_serv = 0;

    for (int i = 0; i < Valores.size(); i++){
            if (Servicio.get(i) == 0){
```

```
                    no_nec_serv++;
                }
            }

            return no_nec_serv/Valores.size();
        }

        public void modificar_pz0_pz1 (double dato){ // "dato" por ser una
            probabilidad debe estar entre 0 y 1.

            this.prob_z0 = dato;
            this.prob_z1 = 1 - dato;
        }

        public UmbralOptimo minima_perdida1Bayes (double L1, double L2){ // calcular
            el umbral que tenga la minima funcion de perdida.

            UmbralOptimo umbral_optimo = new UmbralOptimo();


            // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/6.
            double paso = 0.01;
            double umbral = 0;
            umbral_optimo.set_umbralBayes(0);

            umbral_optimo.riesgoBayes = this.funcion_perdida1Bayes(umbral, L1, L2);

            vectorUmbralBayes.add(umbral);
            vectorRiesgoBayes.add(umbral_optimo.riesgoBayes);

            for( int i = 1; i <= 100; i++){ // intervalo de [0,1] = [0, 0.01, 0.02,
                ..., 0.98, 0.99, 1]
                    umbral = i*paso;

                    vectorUmbralBayes.add(umbral);
                    vectorRiesgoBayes.add(umbral_optimo.riesgoBayes);

                    if (umbral_optimo.riesgoBayes > this.funcion_perdida1Bayes(umbral
                        , L1, L2)){
                            umbral_optimo.umbralBayes = umbral;
                            umbral_optimo.riesgoBayes = this.funcion_perdida1Bayes(
                                umbral, L1, L2);
                    }
            }

            //System.out.println("vectorUmbralBayes = " + vectorUmbralBayes);
            //System.out.println("vectorRiesgoBayes = " + vectorRiesgoBayes);
            return umbral_optimo;
        }


        public double funcion_perdida1Minimax (double umbral, double L1, double L2){
            /* parametros: L1 (perdida cuando se realiza una intervencion que no es
                realmente necesaria)
                                        L2 (perdida cuando no se realiza una
                                            intervencion siendo esta necesaria) */


            // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/(6n).
            double paso = umbral*0.01;
```

```
        // Calculo de la probabilidad de emitir ticket si el sistema no falla. P(
            Yi >= Umbral | Serv = 0)
        double prob_1_0;

        // Integracion compuesta de Simpson.
        double densidad1 = this.funcion_densidad1(0, 0) + this.funcion_densidad1(
            umbral, 0);
        for (int i = 0; i < 50; i++){
                densidad1 += 4*this.funcion_densidad1((2*i+1)*paso, 0);
        }
        for (int i = 1; i < 50; i++){
                densidad1 += 2*this.funcion_densidad1(2*i*paso, 0);
        }

        prob_1_0 = 1 - densidad1*paso/3;


        // Calculo de la probabilidad de no emitir ticket si el sistema falla. P(
            Yi < Umbral | Serv = 1)
        double prob_0_1;

        // Integracion compuesta de Simpson.
        double densidad2 = this.funcion_densidad1(0, 1) + this.funcion_densidad1(
            umbral, 1);
        for (int i = 0; i < 50; i++){
                densidad2 += 4*this.funcion_densidad1((2*i+1)*paso, 1);
        }
        for (int i = 1; i < 50; i++){
                densidad2 += 2*this.funcion_densidad1(2*i*paso, 1);
        }

        prob_0_1 = densidad2*paso/3;


        // Formula para calcular el riesgo Minimax.
        double riesgoMinimax = Math.max(L1*prob_1_0, L2*prob_0_1);

        return riesgoMinimax;
    }

    public UmbralOptimo minima_perdida1Minimax (double L1, double L2){ //
        calcular el umbral que tenga la minima funcion de perdida.

        UmbralOptimo umbral_optimo = new UmbralOptimo();


        // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/6.
        double paso = 0.01;
        double umbral = 0;
        umbral_optimo.set_umbralMinimax(0);

        umbral_optimo.riesgoMinimax = this.funcion_perdida1Minimax(umbral, L1, L2
            );

        vectorRiesgoMinimax.add(umbral_optimo.riesgoMinimax);

        for( int i = 1; i <= 100; i++){ //
                umbral = i*paso;
```

```
            vectorRiesgoMinimax.add(umbral_optimo.riesgoMinimax);

            if (umbral_optimo.riesgoMinimax > this.funcion_perdida1Minimax(
                umbral, L1, L2)){
                    umbral_optimo.umbralMinimax = umbral;
                    umbral_optimo.riesgoMinimax = this.
                        funcion_perdida1Minimax(umbral, L1, L2);
            }
    }

    return umbral_optimo;
}

public ArrayList<Double> devolver_riesgoBayes(){
    return vectorRiesgoBayes;
}

public ArrayList<Double> devolver_riesgoMinimax(){
    return vectorRiesgoMinimax;
}

public ArrayList<Double> devolver_umbral(){
    return vectorUmbralBayes;
}

public void graficar(){
    setTitle("Graficas");
            setSize(1000, 750);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            setVisible(true);

            JPanel panel = new JPanel();
            getContentPane().add(panel);

            DefaultCategoryDataset linea1 = new DefaultCategoryDataset();
            DefaultCategoryDataset linea2 = new DefaultCategoryDataset();

            for (int i=0; i < devolver_riesgoBayes().size(); i++){
                    String umbral = Double.toString(i*0.01);
                    linea1.addValue(devolver_riesgoBayes().get(i), "riesgo de
                        Bayes", umbral);
            }

            for (int i=0; i < devolver_riesgoMinimax().size(); i++){
                    String umbral = Double.toString(i*0.01);
                    linea2.addValue(devolver_riesgoMinimax().get(i), "riesgo
                        Minimax", umbral);
            }

            JFreeChart chart1 = ChartFactory.createLineChart("Criterio de
                Bayes", "Umbral", "riesgo de Bayes", linea1, PlotOrientation.
                VERTICAL, true, true, false);
            JFreeChart chart2 = ChartFactory.createLineChart("Criterio
                Minimax", "Umbral", "riesgo Minimax", linea2, PlotOrientation.
                VERTICAL, true, true, false);

            ChartPanel chartPanel1 = new ChartPanel(chart1, 950, 350, 0, 0,
                900, 300, false, false, false, true, true, false);
```

```
                panel.add(chartPanel1);

                ChartPanel chartPanel2 = new ChartPanel(chart2, 950, 350, 0, 0,
                    900, 300, false, false, false, true, true, false);
                panel.add(chartPanel2);

                setVisible(true);
    }

/*      public void graficarBayes(){
          setTitle("Grafica 1");
                setSize(1000, 700);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setVisible(true);

                JPanel panel = new JPanel();
                getContentPane().add(panel);

                DefaultCategoryDataset linea = new DefaultCategoryDataset();

                for (int i=0; i < devolver_riesgoBayes().size(); i++){
                        String umbral = Double.toString(i*0.01);
                        linea.addValue(devolver_riesgoBayes().get(i), "riesgo de
                            Bayes", umbral);
                }

                JFreeChart chart = ChartFactory.createLineChart("Criterio de
                    Bayes", "Umbral", "riesgo de Bayes", linea, PlotOrientation.
                    VERTICAL, true, true, false);

                ChartPanel chartPanel1 = new ChartPanel(chart, 950, 650, 0, 0,
                    900, 600, false, false, false, true, true, false);
                panel.add(chartPanel1);

                setVisible(true);
    }

    public void graficarMinimax(){
        setTitle("Grafica 2");
                setSize(1000, 700);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setVisible(true);

                JPanel panel = new JPanel();
                getContentPane().add(panel);

                DefaultCategoryDataset linea = new DefaultCategoryDataset();

                for (int i=0; i < devolver_riesgoMinimax().size(); i++){
                        String umbral = Double.toString(i*0.01);
                        linea.addValue(devolver_riesgoMinimax().get(i), "riesgo
                            Minimax", umbral);
                }

                JFreeChart chart = ChartFactory.createLineChart("Criterio Minimax
                    ", "Umbral", "riesgo Minimax", linea, PlotOrientation.VERTICAL
                    , true, true, false);
```

```
                ChartPanel chartPanel2 = new ChartPanel(chart, 950, 650, 0, 0,
                    900, 600, false, false, false, true, true, false);
                panel.add(chartPanel2);

                setVisible(true);
    }
*/
}
```

A.3. Politica2

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;

import javax.swing.JFrame;
import javax.swing.JPanel;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;


public class Politica2 extends JFrame{

        // Listas donde se almacenaran todos los valores recogidos de la BBDD y
            se generaran algunos mas para generar fallos.
        // ValoresX, ValoresY entre [0,1].
        private ArrayList<Double> ValoresX = new ArrayList<Double>();
        private ArrayList<Double> ValoresY = new ArrayList<Double>();


        // Lista que contiene si el servicio se mantiene o no. Necesario para
            estimar la probabilidad de funcionamiento o no.
        /* Se evaluara como 0 (no caida de servicio) y se alargara la lista
            Valores anyadiendo como fallos (Servicio=1) los
        valores de la lista Valores que superen el 80% para operar como en un
            caso real. */
        private ArrayList<Integer> Servicio = new ArrayList<Integer>();


        // Lista de constantes que aparecen al evaluar la funcion de densidad en
            dos dimensiones mediante un Kernel Gaussiano o Normal.
        // Se calcula mediante integracion de Simpson en 50x50 subintervalos.
        private ArrayList<Double> ListaKernel2d = new ArrayList<Double>();


        // Ancho de banda del Kernel2d para la muestra de valores con servicio.
        private double h0;


        // Ancho de banda del Kernel2d para la muestra de valores sin servicio.
    private double h1;
```

```
// Probabilidad de que funcione el servicio (no requiere intervencion
    necesaria).
private double prob_z0;


// Probabilidad de que no funcione el servicio (requiere intervencion
    necesaria).
private double prob_z1;


// Vector con los riesgos y umbrales para el criterio de Bayes.
private ArrayList<Double> vectorUmbralBayes = new ArrayList<Double>();
    private ArrayList<Double> vectorRiesgoBayes = new ArrayList<Double>();


// Vectores con los riesgos y umbrales para el criterio Minimax. El vector
    umbral es igual que el de Bayes.
    private ArrayList<Double> vectorRiesgoMinimax = new ArrayList<Double>();


public Politica2(){
}


public Politica2(ArrayList <Double> valores, int politica){

    if (politica == 2){

            // Longitud de la lista valores (umbrales de la BBDD), antes de
                realizar ninguna simulacion.
            int long_Valores = valores.size();


            // Contador de la cantidad de datos que anyadimos en la lista
                ValoresX e ValoresY, es decir, el numero de datos nuevos que
                anyadimos en las listas.
            int diferencia = 0;


            for(int i = 1; i < long_Valores; i++){
                    ValoresX.add(valores.get(i-1));
                    ValoresY.add(valores.get(i));
                    Servicio.add(0);
            }

            long_Valores = ValoresX.size();

            for (int i = 0; i < long_Valores; i++){
                    if(ValoresX.get(i) >= 0.8 & ValoresY.get(i) >= 0.8){
                            ValoresX.add(ValoresX.get(i));
                            ValoresY.add(ValoresY.get(i));
                            Servicio.add(1);
                            diferencia++;
                    }
            }
```

```
// Inicializar las probabilidades de que funcione o no el
    servicio.
prob_z0=(double) long_Valores/(long_Valores + diferencia);
prob_z1=1 - prob_z0;


// Contador de la suma de los valores con servicio 0 o 1 con
    exponente 1 o 2 (elevados al cuadrado) respectivamente.
double Valores_Serv0_Exp1 = 0, Valores_Serv0_Exp2 = 0;
double Valores_Serv1_Exp1 = 0, Valores_Serv1_Exp2 = 0;


// Contador del numero de valores que hay con servicio 0 o 1
    respectivamente.
int Cont_Serv0 = 0, Cont_Serv1 = 0;


// Recorrer la lista ValoresY para realizar el sumatorio de los
    valores y de sus cuadrados cuando el servicio es 0 o 1.
for (int i = 0; i < ValoresY.size(); i++){
        if (Servicio.get(i) == 0){
                Valores_Serv0_Exp1 += ValoresY.get(i);
                Valores_Serv0_Exp2 += Math.pow(ValoresY.get(i),
                    2);
                Cont_Serv0++;
        }
        else if (Servicio.get(i) == 1){
                Valores_Serv1_Exp1 += ValoresY.get(i);
                Valores_Serv1_Exp2 += Math.pow(ValoresY.get(i),
                    2);
                Cont_Serv1++;
        }
}


// Calcular la media de todos los valores cuando el servicio es 0
     y el exponente es 1 o 2.
double Media_Serv0_Exp1 = Valores_Serv0_Exp1/Cont_Serv0;
double Media_Serv0_Exp2 = Valores_Serv0_Exp2/Cont_Serv0;


// Varianza para los valores cuando el servicio es 0.
double Varianza_Serv0 = Math.sqrt((Media_Serv0_Exp2 - Math.pow(
    Media_Serv0_Exp1, 2)));


// Calcular el ancho de banda del Kernel2d para la muestra de
    valores.
h0 = 0.9*Varianza_Serv0/Math.pow(Cont_Serv0, 0.2);


// Calcular la media de todos los valores cuando el servicio es 1
     y el exponente es 1 o 2.
double Media_Serv1_Exp1 = Valores_Serv1_Exp1/Cont_Serv1;
double Media_Serv1_Exp2 = Valores_Serv1_Exp2/Cont_Serv1;


// Varianza para los valores cuando el servicio es 1.
double Varianza_Serv1 = Math.sqrt((Media_Serv1_Exp2 - Math.pow(
    Media_Serv1_Exp1, 2)));
```

```
// Calcular el ancho de banda del Kernel2d para la muestra de
    valores.
h1 = 0.9*Varianza_Serv1/Math.pow(Cont_Serv1, 0.2);


// el paso entre subintervalo y subintervalo es de 0.01 = (b-a)
    /6.
double paso = 0.01;


// Integracion de Simpson en 50x50 subintervalos. Integracion
    entre [0,1].
double Kernel1d,Kernel2d, a, b;
for (int i = 0; i < ValoresY.size(); i++){
        Kernel1d = 0;
        Kernel2d = 0;

        if (Servicio.get(i) == 0){

                for (int j = 0; j < 50; j++){
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
                        Kernel1d += Math.exp(-0.5*Math.pow((a -
                            ValoresX.get(i))/h0, 2));
                        Kernel1d += 4*Math.exp(-0.5*Math.pow(((a+
                            b)/2 - ValoresX.get(i))/h0, 2));
                        Kernel1d += Math.exp(-0.5*Math.pow((b -
                            ValoresX.get(i))/h0, 2));
                }

                for (int j = 0; j < 50; j++){
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
                        Kernel2d += Math.exp(-0.5*Math.pow((a -
                            ValoresY.get(i))/h0, 2));
                        Kernel2d += 4*Math.exp(-0.5*Math.pow(((a+
                            b)/2 - ValoresY.get(i))/h0, 2));
                        Kernel2d += Math.exp(-0.5*Math.pow((b -
                            ValoresY.get(i))/h0, 2));
                }
        }

        else if (Servicio.get(i) == 1){

                for (int j = 0; j < 50; j++){
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
                        Kernel1d += Math.exp(-0.5*Math.pow((a -
                            ValoresX.get(i))/h1, 2));
                        Kernel1d += 4*Math.exp(-0.5*Math.pow(((a+
                            b)/2 - ValoresX.get(i))/h1, 2));
                        Kernel1d += Math.exp(-0.5*Math.pow((b -
                            ValoresX.get(i))/h1, 2));
                }
                for(int j = 0; j < 50; j++){
                        a = 2*j*paso;
                        b = 2*j*paso + 2*paso;
```

```
                                        Kernel2d += Math.exp(-0.5*Math.pow((a -
                                            ValoresY.get(i))/h1, 2));
                                        Kernel2d += 4*Math.exp(-0.5*Math.pow(((a+
                                            b)/2 - ValoresY.get(i))/h1, 2));
                                        Kernel2d += Math.exp(-0.5*Math.pow((b -
                                            ValoresY.get(i))/h1, 2));
                                }
                        }

                        ListaKernel2d.add(Kernel1d*Kernel2d*paso*paso/9);
                }
        }
}


        public double funcion_densidad2 (double datoX, double datoY, int k){  //
            f(u) = sumatorio[K((u - Y_i)/h)/integral_0_1(K((u - Y_i)/h))]/n.

        // sumatorio de todas las funciones de densidad.
        double f_densidad = 0;


        // numero total de elementos.
        int n = 0;


        // k nos dira si usamos el kernel cuando cae el servicio (k==0) o cuando
            no cae (k==1).
        double h;
        if (k==0){
                h = h0;
        }
        else{
                h = h1;
        }


        // calcular la funcion de densidad.
        for (int i = 0; i < ValoresX.size(); i++){
                if (k == Servicio.get(i)){
                        f_densidad += Math.exp(-0.5*Math.pow((datoX - ValoresX.
                            get(i))/h, 2))*Math.exp(-0.5*Math.pow((datoY -
                            ValoresY.get(i))/h, 2))/ListaKernel2d.get(i);
                        n++;
                }
        }

        return f_densidad/n;
}

public double funcion_perdida2Bayes (double umbral, double L1, double L2){
        /* parametros:  L1 (perdida cuando se realiza una intervencion que no es
            realmente necesaria)
                                        L2 (perdida cuando no se realiza una
                                            intervencion siendo esta necesaria) */


        double paso = (1 - umbral)*0.025;
```

```
// Calculo de la probabilidad de emitir ticket si el sistema no falla. P(
    Xi >= Umbral, Yi >= Umbral | Serv = 0)
double prob_1_0;


// Para intergrar utilizaremos una regla gaussiana bidimensional de 20x20
    nodos.
double densidad1 = 0;
for (int i = 0; i < 20; i++){
        for (int j = 0; j < 20; j++){
                densidad1 += this.funcion_densidad2(umbral + ((2*i+1) -
                    0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                    paso, 0);
                densidad1 += this.funcion_densidad2(umbral + ((2*i+1) -
                    0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                    paso, 0);
                densidad1 += this.funcion_densidad2(umbral + ((2*i+1) +
                    0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                    paso, 0);
                densidad1 += this.funcion_densidad2(umbral + ((2*i+1) +
                    0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                    paso, 0);
        }
}

densidad1 *= paso*paso;

if (densidad1 > 1){
        densidad1 = 1;
}

else if (densidad1 < 0){
        densidad1 = 0;
}

prob_1_0 = densidad1;


// Calculo de la probabilidad de no emitir ticket si el sistema falla. P(
    Xi < Umbral, Yi < Umbral | Serv = 1)
double prob_0_1;


// Para intergrar utilizaremos una regla gaussiana bidimensional de 20x20
    nodos.
double densidad2 = 0;
for (int i = 0; i < 20; i++){
        for (int j = 0; j < 20; j++){
                densidad2 += this.funcion_densidad2(umbral + ((2*i+1) -
                    0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                    paso, 1);
                densidad2 += this.funcion_densidad2(umbral + ((2*i+1) -
                    0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                    paso, 1);
                densidad2 += this.funcion_densidad2(umbral + ((2*i+1) +
                    0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                    paso, 1);
                densidad2 += this.funcion_densidad2(umbral + ((2*i+1) +
                    0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                    paso, 1);
```

```
                }
        }

        densidad2 *= paso*paso;

        if (densidad2 > 1){
                densidad2 = 1;
        }

        else if (densidad2 < 0){
                densidad2 = 0;
        }


        prob_0_1 = 1 - densidad2;


        // Formula para calcular el riesgo de Bayes.
        double riesgoBayes = L1*prob_1_0*prob_z0 + L2*prob_0_1*prob_z1;

        return riesgoBayes;
    }

/*    public double calcular_prob_z0(){ // probabilidad de que la intervencion no
     sea necesaria.

        double no_nec_serv = 0;

        for (int i = 0; i < ValoresY.size(); i++){
                if (Servicio.get(i) == 0){
                        no_nec_serv++;
                }
        }

        return no_nec_serv/ValoresY.size();
    }
*/

    public UmbralOptimo minima_perdida2Bayes (double L1, double L2){ // calcular
        el umbral que tenga la minima funcion de perdida.

        UmbralOptimo umbral_optimo = new UmbralOptimo();


        // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/6.
        double paso = 0.01;
        double umbral = 0;
        umbral_optimo.set_umbralBayes(0);

        umbral_optimo.riesgoBayes = this.funcion_perdida2Bayes(umbral, L1, L2);

        vectorUmbralBayes.add(umbral);
        vectorRiesgoBayes.add(umbral_optimo.riesgoBayes);

        for( int i = 1; i <= 100; i++){ // intervalo de [0,1] = [0, 0.01, 0.02,
            ..., 0.98, 0.99, 1]
                umbral = i*paso;

                vectorUmbralBayes.add(umbral);
                vectorRiesgoBayes.add(umbral_optimo.riesgoBayes);
```

```
            if (umbral_optimo.riesgoBayes > this.funcion_perdida2Bayes(umbral
                , L1, L2)){
                    umbral_optimo.umbralBayes = umbral;
                    umbral_optimo.riesgoBayes = this.funcion_perdida2Bayes(
                        umbral, L1, L2);
            }
    }

    //System.out.println("vectorUmbralBayes = " + vectorUmbralBayes);
    //System.out.println("vectorRiesgoBayes = " + vectorRiesgoBayes);
    return umbral_optimo;
}


public double funcion_perdida2Minimax (double umbral, double L1, double L2){
    /* parametros: L1 (perdida cuando se realiza una intervencion que no es
        realmente necesaria)
                                        L2 (perdida cuando no se realiza
                                            una intervencion siendo esta
                                            necesaria) */


    double paso = (1 - umbral)*0.025;


    // Calculo de la probabilidad de emitir ticket si el sistema no falla. P(
        Xi >= Umbral, Yi >= Umbral | Serv = 0)
    double prob_1_0;


    // Para intergrar utilizaremos una regla gaussiana bidimensional de 20x20
         nodos.
    double densidad1 = 0;
    for (int i = 0; i < 20; i++){
            for (int j = 0; j < 20; j++){
                    densidad1 += this.funcion_densidad2(umbral + ((2*i+1) -
                        0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                        paso, 0);
                    densidad1 += this.funcion_densidad2(umbral + ((2*i+1) -
                        0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                        paso, 0);
                    densidad1 += this.funcion_densidad2(umbral + ((2*i+1) +
                        0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                        paso, 0);
                    densidad1 += this.funcion_densidad2(umbral + ((2*i+1) +
                        0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                        paso, 0);
            }
    }

    densidad1 *= paso*paso;

    if (densidad1 > 1){
            densidad1 = 1;
    }

    else if (densidad1 < 0){
            densidad1 = 0;
    }
```

```
        prob_1_0 = densidad1;


        // Calculo de la probabilidad de no emitir ticket si el sistema falla. P(
            Xi < Umbral, Yi < Umbral | Serv = 1)
        double prob_0_1;


        // Para intergrar utilizaremos una regla gaussiana bidimensional de 20x20
             nodos.
        double densidad2 = 0;
        for (int i = 0; i < 20; i++){
                for (int j = 0; j < 20; j++){
                        densidad2 += this.funcion_densidad2(umbral + ((2*i+1) -
                            0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                            paso, 1);
                        densidad2 += this.funcion_densidad2(umbral + ((2*i+1) -
                            0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                            paso, 1);
                        densidad2 += this.funcion_densidad2(umbral + ((2*i+1) +
                            0.577350269)*paso, umbral + ((2*j+1) - 0.577350269)*
                            paso, 1);
                        densidad2 += this.funcion_densidad2(umbral + ((2*i+1) +
                            0.577350269)*paso, umbral + ((2*j+1) + 0.577350269)*
                            paso, 1);
                }
        }

        densidad2 *= paso*paso;

        if (densidad2 > 1){
                densidad2 = 1;
        }

        else if (densidad2 < 0){
                densidad2 = 0;
        }


        prob_0_1 = 1 - densidad2;


        // Formula para calcular el riesgo Minimax.
        double riesgoMinimax = Math.max(L1*prob_1_0, L2*prob_0_1);

        return riesgoMinimax;
    }

    public UmbralOptimo minima_perdida2Minimax (double L1, double L2){ //
        calcular el umbral que tenga la minima funcion de perdida.

        UmbralOptimo umbral_optimo = new UmbralOptimo();


        // el paso entre subintervalo y subintervalo es de 0.01 = (b-a)/6.
        double paso = 0.01;
        double umbral = 0;
        umbral_optimo.set_umbralMinimax(0);
```

```
        umbral_optimo.riesgoMinimax = this.funcion_perdida2Minimax(umbral, L1, L2
            );

        vectorRiesgoMinimax.add(umbral_optimo.riesgoMinimax);

        for( int i = 1; i <= 100; i++){ //
                umbral = i*paso;

                vectorRiesgoMinimax.add(umbral_optimo.riesgoMinimax);

                if (umbral_optimo.riesgoMinimax > this.funcion_perdida2Minimax(
                    umbral, L1, L2)){
                        umbral_optimo.umbralMinimax = umbral;
                        umbral_optimo.riesgoMinimax = this.
                            funcion_perdida2Minimax(umbral, L1, L2);
                }
        }

        return umbral_optimo;
    }

    public ArrayList<Double> devolver_riesgoBayes(){
        return vectorRiesgoBayes;
    }

    public ArrayList<Double> devolver_riesgoMinimax(){
        return vectorRiesgoMinimax;
    }

    public ArrayList<Double> devolver_umbral(){
        return vectorUmbralBayes;
    }

    public void graficar(){
        setTitle("Graficas");
                setSize(1000, 750);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setVisible(true);

                JPanel panel = new JPanel();
                getContentPane().add(panel);

                DefaultCategoryDataset linea1 = new DefaultCategoryDataset();
                DefaultCategoryDataset linea2 = new DefaultCategoryDataset();

                for (int i=0; i < devolver_riesgoBayes().size(); i++){
                        String umbral = Double.toString(i*0.01);
                        linea1.addValue(devolver_riesgoBayes().get(i), "riesgo de
                            Bayes", umbral);
                }

                for (int i=0; i < devolver_riesgoMinimax().size(); i++){
                        String umbral = Double.toString(i*0.01);
                        linea2.addValue(devolver_riesgoMinimax().get(i), "riesgo
                            Minimax", umbral);
                }

                JFreeChart chart1 = ChartFactory.createLineChart("Criterio de
                    Bayes", "Umbral", "riesgo de Bayes", linea1, PlotOrientation.
```

```
                    VERTICAL, true, true, false);
                JFreeChart chart2 = ChartFactory.createLineChart("Criterio
                    Minimax", "Umbral", "riesgo Minimax", linea2, PlotOrientation.
                    VERTICAL, true, true, false);

                ChartPanel chartPanel1 = new ChartPanel(chart1, 950, 350, 0, 0,
                    900, 300, false, false, false, true, true, false);
                panel.add(chartPanel1);

                ChartPanel chartPanel2 = new ChartPanel(chart2, 950, 350, 0, 0,
                    900, 300, false, false, false, true, true, false);
                panel.add(chartPanel2);

                setVisible(true);
    }

/*    public void graficarBayes(){
      setTitle("Grafica 1");
                setSize(1000, 700);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setVisible(true);

                JPanel panel = new JPanel();
                getContentPane().add(panel);

                DefaultCategoryDataset linea = new DefaultCategoryDataset();

                for (int i=0; i < devolver_riesgoBayes().size(); i++){
                        String umbral = Double.toString(i*0.01);
                        linea.addValue(devolver_riesgoBayes().get(i), "riesgo de
                            Bayes", umbral);
                }

                JFreeChart chart = ChartFactory.createLineChart("Criterio de
                    Bayes", "Umbral", "riesgo de Bayes", linea, PlotOrientation.
                    VERTICAL, true, true, false);

                ChartPanel chartPanel1 = new ChartPanel(chart, 950, 650, 0, 0,
                    900, 600, false, false, false, true, true, false);
                panel.add(chartPanel1);

                setVisible(true);
    }

    public void graficarMinimax(){
        setTitle("Grafica 2");
                setSize(1000, 700);
                setLocationRelativeTo(null);
                setDefaultCloseOperation(EXIT_ON_CLOSE);
                setVisible(true);

                JPanel panel = new JPanel();
                getContentPane().add(panel);

                DefaultCategoryDataset linea = new DefaultCategoryDataset();

                for (int i=0; i < devolver_riesgoMinimax().size(); i++){
                        String umbral = Double.toString(i*0.01);
```

```
                        linea.addValue(devolver_riesgoMinimax().get(i), "riesgo
                            Minimax", umbral);
                }

                JFreeChart chart = ChartFactory.createLineChart("Criterio Minimax
                    ", "Umbral", "riesgo Minimax", linea, PlotOrientation.VERTICAL
                    , true, true, false);

                ChartPanel chartPanel2 = new ChartPanel(chart, 950, 650, 0, 0,
                    900, 600, false, false, false, true, true, false);
                panel.add(chartPanel2);

                setVisible(true);
        }
*/
}
```

A.4. FicheroGere2ABBDD

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Properties;


public class FicheroGere2ABBDD {

        // Lista donde se almacenaran todos los valores del fichero. Y     [0,1].
        private static ArrayList<Double> Y = new ArrayList<Double>();

        // Conexion a Base de Datos e insertar los datos equipo-monitor-umbral
        public static void main(String[] args){

                System.out.println("Prueba de conexion");

                try{
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                }

                catch (ClassNotFoundException e){
                        System.out.println("El driver de JDBC no esta en el
                            Classpath");
                        e.printStackTrace();
                        return;
                }

                System.out.println("Driver registrado");
                Connection connection = null;

                try{
```

```
                String url = "jdbc:oracle:thin:@viver.tissat.es:1521:pr01
                    ";

                Properties props = new Properties();
                props.setProperty("user", "usCARLA");
                props.setProperty("password", "EsdRtK3d");
                props.setProperty("ssl", "true");
                props.setProperty("sslfactory", "org.postgresql.ssl.
                    NonValidatingFactory");

                connection = DriverManager.getConnection(url, props);
                Statement ps = connection.createStatement();

                String equipo = "gere2";

                File fichero = new File("C:/Users/chernando/workspace/
                    PRACTICAS/ConexionBBDD-Criterios/equipoGere2.txt");

                String servicio = "CPU SERVIDOR LINUX";

                ArrayList<Double> listaUmbrales = new ArrayList<Double>()
                    ;
                Y = new ArrayList<Double>();
                listaUmbrales = insertarUmbrales(fichero, equipo,
                    servicio);

                System.out.println(listaUmbrales);

                for (int j=0; j < listaUmbrales.size(); j++){
                        String query1= "INSERT INTO I2TM_PROD.
                            I2TM_UMBRALES_PRACTICA(equipo,monitor,value)
                            values ('"+ equipo +"', '"+ servicio +"', "+
                            listaUmbrales.get(j) +")";
                        ps.executeUpdate(query1);
                }

                String query2 = "commit";
                ps.execute(query2);

                ps.close();
                connection.close();

                System.out.println("Datos insertados correctamente");
        }

        catch (SQLException e){
                System.out.println("Error de conexion");
                e.printStackTrace();
                return;
        }
}

// metodo que lee de fichero e inserta los umbrales para el
    correspondiente equipo-monitor en la base de datos oracle
public static ArrayList<Double> insertarUmbrales(File fichero, String
    equipo, String servicio){

        File archivo = null;
        FileReader fr = null;
        BufferedReader br = null;
```

```
// Lectura del fichero. Creacion de BufferedReader para poder
    hacer una lectura comoda (metodo readLine()).
try {
        archivo = fichero;
        fr = new FileReader(archivo);
        br = new BufferedReader(fr);

        String linea;

        // Las dos primeras lineas de fichero no la estudiaremos.
             La saltamos por ser el fichero un HTML.
        br.readLine();br.readLine();

        // Recorrer el fichero linea a linea.
        while((linea=br.readLine())!=null){

                /* Dos maneras de anyadir. En estado critical y
                    en No critical.
                Ya que los valores se encuentran en posiciones
                    diferentes en el fichero. */

                // Seleccionar las filas donde hay valores (en
                    estado no CRITICAL) y anyadirlos a la lista Y
                    ya en tanto por uno.
                if (linea.length()>30 && linea.length()<65){
                        Y.add(0.01*Double.parseDouble(linea.
                            substring(52,54)));
                }

                // Seleccionar las filas donde hay valores (en
                    estado CRITICAL) y anyadirlos a la lista Y ya
                    en tanto por uno.
                else if(linea.length()>65 && linea.length()<110){
                        Y.add(0.01*Double.parseDouble(linea.
                            substring(54, 57)));
                }

                // OTRA FORMA DE LEER EL FICHERO PARA PASAR A
                    BASE DE DATOS.
                /*String[] palabras = linea.split(" ");
                        String umbral = null;
                        Double umbralNumero = -1.0;
                        for (int i=0; i < palabras.length; i++){
                                if (palabras[i].contains("L") &&
                                    palabras[i].length() < 20){
                                        umbral = palabras[i].
                                            substring(palabras[i].
                                            length() - 3, palabras
                                            [i].length());
                                        //System.out.println(
                                            umbral);
                                        umbralNumero = Double.
                                            parseDouble(umbral);
                                        Y.add(0.01*umbralNumero);
                                }
                                else if (palabras[i].contains("L
                                    ") && palabras[i].length() >
                                    20){
```

```
                                                        umbral = palabras[i].
                                                            substring(palabras[i].
                                                            length() - 6, palabras
                                                            [i].length() - 3);
                                                        //System.out.println(
                                                            umbral);
                                                        umbralNumero = Double.
                                                            parseDouble(umbral);
                                                        Y.add(0.01*umbralNumero);
                                                }
                                        }*/
                                }

                        }

                        catch(Exception e){
                                e.printStackTrace();
                        }

                        finally{

                                // Cerramos fichero, para asegurarnos que se cierra tanto
                                     si salta excepcion como si no.
                                try{
                                        if( null != fr ){
                                                fr.close();
                                        }
                                }

                                catch (Exception e){
                                        e.printStackTrace();
                                }
                        }
                        return Y;
                }
}
```

A.5. ConexionBBDD

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;

public class ConexionBBDD {

        private String url = "jdbc:oracle:thin:@viver.tissat.es:1521:pr01";
        private String user;
        private String pass;
        private Connection con;
        private String driver;

        public String getUrl() {
                return url;
        }
```

```
public void setUrl(String url) {
        this.url = url;
}

public String getUser() {
        return user;
}

public void setUser(String user) {
        this.user = user;
}

public String getPass() {
        return pass;
}

public void setPass(String pass) {
        this.pass = pass;
}

public Connection getCon() {
        return con;
}

public void setCon(Connection con) {
        this.con = con;
}

public String getDriver() {
        return driver;
}

public void setDriver(String driver) {
        this.driver = driver;
}

public ConexionBBDD () {

        try {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                con = DriverManager.getConnection(url, "usCARLA", "
                    EsdRtK3d");

        } catch (SQLException e) {
                e.printStackTrace();

        } catch (ClassNotFoundException e) {
                e.printStackTrace();

        }

}

public ConexionBBDD (String url,String user,String pass,String dver){

        try {
                this.setDriver(dver);
                this.setUrl(url);
                this.setPass(pass);
```

```
                        this.setUser(user);

                        Class.forName(dver);
                        this.con = DriverManager.getConnection(url, user, pass);

                } catch (SQLException e) {
                        e.printStackTrace();

                } catch (ClassNotFoundException e) {
                        e.printStackTrace();

                }
        }

        public ArrayList<Double> ObtenerUmbrales (String equipo, String servicio)
             throws SQLException{

                ArrayList <Double> Y = new ArrayList <Double>();
                try{

                        ResultSet res = con.createStatement().executeQuery("
                            SELECT VALUE FROM I2TM_PROD.I2TM_UMBRALES_PRACTICA
                            WHERE EQUIPO LIKE '"+equipo+"' AND MONITOR LIKE '"+
                             servicio+"' ");

                        while(res.next()){
                                Y.add(res.getDouble("VALUE"));
                        }

                } catch (SQLException e){
                        e.printStackTrace();
                }
                return Y;
        }

        public void closeConnection() {

                try{
                        this.con.close();

                }catch (SQLException e){
                        e.printStackTrace();
                }
        }
}
```

A.6. UmbralOptimo

```
public class UmbralOptimo {

        // Dos variables para cada criterio. Umbral y Riesgo

        double umbralBayes;
        double umbralMinimax;
        double riesgoBayes;
        double riesgoMinimax;
```

```
        public UmbralOptimo(){
        }

        public void set_umbralBayes(double otro_umbralBayes){
                this.umbralBayes = otro_umbralBayes;
        }

        public double get_umbralBayes(){
                return this.umbralBayes;
        }

        public void set_umbralMinimax(double otro_umbralMinimax){
                this.umbralMinimax = otro_umbralMinimax;
        }

        public double get_umbralMinimax(){
                return this.umbralMinimax;
        }

        public void set_riesgoBayes(double otro_riesgoBayes){
                this.riesgoBayes = otro_riesgoBayes;
        }

        public double get_riesgoBayes(){
                return this.riesgoBayes;
        }

        public void set_riesgoMinimax(double otro_riesgoMinimax){
                this.riesgoMinimax = otro_riesgoMinimax;
        }

        public double get_riesgoMinimax(){
                return this.riesgoMinimax;
        }

        public void mostrarBayes(){
                System.out.println("umbral optimo: " + get_umbralBayes() + ",
                    riesgo de Bayes optimo: " + get_riesgoBayes());
        }

        public void mostrarMinimax(){
                System.out.println("umbral optimo: " + get_umbralMinimax() + ",
                    riesgo Minimax optimo: " + get_riesgoMinimax());
        }
}
```