

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA



**Plataforma Web para la Validación de Experimentación en
Aprendizaje Automático y Minería de Datos**

TRABAJO DE FIN DE GRADO

Realizado por:
Adrián Canosa Mouzo

Dirigido por:
Ismael Rodríguez Fernández
Alberto J. Bugarín Díz
Manuel Mucientes Molina



D. Alberto J. Bugarín Diz, D. Ismael Rodríguez Fernández, y D. Manuel Mucientes Molina, Profesores e investigadores del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela,

INFORMAN:

Que la presente memoria, titulada *Plataforma Web para la Validación de Experimentación en Aprendizaje Automático y Minería de Datos*, presentada por **D. Adrián Canosa Mouzo** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación de Grado en Ingeniería Informática, se realizó bajo nuestra dirección en el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

Y para que así conste a los efectos oportunos, se expide el presente informe en Santiago de Compostela, a 8 de septiembre de 2014:

Los directores,

El alumno,

Ismael Rodríguez Fernández,
Alberto J. Bugarín Diz,
Manuel Mucientes Molina

Adrián Canosa Mouzo

Agradecimientos

A mis directores Ismael, Alberto y Manuel, por su ayuda y tiempo dedicado estos meses.

A mi familia por el apoyo recibido estos años.

El presente Trabajo Fin de Grado se ha desarrollado en el marco del proyecto de I+D+i “QTEMP: Descripción lingüística de fenómenos complejos: cuantificadores borrosos generalizados en proposiciones temporales (TIN2011-29827-C02-02)”, coordinado entre el Grupo de Sistemas Inteligentes de la USC y el European Centre for Soft Computing, y financiado por el Ministerio de Economía y Competitividad en el período 2012-2014.

Índice general

1. Introducción	1
1.1. Objetivos del proyecto	2
1.2. Organización del documento	3
2. Contraste de hipótesis	5
2.1. Hipótesis nula y alternativa	6
2.2. Estadístico de contraste	7
2.3. Decisiones y tipos de error	9
2.4. Intervalos de confianza	10
2.5. Decisión final y <i>p-valor</i>	11
2.6. Etapas en la resolución de un contraste de hipótesis	13
2.7. Tests paramétricos	13
2.7.1. Condiciones Paramétricas	14
2.7.2. Test ANOVA	15
2.7.3. t-test	17
2.8. Tests no paramétricos	17
2.8.1. Test de Wilcoxon	18
2.8.2. Test de Friedman	19
2.8.3. Test de Iman-Davenport	20
2.8.4. Test de los Rangos Alineados de Friedman	20
2.8.5. Test de Quade	21
2.8.6. Tests POST-HOC	22
3. Análisis de requisitos	25
3.1. Epics	26
3.2. Historias de usuario	27
3.2.1. Historias de usuario desarrollador	27
3.2.2. Historias de usuario cliente	29
4. Gestión del proyecto	41
4.1. Análisis de riesgos	41
4.2. Metodología de desarrollo	43
4.2.1. Metodologías ágiles	44
4.2.2. Scrum	45
4.3. Gestión de la configuración	48
4.4. Planificación temporal	49
4.5. Estimación de costes	54

5. Arquitectura y herramientas	55
5.1. Arquitectura del sistema	55
5.2. Herramientas de diseño	57
5.2.1. Herramientas	57
5.2.2. Patrones de diseño	58
5.3. Herramientas de desarrollo	59
5.3.1. Análisis de librerías de test estadísticos en Python	59
5.3.2. Análisis de librerías para servicios REST en Python	60
5.3.3. Análisis de frameworks para desarrollo web	62
5.3.4. Otras herramientas	63
6. Diseño e implementación	67
6.1. Módulo Python test estadísticos	67
6.1.1. Test paramétricos	68
6.1.2. Test no paramétricos	69
6.2. API Servicios REST	72
6.3. Aplicación web	74
6.3.1. Prototipo de la interfaz gráfica	74
6.3.2. Heurísticas de Nielsen	76
6.3.3. Diagramas de secuencia	79
7. Validación y pruebas	85
7.1. Pruebas unitarias	86
7.2. Validación de requisitos	88
8. Valoraciones finales	91
8.1. Posibles mejoras	92
8.1.1. Test estadísticos para datos no apareados	92
8.1.2. Datos de entrada	92
8.1.3. Otras mejoras	92
A. Manual técnico	93
A.1. Despliegue de la aplicación en Apache	93
B. Manual de usuario	95
B.1. Página de inicio	95
B.2. Ayuda y consulta de datos	96
B.3. Selección de parámetros / opciones de test	98
B.4. Visualización de resultados	100

Índice de figuras

2.1. Distribuciones de probabilidad.	8
2.2. Decisiones y tipos de error.	9
2.3. Regiones de aceptación y rechazo.	11
2.4. Punto crítico.	11
2.5. Comparativa entre FDP y FDA.	12
2.6. Normalidad vs. No normalidad.	14
2.7. Homocedasticidad vs. Heterocedasticidad.	15
4.1. Ciclos de desarrollo.	45
4.2. Ejemplo gráfico Burn-Down.	47
4.3. Estructura de descomposición del trabajo (EDT).	49
4.4. Cronograma lineal de la planificación de las fases.	52
4.5. Burn-Down Sprint 1, 2, 3 y 4.	52
4.6. Burn-Down Sprint 5, 6, 7 y 8.	53
4.7. Burn-Down Sprint 9.	53
5.1. Arquitectura de la plataforma web.	55
5.2. Patrón MVC implementado.	56
5.3. Patrón <i>Singleton</i>	58
5.4. Servicios web REST.	61
5.5. Organización Twitter Bootstrap.	63
6.1. Prototipo de página principal.	75
6.2. Prototipo página de ayuda.	75
6.3. Prototipo página selección de parámetros/opciones.	76
6.4. Prototipo página visualización de resultados.	76
6.5. Error sin fichero.	78
6.6. Vista test ANOVA.	79
6.7. Diagrama de secuencia de la subida de ficheros.	80
6.8. Diagrama de secuencia de la consulta de ficheros.	81
6.9. Diagrama de secuencia del test de ANOVA.	82
6.10. Diagrama de secuencia de los test no paramétricos de ranking.	83
6.11. Diagrama de secuencia de los test de SciPy.	84
A.1. Directorio del proyecto STAC.	93
B.1. Página de inicio de la plataforma.	95
B.2. Ventana emergente de subida de ficheros.	96
B.3. Mensaje de error.	96

B.4. Listado secciones ayuda.	97
B.5. Sección del formato del fichero de datos.	97
B.6. Pantalla de visualización del fichero de datos.	98
B.7. Descripción de las condiciones paramétricas.	98
B.8. Selección de opciones.	99
B.9. Error cuando no existe ningún fichero.	99
B.10. Error cuando el número de algoritmos es mayor a 2.	100
B.11. Página de visualización de resultados.	100
B.12. Ventana de diálogo exportación resultados.	101
B.13. Formato datos \LaTeX	101
B.14. Formato datos CSV.	101

Capítulo 1

Introducción

En el contexto tecnológico actual, en donde el Big Data es un recurso cada vez más utilizado, el rol del analista de datos (data scientist) se está convirtiendo en una profesión emergente y de elevada demanda. Un analista de datos es aquel profesional que reúne, analiza e interpreta los datos obtenidos con el objetivo de sacar ciertas conclusiones de ellos y así tomar diferentes decisiones, con las que aumentar la productividad en una organización. Relacionado con el analista de datos, un nuevo rol emergente en muchas empresas es el de CDO (“*Chief Data Officer*”). Este rol es el responsable de la gestión y la utilización de la información como un activo para toda la empresa. El analista de datos combina diferentes habilidades, especialmente las técnicas de la minería de datos y del aprendizaje automático (DM&ML).

Según Mitchell [1], una definición de aprendizaje automático sería la siguiente: un programa de ordenador aprende a partir de una experiencia E a realizar una tarea T (de acuerdo con una medida de rendimiento P), si su rendimiento al realizar T , medido con P , mejora gracias a la experiencia E . La minería de datos, por otra parte, es un campo de las ciencias de la computación referido al proceso que trata de descubrir patrones en grandes volúmenes de conjuntos de datos [2]. Para ello utiliza, entre otros métodos, técnicas estadísticas para deducir estos patrones y tendencias que existen en los datos. Por lo general, estos patrones no pueden ser detectados mediante exploración tradicional debido a la complejidad o la gran cantidad de datos.

Una de las tareas más importantes que se deben llevar a cabo en el aprendizaje automático es la validación de resultados obtenidos por los algoritmos de aprendizaje. El método estándar más aceptado en la actualidad es el de la aplicación de test estadísticos sobre los experimentos, que, entre otras utilidades, apoyan la toma de decisiones, como por ejemplo la elección del algoritmo más adecuado.

En este proyecto hemos creado y desarrollado una plataforma para asistir al analista de datos en el proceso de validación de resultados. Para ello, se extendió una librería de test estadísticos, se crearon servicios web para facilitar su consulta y se desarrolló una interfaz web que hace uso de estos servicios. El objetivo es que el analista pueda introducir en la web los datos obtenidos mediante experimentación y seleccionar el test estadístico que desee utilizar para que, de forma automática, la plataforma muestre los resultados de la aplicación del test. Así, la plataforma permitirá de un modo fácil y centralizado la validación de resultados mediante el uso de test estadísticos.

La herramienta se incorporará en la lista de aplicaciones disponibles a través de la web del CiTIUS para su acceso. El impacto y difusión del resultado del proyecto tiene el potencial de ser amplio, ya que en la actualidad no existe ninguna herramienta que centralice la aplicación de los test estadísticos de mayor utilidad para la validación de algoritmos de aprendizaje automático y que, además, resulte fácil de usar.

1.1. Objetivos del proyecto

El proyecto se centra en crear y desarrollar una plataforma web para asistir al analista de datos en el proceso de validación de los resultados obtenidos de diferentes algoritmos de aprendizaje. Para ello, habrá que realizar las siguientes tareas:

1. Completar y extender una librería de test estadísticos, actualmente implementada en el lenguaje Python.

La librería estará formada por test paramétricos, test para evaluar las condiciones de aplicación de los test paramétricos (para la normalidad y homocedasticidad), así como test no paramétricos. Estos test se verán en detalle a lo largo del capítulo 2 en las secciones 2.7, 2.7.1 y 2.8 respectivamente. La librería a extender se denomina “nonparametric.py”, y los test de normalidad y homocedasticidad, así como la prueba \mathcal{T} de Student se tomarán de la librería de estadística de Python SciPy (scipy.stats). El listado de test para el proyecto es el siguientes:

- Normalidad: Shapiro-Wilk, D’Agostino–Pearson y Kolmogorov–Smirnov.
- Homocedasticidad: Levene.
- Paramétricos: t-test, ANOVA, Bonferroni.
- No paramétricos: Wilcoxon, Friedman, Iman-Davenport, Rangos Alineados de Friedman, Quade, Bonferroni-Dunn, Holm, Finner, Hochberg, Li, Shaffer.

2. Crear los servicios web en Python, basados en REST, que hagan disponible el acceso a los test estadísticos vía web.

Los servicios REST están basados en los métodos HTTP (POST y GET para este proyecto). Asimismo, las peticiones a los servicios web de los test incluirán todos los datos necesarios (petición completa e independiente) para que el servidor no tenga que mantener ningún estado para procesar la petición. Los datos a transmitir (datos del analista, resultados obtenidos por los test) con REST se podrán transferir mediante XML, JavaScript Object Notation (JSON), o ambos. Cada servicio dispondrá de varias URIs distintas en función de los parámetros para dar mayor versatilidad a la API.

3. Desarrollar una interfaz web (HTML + JavaScript) para facilitar el uso de los test sobre los datos introducidos por el analista de datos.

Las tecnologías que se emplearán para desarrollarla serán: HTML, JavaScript y CSS. Un requisito para el proyecto es que el analista pueda aplicar los test de la forma más sencilla posible, por lo que se tendrá en cuenta este aspecto en el desarrollo.

1.2. Organización del documento

La finalidad de este documento es la de presentar los desarrollos realizados para resolver correctamente los objetivos definidos, explicando para ello cada una de las partes que componen la plataforma web y las tareas realizadas a lo largo del proceso.

- En el *capítulo 2* se explican los conceptos básicos manejados en el proyecto. Para ello, se realiza un análisis detallado del contraste de hipótesis, tratando los conceptos básicos con ejemplos. Además, se explica la finalidad y funcionamiento de cada uno de los test disponibles en la plataforma.
- El *capítulo 3* hace un análisis de los requisitos identificados en el proyecto, utilizando para ello las historias de usuario empleadas en la metodología del proyecto.
- El *capítulo 4* describe la gestión del proyecto. Incluye el análisis de riesgos, la metodología de desarrollo, la gestión de la configuración, la planificación temporal y el análisis de costes.
- En el *capítulo 5* se propone una arquitectura a alto nivel del sistema y se definen cada una de las partes de las que constará, así como las herramientas de diseño y desarrollo utilizadas.
- El *capítulo 6* explica tanto el diseño como la implementación a bajo nivel de la arquitectura propuesta.
- El *capítulo 7* se definen las pruebas establecidas para poder comprobar que la herramienta es válida y que los objetivos se han cumplido. Se detallan además los resultados de las mismas.
- Por último, en el *capítulo 8* se establecen aquellas conclusiones derivadas de la realización del proyecto, además de una breve indicación de lo que podría ser mejorable o ampliable en un futuro.

Capítulo 2

Contraste de hipótesis

El contraste de hipótesis, también conocido como test estadísticos, se engloba en el ámbito de la Inferencia Estadística, que es la parte de la estadística que estudia cómo sacar conclusiones generales (sujetas a un determinado grado de fiabilidad o significancia) para toda la población a partir del estudio de una muestra. En nuestro caso, se tratará de sacar conclusiones de los resultados obtenidos por diferentes algoritmos sobre distintos conjuntos de datos para determinar, por ejemplo, si los algoritmos tienen un rendimiento significativamente diferente y por lo tanto no se pueden considerar iguales.

El **contraste de hipótesis** es uno de los problemas más comunes dentro de la inferencia estadística. En él se contrasta una hipótesis estadística. Por ejemplo:

Un ingeniero de software afirma que la media de los resultados obtenidos por un algoritmo de aprendizaje automático es 10. ¿Se podría desmentir la afirmación del ingeniero?

El planteamiento del contraste sería el siguiente (μ indica media poblacional):

$$\mu = 10$$

$$\mu \neq 10$$

Para tomar una decisión (desmentir o no la afirmación), hay que basarse en los datos de una muestra, para comprobar si en efecto la media de los resultados es 10 (media muestral). Para ello, se podría establecer una regla de decisión sobre la cual se basaría nuestra decisión final. Por ejemplo: si la media obtenida está próxima a la indicada por el ingeniero (10), entonces se podría afirmar que dice la verdad. Si por el contrario la muestra nos proporciona una media muy distinta a 10, entonces se puede concluir que la evidencia desmiente la afirmación del ingeniero sobre el algoritmo en cuestión. Esto plantea cuándo se puede considerar que la media es lo suficientemente distinta como para determinar que la afirmación del ingeniero es errónea. Por ejemplo si la media de la muestra es 8.5, ¿se podría desmentir la afirmación inicial? El contraste de hipótesis nos proporciona una forma de establecer este criterio y poder rechazar o aceptar la afirmación inicial.

2.1. Hipótesis nula y alternativa

En todo contraste de hipótesis siempre se dan dos posibilidades o hipótesis, las cuales se representan con los siguientes símbolos:

H_0 : hipótesis nula

H_1 : hipótesis alternativa

- H_0 : es la hipótesis que se supone cierta de partida, es decir, es la hipótesis que establece que lo que indica la muestra es solamente debido a la variación aleatoria entre la muestra y la población.
- H_1 : es la hipótesis alternativa y es la que reemplazará a la hipótesis nula si ésta es rechazada. H_1 establece que lo que indica la muestra es verdadero, y representa a toda la población.

A modo de ejemplo, supongamos que unos programadores están trabajando en la optimización de un algoritmo de aprendizaje. El objetivo es mejorar el algoritmo de forma que los resultados que proporcione sean menores de 100. Se toma una muestra de los resultados obtenidos por el nuevo algoritmo optimizado y se observa que la media de la muestra es de 92. Si no hubiera incertidumbre en la media muestral, entonces se podría concluir que la modificación reduciría los resultados a 92. Sin embargo, siempre existe incertidumbre en la media muestral. La media poblacional en realidad será poco mayor o menor a 92.

Los programadores están preocupados de que el nuevo algoritmo en realidad no mejore al anterior, es decir, que la media poblacional pudiera ser mayor o igual a 100. Quieren saber si esta preocupación está justificada. Se ha observado una muestra con media de 92 y existen dos posibles interpretaciones, o como se ha mencionado más arriba, dos tipos de hipótesis que serán contrastadas más adelante mediante un determinado test estadístico:

1. La media poblacional es mayor o igual a 100 (la media muestral es, por tanto, menor debido sólo a la variación aleatoria de la media poblacional). El nuevo algoritmo no mejorará al anterior.
2. La media poblacional es menor que 100, y la media muestral lo refleja. El nuevo algoritmo sí mejorará al anterior.

La primera interpretación sería la hipótesis nula o H_0 . La segunda, la hipótesis alternativa o H_1 , como se comentó más arriba.

En este caso, los programadores están preocupados de que la hipótesis nula sea cierta. Un test estadístico o prueba de hipótesis hallará una medida cuantitativa de la factibilidad de la hipótesis nula (denominado estadístico de contraste, que para este ejemplo viene dado por la media obtenida en la muestra) y se podrá decir a los programadores (después de que el test tome la decisión) si su preocupación está o no justificada. Por tanto, a modo de resumen este ejemplo nos proporciona dos hipótesis:

$$H_0 : \mu \geq 100 \text{ vs. } H_1 : \mu < 100$$

La realización de un contraste de hipótesis no consiste en decidir cuál de las dos hipótesis (H_0 , H_1) es más creíble, sino en decidir si la muestra proporciona o no suficiente evidencia para descartar H_0 . Para realizar la prueba de hipótesis o test estadístico se pone la hipótesis nula en juicio, es decir se empieza suponiendo que H_0 es verdadera. Se podría poner como analogía el supuesto de “*En un juicio, el acusado siempre es inocente hasta que se demuestre lo contrario.*” Esto es:

H_0 : el acusado es inocente

H_1 : el acusado es culpable

y, mientras no se tenga suficiente evidencia para aceptar H_1 , hay que creer que lo que dice H_0 es cierto. La muestra aleatoria proporcionará la evidencia. Si el juicio (test o prueba de hipótesis) determina que el acusado es inocente, sólo se puede decir que no se tiene suficiente evidencia para asegurar que el acusado es culpable, mientras que si aceptamos la hipótesis alternativa, se estará bastante seguro de que el acusado sí es culpable.

2.2. Estadístico de contraste

Los test estadísticos o pruebas de hipótesis, calculan internamente una medida cuantitativa que proporciona la factibilidad de la hipótesis nula. Esta medición se extrae de la muestra proporcionada. Por ejemplo, si queremos contrastar la hipótesis de que la media poblacional es 5, un estadístico a calcular puede ser la media de una muestra. En este caso, la muestra viene determinada por los resultados obtenidos por los algoritmos y cada uno de los test tiene una forma particular de hallar este estadístico mediante una fórmula que lo caracteriza. Estos estadísticos siguen una determinada distribución de probabilidad. Por ejemplo, en este proyecto los test implementados harán uso de estadísticos que siguen distribuciones como:

- Distribución normal \mathcal{N} (p. ej. test de Wilcoxon).
- Distribución chi-cuadrado χ^2 (p. ej. test de Friedman).
- Distribución \mathcal{F} de Fisher-Snedecor (p. ej. test de Iman-Davenport).
- Distribución \mathcal{T} de Student (p. ej. t-test).

La figura 2.1, nos muestra el aspecto que presentan las distintas distribuciones de probabilidad. Las distribuciones dependen de ciertos parámetros para determinar su forma (μ , σ^2 ...): Como podemos ver en la figura 2.1, la distribución normal presenta μ y σ^2 como parámetros. Éstos indican media y varianza respectivamente. La varianza, es una medida de dispersión que indica cómo se distribuye la población. Por ejemplo: en una distribución normal de media 0 y varianza 1 (línea roja en la figura 2.1(a)), aproximadamente el 68 % de la población se encuentra en el intervalo $[-1, 1]$, ya que el área bajo la curva es de 0.68. Por tanto, la probabilidad de que un individuo de la población se encuentre en ese intervalo es del 68 %. Si un estadístico sigue una distribución normal con media μ y varianza σ^2 , se expresa como:

$$\text{Estadístico} \sim N(\mu, \sigma^2)$$

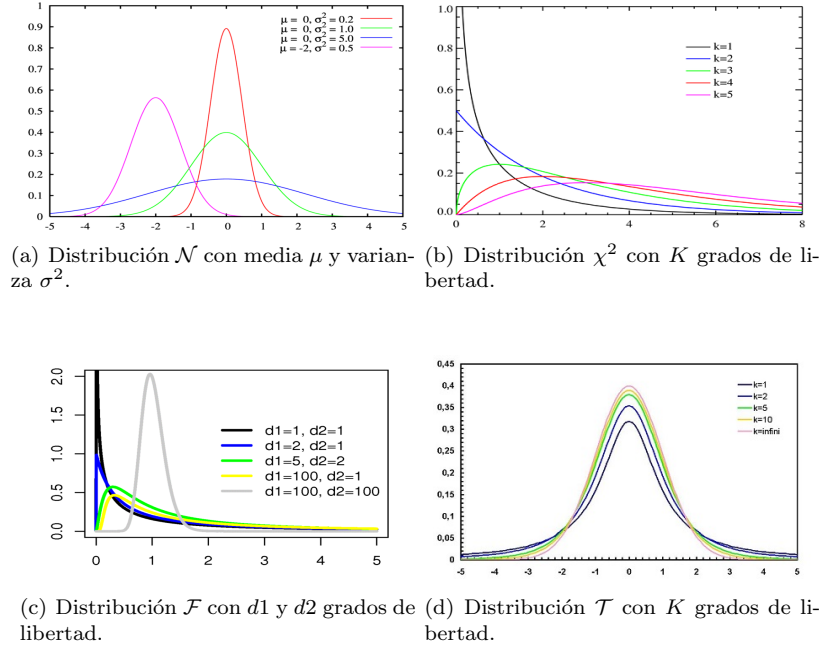


Figura 2.1: Distribuciones de probabilidad.

En las distribuciones χ^2 y \mathcal{T} de Student se habla del parámetro K o grados de libertad ($d1$ y $d2$ en la distribución \mathcal{F} de Fisher-Snedecor). La media y la varianza de estas tres distribuciones vendrán determinadas por el parámetro K . Cuando se habla de grados de libertad se está haciendo referencia al número de valores que se pueden elegir libremente en una muestra. Por ejemplo: una muestra con dos datos y media 5 si el primer dato toma el valor 4 entonces necesariamente el segundo dato debe de ser 6 (para lograr la media de 5). En este caso, se tienen:

$$N - 1 \text{ grados de libertad, donde } N \text{ es el tamaño de la muestra.}$$

Se hallan con la fórmula $N - R$, donde N es el número de individuos en la muestra cuyo valor puede ser elegido de forma libre y R es el número de sujetos cuyo valor dependerá del valor que tengan los individuos de la muestra que son libres. También se puede representar por $K - R$, donde K es el número de grupos (cuando intervienen grupos y no sujetos individuales).

En nuestro caso, N viene determinado por el número de resultados obtenidos por los algoritmos (número de filas de la matriz de la muestra de datos) y K por el número de algoritmos o variables relacionadas que tiene la muestra de datos con la que se están aplicando los test (número de columnas de la matriz). Cada test que use el parámetro de grados de libertad lo calcula de acuerdo a su fórmula característica para el estadístico.

Todas las distribuciones de la figura 2.1 son continuas, pues se puede tomar cualquier valor dentro de un intervalo, a diferencia de las distribuciones discretas. Por otra parte, en la distribución \mathcal{T} de Student a medida que aumentan los grados de libertad se tiende más a una distribución normal estandarizada (de $\mu = 0$ y $\sigma^2 = 1$).

Las distribuciones de probabilidad que pueda seguir un estadístico nos dan un valor diferente de probabilidad para cada valor diferente del estadístico. Este valor de probabilidad indica cuán probable es obtener ese valor del estadístico siendo la hipótesis nula cierta. Por ejemplo, si es cierta la hipótesis nula de que la media de una población es 5, es más probable que obtengamos una media de una muestra igual a 4.5 que a 3.

2.3. Decisiones y tipos de error

Cuando se lleva a cabo un contraste de hipótesis sólo se pueden tomar dos decisiones. Los datos de la muestra, que en este proyecto vendrá dada por los resultados obtenidos por los algoritmos, evidenciarán qué decisión se debe tomar:

1. Aceptar la hipótesis nula (H_0) (rechazar la hipótesis alternativa H_1)
2. Rechazar H_0 (aceptar la hipótesis alternativa)

Sin embargo, cuando se toma la decisión se pueden cometer dos tipos de errores (fig. 2.2):

		Decisión	
		No se rechaza H_0	Se rechaza H_0
Realidad	H_0 es verdadera	Decisión correcta	Error de tipo I
	H_0 es falsa	Error de tipo II	Decisión correcta

Figura 2.2: Decisiones y tipos de error.

La probabilidad de “Error tipo I” se denota por α y se denomina nivel de significación:

$$P(\text{“Error tipo I”}) = P(\text{Rechazar } H_0 | H_0 \text{ es cierta}) = \alpha$$

El nivel de significación consiste en la probabilidad de rechazar la hipótesis nula H_0 cuando verdaderamente es cierta. Este valor α es un parámetro que debe seleccionar la persona que quiere realizar un test estadístico en base a cuán importante es rechazar H_0 cuando es cierta. Normalmente es del 5 %, lo que implicará que 5 de cada 100 veces se acepta la hipótesis alternativa cuando la cierta es la hipótesis nula. Cuanto menor sea el nivel de significación, cada vez es más difícil rechazar la hipótesis nula. Es decir, si queremos equivocarnos menos veces, necesitamos mucha más evidencia para justificar el rechazo. Si es grande es más fácil aceptar la hipótesis alternativa cuando en realidad es falsa.

Por otra parte, la probabilidad de “Error tipo II” se denota por β :

$$P(\text{“Error tipo II”}) = P(\text{Aceptar } H_0 | H_0 \text{ es falsa}) = \beta$$

Este error β consiste en la probabilidad de aceptar la hipótesis nula H_0 cuando verdaderamente es falsa.

Por último, cabe destacar el concepto de “Potencia”.

$$P(\text{"Potencia"}) = P(\text{Rechazar } H_0 | H_0 \text{ es falsa}) = 1 - \beta.$$

La potencia es la probabilidad de detectar que una hipótesis es falsa. Los test estadísticos o pruebas de hipótesis implementados en el presente proyecto se caracterizan por su potencia, siendo esta fija, y dejando como parámetro libre el nivel de significación. Así, cuanto mayor es el nivel de potencia, mejor será el test, ya que se rechazarán más hipótesis nulas cuando se deben rechazar (mayor habilidad en aceptar correctamente hipótesis alternativas).

En este proyecto se pondrá el énfasis en el nivel de significación, ya que es la hipótesis alternativa la que se quiere probar y no se quiere aceptar si en realidad no es cierta es decir, si aceptamos la hipótesis alternativa queremos equivocarnos con un margen de error muy pequeño. Obviamente, lo ideal sería que tanto α como β fuesen nulos y que no se cometiese ningún error, o que ambos valores fuesen muy pequeños. Como no se pueden disminuir ambos errores a la vez, se controla el "Error tipo I".

2.4. Intervalos de confianza

El nivel de significación fijado divide en dos regiones el conjunto de posibles valores del estadístico de contraste: la región de aceptación y la región de rechazo o región crítica. Se denomina región de aceptación a la región que conduce a la aceptación de H_0 y región de rechazo a la región que conduce al rechazo de H_0 en favor de H_1 . Aquí surge el concepto de **Cola**, que indica la porción o porciones de una distribución de probabilidad en la cual se rechaza la hipótesis nula, es decir, la **región de rechazo**.

La determinación de las regiones de aceptación o de rechazo depende de cómo se establezca la hipótesis alternativa H_1 . Por ejemplo, si hablamos de un contraste en el que se esté contrastando una determinada media (μ_0) se podría establecer como H_1 que la media en realidad sea menor, mayor o distinta a (μ_0):

- Media menor (test unilateral con cola a la izquierda):

$$\begin{aligned} H_0 : \mu &= \mu_0 \\ H_1 : \mu &< \mu_0 \end{aligned}$$

- Media mayor (test unilateral con cola a la derecha):

$$\begin{aligned} H_0 : \mu &= \mu_0 \\ H_1 : \mu &> \mu_0 \end{aligned}$$

- Media distinta (test bilateral o de dos colas):

$$\begin{aligned} H_0 : \mu &= \mu_0 \\ H_1 : \mu &\neq \mu_0 \end{aligned}$$

En la figura 2.3, podemos ver cómo quedarían establecidos los intervalos para el ejemplo. En rojo se muestra la región de rechazo:

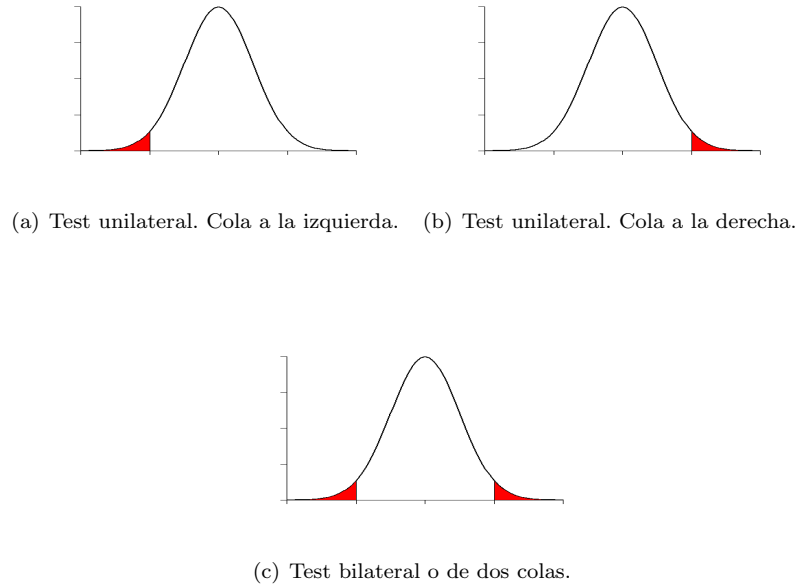


Figura 2.3: Regiones de aceptación y rechazo.

Como se puede observar en el caso del test bilateral o de dos colas, el α se divide en dos porciones iguales: $\alpha/2$, que constituyen la región de rechazo. La región de aceptación tendrá en todos los casos probabilidad $1 - \alpha$.

2.5. Decisión final y *p-valor*

Si el valor del estadístico cae en la región de aceptación, se acepta la hipótesis nula, ya que no existen razones suficientes para rechazar H_0 con el nivel de significación dado. Por tanto, en este caso se diría que el contraste es estadísticamente no significativo, es decir, no existe evidencia estadísticamente significativa en favor de H_1 . La figura 2.4 nos muestra el punto crítico: si el

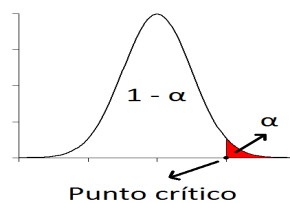


Figura 2.4: Punto crítico.

estadístico obtenido por el test o prueba de hipótesis es 5 y el punto crítico es 4.5, se rechaza H_0 ya que el estadístico pertenece a la región de rechazo.

La decisión de rechazar o aceptar la hipótesis nula, se puede determinar también mediante el *p-valor*, que es el parámetro utilizado para realizar los test estadísticos en este proyecto. El *p-valor* proporciona una forma más eficiente de determinar si el contraste es o no estadísticamente significativo, ya que no sería necesario recalcular regiones de aceptación y rechazo cada vez que el usuario de los test cambia de nivel de significación.

“El *p-valor*, es la probabilidad que hay de obtener un valor al menos tan extremo como el estadístico en cuestión que se ha calculado.”

Para entender mejor el concepto de *p-valor*, conviene hablar de las distribuciones de probabilidad vistas en la figura 2.1 de la sección 2.2 donde se hablaba del estadístico de contraste. Estas distribuciones son funciones que se denominan “funciones de densidad de probabilidad” (FDP). Como expusimos en la sección 2.2, estas funciones proporcionan la probabilidad que existe para cada valor diferente del estadístico (cuán probable es obtener ese valor del estadístico). Si, en vez de trabajar con las funciones de densidad de probabilidad, se trabaja con las funciones de distribución acumuladas (FDA), para cada valor del estadístico éstas devolverían la probabilidad de obtener un valor igual o menor que ese estadístico siendo la hipótesis nula cierta. En la figura 2.5 podemos ver una comparación entre los dos tipos de funciones para el caso de la distribución χ^2 :

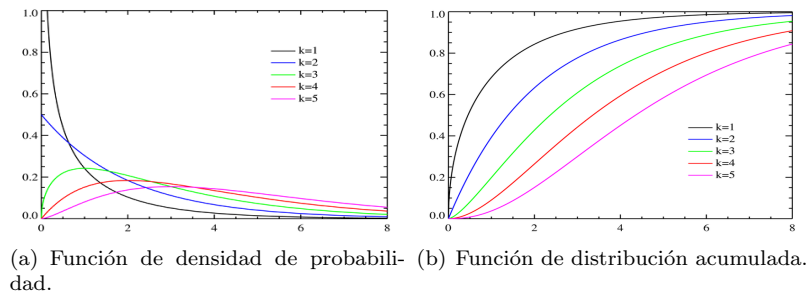


Figura 2.5: Comparativa entre FDP y FDA.

Visto de otro modo, con la distribución acumulada, dado un estadístico, nos devuelve la probabilidad que hay de obtener un valor al menos tan extremo como el estadístico en cuestión. Esto es el *p-valor*. Por ejemplo si el valor del estadístico es 3 y el test da como resultado un *p-valor* igual a 0.1, esto quiere decir que un 10 % de las veces vamos a obtener un valor similar. Si el *p-valor* es muy bajo, es decir, la probabilidad de obtener un valor al menos tan extremo como ese estadístico es muy baja, se puede concluir que la hipótesis nula no es cierta, ya que sería poco probable que siendo cierta se obtuviese ese estadístico.

El criterio para saber si el *p-valor* es lo suficientemente bajo como para rechazar que la hipótesis nula sea cierta es tomado de acuerdo al nivel de significancia establecido. Como se ha mencionado en la sección 2.3, el nivel de significancia o α indica la probabilidad de rechazar la hipótesis nula siendo ésta cierta. Si el *p-valor* es menor que el nivel de significancia se rechaza la hipótesis nula.

Según se disminuye α , cada vez es más difícil de rechazar la hipótesis nula, ya que se necesita mucha más evidencia para justificar el rechazo. Por ejemplo, si α es 0.05 y la probabilidad de obtener un estadístico igual a 1 es 0.03, se rechaza la hipótesis nula. Sin embargo, si el α fuese 0.01, no se podría rechazar: no hay suficiente evidencia de que la hipótesis nula sea falsa.

2.6. Etapas en la resolución de un contraste de hipótesis

En un contraste de hipótesis siempre se siguen una serie de pasos definidos. Como se ha ido viendo a lo largo del capítulo, los pasos para la realización de una prueba de hipótesis o test estadístico son los siguientes [4]:

1. Especificación de la hipótesis nula H_0 y de la hipótesis alternativa H_1 .
2. Suponer que H_0 es verdadera (el test sirve para que a partir de la muestra de datos podamos rechazar H_0 en beneficio de H_1).
3. Calcular un estadístico de prueba o estadístico de contraste. Este estadístico se usa para evaluar la fuerza de la evidencia en contra de H_0 (medir la discrepancia entre la hipótesis y la muestra).
4. Establecer un nivel de significación α en base a cómo de importante se considere rechazar H_0 cuando realmente es verdadera.
5. El nivel de significación fijado divide en dos regiones el conjunto de posibles valores del estadístico de contraste: la región de aceptación y la región de rechazo o región crítica.
6. Si el valor del estadístico cae en la región de rechazo, se rechaza la hipótesis nula, ya que esto evidencia que los datos obtenidos de la muestra no son compatibles con H_0 . Por tanto, en este caso se diría que el contraste es estadísticamente significativo, es decir, existe evidencia estadísticamente significativa en favor de H_1 .
7. Si el valor del estadístico cae en la región de aceptación, se acepta la hipótesis nula, ya que no existen razones suficientes para rechazar H_0 con el nivel de significación dado. Por tanto, en este caso se diría que el contraste es estadísticamente no significativo, es decir, no existe evidencia estadísticamente significativa en favor de H_1 .
8. La decisión de rechazar o aceptar la hipótesis nula, se puede determinar también mediante el *p-valor*, que es el parámetro utilizado para realizar los test estadísticos en este proyecto. El *p-valor* proporciona una forma más eficiente de determinar si el contraste es o no estadísticamente significativo, ya que no sería necesario recalcular regiones de aceptación y rechazo cada vez que el usuario de los test cambia de nivel de significación.

2.7. Tests paramétricos

Uno de los tipos más comunes de test son los test paramétricos. En general, estos test son más robustos y tienen mayor potencia que los test no paramétricos, que se verán más adelante en la sección 2.8. Sin embargo, las pruebas paramétricas se basan en supuestos que muy probablemente

se violan cuando se analiza el rendimiento de algoritmos de optimización y minería de datos [6]. Estas suposiciones o condiciones paramétricas que deben cumplir los resultados de los algoritmos son explicadas a continuación.

2.7.1. Condiciones Paramétricas

Independencia

En estadística, dos eventos son independientes si cuando uno de ellos se da no modifica la probabilidad de la ocurrencia del otro. Dicho de otra forma: cuando las muestras o datos obtenidos por los algoritmos, es decir los resultados de éstos, no dependen unos de otros. La independencia es una característica que en este proyecto debe asumir el usuario de los test, y, por tanto, podrá actuar de una forma u otra bajo su responsabilidad.

Normalidad

Una muestra u observación es normal cuando su comportamiento sigue una distribución normal (o distribución de Gauss) con una cierta media μ y varianza σ^2 . En la figura 2.6 podemos ver que a la izquierda se cumple el supuesto de normalidad, mientras que a la derecha los datos de la muestra no están distribuidos de forma normal:

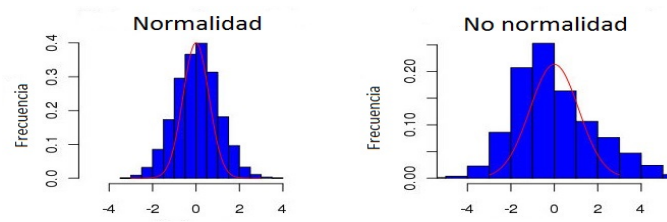


Figura 2.6: Normalidad vs. No normalidad.

En este proyecto se pueden realizar los siguientes test no paramétricos para comprobar el supuesto de normalidad:

- **Shapiro–Wilk:** contrasta la hipótesis nula de que las muestras o poblaciones provienen de una población normalmente distribuida. Analiza la muestra para hallar el nivel de simetría y Kurtosis (forma de la curva) para calcular la diferencia con respecto a una distribución normal, obteniendo el *p-valor* de la suma de los cuadrados de estas discrepancias. Se considera de los más potentes, sobre todo para muestras de menos de 30 elementos. Sin embargo, el rendimiento de esta prueba se ve afectado de forma negativa cuando no existe independencia en los datos.
- **D’Agostino–Pearson:** contrasta la hipótesis nula de que las muestras o poblaciones provienen de una población normalmente distribuida. Primero calcula el coeficiente de asimetría (en qué medida la normal es simétrica ó coeficiente 0) y el coeficiente de Kurtosis (grado de amplitud, donde lo normal es coeficiente 0) para cuantificar cuán lejos se está de

la distribución normal. Luego, calcula cuánto difiere cada uno de los valores de los esperados en una distribución normal, para obtener el p -valor de la suma de estas discrepancias. Es menos potente que el test de Shapiro-Wilk, pero no se ve afectado cuando los datos no son independientes.

- **Kolmogorov–Smirnov:** realiza una prueba de bondad de ajuste, para determinar si los datos observados de la muestra se ajustan a la distribución normal. Tiene como H_0 que la distribución obtenida de los datos observados es idéntica a la distribución normal. Es la prueba que menos potencia presenta de los tres, y por tanto es la que peor funciona.

Homocedasticidad

La homocedasticidad es la condición que dice que las poblaciones de entrada o datos obtenidos por los algoritmos proceden de poblaciones con varianzas iguales. Es decir, esta propiedad indica la hipótesis de igualdad de varianzas. El caso contrario sería heterocedasticidad. En la figura 2.7 se puede apreciar la diferencia existente entre unos datos que presentan homocedasticidad y otros que no, donde el eje de ordenadas representa la varianza del error (la varianza dentro de cada tratamiento o la varianza de los resultados obtenidos por un algoritmo en los distintos conjuntos de datos), y el eje de abscisas representa a las observaciones. En el primer caso, la varianza del error se mantiene constante a lo largo de las observaciones, mientras que en el otro caso no:

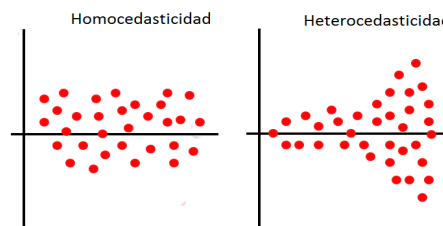


Figura 2.7: Homocedasticidad vs. Heterocedasticidad.

En este proyecto se puede realizar el siguiente test no paramétrico para comprobar el supuesto de homocedasticidad:

- **Test de Levene:** contrasta la hipótesis nula de que todas las poblaciones de entrada proceden de poblaciones con varianzas iguales. Se utiliza para comprobar si K muestras pertenecientes a datos obtenidos por K algoritmos presentan o no homogeneidad de varianzas.

2.7.2. Test ANOVA

El análisis de la varianza (ANOVA) es uno de los test estadísticos más ampliamente utilizados para probar la igualdad de más de dos medias de la población. Se trata de una versión más general del t-test, ya que permite comparar más de 2 poblaciones (en este proyecto resultados de más de dos algoritmos). Dado que es un test paramétrico, se asume que se dan las condiciones de independencia, normalidad y homocedasticidad en su aplicación. De no ser el caso, los resultados de esta prueba no son fiables.

Hipótesis

- Hipótesis nula $H_0: \mu_1 = \mu_2 = \mu_3 \dots = \mu_K$.
- Hipótesis alternativa $H_1: \exists \mu_j \neq \mu \quad j = 1, 2, \dots, K$.

La hipótesis nula indica que las medias de distintas poblaciones o muestras coinciden ($K > 2$), frente a la hipótesis alternativa de que por lo menos una de las poblaciones tiene una media que difiere de las demás. Es, por tanto, un contraste o prueba unilateral con cola a la derecha. En este proyecto el parámetro K viene determinado por el número de algoritmos que existen en la muestra de datos.

Pasos a desarrollar

Los pasos a seguir para realizar el test de ANOVA son los siguientes:

- Se analiza la variación total (respecto a la media general o media de medias de los resultados de cada algoritmo):

$$variacion_t = \sum_{i=1}^N \sum_{j=1}^K (X_{ij} - \bar{X})^2,$$

donde \bar{X} es la media general, N es el número de conjuntos de datos o problemas sobre los que se aplican los algoritmos y X_{ij} es un resultado específico obtenido por un algoritmo.

- Se halla también la variación entre los diferentes tratamientos o algoritmos (efecto de la media de cada tratamiento respecto a la media general):

$$variacion_{tr} = \sum_{i=1}^K N(\bar{X}_i - \bar{X})^2,$$

donde \bar{X}_i representa la media del tratamiento o algoritmo.

- La variación dentro del tratamiento o variación del error (cada valor respecto a la media de su tratamiento):

$$variacion_e = \sum_{i=1}^N \sum_{j=1}^K (X_{ij} - \bar{X}_j)^2,$$

donde \bar{X}_j representa la media de un tratamiento o algoritmo.

- Se calculan los grados de libertad totales, del tratamiento y del error como:

$$GLT = (NK) - 1$$

$$GLTR = K - 1$$

$$GLE = GLT - GLTR$$

- Luego se determinan los cuadrados medios totales (CMT) del tratamiento ($CMTR$) y del error (CME), que son las variaciones divididas entre los grados de libertad correspondientes.

- Se halla el estadístico, que se distribuye como una distribución \mathcal{F} con $K - 1$ y $(KN) - K$ grados de libertad:

$$anova = CMT/CME$$

- Por último se halla el p -valor y se toma la decisión en función del nivel de significancia. A modo de ejemplo para el cálculo del p -valor en este caso, primero se haya la probabilidad de obtener un estadístico menor o igual que el calculado previamente (es decir, el valor que proporciona la función de distribución acumulada \mathcal{F} de Fisher-Snedecor para el estadístico con $K - 1$ y $(KN) - K$ grados de libertad, como vimos en la sección 2.5). Esto representaría un área (probabilidad acumulada) en la función de densidad de probabilidad: el área a la izquierda del valor del estadístico. Dado que se trata de un contraste unilateral con cola a la derecha, se resta esta probabilidad a la probabilidad total, es decir, a 1, para obtener así el área que queda a la derecha del estadístico. Si este valor resultante (p -valor) es menor que el nivel de significancia, se rechazaría la hipótesis nula.

En un contraste bilateral, para hallar el p -valor habría que multiplicar el valor devuelto en la función de distribución acumulada por 2, ya que en este caso habría dos colas.

2.7.3. t-test

El caso más simple de ANOVA donde intervienen únicamente 2 muestras o algoritmos es realizado por este test, también conocido como la prueba \mathcal{T} de Student. Dado que es un test paramétrico, se asume que se dan las condiciones de independencia, normalidad y homocedasticidad en su aplicación. De no ser el caso, los resultados de esta prueba no son fiables.

Hipótesis

- Hipótesis nula $H_0: \mu_1 = \mu_2$.
- Hipótesis alternativa $H_1: \mu_1 \neq \mu_2$.

La hipótesis nula indica que las medias de las 2 poblaciones o muestras coinciden, frente a la hipótesis alternativa de que son distintas. Se trata, por tanto, de un contraste o prueba bilateral. El estadístico de este test, sigue una distribución \mathcal{T} de Student con $2N - 2$ grados de libertad.

2.8. Tests no paramétricos

Cuando los datos obtenidos de la aplicación de los algoritmos de aprendizaje automático no cumplen las características de independencia, normalidad u homocedasticidad total o parcialmente, es necesario aplicar test no paramétricos. La validación de nuevos algoritmos requiere con frecuencia la definición de un marco experimental exhaustivo y la parte crítica de estas comparaciones recae en la validación estadística de los resultados, contrastando las diferencias encontradas entre métodos. Dentro de las técnicas disponibles, destacan los test no paramétricos debido a su flexibilidad y a las pocas restricciones de uso que presentan (a diferencia de los test paramétricos, los cuales sufren a menudo problemas derivados de la imposibilidad de cumplir las condiciones paramétricas para su uso) [5].

2.8.1. Test de Wilcoxon

La prueba de los rangos con signo de Wilcoxon también conocida como el test de Wilcoxon es una prueba no paramétrica que se utiliza como alternativa a la prueba \mathcal{T} de Student cuando no se puede suponer la normalidad de las muestras. Es, por tanto, menos potente que la prueba \mathcal{T} de Student. El test de Wilcoxon fue creado por Frank Wilcoxon y publicado en 1945 [3].

Sirve para comparar dos métodos o tratamientos (en este proyecto interesa comparar dos algoritmos). Por tanto, los individuos (los problemas) donde se aplican los algoritmos tienen que ser los mismos. Es decir, a un mismo individuo se le efectúa la medición de dos variables: las muestras son apareadas. Para tomar la decisión hay que basarse en las observaciones de N individuos independientes (sin relación existente entre ellos). Para tamaños muestrales pequeños, se puede determinar mediante la comparación del estadístico con el valor crítico de la tabla de Wilcoxon. Para tamaños muestrales grandes (> 25), el test se puede aproximar con la distribución normal.

Hipótesis

- Hipótesis nula H_0 : la mediana de las diferencias = 0.
- Hipótesis alternativa H_1 : la mediana de las diferencias $\neq 0$.

La mediana, es el valor que ocupa el lugar central de todos los datos cuando éstos están ordenados de menor a mayor. H_0 indica que la mediana de las diferencias de dos muestras (resultados de dos algoritmos) relacionadas son iguales, es decir, las dos medianas son iguales (los resultados obtenidos no dependen del algoritmo). H_1 indica, por otra parte, que las medianas son diferentes. Se trata, por tanto, de una prueba bilateral.

Pasos a desarrollar

Los pasos a seguir para realizar la prueba de los rangos de Wilcoxon son los siguientes:

- Se calculan las diferencias entre las muestras. Por ejemplo: diferencias entre los resultados del algoritmo A y B.
 - Se eliminan los elementos que tengan diferencias nulas.
- Se ordenan las diferencias en valor absoluto (independientemente del signo).
- Se asignan rangos de orden 1,2,...,N. Si hay empates se calcula la media del rango de cada uno de los elementos repetidos.
- Suma de los rangos según los signos que tengan las diferencias para obtener los estimadores:
 - $T(+) =$ Suma de los rangos correspondientes a diferencias positivas.
 - $T(-) =$ Suma de los rangos correspondientes a diferencias negativas.
- Definir el estadístico:
 - $T = \min[T(+), T(-)]$

- Si $N \leq 25$, se examina la tabla de Wilcoxon que nos da los valores críticos (el intervalo de aceptación) para cada valor de N y cada nivel de significancia. El contraste será estadísticamente significativo si: $T \leq$ límite inferior correspondiente.
- Si $N > 25$, el estadístico se ajusta a la distribución normal. Por tanto, se calcula el estadístico Z y se toma la decisión en función del p -valor y del nivel de significancia:

$$Z = \frac{T - \frac{N(N+1)}{4}}{\sqrt{\frac{N(N+1)(2N+1)}{24}}}$$

2.8.2. Test de Friedman

El test de Friedman es una prueba no paramétrica que puede realizar comparaciones entre dos o más algoritmos, es decir, se trata de una prueba de comparaciones múltiples. Fue desarrollada por el economista Milton Friedman y trabaja asignando rankings para establecer cuál es el mejor algoritmo de la muestra de datos proporcionada.

Hipótesis

- Hipótesis nula H_0 : no existen diferencias entre los algoritmos.
- Hipótesis alternativa H_1 : existen diferencias entre los algoritmos.

La hipótesis nula quiere decir que todos los algoritmos se comportan de la misma forma, por lo que los rankings que poseen deben de ser similares. La hipótesis alternativa, por el contrario, afirma que existen diferencias, lo cual quiere decir que, al menos, el rendimiento de un algoritmo es diferente al rendimiento que presentan los demás. Se trata, por tanto, de un contraste o prueba unilateral con cola a la derecha.

Pasos a desarrollar

Los pasos a seguir para realizar el test de Friedman son los siguientes:

- En primer lugar se asignan rankings r_{ij} a los resultados obtenidos por cada algoritmo j en cada problema i . Es decir, para cada problema o conjunto de datos se asigna un ranking cuyos valores están comprendidos entre 1 y K , donde K representa el número de algoritmos que se están comparando. Los rankings se asignan de forma ascendente (1 al mejor resultado, 2 al segundo mejor, etc.) y se tiene en cuenta la función objetivo de los algoritmos, es decir, si lo que se pretende es minimizar o maximizar resultados.
- En caso de que haya empates en la asignación de rankings anterior, se asignan rankings medios:

$$r_{ij} = \frac{rep + (2pos) + 1}{2},$$

donde rep representa el número de veces que se repite el dato y pos representa la posición que ocupa el dato repetido.

- A continuación, se calculan los rankings medios de cada algoritmo en los N problemas:

$$R_j = \frac{\sum_{i=1}^N r_{ij}}{N}$$

- El estadístico de Friedman sigue una distribución χ^2 con $K - 1$ grados de libertad:

$$friedman = \frac{12N}{K(K+1)} \left[\sum R_j^2 - \frac{K(K+1)^2}{4} \right]$$

- Por último se halla el p -valor y se toma la decisión en función del nivel de significancia.

2.8.3. Test de Iman-Davenport

El estadístico de Friedman fue mejorado por Iman y Davenport, que demostraron que tenía un comportamiento demasiado conservador (se tiende a aceptar la hipótesis nula y, por tanto, la potencia del test es menor).

Hipótesis

- Igual al test de Friedman.

Pasos a desarrollar

- El test de Iman-Davenport hace las mismas operaciones que el test de Friedman pero en él se calcula un estadístico más ajustado (en el que también interviene el estadístico de Friedman). Este nuevo estadístico, sigue una distribución \mathcal{F} con $(K - 1)$ y $(K - 1) * (N - 1)$ grados de libertad, donde N representa el número de problemas o conjuntos de datos y K el número de algoritmos:

$$iman_d = \frac{(N - 1)friedman}{N(K - 1) - friedman}$$

2.8.4. Test de los Rangos Alineados de Friedman

El test de los rangos alineados de Friedman realiza comparaciones y asigna rankings teniendo en cuenta a todos los conjuntos de datos, a diferencia del test de Friedman, que asigna rankings dentro de cada conjunto (es decir, dentro de los resultados obtenidos por los algoritmos para cada problema en particular). Por tanto, en este caso los valores de los rankings irán desde 1 hasta $K * N$. Suele emplearse cuando el número de algoritmos en la comparación es pequeño y cuando se quiere realizar una comparación entre conjuntos de datos.

Hipótesis

- Igual al test de Friedman.

Pasos a desarrollar

- Cálculo de las observaciones alineadas: primero se halla el valor de localización, que es el rendimiento medio alcanzado por los algoritmos en cada conjunto de datos y luego se calculan las diferencias entre el rendimiento obtenido por cada algoritmo con respecto al valor de localización dentro de un mismo conjunto de datos.
- Se repite el primer paso para los N conjuntos de datos.
- Se juntan todas las observaciones alineadas y se ordenan para asignar los rankings alineados desde 1 hasta $K * N$. En caso de empates se procede asignando valores medios igual que en el test de Friedman.
- Se calculan los rankings medios de cada algoritmo en los N problemas.
- El estadístico para esta prueba sigue una distribución χ^2 con $K - 1$ grados de libertad:

$$rangos_{al} = \frac{(K - 1) \left[\sum_{j=1}^K \hat{R}_j^2 - \left(\frac{KN^2}{4} \right) (KN + 1)^2 \right]}{\left[\frac{KN(KN+1)(2KN+1)}{6} \right] - \left(\frac{1}{K} \right) \sum_{i=1}^N \hat{R}_i^2},$$

donde \hat{R}_i y \hat{R}_j son la suma total de los rankings del problema i y del algoritmo j respectivamente.

- Por último se halla el p -valor y se toma la decisión en función del nivel de significancia.

2.8.5. Test de Quade

El test de Quade tiene en cuenta, a diferencia del test de Friedman que considera que todos los problemas son iguales en importancia, que algunos problemas son más difíciles o que los resultados que obtienen los algoritmos sobre ellos son más distantes (se realiza una ponderación).

Hipótesis

- Igual al test de Friedman.

Pasos a desarrollar

- Se obtienen los rankings de cada conjunto de datos de la misma forma que en Friedman.
- Asignación de rankings a los problemas en función del tamaño del rango de la muestra en cada uno (diferencia entre el valor observado más alto y el más bajo). Este ranking de 1 a N usa también rankings medios en caso de empate.
- A partir de estos datos se pueden obtener los rankings medios finales para cada algoritmo y obtener el estadístico, que sigue como una distribución \mathcal{F} con $(K - 1)$ y $(K - 1) * (N - 1)$ grados de libertad.
- Por último se halla el p -valor y se toma la decisión en función del nivel de significancia.

2.8.6. Tests POST-HOC

Los test no paramétricos de ranking (test de Friedman, Iman-Davenport, Rangos Alineados de Friedman y Quade), dan como resultado la existencia o no de diferencias significativas entre los algoritmos sobre los que se han aplicado. Es decir, nos dice si el contraste de hipótesis es o no estadísticamente significativo. Si se rechaza la hipótesis nula de “todos los algoritmos son iguales”, sabremos que entre los algoritmos existen diferencias. Sin embargo, puede ocurrir que un algoritmo presente un rendimiento similar a otro u otros y por tanto se pueda considerar igual.

Estos test comparan los algoritmos y realizan contrastes de hipótesis entre ellos para determinar diferencias.

Hipótesis

- Hipótesis nula H_0 : el algoritmo i y j son iguales.
- Hipótesis alternativa H_1 : el algoritmo i y j son distintos.

Se trata, por tanto, de test que realizan contrastes bilaterales, ya que están destinados a encontrar diferencias a posteriori en caso de que el test de ranking sea estadísticamente significativo. Se distinguen dos tipos de comparación:

- Método de control: se compara el primer algoritmo del ranking devuelto por el test de ranking con el resto de algoritmos y por tanto habrá $K - 1$ comparaciones o contrastes.
- Comparación múltiple: compara todos los algoritmos entre sí. El número de comparaciones o contrastes por tanto es:

$$m = \frac{K(K - 1)}{2}$$

Todos los test POST-HOC aproximan los valores Z (estadísticos) de una distribución normal a partir de las diferencias entre dos rankings. La forma de aproximar estos valores Z varía en función del test de ranking de donde se provenga:

- Test de Friedman / Iman-Davenport:

$$Z = \frac{(R_i - R_j)}{\sqrt{\frac{K(K+1)}{6N}}},$$

donde R_i y R_j son los rankings medios obtenidos por el algoritmo i y j respectivamente en el test de Friedman.

- Test de los Rangos Alineados de Friedman:

$$Z = \frac{\hat{R}_i - \hat{R}_j}{\sqrt{\frac{K(K+1)}{6N}}},$$

donde \hat{R}_i y \hat{R}_j son los rankings medios obtenidos por el algoritmo i y j respectivamente en el test de los Rangos Alineados de Friedman.

- Test de Quade:

$$Z = \frac{T_i - T_j}{\sqrt{\frac{K(K+1)(2N+1)(K-1)}{18N(N+1)}}},$$

donde T_i y T_j son los rankings medios obtenidos por el algoritmo i y j respectivamente en el test de Quade.

Luego, calculan los *p-valores* y ordenan todos los datos en función de éstos de mayor a menor significancia (de menor a mayor). El contraste de las hipótesis (los resultados), así como el cálculo del valor α y los *p-valores* ajustados varían en función de los test aplicados. Los *p-valores* ajustados son *p-valores* que dependen de toda la familia de comparaciones y no sólo de una comparación, es decir, consideran la familia de hipótesis completa para cada pareja de algoritmos y se pueden comparar con el nivel de significancia proporcionado sin ajustar.

Tests

- **Test de Bonferroni-Dunn:** el contraste de las hipótesis se realiza comparando cada *p-valor* con el nivel de significancia ajustado:

$$\alpha_{ajustado} = \frac{\alpha}{(K-1)}$$

La prueba de Bonferroni-Dunn no debe utilizarse a pesar de su simplicidad, ya que es una prueba muy conservadora y muchas diferencias pueden no ser detectadas.

- **Test de Holm:** compara cada *p-valor* (empezando por el más significativo) con:

$$\alpha_{ajustado_i} = \frac{\alpha}{(K-i)}$$

Si se rechaza una hipótesis continúa contrastando. En el caso de que una hipótesis se rechace se rechazan todas las demás.

Procedimiento de Holm siempre puede considerarse mejor (mayor potencia) que el de Bonferroni-Dunn, porque controla adecuadamente la FWER (*Familywise error rate*), que es la probabilidad de cometer uno o más errores de tipo I entre todas las hipótesis.

- **Test de Hochberg:** compara en la dirección opuesta a Holm. En el momento que encuentra una hipótesis que pueda aceptar, acepta todas las demás.

Procedimiento de Hochberg presenta mayor potencia que el de Holm, pero las diferencias entre ellos en la práctica son pequeñas.

- **Test de Finner:** compara cada *p-valor* (empezando por el más significativo) con:

$$\alpha_{ajustado_i} = 1 - (1 - \alpha)^{\frac{(K-1)}{i}}$$

Al igual que el test de Holm, si se rechaza una hipótesis continúa contrastando. En el caso de que una hipótesis se rechace se rechazan todas las demás.

Una de las mejores alternativas es la prueba Finner, ya que es fácil de aplicar y ofrece mejores resultados que los otros procedimientos, excepto la prueba de Li en algunos casos.

- **Test de Li:** rechaza todas las hipótesis si el p -valor menos significativo es menor que α . En otro caso, acepta dicha hipótesis y rechaza cualquier hipótesis restante cuyo p -valor sea menor que un valor específico:

$$valor = \frac{(1 - p\text{-valor}_{k-1})}{(1 - \alpha)\alpha}$$

El test POST-HOC de Li es más simple incluso que el test de Finner (ya que sólo requiere dos pasos). El autor declara que la potencia de esta prueba está altamente influenciada por el p -valor (mayor potencia cuando el p -valor es inferior a 0.5) [7].

Los test anteriores son para el caso de comparaciones simples (utilizan un método de control), con lo que interviene el parámetro K para realizar las $K - 1$ comparaciones. Para el caso de comparación múltiple, habría que reemplazar K por m (número de comparaciones múltiples). Para los test POST-HOC anteriores existe, por tanto, un test POST-HOC de comparación múltiple que se obtiene de forma análoga, excepto para el test de Li. En lugar del test de Li para comparaciones múltiples en este proyecto se implementa el test de Shaffer:

- **Test de Shaffer:** rechaza cada H_i si:

$$p\text{-valor}_i \leq \frac{\alpha}{t_i},$$

donde t_i puede ser obtenido mediante:

$$S(K) = \bigcup_{j=1}^K \left\{ \binom{j}{2} + x : x \in S(K - j) \right\},$$

que calcula la secuencia de número máximo de hipótesis que pueden ser ciertas en una comparación secuencial entre K distribuciones, donde $K \geq 2$ y $S(0) = S(1) = \{0\}$

Los test de la lista anterior están ordenados de menor a mayor potencia, siendo, como se puede apreciar, el test de Bonferroni-Dunn el menos potente de todos y el test de Li el que mayor potencia presenta. Sin embargo, las diferencias de potencia entre los métodos son bastante pequeñas en general, con algunas excepciones, como se ha comentado anteriormente [7].

Capítulo 3

Análisis de requisitos

El análisis de requisitos es el primer paso técnico del proceso de ingeniería del software. Es aquí donde se refina la declaración general del ámbito del software en una especificación concreta que se convierte en la base de todas las actividades de ingeniería del software que siguen.

En este proyecto, hemos utilizado el enfoque de metodología de desarrollo ágil Scrum [10]. Los detalles de esta metodología y la razón de su utilización en este proyecto se detallan en la sección 4.2. Para realizar el análisis de requisitos, a diferencia del enfoque de tradicional en el que los requisitos son descritos de una forma muy estricta y formal, esta metodología permite describir las necesidades del usuario de una manera más simple. Para ello, se establecen los requisitos mediante las **historias de usuario**. Las historias de usuario son requisitos escritos en un lenguaje coloquial bien directamente por el mismo cliente, o bien como un recordatorio posterior de las conversaciones mantenidas con el cliente. Consisten en una o dos frases en donde de una forma no precisa se detalla lo que el usuario requiere de la aplicación. Además, deben ser:

- **Independientes:** no depender de otras para su compleción.
- **Negociables:** no son del todo claras, y por tanto se necesita discutir con los usuarios. Se concretan en los criterios de aceptación.
- **Valoradas por el cliente:** esto permite conocer en qué está más interesado el cliente y qué es más importante para la aplicación.
- **Estimables:** se puede establecer una valoración del tiempo que llevará completarlas.
- **Pequeñas:** para poder hacer una mejor estimación. Normalmente más de 2 días y menos de 1 semana.
- **Verificables:** se necesitan poder probar para saber si se ha completado con éxito.

El formato a seguir es el siguiente:

Como <tipo de usuario>, me gustaría <objetivo>, ya que <razón>

Además, a las historias de usuario se añaden criterios de aceptación y una prioridad.

Entre los beneficios de usar historias de usuario para elaborar los requisitos destaca que no se requiere elaborar una gran cantidad de documentos formales y por lo tanto se requiere menos tiempo para su administración. Por ello, permiten responder rápidamente a los requisitos cambiantes, algo a tener en cuenta sabiendo que normalmente los clientes o los usuarios finales con frecuencia no saben lo que necesitan desde un principio y es algo que se debe ir refinando a lo largo del proyecto.

Un nivel de abstracción mayor a las historias de usuario es dado por los denominados Epics. Los Epics son historias de usuario mucho más generales (a más alto nivel) que nos dan una primera idea del trabajo que puede estar involucrado en su definición. Los Epics se pueden desglosar en varias historias de usuario y están compuestos por un título o requerimiento muy general.

En este proyecto, primeramente se detallan los Epics y luego se describen las historias de usuario en las que se pueden desglosar. Para la prioridad de las historias de usuario, se han definido tres niveles en función del interés del cliente en las mismas:

- **Alta:** Indica que el cliente está muy interesado en el requerimiento en cuestión y que lo considera clave para la aplicación.
- **Media:** Indica que el cliente está interesado en el requerimiento, pero no es tan importante para la aplicación.
- **Baja:** Indica aquellos requerimientos que, siendo favorables, son prescindibles.

3.1. Epics

Los Epics son identificados mediante EP-x, donde x representa un número natural único.

- **EP-1:** Proporcionar una API REST de los test estadísticos.
- **EP-2:** Permitir realizar test estadísticos paramétricos.
- **EP-3:** Permitir realizar test estadísticos no paramétricos.
- **EP-4:** Permitir realizar test estadísticos no paramétricos para evaluar las condiciones paramétricas.
- **EP-5:** Permitir gestionar un fichero.
- **EP-6:** Visualizar los resultados de los test.
- **EP-7:** Permitir modificar opciones de los test.
- **EP-8:** Proporcionar una interfaz usable.

3.2. Historias de usuario

Las historias de usuario se identifican mediante HU-x, donde x representa un número natural único. Cabe destacar que en este proyecto se consideran dos tipos de usuario: desarrollador y cliente. El primero, se considera debido a que una de las partes del proyecto consiste en desarrollar una API REST que hace disponible los test vía web. Esta API la puede utilizar un desarrollador para realizar la interfaz web, con lo que es necesario que un desarrollador diga las necesidades que requiere de la API. El cliente, por otra parte, se refiere al usuario final de la plataforma, es decir, el analista de datos que quiere validar algoritmos de aprendizaje automático. Ambos tienen intereses diferentes y por tanto se pueden considerar por separado.

3.2.1. Historias de usuario desarrollador

Desglosando el Epic **EP-1**: Proporcionar una API REST de los test estadísticos, se obtienen las siguientes historias de usuario:

HU-1 - Acceder a los test	
Como:	Desarrollador
Me gustaría:	acceder a los test como recursos independientes con parámetros opcionales.
Ya que:	esto facilita su utilización y los hace más flexibles.
C. Aceptación:	<ul style="list-style-type: none"> - Los test estadísticos son accesibles individualmente como un recurso. - Los métodos de POST-HOC pueden ser también accedidos individualmente como un sub recurso dentro del recurso de acceso del test principal. - Se proporcionan varias URIs (identificador de recursos uniforme) para cada test cuyos parámetros (como el nivel de significación, la función objetivo o el test POST-HOC) son opcionales, teniendo un valor común por defecto.
Prioridad:	alta

HU-2 - Gestionar ficheros	
Como:	Desarrollador
Me gustaría:	poder gestionar un fichero como un recurso independiente.
Ya que:	esto facilita la subida y consulta de ficheros.
C. Aceptación:	<ul style="list-style-type: none"> - Los ficheros son accesibles individualmente como un recurso. - Los recursos de fichero son creados de forma individual.
Prioridad:	alta

HU-3 - Devolver datos JSON	
Como:	Desarrollador
Me gustaría:	que los datos devueltos por los servicios de la API REST fuesen en formato JSON (JavaScript Object Notation).
Ya que:	es un formato ligero para el intercambio de datos y además es muy simple, con lo que el análisis sintáctico sería más sencillo.
C. Aceptación:	- Los servicios de la API REST devuelven datos en formato JSON.
Prioridad:	media

HU-4 - Analizar datos subidos	
Como:	Desarrollador
Me gustaría:	que el fichero de datos subido al servidor tuviese el formato csv.
Ya que:	es un formato abierto y sencillo para representar datos en forma de tabla y los test reciben siempre los datos en forma de matriz, donde las filas representan los conjuntos de datos de cada problema y las columnas los resultados obtenidos de cada algoritmo.
C. Aceptación:	- El servicio de subida comprueba que los datos subidos siguen el estándar csv y están dispuestos de acuerdo a una convención establecida para el proyecto. - En caso de no cumplir el estándar se devuelve un error.
Prioridad:	alta

HU-5 - Limitar ficheros subidos	
Como:	Desarrollador
Me gustaría:	establecer un límite de ficheros subidos.
Ya que:	así se evita tener en memoria ficheros muy antiguos que ya no se usan.
C. Aceptación:	- Los ficheros se almacenan en un diccionario con límite de elementos, de forma que cuando se llega al límite, el elemento con más tiempo en el diccionario se elimina.
Prioridad:	baja

HU-6 - Visualizar información test	
Como:	Desarrollador
Me gustaría:	obtener información acerca de los test estadísticos.
Ya que:	esto permite conocer qué hace cada test y evita confusiones a la hora de utilizar el servicio.
C. Aceptación:	- Los servicios de la API REST tienen información relativa al test (uso e hipótesis). - El módulo de Python donde están implementados los test tiene documentación en base a un generador de documentación relativa al test (uso, hipótesis, argumentos de entrada / salida, ...).
Prioridad:	media

3.2.2. Historias de usuario cliente

Desglosando el Epic **EP-2**: Permitir realizar test estadísticos paramétricos, se obtienen las siguientes historias de usuario:

HU-7 - Realizar test de ANOVA	
Como:	Cliente
Me gustaría:	poder aplicar el test de ANOVA a mis datos.
Ya que:	es una prueba muy utilizada para determinar si las medias de más de dos muestras son similares.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test ANOVA. - La plataforma muestra una sección para test paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i> y el resultado. En caso de ser estadísticamente significativo, también aparecerán los resultados del test POST-HOC de Bonferroni (<i>p-valores</i>, <i>p-valores</i> ajustados, etc.)
Prioridad:	alta

HU-8 - Realizar test t-test	
Como:	Cliente
Me gustaría:	poder aplicar el test t-test a mis datos.
Ya que:	aunque contrasta la misma hipótesis que ANOVA éste tiene otro estadístico y funciona únicamente para dos algoritmos, con lo que puede resultar interesante tener otro punto de vista.
C. Aceptación:	<ul style="list-style-type: none"> - La plataforma muestra una sección para test paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i> y el resultado. - Devuelve error en caso de que los datos tengan resultados de más de 2 algoritmos.
Prioridad:	media

Desglosando el Epic **EP-3**: Permitir realizar test estadísticos no paramétricos, se obtienen las siguientes historias de usuario:

HU-9 - Realizar test de Wilcoxon	
Como:	Cliente
Me gustaría:	poder aplicar el test de Wilcoxon a mis datos.
Ya que:	es útil como alternativa al test t-test cuando los datos no cumplen con el supuesto de normalidad.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Wilcoxon. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve el estadístico, el <i>p-valor</i>,... incluyendo también la suma de los rangos positivos y la suma de los rangos negativos característicos del test de Wilcoxon. - Devuelve error en caso de que los datos tengan resultados de más de 2 algoritmos.
Prioridad:	alta

HU-10 - Realizar test de Friedman	
Como:	Cliente
Me gustaría:	poder aplicar el test de Friedman a mis datos.
Ya que:	es una prueba muy útil a la hora de comparar algoritmos (ya que trabaja asignando rankings) y determinar si existen diferencias entre ellos.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Friedman. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i>, el resultado y el ranking.
Prioridad:	alta

HU-11 - Realizar test de Iman-Davenport	
Como:	Cliente
Me gustaría:	poder aplicar el test de Iman-Davenport a mis datos.
Ya que:	es una prueba cuyo estadístico es más ajustado que el de Friedman y por lo tanto es más potente y puede servir mejor para mis datos.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Iman-Davenport. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i>, el resultado y el ranking.
Prioridad:	alta

HU-12 - Realizar test de los Rangos Alineados de Friedman	
Como:	Cliente
Me gustaría:	poder aplicar el test de los Rangos Alineados de Friedman.
Ya que:	es una prueba que a diferencia de la de Friedman establece los rankings teniendo en cuentas las posibles interrelaciones que pueden haber entre los distintos conjuntos de datos o problemas, con lo que puede ser muy útil en ese sentido.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Rangos Alineados de Friedman. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i>, el resultado y el ranking.
Prioridad:	alta

HU-13 - Realizar test de Quade	
Como:	Cliente
Me gustaría:	poder aplicar el test de Quade.
Ya que:	es una prueba que a diferencia de la de Friedman considera que algunos problemas son más difíciles, o que los resultados que obtienen los algoritmos sobre ellos son más distantes, con lo cual es de las pruebas de ranking más potentes y mejores que puedo aplicar sobre mis resultados.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Quade. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los datos: estadístico, el <i>p-valor</i>, el resultado y el ranking.
Prioridad:	alta

HU-14 - Realizar test de Bonferroni-Dunn	
Como:	Cliente
Me gustaría:	poder aplicar el test de Bonferroni-Dunn.
Ya que:	es una prueba muy común que se realiza en caso de que los test de ranking obtengan diferencias significativas para detectar diferencias concretas entre pares de algoritmos.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Bonferroni-Dunn con método de control y el test para comparaciones múltiples. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba tanto simple como de comparación múltiple. - El test devuelve los esperados, como los <i>p-valores</i> ajustados, los estadísticos, el método de control, etc., en caso de que el test de ranking principal sea estadísticamente significativo. Si se trata del POST-HOC simple, se realizarán $K - 1$ comparaciones. En el caso de la prueba multitest, se realizarán $\frac{K(K-1)}{2}$ comparaciones.
Prioridad:	alta

HU-15 - Realizar test de Holm	
Como:	Cliente
Me gustaría:	poder aplicar el test de Holm.
Ya que:	se trata de una prueba de comparación a posteriori con más potencia que la de Bonferroni-Dunn y puede resultar más útil al rechazar posiblemente más hipótesis.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Holm con método de control y el test para comparaciones múltiples. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba tanto simple como de comparación múltiple. - El test devuelve los resultados esperados, como los <i>p-valores</i> ajustados, los estadísticos, el método de control, etc., en caso de que el test de ranking principal sea estadísticamente significativo. Si se trata del POST-HOC simple, se realizarán $K - 1$ comparaciones. En el caso de la prueba multitest, se realizarán $\frac{K(K-1)}{2}$ comparaciones.
Prioridad:	alta

HU-16 - Realizar test de Finner	
Como:	Cliente
Me gustaría:	poder aplicar el test de Finner.
Ya que:	se trata de una prueba de comparación a posteriori con más potencia que la de Holm y puede resultar más útil al rechazar posiblemente más hipótesis.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Finner con método de control y el test para comparaciones múltiples. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba tanto simple como de comparación múltiple. - El test devuelve los resultados esperados, como los <i>p-valores</i> ajustados, los estadísticos, el método de control, etc., en caso de que el test de ranking principal sea estadísticamente significativo. Si se trata del POST-HOC simple, se realizarán $K - 1$ comparaciones. En el caso de la prueba multitest, se realizarán $\frac{K(K-1)}{2}$ comparaciones.
Prioridad:	alta

HU-17 - Realizar test de Hochberg	
Como:	Cliente
Me gustaría:	poder aplicar el test de Hochberg.
Ya que:	se trata de una prueba de comparación a posteriori con más potencia que la de Finner y puede resultar más útil al rechazar posiblemente más hipótesis.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Hochberg con método de control y el test para comparaciones múltiples. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba tanto simple como de comparación múltiple. - El test devuelve los resultados esperados, como los <i>p-valores</i> ajustados, los estadísticos, el método de control, etc., en caso de que el test de ranking principal sea estadísticamente significativo. Si se trata del POST-HOC simple, se realizarán $K - 1$ comparaciones. En el caso de la prueba multitest, se realizarán $\frac{K(K-1)}{2}$ comparaciones.
Prioridad:	alta

HU-18 - Realizar test de Li	
Como:	Cliente
Me gustaría:	poder aplicar el test de Li.
Ya que:	se trata de una prueba de comparación a posteriori con más potencia que la de Hochberg y puede resultar más útil al rechazar posiblemente más hipótesis.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Li simple con método de control. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los resultados esperados, como los $K - 1$ <i>p-valores</i> ajustados, estadísticos, resultados, etc.
Prioridad:	alta

HU-19 - Realizar test de Shaffer	
Como:	Cliente
Me gustaría:	poder aplicar el test de Shaffer para comparaciones múltiples.
Ya que:	se trata de una prueba multitest con mucha potencia y puede que rechace muchas más hipótesis.
C. Aceptación:	<ul style="list-style-type: none"> - El módulo de Python tiene implementado el test de Shaffer de comparaciones múltiples. - La plataforma muestra una sección para test no paramétricos donde se puede aplicar la prueba. - El test devuelve los resultados esperados, como los $\frac{K(K-1)}{2}$ <i>p-valores</i> ajustados, estadísticos, resultados, etc.
Prioridad:	alta

Desglosando el Epic **EP-4**: Permitir realizar test estadísticos no paramétricos para evaluar las condiciones paramétricas, se obtienen las siguientes historias de usuario:

HU-20 - Realizar test de normalidad	
Como:	Cliente
Me gustaría:	poder realizar distintos test para ver si mis muestras de datos obtenidas por los algoritmos están distribuidas de forma normal.
Ya que:	esto ayuda a determinar si sobre mis datos puedo aplicar test paramétricos o si por el contrario lo más adecuado (si no existe normalidad) es aplicar un test no paramétrico.
C. Aceptación:	<ul style="list-style-type: none"> - La plataforma muestra una sección para test de condiciones paramétricas donde se pueden aplicar las pruebas de normalidad de Shapiro-Wilk, D'Agostino-Pearson y Kolmogorov-Smirnov para determinar si los datos siguen una distribución normal, siendo el test de Shapiro-Wilk el más potente y Kolmogorov-Smirnov el menos potente. - El test devuelve, para cada muestra de resultados obtenida por cada algoritmo, los estadísticos, <i>p-valores</i> y resultados.
Prioridad:	alta

HU-21 - Realizar test de Levene	
Como:	Cliente
Me gustaría:	poder realizar el test de Levene para ver si mis muestras de datos obtenidas por los algoritmos están distribuidas de forma normal.
Ya que:	se trata de una prueba para determinar la homocedasticidad, condición requerida para aplicar correctamente un test paramétrico o no paramétrico en caso de que los datos no presenten homocedasticidad.
C. Aceptación:	<ul style="list-style-type: none"> - La plataforma muestra una sección para test de condiciones paramétricas donde se puede aplicar la prueba de homocedasticidad de Levene. - El test devuelve el estadístico, el <i>p-valor</i> y el resultado de la prueba.
Prioridad:	alta

Desglosando el Epic **EP-5**: Permitir gestionar un fichero, se obtienen las siguientes historias de usuario:

HU-22 - Subir fichero de datos	
Como:	Cliente
Me gustaría:	poder subir en un fichero los resultados obtenidos por los algoritmos de aprendizaje.
Ya que:	es la forma más cómoda de proporcionar los datos y realizar los test sobre ellos.
C. Aceptación:	<ul style="list-style-type: none"> - La plataforma muestra en la página principal un botón para subir el fichero. - En la parte superior de la plataforma también hay disponible un botón para poder realizar la subida de datos.
Prioridad:	alta

HU-23 - Consultar fichero de datos	
Como:	Cliente
Me gustaría:	poder consultar los datos de mi fichero de resultados obtenidos por los algoritmos.
Ya que:	es una forma de poder visualizar los datos sin tener que abrir el fichero en el escritorio local.
C. Aceptación:	<ul style="list-style-type: none"> - La plataforma en la parte superior muestra un botón para consultar el fichero en caso de que se haya subido uno previamente. - Al subir un fichero, la plataforma redirige automáticamente a la página de visualización del archivo.
Prioridad:	media

Desglosando el Epic **EP-6**: Visualizar los resultados de los test, se obtienen las siguientes historias de usuario:

HU-24 - Visualizar resultados de los test	
Como:	Cliente
Me gustaría:	poder ver los resultados en forma de tabla señalando el lugar concreto de la tabla por donde se pasa el ratón.
Ya que:	de esta forma, la lectura de resultados será más simple.
C. Aceptación:	<ul style="list-style-type: none"> - Los resultados se muestran en forma de tabla, teniendo en la primera fila de las columnas el nombre que identifica a los datos de la columna. - Cuando se pasa el ratón por un elemento de la tabla, toda la fila se destaca para poder visualizar mejor los datos relacionados.
Prioridad:	alta

HU-25 - Exportar los resultados en formato csv	
Como:	Cliente
Me gustaría:	poder exportar los resultados a fichero en formato csv.
Ya que:	así puedo guardar de forma automática los resultados en local en un formato sencillo y legible.
C. Aceptación:	<ul style="list-style-type: none"> - Cuando se muestran los resultados en una tabla, encima de ésta aparece un botón para exportar a formato csv. - En caso de que se aplique un test de comparación POST-HOC también se incluirá un botón para exportar a csv los resultados de éste.
Prioridad:	media

HU-26 - Exportar los resultados en formato \LaTeX	
Como:	Cliente
Me gustaría:	poder exportar los resultados a fichero en formato \LaTeX .
Ya que:	así puedo guardar de forma automática los resultados para luego poder utilizarlos en un documento de \LaTeX , sin necesidad de crear manualmente un tabla.
C. Aceptación:	<ul style="list-style-type: none"> - Cuando se muestran los resultados en una tabla, encima de ésta aparece un botón para exportar a formato \LaTeX. - En caso de que se aplique un test de comparación POST-HOC también se incluirá un botón para exportar a \LaTeX los resultados de éste.
Prioridad:	media

Desglosando el Epic **EP-7**: Permitir modificar opciones de los test, se obtienen las siguientes historias de usuario:

HU-27 - Seleccionar nivel de significancia	
Como:	Cliente
Me gustaría:	poder elegir el nivel de significancia de los test.
Ya que:	el nivel es un parámetro muy importante a la hora de calcular los resultados de los test y conviene que sea modificable.
C. Aceptación:	- En la elección del test se da la posibilidad de seleccionar un nivel de significación. Por defecto se establece el más común: 0.05.
Prioridad:	Alta

HU-28 - Seleccionar la función objetivo	
Como:	Cliente
Me gustaría:	poder elegir la función objetivo de los algoritmos con los que voy a aplicar los test.
Ya que:	esto modifica el orden del ranking en los test no paramétricos de ranking.
C. Aceptación:	- En la elección del test se da la posibilidad de seleccionar minimización (establecido por defecto) o maximización.
Prioridad:	media

Desglosando el Epic **EP-8**: Proporcionar una interfaz usable, se obtienen las siguientes historias de usuario:

HU-29 - Ver ayuda	
Como:	Cliente
Me gustaría:	poder acceder a la ayuda de la plataforma.
Ya que:	siempre es útil tener un acceso a información que puede ayudar a entender mejor los test, y los conceptos relacionados con los resultados obtenidos, etc.
C. Aceptación:	<ul style="list-style-type: none"> - En la parte superior derecha de la plataforma aparece un botón para acceder a la ayuda. - En los lugares donde sea preciso (en el panel de subida de fichero, en los test, etc.) hay disponible un enlace al concepto relacionado en la ayuda.
Prioridad:	media

HU-30 - Recordar el test de ranking	
Como:	Cliente
Me gustaría:	poder visualizar, cuando estoy eligiendo un test de comparación POST-HOC, el test de ranking previamente seleccionado.
Ya que:	así me podría dar cuenta si me equivoco en la elección.
C. Aceptación:	- En la selección del test POST-HOC aparece encima el test de ranking previamente seleccionado.
Prioridad:	baja

HU-31 - Avisar de las condiciones paramétricas	
Como:	Cliente
Me gustaría:	que la plataforma me diese un aviso en caso de que intente aplicar un test paramétrico sin haber hecho previamente las condiciones paramétricas.
Ya que:	aplicar un test paramétrico sobre datos que no siguen las suposiciones paramétricas no da resultados fiables.
C. Aceptación:	<ul style="list-style-type: none"> - la plataforma da un aviso si se intenta hacer un test paramétrico sin haber comprobado antes las condiciones sobre esos datos. - No se impide hacer el test si así lo desea el usuario.
Prioridad:	media

HU-32 - Ver una breve información de los test	
Como:	Cliente
Me gustaría:	que debajo de las opciones para realizar los test, apareciese una breve descripción de los mismos.
Ya que:	conviene tener, para el que no lo sepa, una mínima idea de qué hipótesis se contrastan en él.
C. Aceptación:	- Debajo de las opciones de los test aparece una breve descripción de las hipótesis que se contrastan.
Prioridad:	media

HU-33 - Enlazar con ficheros de ejemplo	
Como:	Cliente
Me gustaría:	que en la sección de ayuda de fichero hubiese enlaces a archivos de ejemplo.
Ya que:	así podría probar la aplicación sin necesidad de crear un archivo y además podría ver mejor cómo tiene que ser el formato del archivo de resultados a contrastar.
C. Aceptación:	- En la sección de ayuda de fichero aparecen enlaces a ficheros de ejemplo.
Prioridad:	baja

HU-34 - Aplicación en diferentes pestañas del navegador	
Como:	Cliente
Me gustaría:	poder utilizar de forma independiente la plataforma en diferentes pestañas del navegador.
Ya que:	así puedo estar trabajando a la vez con diferentes archivos.
C. Aceptación:	- La plataforma puede operar independientemente en diferentes pestañas con diferentes archivos de datos.
Prioridad:	alta

HU-35 - Diseño adaptable	
Como:	Cliente
Me gustaría:	poder utilizar la plataforma en dispositivos de pantalla pequeña.
Ya que:	esto haría la plataforma más usable y podría trabajar en más dispositivos, como una tableta o móvil.
C. Aceptación:	- El diseño de la plataforma se adapta a dispositivos cuya resolución es más baja de la normal.
Prioridad:	media

HU-36 - Idioma	
Como:	Cliente
Me gustaría:	que el idioma de la aplicación fuese el inglés.
Ya que:	esto haría que la aplicación fuese mas fácilmente divulgable y potencialmente utilizada por más personas.
C. Aceptación:	- Los textos de la plataforma están escritos en inglés.
Prioridad:	baja

HU-37 - Flujo de trabajo	
Como:	Cliente
Me gustaría:	poder visualizar el flujo de trabajo de la aplicación de los test al entrar en la plataforma.
Ya que:	esto me permitiría tener a primera vista una referencia clara de los pasos que debo seguir y las cosas que puedo hacer.
C. Aceptación:	<ul style="list-style-type: none">- La página principal de la plataforma muestra el flujo de trabajo a través una imagen donde se detalla de forma esquemática el flujo.- En la página principal se muestra un botón para subir un fichero, primer paso del flujo de trabajo.
Prioridad:	baja

Capítulo 4

Gestión del proyecto

La gestión de proyectos de software es una parte esencial de la ingeniería del software. La buena gestión no puede garantizar el éxito del proyecto. Sin embargo, la mala gestión normalmente lleva al fracaso del proyecto. La gestión del proyecto es una etapa llevada a cabo a lo largo de todo el ciclo de vida del proyecto como método para lograr que el producto final se ajuste a las necesidades y restricciones impuestas (tiempo, costes, requerimientos, etc.).

En este capítulo se realiza la gestión de costes del proyecto, donde se detalla el análisis de riesgos, la metodología de desarrollo empleada, la gestión de configuración, la planificación temporal y la estimación de costes del proyecto.

4.1. Análisis de riesgos

Una tarea importante del gestor de proyectos es anticipar los riesgos que podrían afectar a la programación del proyecto o a la calidad del software a desarrollar y emprender acciones para evitar esos riesgos.

Un riesgo de un proyecto es un evento o condición incierto que, si se produce, tendrá un efecto positivo o negativo sobre al menos un objetivo del proyecto, como tiempo, coste, alcance o calidad, es decir, cuando el objetivo de tiempo de un proyecto es cumplir con el cronograma acordado; cuando el objetivo de coste del proyecto es cumplir con el coste acordado, etc.

El riesgo está compuesto de tres componentes esenciales:

- Un evento definible.
- Probabilidad de ocurrencia.
- Consecuencia de la ocurrencia (impacto).

En este proyecto, se consideran los riesgos que, en caso de producirse, tendrían un efecto negativo sobre el proyecto.

En las metodologías ágiles y en concreto en Scrum, que es la metodología utilizada en este proyecto (explicada en 4.2.2), los ciclos de desarrollo son cortos, se hacen constantes revisiones y se considera la idea de incertidumbre desde el principio (de la que tienen origen normalmente los riesgos). Scrum se encarga del riesgo a principios del ciclo de vida del proyecto y sustituye la gestión del riesgo explícita (la gestión de riesgos utilizada en las metodologías tradicionales, donde la burocracia dificulta que se lleve a cabo la disciplina y actualización adecuada) por la gestión del riesgo continua. Esta gestión continua es realizada en las distintas reuniones que se llevan a cabo a lo largo del desarrollo del proyecto.

A continuación, se enumeran los posibles riesgos en dos grupos. Para ello, se identifica el riesgo, se da una breve descripción, se analiza (cuál sería la probabilidad y el impacto) y se indican posibles estrategias de prevención, minimización o contingencia. Esta descripción está basada en la gestión de riesgos propuesta por el ingeniero de software Ian Sommerville en su libro [9].

Riesgos del proyecto: Afectan a la calendarización o los recursos.

RPY-1 - Retraso respecto a la planificación temporal	
Descripción:	No se consigue seguir la planificación prevista debido a una mala organización, planificación o debido a que tiene lugar algún otro riesgo descrito a continuación.
Probabilidad:	Alta
Impacto:	Serio
Estrategia de prevención:	Las reuniones de planificación y revisión de los incrementos (sprints) con los directores del proyecto llevadas a cabo en la metodología Scrum sirven de estrategia de prevención, ya que se puede validar el progreso.
Plan de contingencia:	Ajustar la planificación inicial.

Riesgos del producto: Afectan a la calidad o al rendimiento del software.

RPD-1 - Alteración de los requisitos	
Descripción:	No se definen todos los requisitos al inicio del desarrollo o algunos requisitos cambian debido por ejemplo a que una prueba de hipótesis no devuelve los resultados esperados o no existe suficiente información para llevar a cabo su implementación.
Probabilidad:	Moderada
Impacto:	Tolerable
Estrategia de minimización:	La metodología Scrum utilizada en este proyecto asume la incertidumbre y por tanto considera que los requisitos pueden cambiar a lo largo del desarrollo, con lo cual constituye una forma de minimizar el impacto.

RPD-2 - Escasa información de un determinado test estadístico	
Descripción:	La información existente de una determinada prueba de hipótesis es escasa, debido a que el test es reciente y aún no existe mucha documentación, lo que dificulta e impide que se pueda implementar de forma correcta.
Probabilidad:	Baja
Impacto:	Tolerable
Estrategia de prevención:	Buscar información en artículos relacionados con el autor o test similares que puedan ayudar a entender qué operaciones realiza el test.
Plan de contingencia:	Sustituir el test estadístico por otro de similares características para el que sí haya documentación.

RPD-3 - Resultados de test no esperados	
Descripción:	Los resultados obtenidos de la aplicación de un test no son los esperados. Cabe destacar que puede que para un determinado conjunto de datos de entrada los resultados obtenidos sean mejores que los resultados obtenidos para otros datos.
Probabilidad:	Moderada
Impacto:	Tolerable
Estrategia de prevención:	Programar paso a paso contrastando los resultados de se obtienen y probar con diferentes conjuntos de datos.
Plan de contingencia:	Sustituir el test estadístico por otro de similares características que funcione mejor.

RPD-4 - Mala usabilidad de la plataforma	
Descripción:	La web no es lo suficientemente clara y fácil de usar para los usuarios finales.
Probabilidad:	Baja
Impacto:	Serio
Estrategia de prevención:	Las reuniones de revisión de los incrementos (sprints) con los directores del proyecto (“representantes” de la voz del cliente) llevadas a cabo en la metodología Scrum sirven de estrategia de prevención, haciendo que la probabilidad de ocurrencia de este riesgo sea mínima.
Plan de contingencia:	Establecer una reunión urgente con los directores para tratar de determinar las causas y reconstruir las características que fallan.

4.2. Metodología de desarrollo

En todo proyecto software la elección de la metodología de desarrollo juega un papel crucial a la hora de finalizar con éxito el proyecto. El diseño y el desarrollo de software no es una tarea sencilla y cada proyecto tiene características personales que propician la selección de una determinada metodología u otra.

Al principio, cuando el desarrollo de software todavía se estaba iniciando, surgió la necesidad de mejorar el proceso para poder finalizar con éxito los proyectos, con lo que se implantaron y adaptaron metodologías existentes en otras áreas al desarrollo de software. Estas metodologías tradicionales, también denominadas pesadas, dividen el proceso de desarrollo en varias etapas que se van realizando de manera secuencial, siendo la primera fase aquella que sienta las bases del proyecto. Uno de los principales inconvenientes que tienen estas metodologías es su baja tolerancia a los cambios, así como el no ofrecer una buena solución en entornos volátiles.

4.2.1. Metodologías ágiles

En un escenario rápido e inestable, encajan mal las estrategias de negocio predictivas (metodologías pesadas), que diseñan un producto y trazan un plan de negocio. Es en este ámbito donde surgen las metodologías ágiles, que ofrecen una mayor libertad al equipo de trabajo cuando existe incertidumbre. No se trata de elegir un modelo como el mejor, simplemente habrá casos en los que convendrá una gestión predictiva (p.ej. cuando los requisitos están completamente determinados desde el principio y no habrá cambios) y otros (p.ej. entornos volátiles) en los que la opción ágil puede ser más beneficiosa.

En marzo de 2001, 17 críticos de los modelos de producción basados en procesos, convocados por Kent Beck, que había publicado un par de años antes el libro en el que explicaba la nueva metodología Extreme Programming Beck [8]) se reunieron en Salt Lake City para discutir sobre el desarrollo de software. En la reunión se acuñó el término “Métodos Ágiles” para definir a aquellos que estaban surgiendo como alternativa a las metodologías formales, a las que consideraban excesivamente pesadas y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas, previas al desarrollo. Los integrantes de la reunión resumieron en cuatro postulados lo que ha quedado denominado como “Manifiesto Ágil”, que son los valores sobre los que se asientan estos métodos:

- Se valora más a los individuos y su interacción que a los procesos y las herramientas.
- Se valora más el software que funciona que la documentación exhaustiva.
- Se valora más la colaboración con el cliente que la negociación contractual.
- Se valora más la respuesta al cambio que el seguimiento de un plan.

Sobre estos valores, se establecen una serie de principios de las metodologías ágiles, como son:

- Requisitos cambiantes bienvenidos.
- Potenciar las conversaciones en persona.
- Producto funcional como medida de progreso.
- Satisfacer al cliente mediante entregas tempranas.

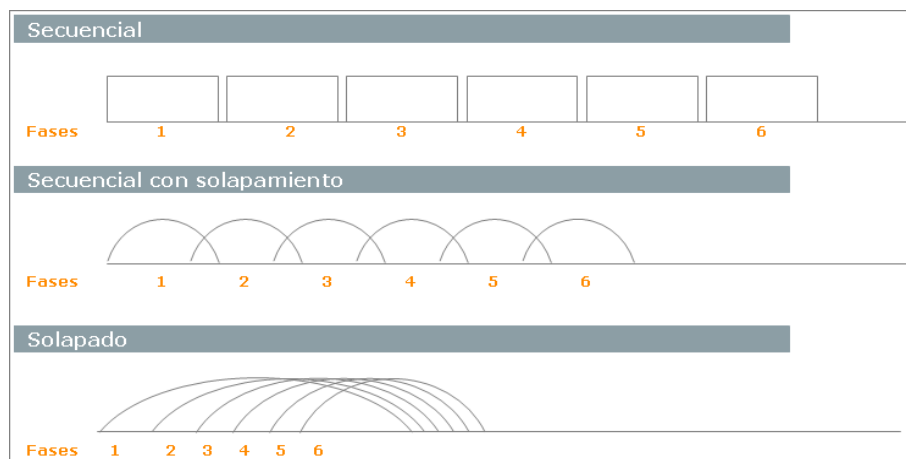


Figura 4.1: Ciclos de desarrollo.

4.2.2. Scrum

Existen muchas metodologías ágiles disponibles, como el método de desarrollo de sistemas dinámicos (DSDM), la programación extrema (XP), etc. Sin embargo, en este proyecto se hace uso de la metodología Scrum. Actualmente, Scrum es una de las metodologías más populares para la gestión de proyectos.

La razón principal de la elección de esta metodología es su flexibilidad a la hora de realizar las diferentes tareas del proyecto, pudiendo seleccionar aquellas tareas que más convienen hacer en un momento dado (las tareas de mayor importancia al principio, para presentar versiones utilizables lo antes posible). Además, en este proyecto existe cierta incertidumbre a la hora de establecer requisitos fijos de antemano, como se puede ver en la sección 4.1 en el riesgo **RPD-1**. Así, se desarrollan tareas en función de las necesidades cambiantes durante todo el proyecto y el solapamiento de fases ilustrado en la figura 4.1 evita que no se tenga que esperar a la finalización de una fase para poder comenzar otra (una de las desventajas de las metodologías pesadas). Otra razón por la que se eligió esta metodología es por la necesidad de ir presentando periódicamente versiones utilizables de la aplicación al cliente (los directores), para que éstos puedan empezar a utilizar la aplicación desde el principio e ir sugiriendo cambios y mejoras a realizar (característica de entornos con incertidumbre). Por ello, las reuniones de planificación y revisión de los sprints (incrementos) son tan importantes en este proyecto.

Scrum fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica en esa época. Nonaka y Takeuchi compararon la nueva forma de trabajo en equipo, con el avance en formación de scrum de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Entre las características que describen los entornos de desarrollo típicos en los que se puede aplicar la metodología de desarrollo ágil Scrum, destacan las siguientes:

- La incertidumbre: Es asumida en el desarrollo del proyecto, en donde se apunta cuál es la visión genérica que se quiere conseguir, o la dirección estratégica que hay que seguir,

pero no un plan detallado del producto y su desarrollo (para lo cual se da un margen de libertad).

- **Fases del desarrollo solapadas:** Las fases no existen como tal sino que se desarrollan tareas/actividades en función de las necesidades cambiantes durante todo el proyecto. De hecho, en muchas ocasiones no es posible realizar un diseño técnico detallado antes de empezar a desarrollar y ver algunos resultados. En la figura 4.1, podemos ver gráficamente en el tercer caso el solapamiento de fases, mientras que en el primer caso el retraso en una fase hace de cuello de botella en el proyecto. El solapamiento diluye el ruido y los problemas entre fases.
- **Control sutil:** El equipo trabaja con autonomía en un entorno de ambigüedad, inestabilidad y tensión. La gestión establece puntos de control suficientes para evitar que el ambiente de ambigüedad, inestabilidad y tensión del derive hacia descontrol, pero no ejerce un control rígido que impediría la creatividad y la espontaneidad.

En Scrum la gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. El desarrollo se inicia desde la visión general de producto, dando detalle solo a las funcionalidades que, por ser las de mayor prioridad para el negocio, se van a desarrollar en primer lugar, y pueden llevarse a cabo en un periodo de tiempo breve (entre 15 y 30 días). Cada uno de los ciclos de desarrollo es una iteración (sprint) que produce un incremento terminado y operativo del producto. Estas iteraciones son la base del desarrollo ágil, y Scrum gestiona su evolución a través de reuniones breves de seguimiento en las que todo el equipo revisa el trabajo realizado desde la reunión anterior y el previsto hasta la reunión siguiente.

Roles

- **Propietario del producto (*Product Owner*):** Representa la voz del cliente. Se asegura de que el equipo Scrum trabaje de forma adecuada desde la perspectiva del negocio. Tiene entre sus responsabilidades la financiación, la decisión para efectuar el lanzamiento, etc. También puede escribir historias de usuario, priorizarlas y colocarlas en el *Product Backlog* (descripciones genéricas de todos los requisitos priorizadas de mayor a menor importancia).
- **Scrum Master:** Responsable de garantizar que se aplica correctamente la metodología. Su trabajo principal es eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint.
- **Equipo desarrollo:** Tiene la responsabilidad de entregar el producto y capacidades para análisis, diseño, desarrollo, pruebas, documentación, etc.

Componentes básicos

- **Product Backlog:** Listado general de requisitos que evoluciona durante todo el desarrollo. Todos pueden contribuir y aportar sugerencias y están ordenados por prioridad según el cliente.
- **Sprint Backlog:** Lista de tareas a realizar en el sprint para generar el incremento. Se asignan los recursos y se estiman las unidades de tiempo para realizar cada tarea.

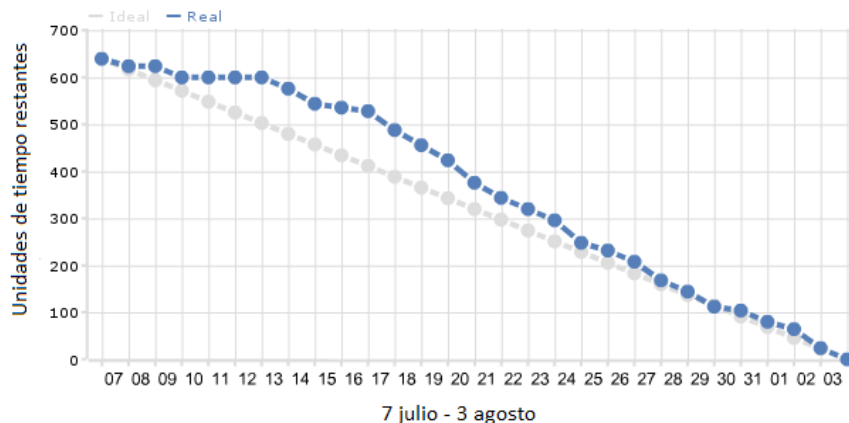


Figura 4.2: Ejemplo gráfico Burn-Down.

- **Incremento:** Es el resultado de cada sprint listo enseñar al cliente (está terminado y probado) y por tanto en condiciones de ser usado.

Reuniones

- **Planificación del sprint:** Jornada de trabajo previa al inicio de cada sprint en la que se determina cuál es el trabajo y los objetivos que se deben cubrir con esa iteración. Asiste el propietario del producto, el *Scrum Master* y el equipo de trabajo. Esta reunión genera el *Sprint Backlog*.
- **Monitorización del sprint:** Reunión diaria donde sólo interviene el equipo y en la que se hace un repaso al avance de cada tarea y al trabajo previsto para la jornada. La duración máxima debe ser de 15 minutos y se recomienda hacerla de pie con los elementos del Sprint Backlog en forma de post-it en una pizarra.
- **Revisión del sprint:** Análisis y revisión del incremento generado. Se presenta al cliente el incremento desarrollado (terminado, probado y operando en el entorno del cliente). Se puede obtener feedback para mejorar e incorporar en sucesivos sprints.

Gráfico *Burn-Down* Herramienta para gestionar y seguir el trabajo de cada sprint, donde se representa gráficamente del avance del sprint. En la figura 4.2 podemos ver este gráfico a modo de ilustración.

Las **modificaciones** realizas sobre la metodología Scrum para realizar este proyecto son dos:

- Los roles de *Product Owner* y *Scrum Master* en este proyecto son los mismos, es decir, corresponden a los directores del proyecto, que hacen de voz representativa del cliente y garantizan que se aplique correctamente la metodología. Además, el equipo de desarrollo está representado por una persona.
- No se realizan las reuniones de monitorización o seguimiento del sprint, ya que en este caso no es necesario realizar un seguimiento tan exhaustivo en el desarrollo y por lo que sólo se

llevan a cabo cuando existe algún impedimento para continuar con el sprint. Sí se realizan las reuniones de planificación y de revisión del sprint.

Como herramienta de gestión de proyectos, se ha utilizado Acunote [11]. Se trata de una herramienta de gestión de proyectos ágiles y software Scrum. Es utilizada ampliamente en el mundo empresarial como parte de la gestión de proyecto TIC. Entre las razones de su utilización cabe destacar su sencillez y facilidad de uso. Asimismo, permite mostrar el progreso actual y proporciona potentes analíticas, como el gráfico Burn-Down, del que hemos hablado anteriormente.

4.3. Gestión de la configuración

La configuración del software es el conjunto de características funcionales y físicas del software detalladas en la documentación técnica o alcanzadas en un producto. (IEEE610.12-90). La gestión de la configuración, por su parte, es un proceso cuyo propósito es establecer y mantener la integridad de los elementos de trabajo. Para ello, se identifican los elementos y se controlan los cambios de éstos a lo largo de su ciclo de vida, registrando el estado y verificando que estén completos y que sean los correctos.

Para llevar a cabo la gestión de la configuración en este proyecto, se utiliza un sistema de control de versiones mediante la herramienta GitLab que ofrece el CiTIUS [18] y que facilita la gestión de repositorios. Esta herramienta, se basa en GIT, que es un SCV (sistema de control de versiones) multiplataforma y distribuido, donde hay un repositorio central con el cual se puede sincronizar todo el mundo. Se sitúa en una máquina en concreto y es el repositorio que contiene todo el histórico, etiquetas y ramas. La razón de utilizar esta herramienta es su velocidad, su diseño sencillo y la posibilidad de visualizar gráficamente la interacción (gráficas de modificaciones, etc.) con el repositorio.

En este proyecto los elementos de configuración implican tanto los archivos de desarrollo como los archivos de documentación:

- Módulo de Python de los test estadísticos (paramétricos y no paramétricos).
- API REST.
- Archivos de la web: archivos HTML, CSS, JavaScript, imágenes, etc.
- Memoria.
- Documentación del módulo de Python de los test estadísticos.

El proceso de gestión de configuración es el siguiente:

1. Se dispone de una rama principal denominada *master* que hace la labor de **línea base** (donde se encuentran los archivos de los sprints finalizados, es decir, los archivos con los cambios del último sprint).
2. Para mantener organizada y de forma accesible la información, cada sprint se realiza en una rama de desarrollo separada de la rama *master*. Cuando se finaliza el sprint se fusiona en la rama principal.

- La nomenclatura de las ramas será la siguiente: “sprintX”, donde X representa el número de sprints actual.
 - En la rama de desarrollo del sprint se van realizando los cambios a los archivos, indicando con una breve descripción los cambios que se hacen.
3. Al finalizar cada sprint, también se crea una etiqueta o *tag* de la rama de desarrollo para tener disponibles todas las versiones de la aplicación (todos los sprints) en un archivo comprimido.
- La nomenclatura de las etiquetas será la siguiente: “v0.X”, donde X representa el número de sprints actual.

4.4. Planificación temporal

La planificación temporal consiste en la identificación de tareas, asignación de tiempos y recursos a dichas tareas y en la planificación de la secuencia de ejecución de forma que el proyecto pueda finalizarse a tiempo.

El trabajo del presente proyecto se divide en varias fases, especificadas previamente en la estructura de descomposición del trabajo del anteproyecto, que se puede ver en la figura 4.3. El EDT consiste en una representación jerárquica del trabajo a realizar, con el fin de organizar y definir el alcance del proyecto. Debido al carácter iterativo de la metodología Scrum, las fases que a continuación se detallan se organizarán de una manera no secuencial (recordemos que las fases en esta metodología se solapan, como se vio en la sección 4.2).

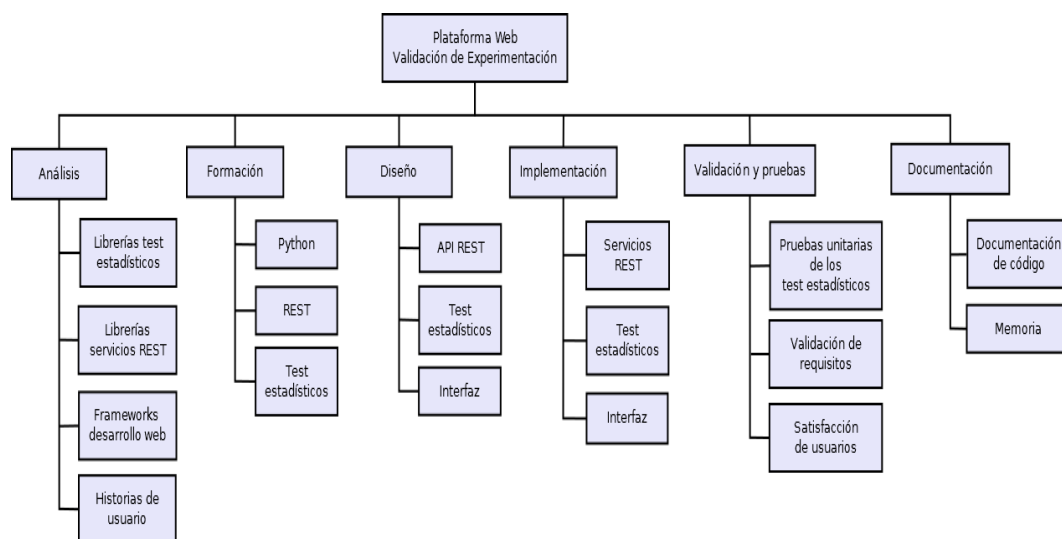


Figura 4.3: Estructura de descomposición del trabajo (EDT).

A continuación se describen las diferentes etapas de una forma más detallada, donde se indica también el tipo de recurso que realizada cada etapa y la duración estimada de las mismas. Las estimaciones son iniciales y no estimaciones realizadas al inicio de cada sprint, las cuales son

mucho más cercanas a la realidad, y con las actividades más desglosadas. La dedicación será de 20 horas semanales. En total, la duración estimada del proyecto es de aproximadamente 21 semanas de duración, lo que supone aproximadamente 425 horas de trabajo.

Etapas de Análisis

En esta etapa se estudiará el problema a resolver y se realizará un análisis de los requisitos del proyecto mediante la técnica de desarrollo ágil vista en el capítulo 3: las historias de usuario, que serán tratadas y discutidas con el rol del Product Owner y del Scrum Master, es decir, los directores del proyecto. También se dedicará parte del tiempo a estudiar las librerías (test estadísticos, servicios web REST), frameworks para el desarrollo web (parte visual de la plataforma) y herramientas existentes en la actualidad (IDEs, documentación, etc.) que puedan facilitar la labor de desarrollo.

Tiempo estimado
1 semana

Etapas de Formación

Con la formación se tratará de adquirir los conocimientos necesarios en el lenguaje Python, incluyendo la lectura de manuales, tutoriales, etc. También será objeto de estudio la tecnología de los servicios web REST, que tienen un enfoque diferente al tradicional (servicios web SOAP). Aquí se realizarán por tanto diferentes lecturas sobre servicios web basados en REST para comprender esta tecnología. Asimismo, la etapa de formación también incluye el estudio del contraste de hipótesis (test estadísticos), ya que el proyecto en sí está dedicado a este propósito. Se realizará un estudio de cada test, para comprender su funcionamiento de cara a su implementación en las fases de codificación.

Tiempo estimado
1 semana

Además del tiempo estimado para esta etapa donde se dedicará el 100 % del tiempo asignado, esta etapa también se realizará en paralelo al resto de etapas del proyecto hasta el momento en que la parte de la implementación de los test estadísticos y el API REST esté completa. Esto es debido a que en muchas ocasiones resulta difícil tener los conceptos claros o saber qué es lo que se necesita hasta que no se empieza el diseño o la implementación. Sin embargo, la dedicación cuando se realiza en paralelo a otras etapas es mucho menor (en torno al 15 %).

Etapas de Diseño

En esta fase se definirá en primer lugar la arquitectura del proyecto, para luego realizar el diseño de los test estadísticos, así como de la API REST y la interfaz web, para la cual se diseñará un prototipo inicial conocido como *storyboard*, que es conjunto de ilustraciones con el objetivo de mostrar gráficamente las partes de las que se compone la interfaz. Dada la característica iterativa de la metodología elegida, el diseño también se irá refinando a medida que avance el proyecto.

Tiempo estimado
3 semanas

Etapas de Implementación

Esta es la fase que requiere un mayor tiempo en proporción a las demás. Durante esta etapa se realizará la implementación de los test estadísticos y de la API REST, así como de la interfaz web de usuario, lo que constituye un importante espacio de tiempo.

Tiempo estimado
12 semanas

Etapas de Validación y pruebas

La etapa de validación y pruebas incluye la realización de pruebas unitarias para comprobar el correcto funcionamiento de cada uno de los test estadísticos, así como la validación de los requisitos acordados y la elaboración de una encuesta de satisfacción de usuarios para obtener el feedback por parte de los usuarios que empiecen a utilizar la plataforma.

Tiempo estimado
2 semanas

Etapas de Documentación

En esta fase se incluye la documentación relativa al desarrollo y gestión del proyecto, la realización de la memoria del proyecto y los manuales técnicos y de usuario.

Tiempo estimado
2 semanas

Además del tiempo estimado para esta etapa donde se dedicará el 100 % del tiempo asignado, esta fase se realizará en paralelo al resto de fases del proyecto (con un 15 % de tiempo dedicado), ya que durante todo el ciclo de vida del proyecto se está realizando la documentación del código, además de algunas partes de la memoria.

A continuación en la figura 4.4 se puede ver el cronograma de la planificación de las distintas fases del proyecto. En la imagen se puede ver el número de sprints llevados a cabo así como cada una de las fases involucradas en cada sprint y el tiempo empleado:

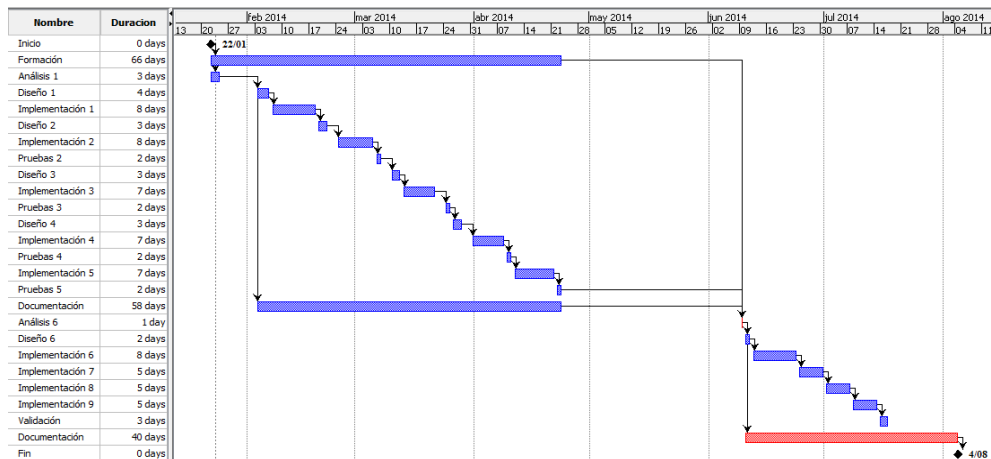


Figura 4.4: Cronograma lineal de la planificación de las fases.

En las figuras 4.5, 4.6 y 4.7 podemos los gráficos Burn-Down correspondientes a los 9 sprints realizados. A modo de resumen, en el primero de ellos, se realizó un caso de prueba con el test de Wilcoxon. En el segundo, fue donde se llevó a cabo el desarrollo de los test no paramétricos de ranking, además de los servicios web correspondientes de consulta y subida de ficheros. El sprint 3 incluye mejoras del anterior y el desarrollo de los test POST-HOC. El sprint 4, incluye mejoras del sprint anterior y el desarrollo de los test paramétricos.

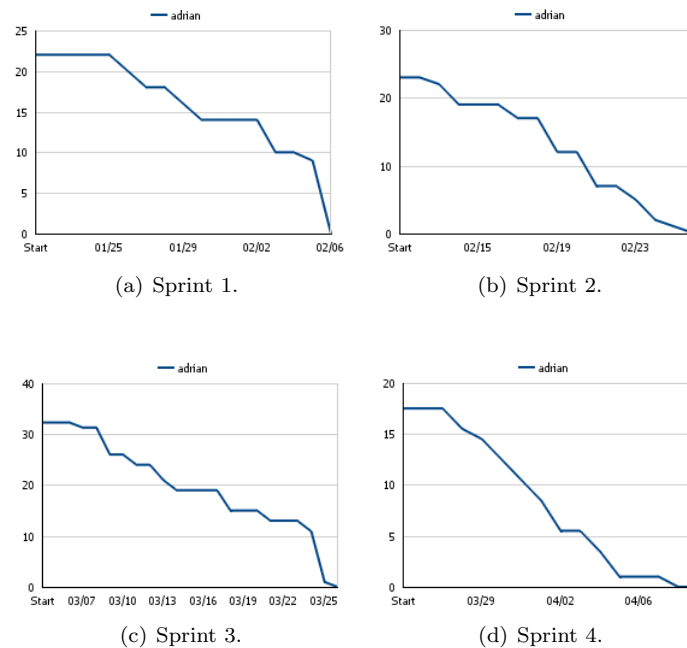


Figura 4.5: Burn-Down Sprint 1, 2, 3 y 4.

El sprint 5, incluye corrección de errores, mejoras del anterior sprint y algunos POST-HOC

de comparación múltiple. En el sprint 6, se empieza a desarrollar la interfaz web, además de alguna mejora del anterior sprint. Los sprints 7, 8 y 9 son básicamente mejoras realizadas sobre la interfaz y la exportación de resultados (formato \LaTeX y CSV), centrándose un poco más en la elaboración de la memoria.

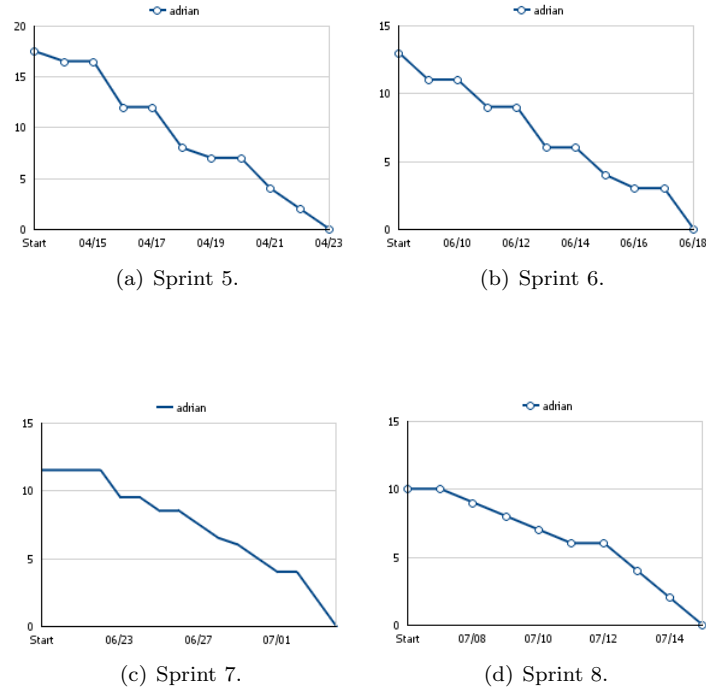


Figura 4.6: Burn-Down Sprint 5, 6, 7 y 8.

Cabe destacar que pese a que tuvo lugar la ocurrencia del riesgo **RPD-3**, y por lo tanto del riesgo **RPD-1** debido a que el test de Li para comparaciones múltiples no funcionaba correctamente y se tuvo que implementar en su lugar el test de Finner, esto no llevó a un retraso respecto al tiempo estimado.

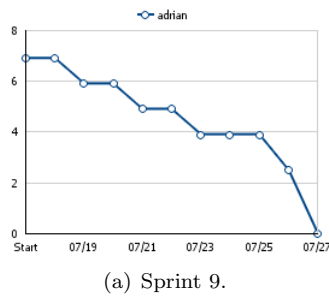


Figura 4.7: Burn-Down Sprint 9.

4.5. Estimación de costes

Los costes se han dividido en tres categorías: coste de personal, costes hardware y los costes software.

Coste de personal

El coste del personal establecido es fruto de la investigación de salarios (brutos) de un analista programador en la Comunidad de Galicia [12], así como del examen de costes de otros proyectos de fin de grado desarrollados. En este apartado habría que incluir también el coste de personal asociado al rol de dirección. Sin embargo, por cuestiones de simplicidad este análisis sólo considera el coste asociado al analista programador.

Horas Trabajadas	Coste/Hora	Coste Total
428	13 €	5564 €

Costes hardware

Estos costes se refieren al material adquirido para poder realizar el proyecto. Teniendo en cuenta que la vida útil de una estación de trabajo es de 5 años y que las horas de utilización cada año son 1700, si el precio del equipo es de 800 €, el coste total amortizado durante el proyecto suma un total de 40.28 €.

Componente Hardware	Coste
Ordenador HP	40.28 €

Costes software

Durante el desarrollo del proyecto se utilizó únicamente software gratuito para reducir el coste.

Componente Software	Coste	Componente Software	Coste
Ubuntu 12.04	0 €	ProjectLibre 1.5.9	0 €
Spyder 2	0 €	StarUML 5.0	0 €
Bottle 0.12	0 €	Dia 0.97.2	0 €
Twitter Bootstrap 3.1.1	0 €	Lumzy	0 €
Sphinx 1.2.2	0 €	Librerías	0 €
TeXstudio 2.8.2	0 €	Apache 2.4	0 €

Coste total

Si se asumen gastos indirectos de luz, conexión a Internet, etc. durante el desarrollo (en la USC para proyectos TIC se estima en un 20 % adicional del coste total del proyecto), el coste final es el siguiente:

$$personal + hardware + software + indirecto = 5564 + 40,28 + 0 + 1120,86 = 6725,14 \text{ Euros.}$$

Capítulo 5

Arquitectura y herramientas

El proceso de diseñar es definido por el IEEE como el esfuerzo para definir la arquitectura, componentes, interfaces y otras características de un sistema o componente. También se podría definir como la etapa del ciclo de vida del software en la cual se produce una descripción de la estructura interna del software que sirve de base para su construcción. El estándar ISO 12207 identifica un diseño a alto nivel o arquitectónico y un diseño detallado. El primero describe la estructura y organización, es decir, los subsistemas o componentes y sus relaciones. El segundo, describe cada componente y su comportamiento específico, para proceder a su construcción.

Después de tener una idea formada de los objetivos y requisitos del presente proyecto, en este capítulo nos centraremos en el diseño de alto nivel o la arquitectura, donde además de utilizar representaciones gráficas a alto nivel del sistema se detallarán las tecnologías y elementos arquitectónicos básicos del mismo, así como las herramientas utilizadas para llevar a cabo el proyecto.

5.1. Arquitectura del sistema

Analizando los requisitos y los objetivos del proyecto se identifican tres partes críticas en el sistema: la interfaz gráfica de usuario (interfaz web), el API de los servicios web de los test estadísticos y el módulo STAC (*Statistical Tests for Algorithm Comparison*), donde están incluidos los propios test estadísticos (tanto los test paramétricos como los test no paramétricos). En la figura 5.1 podemos ver la arquitectura de la plataforma web:

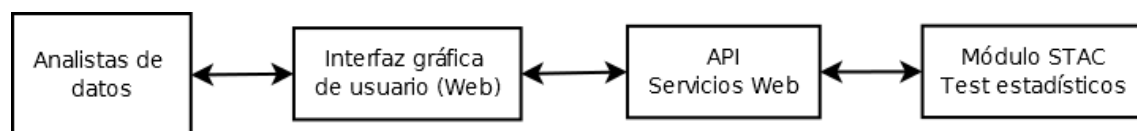


Figura 5.1: Arquitectura de la plataforma web.

Como podemos ver, los usuarios están representados como analistas de datos, y hacen uso

de la interfaz gráfica de usuario (interfaz web), donde pueden realizar tareas tales como subir sus resultados y visualizarlos para posteriormente poder aplicar sobre ellos alguno de los test disponibles en el sistema. Esta interfaz representa la cara visible hacia el usuario, y por tanto será lo más usable posible para facilitar la tarea de validación.

La interfaz gráfica a su vez se comunica con el API de los servicios web, que son los encargados de ejecutar los test estadísticos con los parámetros u opciones elegidas por el analista en la interfaz (nivel de significancia, etc.), y devolver los datos obtenidos por el test a la interfaz para que ésta represente los datos de la forma más cómoda posible.

Por último, la parte más importante de la plataforma está representada por el módulo STAC, compuesto tanto por test paramétricos como no paramétricos. Aquí el usuario dispone de varios test, entre los que se encuentran por ejemplo aquellos que verifican si se cumplen las condiciones de homocedasticidad o normalidad para determinar si se pueden aplicar con seguridad (fiabilidad) test paramétricos.

Con el objetivo de desacoplar los datos y la lógica de negocio (modelo) de la interfaz de usuario (vista) y el módulo encargado de gestionar los eventos (controlador), se ha aplicado el patrón MVC (Modelo-vista-controlador) para construir la plataforma web. El concepto de patrón de diseño se explicará en más profundidad en secciones posteriores. En la figura 5.2 se muestra una representación gráfica de esta arquitectura, que ha sido en la que se ha basado este proyecto y en la que se distinguen las siguientes partes:

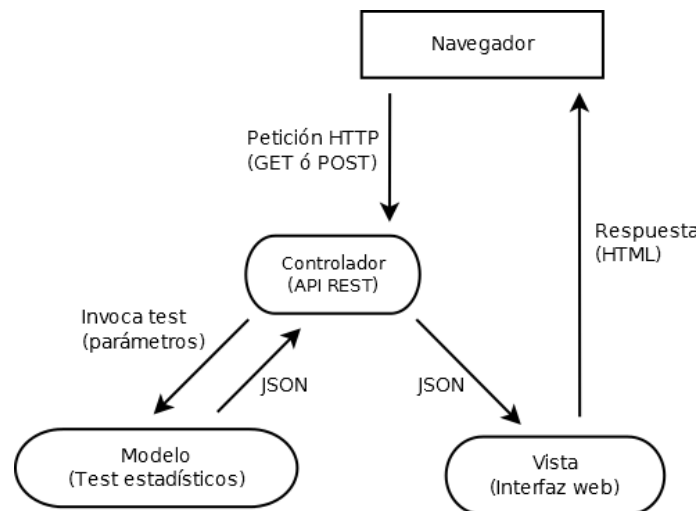


Figura 5.2: Patrón MVC implementado.

- El controlador se encarga de recibir las peticiones HTTP (GET o POST) del navegador y decide qué test estadístico quiere invocar (para lo cual realizará una llamada con ciertos parámetros). La subida y visualización de ficheros de datos son tratados por el propio controlador. Éste una vez obtenga los datos se los devolverá a la vista (en formato JSON). En este proyecto el controlador está representado por el API de servicios web REST.
- Las vistas reciben los datos del modelo a través del controlador y se los muestran a los usuarios (en formato HTML), en ellas únicamente se realizan operaciones simples. Representa

la interfaz gráfica de usuario (Web).

- El modelo es el responsable de realizar una gran parte de las funcionalidades del sistema, ya que en él se lleva a cabo la realización de las distintas pruebas de hipótesis. Esta parte representaría al módulo STAC, en donde se localizan los test paramétricos y no paramétricos.

Las tecnologías mencionadas (HTTP, REST, JSON) se detallan en secciones posteriores dentro del presente capítulo.

5.2. Herramientas de diseño

En este proyecto se utiliza la programación modular como paradigma de programación. La programación modular consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable. Un módulo es cada una de las partes de un programa que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. La razón de realizarlo de esta forma y no orientado a objetos es debido a que los test estadísticos implementados son funciones desde un punto de vista más matemático (aplicación de un proceso a unas entradas para obtener una salida), que no contienen estado, y por lo tanto se adaptan más a una programación más procedural o funcional. Para el caso de la API REST, el framework elegido que se verá en secciones posteriores hace también que los servicios web tengan una representación en forma de función, con lo que sería añadir una complejidad extra al proyecto innecesaria.

Para poder modelar el software de un modo formal se utilizará UML (*Unified Modeling Language*). UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Permite especificar diferentes ámbitos del sistema desde la lógica de negocio hasta la estructura hardware, sin ligarse a ningún lenguaje de desarrollo en particular. Para ello, hace uso de modelos que representan el sistema desde un punto de vista específico.

UML proporciona 13 tipos de diagramas, que se dividen en tres categorías: estructura, comportamiento e interacción. Pese a que UML está por lo general ligado a la programación orientada a objetos, en este proyecto se utilizará para realizar un diagrama de la categoría de interacción, que servirá para definir acciones que se pueden realizar en la plataforma:

- **Diagramas de secuencia:** indican los componentes que forman parte del sistema y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. Los mensajes o llamadas intercambiados están ordenados temporalmente (en secuencia).

5.2.1. Herramientas

Para poder realizar los diagramas UML y el prototipo de la interfaz gráfica del usuario, se han utilizado las siguientes herramientas:

- **StarUML:** herramienta que soporta la mayoría de los tipos de diagramas especificados en UML 2.0 y que servirá para realizar los diagramas en este proyecto.
- **Lumzy:** herramienta de creación de prototipos para sitios web y aplicaciones.

5.2.2. Patrones de diseño

Un patrón es una solución a un problema en un contexto particular. Es recurrente (lo que hace la solución relevante a otras situaciones), enseña (permite entender cómo adaptarlo a la variante particular del problema donde se quiere aplicar) y tiene un nombre para referirse al patrón. Los patrones facilitan la reutilización de diseños y arquitecturas software que han tenido éxito. Existen tres categorías de patrones de diseño:

- **Patrones de creación:** tratan de la inicialización y configuración de clases y objetos.
- **Patrones estructurales:** tratan de desacoplar interfaz e implementación de clases y objetos.
- **Patrones de comportamiento:** tratan de las interacciones dinámicas entre sociedades de clases y objetos.

Debido a que los patrones de diseño están fuertemente ligados al paradigma orientado a objetos (clases y objetos), en este proyecto se utilizan solamente dos patrones:

- **El patrón MVC:** visto en la sección 5.1, encaja dentro de los denominados patrones estructurales, y sirve para desacoplar interfaz e implementación
- **El patrón *Singleton* (Fig. 5.3):** para la implementación de la historia de usuario **HU-5** vista en la sección 3.2.1. Este es el único punto del proyecto donde se utiliza programación orientada a objetos. La razón de utilizar una clase para implementar la **HU-5** en este caso se facilita la programación de esta característica del proyecto.

Patrón Singleton

Este patrón está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella (Fig. 5.3). El patrón *singleton* se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

Singleton	
-	<u>singleton : Singleton</u>
-	Singleton()
+	<u>getInstance() : Singleton</u>

Figura 5.3: Patrón *Singleton*.

5.3. Herramientas de desarrollo

5.3.1. Análisis de librerías de test estadísticos en Python

Librería SciPy - *Scientific Computing Tools for Python*

SciPy [13] cuenta con una colección denominada *The SciPy Stack*, que consta de un conjunto de paquetes principales de software de código abierto para la computación científica en Python, donde la comunidad puede utilizar y desarrollar esta colección. Entre los paquetes básicos de esta colección se encuentran:

- **NumPy:** paquete fundamental para computación numérica, donde se puede encontrar álgebra lineal, operaciones estadísticas básicas (como la media, la varianza, ...), aleatoriedad, etc.
- **Librería SciPy:** una colección de algoritmos numéricos y herramientas específicas de dominio, incluyendo el procesamiento de señales, la optimización, estadística, etc.

SciPy y NumPy tienen versiones de documentación en formato HTML y PDF [14], que cubren casi toda la funcionalidad disponible. La librería SciPy es uno de los paquetes principales que componen *The SciPy Stack*. Proporciona muchas rutinas numéricas fáciles de usar y eficientes, como las rutinas de integración numérica y optimización. SciPy está organizada en subpaquetes o módulos que cubren diferentes dominios de computación científica. Para este proyecto, el subpaquete que nos interesa es *stats*, que consta de distribuciones y funciones estadísticas.

El subpaquete *stats* contiene un gran número de distribuciones de probabilidad (tanto continuas como discretas). Las continuas (como la distribución χ^2 ó \mathcal{T} vistas en la sección 2.2) son las que nos interesan. Estas distribuciones continuas definidas como clases constan de métodos comunes, como son la función de densidad de probabilidad o la función de distribución acumulada para una distribución determinada, de las que ya hemos hablado en la figura 2.5. Estas funciones sirven de utilidad para realizar los test estadísticos y obtener valores necesarios. Asimismo, este módulo proporciona también una biblioteca de test estadísticos. Entre estos test están aquellos para hallar las condiciones paramétricas de normalidad (Shapiro-Wilk, D'Agostino-Pearson y Kolmogorov-Smirnov), y homocedasticidad (Levene), así como la prueba \mathcal{T} de Student (vistos en la sección 2.7 del capítulo 2).

Por tanto, para este proyecto nos serviremos de los test de las condiciones paramétricas, así como de la prueba \mathcal{T} de Student incluidos en esta librería, ya que forman parte de los objetivos del proyecto. Sin embargo, esta librería no cubre todas las necesidades de test detalladas en los objetivos para el proyecto.

Librería *nonparametric.py*

Este módulo, creado por Ismael Rodríguez, contiene algunos de los test marcados como objetivos en este proyecto: el test de Wilcoxon, el test de Friedman, el test de Iman-Davenport, el test de Bonferroni-Dunn, el test de Holm y el test Hochberg. Sin embargo, no se ha probado su correcto funcionamiento y por tanto algunos de ellos necesitan ser rediseñados e implementados de nuevo. Para ello será necesario:

- Probar que funcionan correctamente mediante la realización de test unitarios y corregir los fallos encontrados.
- Comprobar que devuelven los datos necesarios: rankings, estadísticos, *p-valor* (en caso de los test de ranking); y los parámetros usados para cada comparación, *p-valores* ajustados (en el caso de los test de comparación).
- Hacer que reciban y devuelvan los datos de la misma manera.

Este módulo tiene dependencias con la librería SciPy (con el módulo *stats*), así como con el paquete NumPy. Como se ha dicho anteriormente, la funcionalidad de SciPy sirve a este proyecto para poder hallar parámetros básicos que serán necesarios para obtener todos los datos necesarios en los test.

Dado que será necesario corregir y volver a implementar muchas de las funcionalidades presentes en este módulo, y debido a que muchas otras funcionalidades aún quedan por implementar, este módulo tampoco cumple con las expectativas del proyecto. La extensión y corrección de este módulo permitirá la obtención de un módulo acorde a los objetivos del proyecto con todos los test disponibles.

5.3.2. Análisis de librerías para servicios REST en Python

Servicios REST

Los test estadísticos, así como la subida y consulta de ficheros serán accesibles vía web mediante servicios web en Python y basados en REST. La Transferencia de Estado Representacional (*Representational State Transfer*) o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo. Aunque REST no es un estándar, está basado en los estándares: HTTP, URL, representación de los recursos (XML, HTML, GIF, JPEG, JSON ...), y tipos MIME (text/xml, text/html, application/json, ...)

Por ejemplo, en la figura 5.4 se puede ver como ejemplo una petición para consultar el contenido de un archivo. Para ello, la respuesta indica en la cabecera con “Content-Type: application/json”, el tipo de dato a devolver, por lo que el cliente sabe que el contenido de la respuesta es una cadena en formato JSON, y puede procesarla como prefiera. Los servicios web REST están basados en los siguientes principios:

- **Utilización de métodos HTTP:** por tanto, los métodos a utilizar para la comunicación con los servicios web de los test podrán ser los comunes de HTTP: GET, POST, PUT, DELETE.
- **Servicios sin estado:** las peticiones a los servicios web de los test incluirán todos los datos necesarios (petición completa e independiente) para que el servidor no tenga que mantener ningún estado para procesar la petición. En otras palabras, el servidor no puede almacenar información proporcionada por el cliente en una solicitud y usarlo en otra solicitud.

- **Exposición de URIs (identificador recursos uniforme) con forma de directorios:** por ejemplo, se podrían hacer URIs del estilo: `http://localhost/api/nombre_test/...`

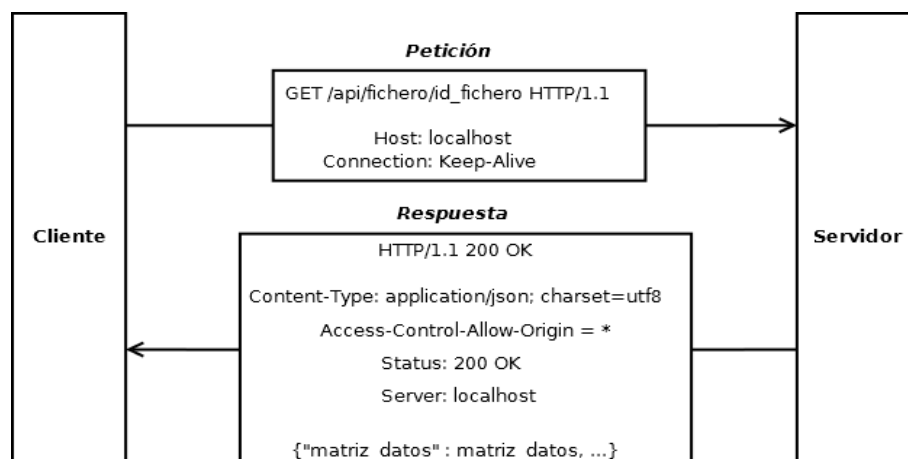


Figura 5.4: Servicios web REST.

Cabe destacar que para el presente proyecto se utilizarán los métodos HTTP GET y POST, el primero para obtener información de un recurso (esto es, obtener los resultados de los test, así como el contenido del fichero) y el segundo para crear un nuevo recurso (para la creación de un nuevo fichero). Asimismo, los recursos estarán representados por el tipo MIME “application/json”. La figura 5.4 muestra estas características.

Librería Bottle

Para desarrollar los servicios web que diesen acceso a los test estadísticos, así como a la gestión del fichero, en el anteproyecto se había barajado la utilización de alguno de los siguientes frameworks: Flask, Web.py y Bottle. El framework elegido finalmente fue Bottle [15]. Este framework web es conocido más bien como un micro framework, y las principales razones que propiciaron su elección frente a las demás es que se trata de un framework rápido, sencillo y ligero para Python. Se distribuye como un módulo y está formado por un único archivo y no tiene dependencias distintas de la biblioteca estándar de Python. Además, dispone de las siguientes características:

- **Enrutamiento:** mapeo de solicitudes de llamadas a función con soporte para URLs limpias y dinámicas.
- **Plantillas:** motor de plantillas integrado.
- **Utilidades:** la facilidad de acceso a datos de formulario, subida de ficheros, cookies, cabeceras y otros metadatos relacionados con HTTP.
- **Servidor:** servidor HTTP propio para desarrollo integrado y soporte para cualquier servidor HTTP compatible con la especificación WSGI (interfaz entre servidores web y aplicaciones web o frameworks para el lenguaje Python).

Bottle permite la implementación de servicios web accesibles mediante una o más URIs. El decorador *route()* asigna una URI a un trozo de código (el propio servicio) en lo que se denomina ruta. El decorador decora a lo que realmente referencia, que es un trozo de código. Cada vez que el navegador llama a una URL especificada con *route()* la función o servicio asociado es llamado y el valor de retorno se envía al navegador. A un mismo servicio se pueden unir tantas URIs como se desean. El siguiente código muestra cómo de una forma sencilla se puede implementar un servicio REST:

```
@route('/fichero/<id_fichero>', method='GET')
def consultar_fichero(id_fichero):
    response.headers['Access-Control-Allow-Origin'] = '*'
    response.content_type = "application/json"
    ...
    return datos
```

Como se puede ver en ejemplo, por medio de *@route* se indica la URL. El método utilizado en este caso es GET y la respuesta será dada en formato JSON. Con *id_fichero*, se indica que se va a pasar el identificador del fichero como parte de la URL.

5.3.3. Análisis de frameworks para desarrollo web

Twitter Bootstrap

El framework para el desarrollo de la interfaz web utilizado en la plataforma es Twitter Bootstrap [16]. Bootstrap fue desarrollado por Mark Otto y Jacob Thornton de Twitter, como solución interna para solucionar las inconsistencias en el desarrollo dentro del equipo de ingeniería de Twitter, ya que hasta ese momento no había establecida ninguna convención sobre las formas en las que los ingenieros desarrollaban la plataforma. En agosto del 2011, Twitter liberó a Bootstrap como código abierto. En febrero del 2012, se convirtió en el proyecto de desarrollo más popular de GitHub [17] y es utilizado por organizaciones como la NASA.

Se trata de un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web (*Front-end* o interfaz). Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML5 y CSS3, así como, extensiones de JavaScript opcionales adicionales. Una de sus principales características es que utiliza LESS, que es una ampliación de las hojas de estilo CSS, pero a diferencia de éstas, funciona como un lenguaje de programación, permitiendo el uso de variables, funciones, operaciones aritméticas, entre otras, para acelerar y enriquecer los estilos en un sitio web.

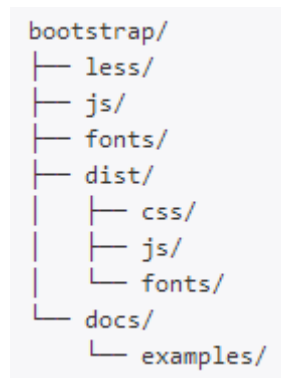


Figura 5.5: Organización Twitter Bootstrap.

La figura 5.5 muestra la organización de directorios de Bootstrap. Los directorios `less/`, `js/` y `fonts/` contienen el código fuente utilizado para generar los archivos CSS, JavaScript y las fuentes. El directorio `dist/` contiene los mismos archivos que se han mostrado en la sección anterior para la versión compilada de Bootstrap. El directorio `docs/` incluye el código fuente de la documentación de Bootstrap y el directorio `examples/`, que contiene varios ejemplos de muestra. El resto de archivos incluidos proporcionan información de licencia y desarrollo.

Las razones de la utilización de este framework se basan en la sencillez y en la amplia documentación existente. Sin embargo, la razón principal es la capacidad para diseño *responsive* o fluido (*responsive design*). Mediante el diseño responsivo, se consigue que la web se adapte según la resolución del dispositivo o ventana del navegador. Esto es necesario para poder desarrollar de un modo fácil y adecuado la historia de usuario **HU-5** vista en la sección 3.2.2 del capítulo 3. Para este propósito, Bootstrap cuenta con un diseño de páginas basado en rejillas, que están formadas por de filas y columnas donde se colocan los contenidos. Las rejillas crecen hasta 12 columnas a medida que crece el tamaño de la pantalla del dispositivo.

5.3.4. Otras herramientas

Desarrollo

Con el objetivo de disminuir los costes del proyecto, y teniendo en cuenta el uso de herramientas que requieran el menor tiempo de aprendizaje, el entorno de trabajo seleccionado para el desarrollo del proyecto fue el siguiente:

- **Sistema operativo:** Ubuntu 12.04, basado GNU/Linux.
- **IDE de programación:** Spyder 2, entorno multiplataforma de código abierto para la programación científica en el lenguaje Python. La razón de su utilización es que Spyder integra NumPy y SciPy, entre otras librerías.

Documentación

Para el desarrollo de la documentación se han utilizado las siguientes herramientas:

- **Sphinx 1.2.2:** Sphinx se trata de una herramienta de código abierto para generación de documentación de código Python. Es utilizada para documentar el módulo de Python STAC perteneciente a los test estadísticos. La razón de su utilización es que permite la generación de documentación en formato HTML para poder acceder a ella desde el navegador. Además, permite la escritura de fórmulas mediante notación $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, lo cual es importante en este proyecto.
- **TeXstudio 2.8.2:** IDE de código abierto de $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ que proporciona un soporte moderno de escritura, corrección ortográfica interactiva, plegado de código y resaltado de sintaxis. Es utilizado para la realización de la documentación.
- **ProjectLibre 1.5.9:** herramienta multiplataforma de código abierto utilizada para la elaboración de diagramas de Gantt.
- **Dia 0.97.2:** herramienta multiplataforma de código abierto utilizada para la elaboración de diagramas en general. Es utilizada en este proyecto para elaborar el EDT, así como diagramas genéricos de arquitectura, etc.

Tecnologías

A parte de las tecnologías anteriormente mencionadas, el desarrollo del proyecto también está marcado por la utilización de las siguientes otras:

- **HTML:** *HyperText Markup Language*, hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que, en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc.
- **JavaScript:** es un lenguaje de programación interpretado. Se utiliza en páginas web HTML para realizar operaciones en el marco de la aplicación cliente, sin acceso a funciones del servidor.
- **CSS:** *Cascading style sheets* u hojas de estilo en cascada es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML.
- **Apache y el módulo WSGI:** a pesar de que el framework Bottle dispone de su propio servidor HTTP para desarrollo integrado, y aprovechando que soporta cualquier servidor HTTP compatible con la especificación WSGI, en este proyecto se optó por la utilización del servidor web Apache. La principal razón de realizarlo de este modo es que el tiempo de respuesta del servidor de Bottle era de 5 segundos cuando se accedía desde otra red, mientras que utilizando Apache se llega a bajar a 2-3 milisegundos. Además, con esto se consigue hacer disponible la API REST por el puerto 80.
 - **Apache** es un servidor web HTTP de código abierto, que implementa el protocolo HTTP/1.12. Desde 1996, Apache es el servidor HTTP más usado. Alcanzó su máxima

cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en Internet.

- **WSGI** es una interfaz o especificación entre servidores web y aplicaciones web o frameworks para el lenguaje Python. El módulo WSGI de Apache implementa la interfaz WSGI, lo cual permite servir las aplicaciones Python (nuestro caso el API REST).

- **AJAX:** *Asynchronous JavaScript + XML*, está formada varias por tecnologías que permiten realizar peticiones HTTP y modificaciones en las páginas de forma asíncrona sin necesidad de recargar la página. Entre los beneficios del uso de AJAX destaca la reducción del tráfico con el servidor. Es utilizada en este proyecto para poder llamar a los servicios REST y gestionar los datos devueltos por éstos (interacción con las páginas HTML).
- **JSON:** *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. Es utilizado por AJAX y los servicios REST. La razón de la utilización de este formato es que puede ser interpretado por cualquier lenguaje (incluido Python). En la figura 5.4 podemos ver un ejemplo de formato JSON en la respuesta a la petición.

Librerías

En la siguiente tabla se muestran las distintas librerías de JavaScript utilizadas en el desarrollo del proyecto:

Librería	Versión	Página Oficial
jQuery	1.11.0	http://jquery.com/
MathJax	2.4	http://www.mathjax.org/

- **jQuery:** es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es utilizada en este proyecto para trabajar junto con el framework Bootstrap y para realizar las llamadas AJAX a los servicios web.
- **MathJax:** es un motor de visualización de código libre de JavaScript que permite visualizar fórmulas matemáticas en navegadores web, utilizando diferentes lenguajes de marcado (entre ellos \LaTeX). MathJax tiene licencia libre y funciona en todos los navegadores modernos. La razón de su utilización en el proyecto es que en alguna ocasión se necesita escribir fórmulas en documentos HTML, y esta librería constituye una manera fácil de llevar a cabo esta tarea.

Capítulo 6

Diseño e implementación

Una vez diseñada la arquitectura de la plataforma, es necesario detallar el proceso de creación de cada una de las unidades de las que consta la aplicación. Para ello, en primer lugar, se explicará como está organizado y desarrollado el módulo de Python de los test estadísticos. En segundo lugar, se detallará el diseño de la API REST desde el punto de vista de las URIs que conforman cada servicio web. Por último, se detallará la capa superior de la arquitectura, explicando cómo será la interfaz gráfica mediante prototipado y analizando su nivel de usabilidad mediante los principios heurísticos de Nielsen. Asimismo, se hará uso de UML para especificar los diagramas de secuencia de las funcionalidades más relevantes de la plataforma.

6.1. Módulo Python test estadísticos

El módulo de Python de los test estadísticos o módulo STAC (Statistical Tests for Algorithm Comparison) está formado por los siguientes archivos:

- **tests_parametricos.py:** contiene el test paramétrico ANOVA y el test POST-HOC de Bonferroni para el test de ANOVA.
- **tests_no_parametricos.py:** contiene todos los test no paramétricos (Wilcoxon, Friedman, etc.) indicados en los objetivos del proyecto, excepto aquellos que ya se encuentran en la librería SciPy, tal y como se indicó en la sección 5.3.1.
- **__init__.py:** para que los archivos de test puedan ser importados todos a la vez, es posible empaquetarlos juntos. En este fichero se indica qué se importará al importar el módulo.

Este módulo, como se había comentado en la sección 5.3.1 del capítulo 5, tiene dependencias con la librería SciPy (con el módulo *stats*), así como con el paquete NumPy. La funcionalidad de SciPy sirve a este proyecto para poder hallar parámetros básicos que serán necesarios para obtener todos los datos necesarios en los test.

El diseño de los test estadísticos está determinado por los argumentos de entrada y valores de salida de cada test.

6.1.1. Test paramétricos

Prototipo de la función del test ANOVA:

```
anova_test(matriz_datos, alpha=0.05)
```

- **“matriz_datos”**: lista de listas de conjuntos de datos. Cada conjunto de datos representa los resultados obtenidos por los algoritmos al ser aplicados sobre un problema concreto.
- **“alpha”**: nivel de significación o error tipo I (probabilidad de rechazar la hipótesis nula siendo cierta).

Datos devueltos:

Diccionario que contiene los siguientes datos:

- **“resultado”**: *true* ó *false*, que indica que el contraste es estadísticamente significativo o estadísticamente no significativo, dependiendo de si el p_valor es menor que el nivel de significación.
- **“p_valor”**: probabilidad de obtener un valor al menos tan extremo como el estadístico hallado suponiendo la hipótesis nula cierta.
- **“estadístico”**: valor del estadístico en cuestión.
- **“variaciones”**: lista con las variaciones o sumas de cuadrados total, entre tratamientos y variación del error (SCT, SCTR, SCE).
- **“grados_libertad”**: lista con los grados de libertad totales, entre tratamientos y del error (GLT, GLTR, GLE).
- **“cuadrados_medios”**: lista con los cuadrados medios (suma cuadrados / grados libertad) total, entre tratamientos y variación del error (SCT, SCTR, SCE).
- **“medias_algoritmos”**: lista con las medias de los datos de cada tratamiento o algoritmo.
- **“media_general”**: media de la lista de medias de los algoritmos.

Prototipo de la función del test POST-HOC Bonferroni:

```
bonferroni_test(nombres_algoritmos, medias_algoritmos, cuadrado_medio_error, N, alpha=0.05)
```

- **“nombres_algoritmos”**: lista de los nombres de los algoritmos.
- **“medias_algoritmos”**: lista con las medias de los datos de cada tratamiento o algoritmo.
- **“cuadrado_medio_error”**: valor del cuadrado medio del error (suma cuadrados error / grados libertad error).
- **“N”**: número de conjuntos de datos.

- **“alpha”**: nivel de significación o error tipo I (probabilidad de rechazar la hipótesis nula siendo cierta).

Datos devueltos:

Diccionario que contiene los siguientes datos:

- **“resultado”**: lista de valores *true* ó *false*, que indica si cada uno de los contrastes es o no estadísticamente significativo, dependiendo de si cada uno de los p-valores es menor que el nivel de significación.
- **“p_valores”**: lista de probabilidades de obtener un valor al menos tan extremo como cada uno de los estadísticos hallados suponiendo la hipótesis nula cierta.
- **“valores_t”**: lista de valores o estadísticos para cada comparación.
- **“p_valores ajustados”**: lista de los p-valores de las comparaciones ajustados a toda la familia de comparaciones.
- **“alpha”**: nivel de significación (modificado según el valor de m).
- **“comparaciones”**: lista donde cada elemento es un texto que contiene los nombres de los dos algoritmos involucrados en la comparación tal como “algoritmoA vs algoritmoB”. Se ordena según del p-valor de la comparación.

6.1.2. Test no paramétricos

Prototipos de la función del test no paramétricos de Wilcoxon:

```
wilcoxon_test(matriz_datos, alpha=0.05)
```

- **“matriz_datos”**: lista de listas de conjuntos de datos. Cada conjunto de datos representa los resultados obtenidos por los algoritmos al ser aplicados sobre un problema concreto.
- **“alpha”**: nivel de significación o error tipo I (probabilidad de rechazar la hipótesis nula siendo cierta).

Datos devueltos:

Diccionario que contiene los siguientes datos:

- **“resultado”**: *true* ó *false*, que indica que el contraste es estadísticamente significativo o estadísticamente no significativo, dependiendo de si el p-valor es menor que el nivel de significación.
- **“estadístico”**: valor del estadístico en cuestión.
- **“suma rangos pos”**: suma de los rangos de las diferencias mayores que 0.
- **“suma rangos neg”**: suma de los rangos de las diferencias menores que 0.

- Si $N \leq 25$ (tamaño muestral):
 - “**punto crítico**” límite inferior del intervalo de aceptación. El contraste será estadísticamente significativo si: estadístico \leq límite inferior correspondiente.
- Si $N > 25$:
 - “**p_valor**”: probabilidad de obtener un valor al menos tan extremo como el estadístico hallado suponiendo la hipótesis nula cierta.

Prototipos de las funciones test no paramétricos de ranking:

```
friedman_test(nombres_algoritmos, matriz_datos, alpha=0.05, tipo=0)
```

```
iman_davenport_test(nombres_algoritmos, matriz_datos, alpha=0.05, tipo=0)
```

```
friedman_rangos_alineados_test(nombres_algoritmos, matriz_datos, alpha=0.05, tipo=0)
```

```
quade_test(nombres_algoritmos, matriz_datos, alpha=0.05, tipo=0)
```

- “**nombres_algoritmos**”: lista que contiene los nombres de los algoritmos y que será empleada para devolver el ranking de nombres de los algoritmos.
- “**matriz_datos**”: lista de listas de conjuntos de datos. Cada conjunto de datos representa los resultados obtenidos por los algoritmos al ser aplicados sobre un problema concreto.
- “**alpha**”: nivel de significación o error tipo I (probabilidad de rechazar la hipótesis nula siendo cierta).
- “**tipo**”: indica si lo que se pretende es minimizar (en cuyo caso se establece a 0) o maximizar (se establece a 1).

Datos devueltos:

Diccionario que contiene los siguientes datos:

- “**resultado**”: *true* ó *false*, que indica que el contraste es estadísticamente significativo o estadísticamente no significativo, dependiendo de si el p_valor es menor que el nivel de significación.
- “**p_valor**”: probabilidad de obtener un valor al menos tan extremo como el estadístico hallado suponiendo la hipótesis nula cierta.
- “**estadístico**”: valor del estadístico en cuestión.
- “**nombres**”: lista de los nombres de los algoritmos ordenados según los valores numéricos de los rankings medios obtenidos por los distintos algoritmos.
- “**ranking**”: lista de los valores de los rankings medios ordenados de menor a mayor (cuanto menor es el valor mejor es el dato.)

Prototipos de las funciones de los test POST-HOC:

```

bonferroni_dunn_test(K, nombres, valores_z, p_valores, metodo_control, alpha=0.05)
holm_test(K, nombres, valores_z, p_valores, metodo_control, alpha=0.05)
hochberg_test(K, nombres, valores_z, p_valores, metodo_control, alpha=0.05)
li_test(K, nombres, valores_z, p_valores, metodo_control, alpha=0.05)
finer_test(K, nombres, valores_z, p_valores, metodo_control, alpha=0.05)
nemenyi_multitest(m, comparaciones, valores_z, p_valores, alpha=0.05)
holm_multitest(m, comparaciones, valores_z, p_valores, alpha=0.05)
hochberg_multitest(m, comparaciones, valores_z, p_valores, alpha=0.05)
finer_multitest(m, comparaciones, valores_z, p_valores, alpha=0.05)
shaffer_multitest(m, comparaciones, valores_z, p_valores, alpha=0.05)

```

- **“K”**: número de algoritmos (incluyendo método de control).
- **“valores_z”**: estadísticos calculados en función del test de ranking y del ranking devuelto por el test principal. Siguen una normal (0, 1) y están ordenados según los p-valores.
- **“p_valores”**: p-valores de cada uno de los estadísticos para comparar con los niveles de significancia ajustados.
- **“nombres”**: lista de nombres de los algoritmos (con los que el método de control se compara) ordenados según los p-valores.
- **“metodo_control”**: método de control del test, por convención es el test de menor ranking.
- **“alpha”**: nivel de significancia (probabilidad de error tipo 1) del test de ranking principal.
- **“m”**: número de comparaciones.
- **“comparaciones”**: nombres de las hipótesis contrastadas. Por ejemplo “algoritmoA vs algoritmoB”.

Datos devueltos:

Diccionario que contiene los siguientes datos:

- **“valores_z”** y **“p_valores”**.
- **“alpha”** ó **“alphas”**: nivel de significación. Bonferroni-Dunn y Li devuelven un alpha modificado. El resto de test devuelven una lista de alphas cuyos elementos son diferentes para cada comparación.
- **“resultado”**: lista de valores *true* ó *false*, que indica si cada uno de los contrastes es o no estadísticamente significativo, dependiendo de si cada uno de los p-valores es menor que el nivel de significación.

- **“p_valores ajustados”**: lista de los p_valores de las comparaciones ajustados a toda la familia de comparaciones.

Para los test con método de control, el diccionario también incluye **“metodo_control”** y **“nombres”**. Para los test de comparación múltiple, éste también incluye: **“comparaciones”**.

6.2. API Servicios REST

Como se había comentado en la sección 5.3.2 del capítulo 5, Bottle permite la implementación de servicios web accesibles mediante una o más URIs (*Uniform Resource Identifier* o identificador de recursos uniforme). El decorador `route()` asigna una URI a un trozo de código (el propio servicio web) en lo que se denomina ruta. El diseño de la API REST está determinado por las URIs mediante las cuales se puede acceder a los servicios.

La plataforma cuenta con una API REST en la que se distinguen dos tipos de servicios web, diferenciados por la utilidad que tienen:

- Servicios de subida / consulta de ficheros de datos.
- Servicios de acceso / ejecución de los test.

Para los ejemplos de URIs en el diseño se supone que todos los servicios escuchan en `/api/`. Esto se pudo hacer modificando el archivo que utiliza el módulo WSGI para cargar la aplicación (API REST) en el servidor web Apache. En el manual técnico del presente documento (manual de despliegue) se detallará el contenido de este archivo.

Servicios de subida y consulta de ficheros: Podemos ver el diseño de este tipo de servicios en las tablas 6.1 y 6.2. El primer decorador permite la subida y el almacenamiento del contenido de un fichero en el servidor. El método usado es “POST”, ya que se envían datos al servidor (el fichero) y éste devuelve el resumen HASH del contenido del fichero o error en el caso de que se haya producido algún error en el procesamiento del archivo o éste no se encuentre en el servidor. El segundo decorador, permite consultar los datos de un fichero en concreto. Tiene un parámetro de entrada no opcional `id_fichero` (resumen HASH del contenido del fichero). En caso de no existir un fichero con esa clave, se devuelve error. El método empleado es “GET”, ya que únicamente se reciben datos del servidor.

Decorador
<code>@route('/fichero', method="POST")</code>
<code>@route('/fichero/<id_fichero>', method="GET")</code>

Cuadro 6.1: Decoradores Bottle.

Ejemplo URI	Método
<code>/api/fichero</code>	POST
<code>/api/fichero/id_fichero</code>	GET

Cuadro 6.2: Ejemplos de URIs y métodos empleados.

Servicios de acceso y ejecución de test: Estos servicios son los encargados de dar acceso a la ejecución de los test estadísticos y los que devuelven los datos proporcionados por éstos (en formato JSON) al navegador. La función AJAX de jQuery se encargará de mostrar estos datos en la interfaz de usuario. El método empleado es GET, ya que los servicios web de los test únicamente devuelven datos (resultado de su ejecución).

Las tablas 6.3 y 6.4 muestran el diseño para los test de evaluación de las condiciones paramétricas, los test paramétricos y el test de Wilcoxon (no paramétrico):

Decorador
@route('/nombre/<id_fichero>', method="GET")
@route('/nombre/<id_fichero>/<alpha:float>', method="GET")

Cuadro 6.3: Decoradores Bottle.

Ejemplo URI	Método
/api/ttest/id_fichero	GET
/api/ttest/id_fichero/0.05	GET

Cuadro 6.4: Ejemplos de URIs y métodos empleados.

El valor *nombre* indica el nombre del test referenciado (p. ej. /ttest/...). Los parámetros utilizados son los siguientes:

- **alpha:** nivel de significación. Parámetro opcional cuyo valor por defecto es 0.05 (probabilidad de error tipo 1 más común).
- **id.fichero:** resumen HASH del contenido del fichero que identifica al mismo inequívocamente. Se trata de un parámetro no opcional.

Las tablas 6.5 y 6.6 muestran el diseño para los test no paramétricos de ranking (como el test de Friedman, el test de Iman-Davenport, etc.):

Decorador
@route('/nombre/<id_fichero>/<test_comparacion>', method="GET")
@route('/nombre/<id_fichero>/<alpha:float>/<test_comparacion>', method="GET")
@route('/nombre/<id_fichero>/<tipo:int>/<test_comparacion>', method="GET")
@route('/nombre/<id_fichero>/<alpha:float>/<tipo:int>/<test_comparacion>', method="GET")
@route('/nombre/<id_fichero>', method="GET")
@route('/nombre/<id_fichero>/<alpha:float>', method="GET")
@route('/nombre/<id_fichero>/<tipo:int>', method="GET")
@route('/nombre/<id_fichero>/<alpha:float>/<tipo:int>', method="GET")

Cuadro 6.5: Decoradores Bottle.

Ejemplo URI	Método
/api/friedman/id_fichero/li_test	GET
/api/friedman/id_fichero/0.05/li_test	GET
/api/friedman/id_fichero/1/li_test	GET
/api/friedman/id_fichero/0.05/1/li_test	GET
/api/friedman/id_fichero	GET
/api/friedman/id_fichero/0.05	GET
/api/friedman/id_fichero/1	GET
/api/friedman/id_fichero/0.05/1	GET

Cuadro 6.6: Ejemplos de URIs y métodos empleados.

El valor *nombre* indica el nombre del test de ranking referenciado (p. ej. /friedman/...) Los parámetros utilizados son los siguientes:

- **alpha:** nivel de significación. Parámetro opcional cuyo valor por defecto es 0.05 (probabilidad de error tipo 1 más común).
- **tipo:** indica la función objetivo de los algoritmos a contrastar: minimización (0) / maximización (1). Es un parámetro opcional cuyo valor por defecto es 0 (minimización).
- **test_comparacion:** indica el nombre del test POST-HOC a aplicar en caso de que el resultado del test de ranking determine que existen diferencias significativas. Se trata de un parámetro opcional cuyo valor por defecto es “bonferroni_dunn_test”.
- **id_fichero:** resumen HASH del contenido del fichero que identifica al mismo inequívocamente. Se trata de un parámetro no opcional.

6.3. Aplicación web

6.3.1. Prototipo de la interfaz gráfica

Para la implementación de la interfaz gráfica se construyeron inicialmente varios prototipos visuales de diferentes aspectos de la plataforma para luego poder proceder a su implementación. Para elaborar estos prototipos, se tuvieron en cuenta las historias de usuario del cliente vistas en la sección 3.2.2 del capítulo 3. En las figuras 6.1, 6.2, 6.3 y 6.4 podemos ver los prototipos de la página principal, la página donde se muestra la ayuda, la página de selección de parámetros de los test y la página de visualización de los resultados respectivamente. Por otro lado, en el manual de usuario al final del presente documento se muestra en las figuras B.1, B.5, B.8 y B.11 el resultado final de la implementación.

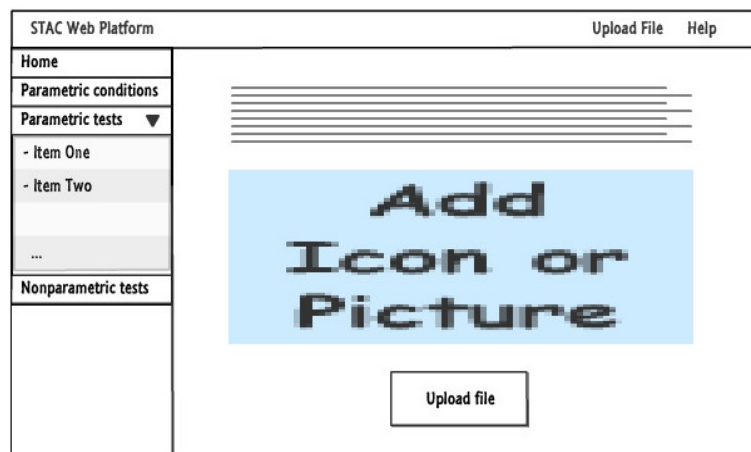


Figura 6.1: Prototipo de página principal.

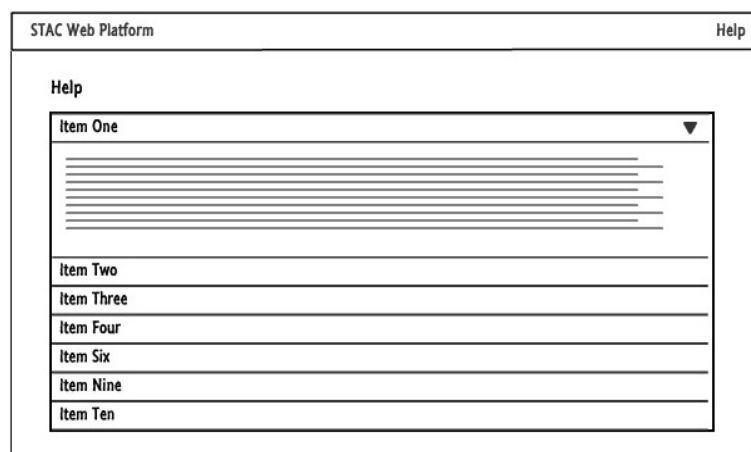


Figura 6.2: Prototipo página de ayuda.

Figura 6.3: Prototipo página selección de parámetros/opciones.

parameter1	parameter2	parameter3	...
...

Figura 6.4: Prototipo página visualización de resultados.

6.3.2. Heurísticas de Nielsen

En la interacción persona-ordenador, se siguen varios pasos para crear sistemas que sean amigables para el usuario. En el paso de evaluación, se pueden realizar pruebas de expertos, en las cuales lo más común es utilizar las heurísticas creadas por Jakob Nielsen para evaluar el diseño de la interfaz de usuario. Los 10 principios de diseño basados en el usuario, que definió Jakob Nielsen en 1990, siguen siendo un referente importante para evaluar la usabilidad de un sitio web. A continuación se detallan los principios heurísticos de Nielsen, y se comenta en qué medida el sistema cumple con estos principios.

Visibilidad del estado del sistema

El sistema debe siempre mantener a los usuarios informados del estado del sistema, con una

realimentación apropiada y en un tiempo razonable.

El sistema siempre trata de indicar el lugar donde se encuentra el usuario, así como las acciones que puede realizar en el lugar en el que se encuentre. La parte superior de la web trata de indicar estos aspectos mediante títulos / breves descripciones. El menú desplegable de la izquierda también proporciona información del lugar en el que se encuentra el usuario.

Lenguaje de los usuarios

El sistema debe hablar el lenguaje de los usuarios, con las palabras, las frases y los conceptos familiares, en lugar de que los términos estén orientados al sistema. Utilizar convenciones del mundo real, haciendo que la información aparezca en un orden natural y lógico.

Si bien esta plataforma web está dirigida a gente que posee ciertos conocimientos en algoritmos de aprendizaje automático, puede que estos usuarios no tengan conocimientos suficientes acerca de la validación de resultados mediante la aplicación de test estadísticos y el contraste de hipótesis en general. Por este motivo, el sistema pretende de la forma más amigable posible explicar para qué sirve cada test, los conceptos básicos relacionados con el contraste de hipótesis y cómo interpretar los resultados obtenidos.

Control y libertad para el usuario

Los usuarios eligen a veces funciones del sistema por error y necesitan a menudo una salida de emergencia claramente marcada, esto es, salir del estado indeseado sin tener que pasar por un diálogo extendido. Es importante disponer de deshacer y rehacer.

El sistema proporciona un botón para volver atrás después de la aplicación de los test dando la posibilidad de cambiar los parámetros u opciones de los mismos. Además, la barra superior contiene en el título de la plataforma un enlace directo a la página principal (de forma similar a la mayoría de las páginas web actuales). En cuanto a la subida de ficheros, el panel emergente que se despliega puede ser cerrado mediante un botón del propio panel o haciendo clic en cualquier lugar de la pantalla, lo cual favorece la cancelación de la acción. Por otra parte, el menú desplegable a la izquierda tiene enlaces directos a los diferentes aspectos de la plataforma (incluido también a la página principal).

Consistencia y estándares

Los usuarios no deben tener que preguntarse si las diversas palabras, situaciones, o acciones significan la misma cosa. En general siga las normas y convenciones de la plataforma sobre la que se está implementando el sistema.

Se mantiene un lenguaje homogéneo intentando no ser repetitivo y procurando no expresar en distintos lugares las mismas cosas de forma diferente. Además, la simplicidad del sistema favorece que no se produzcan este tipo de dudas en los usuarios.

Ayuda a los usuarios para reconocimiento, diagnóstico y recuperación de errores

Los mensajes de error se deben expresar en un lenguaje claro (no haya códigos extraños), se debe indicar exactamente el problema, y deben ser constructivos.

En caso de error, se muestra un mensaje indicado claramente el motivo, de forma que el usuario pueda volver a realizar la acción corrigiendo el problema. Estos mensajes no contienen ningún tipo de código e intentan indicar cómo solucionar el problema. Por ejemplo, si se aplican

test paramétricos sin haber realizado los test para determinar si los datos cumplen con las condiciones paramétricas, el sistema muestra un mensaje indicando que los resultados devueltos puede que no sean fiables e indica los test que se deberían aplicar.

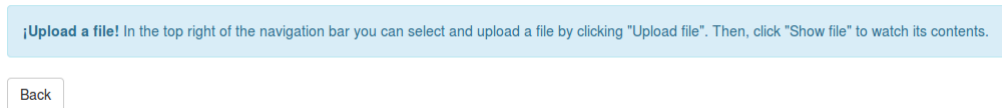


Figura 6.5: Error sin fichero.

Prevención de errores

Es importante prevenir la aparición de errores, mejor que generar buenos mensajes de error.

Se evita que los usuarios tengan que usar campos de texto en los que introducir información, ya que son una de las mayores fuentes de errores. En su lugar se utilizan botones y menús desplegables. El sistema también cuenta con enlaces a la ayuda como una forma de evitar que el usuario cometa errores.

Reconocimiento antes que cancelación

El usuario no debería tener que recordar la información de una parte de diálogo para otra. Es mejor mantener objetos, acciones, y las opciones visibles que memorizar.

El usuario siempre tiene la información directamente disponible, sin necesidad de tener que recordarla. Para ello, en cualquier lugar de la web debajo de la barra superior aparece un título o breve descripción que indica claramente en qué lugar se encuentra el usuario. En el único procesamiento con más de un paso, la aplicación de un test POST-HOC después de un test de ranking, se muestra el test previamente seleccionado.

Flexibilidad y eficiencia de uso

Las instrucciones para el uso del sistema deben ser visibles o fácilmente accesibles siempre que se necesiten. Los aceleradores no vistos por el usuario principiante, mejoran la interacción para el usuario experto de tal manera que el sistema puede servir para usuarios inexpertos y experimentados. Es importante que el sistema permita personalizar acciones frecuentes.

La plataforma dispone de servicios web basados en REST que escuchan permanentemente en `/api/`, permitiendo a un usuario experto (p. ej. un desarrollador conocedor del sistema) poder introducir en la propia URL los parámetros de los test y el servicio al que desea acceder sin necesidad de interactuar con la interfaz web tal y como lo haría un usuario normal (p. ej. `http://localhost/api/ttest/id_fichero/0.05`). Los resultados en este caso se mostrarían en formato JSON, lo cual no resulta visualmente atractivo para un usuario corriente.

Además, en todos los pasos hay enlace a la ayuda para usuarios inexpertos, mientras que para los que ya conocen la mecánica, los pasos a seguir serían los mínimos posibles.

Estética de diálogos y diseño minimalista

No deben contener información que sea inaplicable o se necesite raramente. Cada unidad adicional de información en un diálogo compite con las unidades relevantes de información y

disminuye su visibilidad relativa.

La web proporciona descripciones generales o resúmenes de cada test, que pueden resultar muy útiles al usuario no experimentado que necesite tener una idea clara del uso de cada test. Con el objetivo de que esta información extra no interfiera con la selección de opciones y aplicación de los test, se posiciona en la parte inferior de la pantalla (debajo de las opciones).

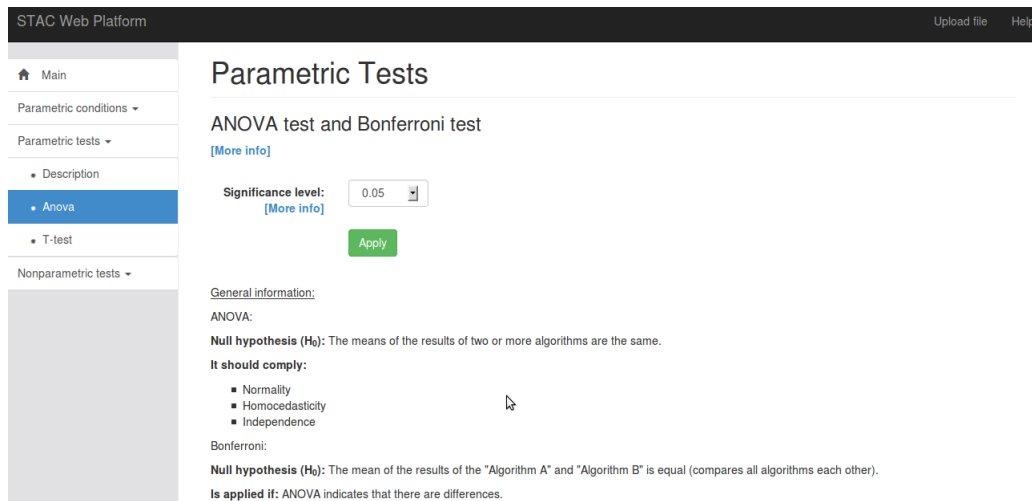


Figura 6.6: Vista test ANOVA.

Ayuda general y documentación

Aunque es mejor si el sistema se puede usar sin documentación, puede ser necesario disponer de ayuda y documentación. Esta ha de ser fácil de buscar, centrada en las tareas del usuario, tener información de las etapas a realizar y que no sea muy extensa.

La plataforma web cuenta con un botón en la barra superior mediante el cual se puede acceder a la ayuda. En esta ayuda se puede encontrar información acerca del formato del fichero, conceptos básicos, información de test, autoría, etc. Además de este botón, el sistema dispone de diferentes enlaces en lugares clave (opciones de los test, visualización de resultados, etc.) que enlazan con diferentes secciones de la ayuda.

6.3.3. Diagramas de secuencia

Como se ha comentado en el capítulo 5, los diagramas de secuencia indican los componentes que forman parte del sistema y las llamadas que se hacen en cada uno de ellos para realizar una tarea determinada. Los mensajes o llamadas intercambiados están ordenados temporalmente (en secuencia).

Los objetos Interfaz Web y API REST representan a aquellos ficheros que componen la interfaz web (documentos HTML, CSS, JavaScript etc.) y aquellos que constituyen los servicios web REST (fichero de los servicios y la librería Bottle) respectivamente. Asimismo, la clase *LimitedSizeDict* representa la clase utilizada para limitar el número de ficheros subidos (diccio-

nario con límite de elementos). A continuación se explicarán los diagramas de secuencia para las funcionalidades más relevantes del sistema.

Diagrama para subir fichero de datos (HU-22)

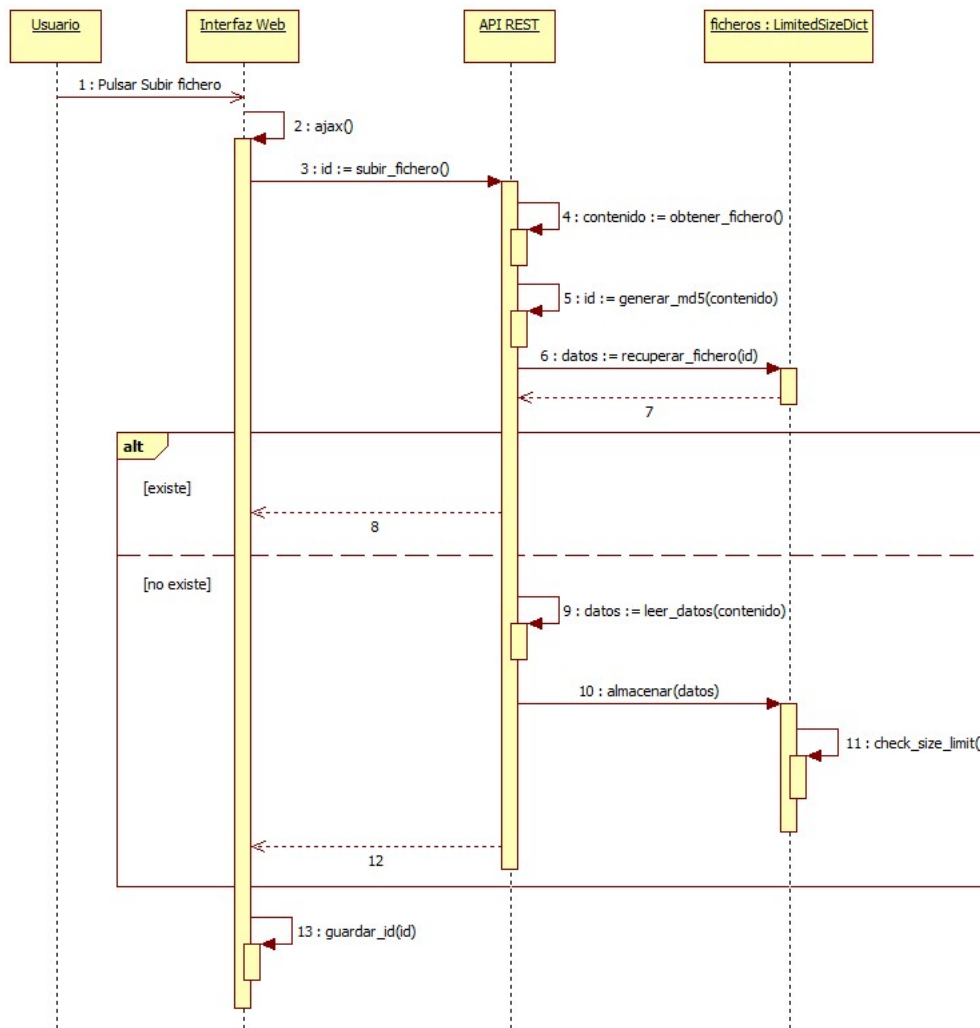


Figura 6.7: Diagrama de secuencia de la subida de ficheros.

En la figura 6.7 podemos ver el diagrama de secuencia para la subida de ficheros de datos. En primer lugar, el usuario interactúa con la interfaz, donde pulsa el botón para subir un fichero. En la ventana emergente, selecciona el fichero y pulsa el botón *Subir fichero*. Esto desencadena una llamada AJAX al servicio de subida de ficheros del API REST: *subir_fichero()*. Esta llamada AJAX añade varios parámetros, como el contenido del fichero, el tipo de llamada a realizar (en

este caso POST, pues se envían datos al servidor) o el tipo de datos que se esperan de vuelta (que para este proyecto siempre será JSON, como se indicó en el capítulo 5).

Una vez en el servicio web, se obtiene el contenido del fichero especificado en la llamada AJAX con *obtener_fichero()*. Sobre este contenido, se aplica un algoritmo de reducción criptográfico denominado MD5: *generar_md5()*, que nos proporciona una clave única para el fichero que lo diferenciará de cualquier otro. Una vez obtenida esta clave, se consulta si existe dicho fichero: *recuperar_fichero(id)*. Si existe, se devuelve el identificador (en formato JSON) a la aplicación web, que se encargará de guardar en la sesión este identificador para su posterior uso: *guardar_id(id)*. Si no existe en el diccionario, entonces se invoca a la función *leer_datos(contenido)*, que determinará si el formato es adecuado y generará un JSON del contenido.

Si el formato es adecuado este JSON es almacenado en el diccionario de ficheros con *almacenar(datos)* y finalmente se devolverá el identificador a la interfaz web.

Diagrama para consultar fichero de datos (HU-23)

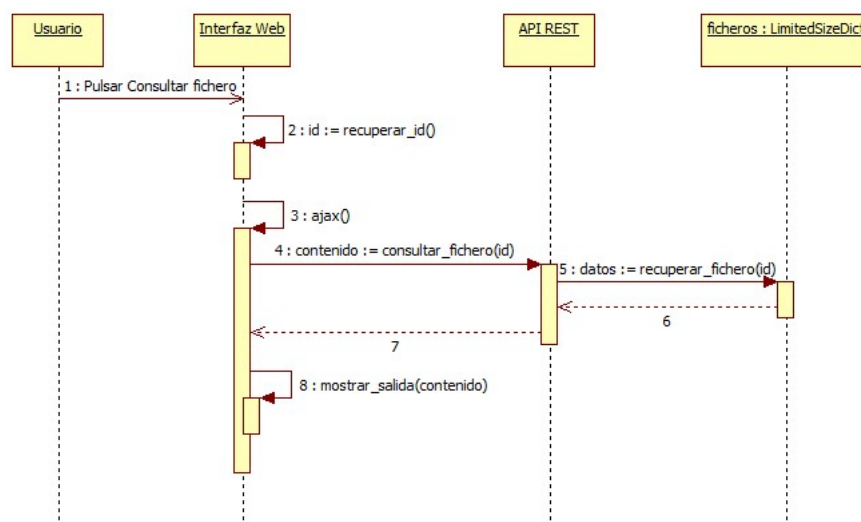


Figura 6.8: Diagrama de secuencia de la consulta de ficheros.

En la figura 6.8 se muestra el diagrama de secuencia para la consulta del fichero. Aquí, en primer lugar el usuario pulsa el botón de *Consultar fichero*. Luego, la interfaz web recupera el identificador de fichero actual almacenado en la sesión: *recuperar_id()*.

A continuación, se realiza la llamada AJAX al servicio web de consulta de ficheros. Esta llamada se diferencia de la anterior en que el método empleado es GET y no POST, pues únicamente se requiere un JSON (contenido del fichero). La llamada al servicio lleva como parámetro obligatorio el identificador del fichero: *consultar_fichero(id)*. El servicio recupera los datos del fichero: *recuperar_fichero(id)*, y devuelve el contenido a la interfaz, que se encarga de representar los datos en una tabla y mostrarlos mediante *mostrar_salida(contenido)*.

Diagrama para realizar test de ANOVA (HU-7)

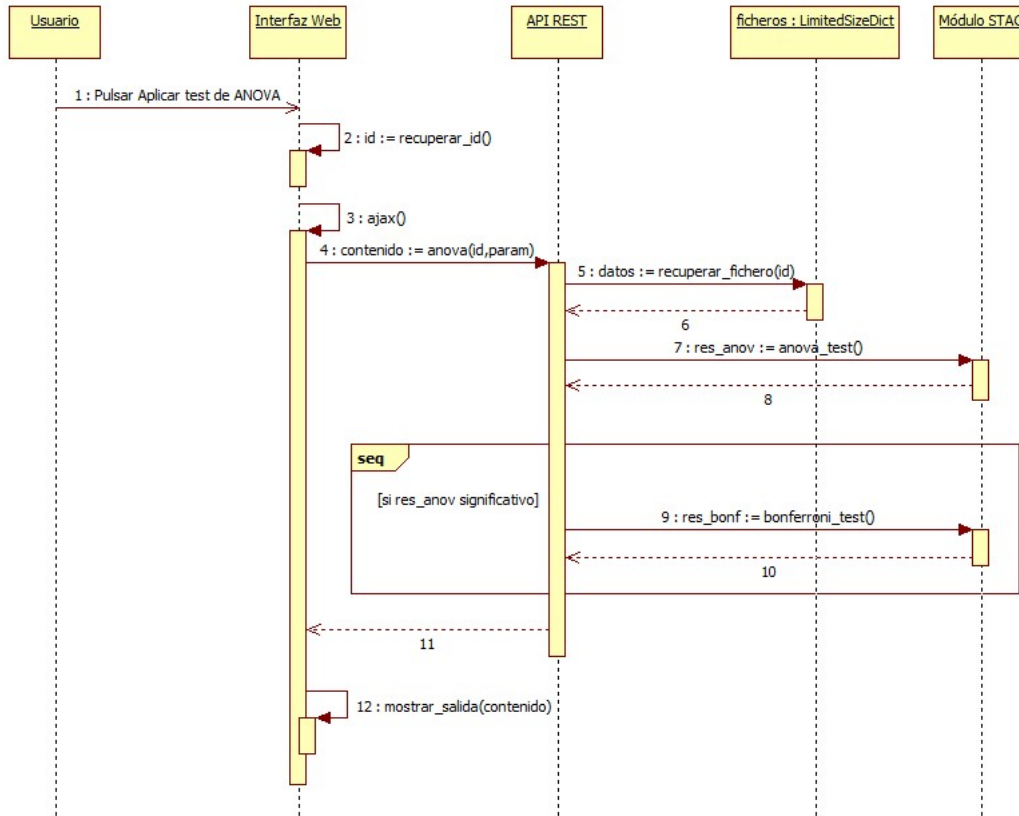


Figura 6.9: Diagrama de secuencia del test de ANOVA.

La realización del test paramétrico ANOVA se muestra en el diagrama 6.9. En este caso, el usuario accede a la sección de ANOVA en el apartado de test paramétricos, selecciona las opciones disponibles y aplica el test. La llamada al servicio, además del identificador del fichero, en este caso podría llevar más parámetros, como se muestra en la llamada a la función del servicio: *anova(id,param)*. Estos parámetros se muestran en la sección 6.2. En este caso, se tiene como parámetro opcional únicamente el nivel de significancia “alpha”.

Una vez dentro del servicio web para ANOVA se accede al contenido del fichero de datos almacenado en el diccionario: *recuperar_fichero(id)* y se procede a realizar la llamada al test estadístico del módulo STAC: *anova_test()*. Los argumentos para los test se muestran en la sección 6.1.

Una vez obtenidos los datos (siempre en formato JSON), si el resultado de la prueba de ANOVA es estadísticamente significativa, se procede a aplicar el test POST-HOC de Bonferroni. Una vez aplicados los test se devuelve un JSON desde la API REST a la interfaz web con los resultados. Aquí nuevamente mediante *mostrar_salida(contenido)* se muestran los resultados del

test de ANOVA en una tabla y, si hay resultados para el test POST-HOC de Bonferroni, se genera otra tabla para estos resultados.

Diagrama para realizar los test no paramétricos de ranking

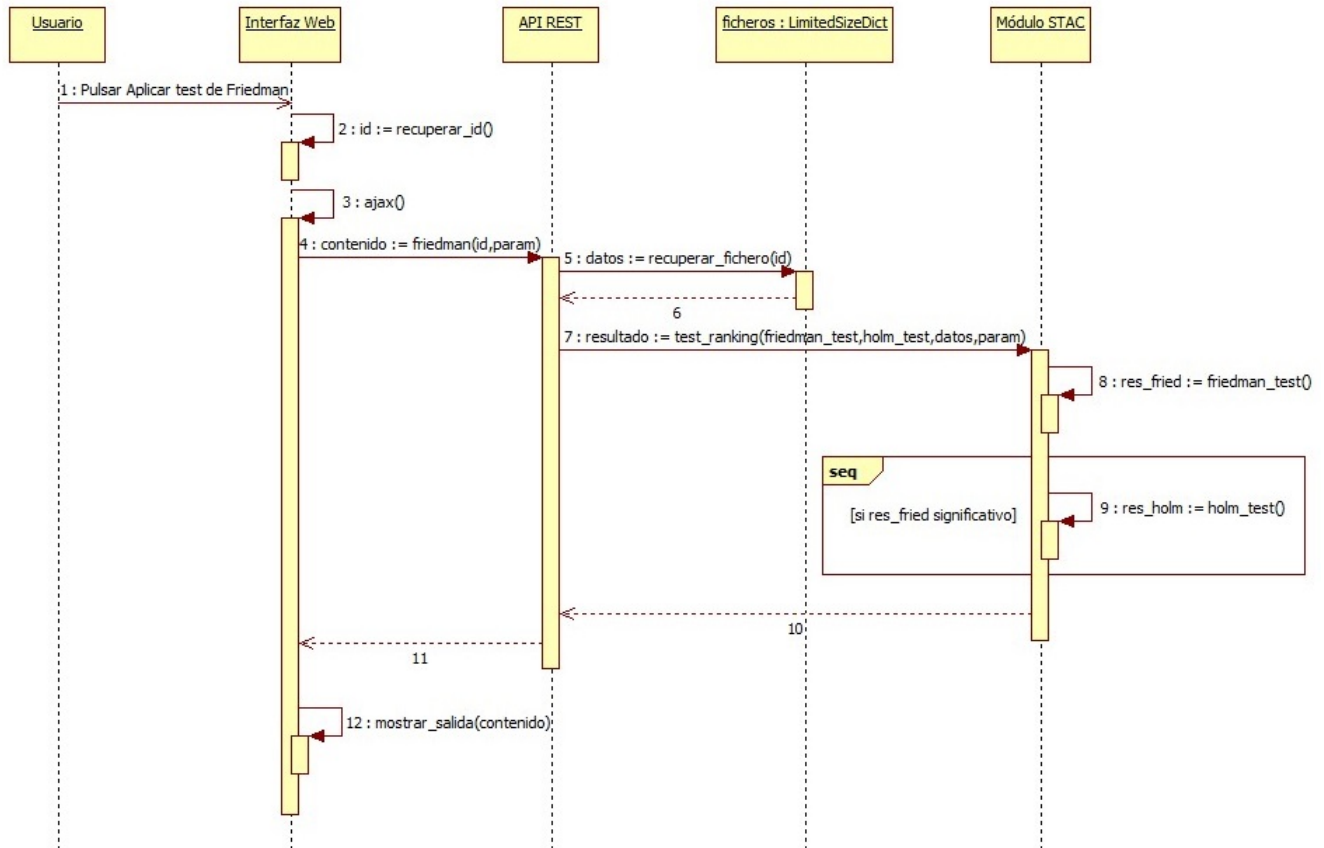


Figura 6.10: Diagrama de secuencia de los test no paramétricos de ranking.

En el diagrama de la figura 6.10 podemos ver el diagrama de secuencia para la realización de test de ranking. En este caso, la interacción del usuario con la interfaz implica posicionarse en la sección de test de ranking dentro de los test no paramétricos y seleccionar las opciones de aplicación de un test determinado.

En este diagrama se muestra el ejemplo para la realización del test de Friedman junto con el test POST-HOC de Holm. Sin embargo, esto sería aplicable para todos los test de ranking. Como se ha dicho anteriormente, tanto los parámetros de los servicios como los parámetros de las funciones de los test se muestran en las secciones 6.2 y 6.1 respectivamente.

Este tipo de test se caracterizan por la invocación de la función *test_ranking(friedman_test, holm_test,datos,param)*, que permite separar los test POST-HOC de los test de ranking, evitando

tener que pasar como argumento al test POST-HOC el test de ranking realizado previamente. Dentro de esta función, se realiza el test de Friedman: *friedman_test()* y si su resultado es estadísticamente significativo, se realiza el test POST-HOC de Holm: *holm_test()*, de forma similar al test de ANOVA.

Diagrama para realizar test de SciPy

Por último, en la figura 6.11 se muestra el caso de la aplicación de test pertenecientes a la librería SciPy. Por tato, se incluye el objeto Módulo SciPy. El diagrama muestra como caso de ejemplo la aplicación del test de normalidad de Shapiro-Wilk: *shapiro()*. Este test se aplica para cada uno de los algoritmos de aprendizaje automático presentes en el fichero de entrada, con el objetivo de determinar si los datos obtenidos por cada algoritmo siguen una distribución normal.

Se muestra también una segunda interacción del usuario, en la cual éste en la pantalla de visualización de resultados de los test pulsa el botón de *Exportar a formato $LaTeX$* . Como se puede apreciar, la interfaz web en este caso invocaría a la función *exportTableToLaTeX()* para generar el contenido. Ocurriría lo mismo en caso de la exportación de resultados a formato CSV (con la función *exportTableToCSV()*).

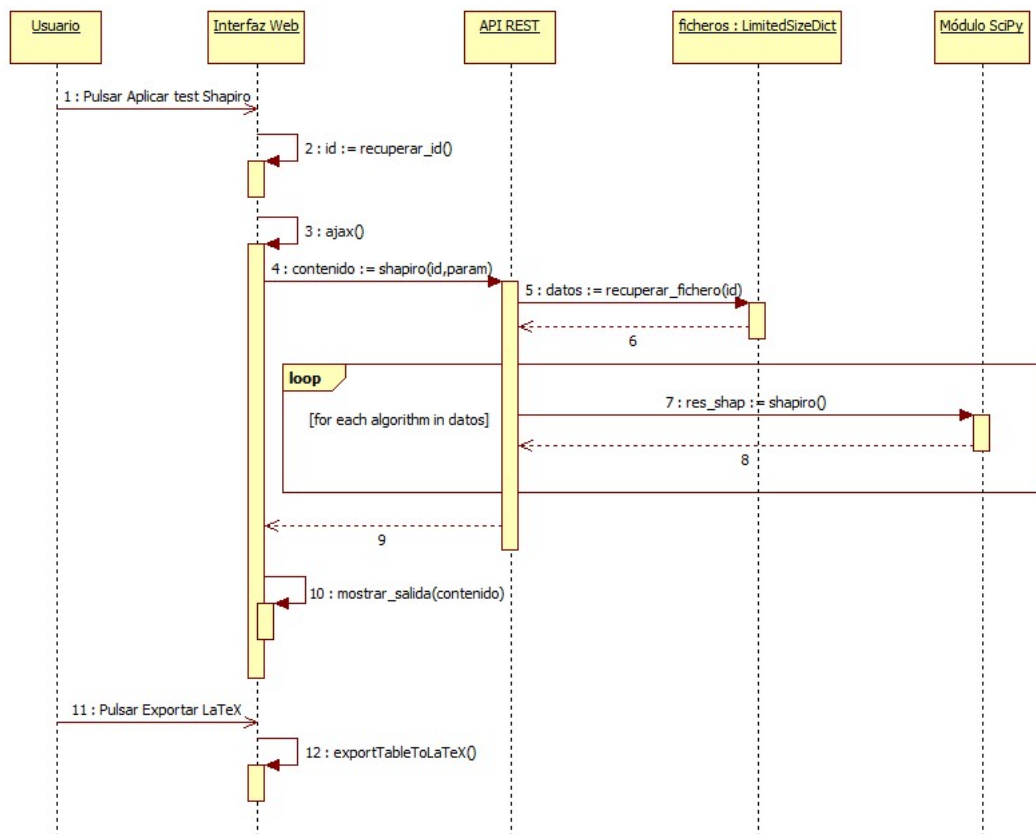


Figura 6.11: Diagrama de secuencia de los test de SciPy.

Capítulo 7

Validación y pruebas

Durante y después del proceso de implementación, el programa que se está desarrollando debe ser comprobado para asegurar que satisface su especificación y entrega la funcionalidad esperada por las personas interesadas en el software. La verificación y la validación es el nombre dado a estos procesos de análisis y pruebas. Tienen lugar en cada etapa del proceso del software. Comienza con revisiones de los requerimientos y continúa con revisiones del diseño e inspecciones de código hasta la prueba del producto [9].

La verificación y la validación no son lo mismo, aunque a menudo se confunden. Boehm [19] expresó de forma breve la diferencia entre ellas:

- **Verificación:** *¿estamos construyendo el producto correctamente?*
- **Validación:** *¿estamos construyendo el producto correcto?*

En este capítulo, se detallarán las pruebas de verificación y validación realizadas, para lo cual se llevarán a cabo:

- **Pruebas unitarias:** para verificar si los test implementados calculan y devuelven de forma correcta todos los datos necesarios.
- **Validación de requisitos (historias de usuario):** para documentar el balance de éxito del proyecto desde el punto de vista del grado de cumplimiento de las historias de usuario.

Dada la metodología Scrum (sección 4.2.2), que ha sido la utilizada en este proyecto, las pruebas cobran mayor importancia a lo largo de todo el ciclo de vida del software (no sólo al final). Esto es debido a que el carácter solapado de sus fases “impone” realizar pruebas para verificar que el código se comporta de la manera esperada y en base a las historias de usuario antes de poner fin a cada sprint, ya que es después de la finalización del sprint cuando se establece la reunión de análisis y revisión del incremento generado. En esta reunión se presenta al cliente (los directores del proyecto) el incremento desarrollado (terminado, probado y operando en un entorno), con el fin de obtener realimentación para mejorar e incorporar en sucesivos sprints e ir determinando el balance de éxito en el cumplimiento de las historias de usuario (validación de historias de usuario).

7.1. Pruebas unitarias

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Dicho de otra forma, las pruebas unitarias se basan en hacer pruebas en pequeños fragmentos de un programa. Estos fragmentos deben ser unidades estructurales de un programa encargados de una tarea específica. En programación procedural u orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones que tenemos definidos. En nuestro caso, se han utilizado para comprobar que cada uno de los test estadísticos implementados calcula y devuelve correctamente todos los datos requeridos.

El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores. Además, una vez realizadas las pruebas unitarias, en caso de que haya que refactorizar algún test estadístico las mismas pruebas pueden servir para probar el nuevo código asegurándonos de que éste sigue siendo válido bajo la nueva implementación.

Durante el ciclo de vida del proyecto se han realizado diferentes fases de pruebas dedicadas especialmente a verificación del correcto funcionamiento de los test. Para aquellos test provenientes de la librería SciPy (de los que hemos hablado en capítulos anteriores), no se han realizado pruebas unitarias, pues ya están verificados por dicha librería.

Para realizar las pruebas unitarias sobre los test implementados se ha utilizado el siguiente marco experimental [7]:

- **Test no paramétricos de ranking y POST-HOC de comparación simple (con método de control):** 24 problemas o conjuntos de datos de los repositorios UCI [20] (repositorio de aprendizaje automático) y KEEL [22] (que además de incluir un repositorio de conjuntos de datos es una herramienta para evaluar algoritmos evolutivos para problemas de minería de datos, incluyendo la clasificación, etc.), sobre los que se aplicaron 4 algoritmos de clasificación de la herramienta KEEL: PDFC (*Positive Definite Fuzzy Classifier*), NNEP (*Neural Network Evolutionary Programming*), IS-CHC + 1NN (*CHC Adaptive Search for Instance Selection*), FH-GBML (*Fuzzy Hybrid Genetics-Based Machine Learning*).
- **POST-HOC de comparación múltiple:** Ranking obtenido por el test de Friedman aplicado en 30 conjuntos de datos de UCI y KEEL sobre los que se aplicaron otros 5 algoritmos de clasificación de la herramienta KEEL: C4.5, 1NN, Naïve Bayes, Kernel y CN2.

Asimismo, para el caso del test de Wilcoxon, el test de Anova y el test de Bonferroni, se han empleado datos obtenidos de Internet con los que se han podido verificar su correcto funcionamiento [21]. Además, con algunos test se empleó la herramienta STATService 2.0 [23] para tener una referencia adicional de verificación de resultados.

A continuación, en la tabla 7.1 se detallan las pruebas unitarias realizadas, las cuales se han llevado a cabo con el valor 0.05 como de nivel de significancia, ya que es el valor más común:

Test estadístico	Resultado esperado	Resultado obtenido
Test de los Rangos Signados de Wilcoxon	Devuelve correctamente el estadístico, el <i>p-valor</i> y el resultado, incluyendo también la suma de los rangos positivos y la suma de los rangos negativos. Devuelve error en caso de que los datos tengan resultados de más de 2 algoritmos.	Correcto
Test de Friedman	Devuelve correctamente el estadístico, el <i>p-valor</i> y el resultado, así como los rankings (tanto en maximización como en minimización).	Correcto
Test de Iman-Davenport	Devuelve correctamente el estadístico (más ajustado que el Friedman) y su <i>p-valor</i> correspondiente.	Correcto
Test de los Rangos Alineados de Friedman	Los mismos datos que Friedman pero para este test.	Correcto
Test de Quade	Los mismos datos que Friedman pero para este test.	Correcto
Datos comunes a los test POST-HOC con método de control	Devuelve correctamente los estadísticos (valores Z), los <i>p-valores</i> asociados, los nombres de los algoritmos ordenados según los <i>p-valores</i> , el método de control y el valor K (número de algoritmos involucrados).	Correcto
Test de Bonferroni-Dunn	Devuelve correctamente los <i>p-valores</i> ajustados, los resultados y el nivel de significación ajustado a partir de los datos obtenidos en la función anterior.	Correcto
Test de Holm	Devuelve correctamente los <i>p-valores</i> ajustados, los resultados y los niveles de significancia ajustados.	Correcto
Test de Hochberg	Los mismos datos que Holm pero para este test.	Correcto
Test de Li	Devuelve correctamente los <i>p-valores</i> ajustados y los resultados.	Correcto
Test de Finner	Los mismos datos que Holm pero para este test.	Correcto
Datos comunes a los test POST-HOC de comparaciones múltiples	Devuelve correctamente los estadísticos (valores Z), los <i>p-valores</i> asociados, los nombres de las comparaciones ordenados según los <i>p-valores</i> y el número total de comparaciones.	Correcto
Multitest de Bonferroni-Dunn	Devuelve correctamente los <i>p-valores</i> ajustados, los resultados y el nivel de significación ajustado a partir de los datos obtenidos en la función anterior.	Correcto

Cuadro 7.1: Pruebas unitarias realizadas.

Test estadístico	Resultado esperado	Resultado obtenido
Multitest de Holm	Devuelve correctamente los <i>p-valores</i> ajustados, los resultados y los niveles de significancia ajustados.	Correcto
Multitest de Hochberg	Los mismos datos que el multitest de Holm pero para este test.	Correcto
Multitest de Finner	Los mismos datos que el multitest de Holm pero para este test.	Correcto
Test de Shaffer	Los mismos datos que el multitest de Holm pero para este test.	Correcto
Test ANOVA	Devuelve correctamente el resultado, el estadístico, el <i>p-valor</i> , las variaciones y los grados de libertad (totales, del tratamiento y del error), los cuadrados medios y los valores medios (algoritmos y media general).	Correcto.
Test de Bonferroni	Devuelve correctamente el nivel de significancia ajustado, los estadístico (valores <i>t</i>), sus <i>p-valores</i> asociados, los resultados y los <i>p-valores</i> ajustados.	Correcto

Cuadro 7.1: Pruebas unitarias realizadas.

7.2. Validación de requisitos

Como se comentó anteriormente, la validación es el proceso de comprobar que el sistema software producido es lo que el usuario realmente quería. Las historias de usuario, descritas en las secciones 3.2.1 y 3.2.2, correspondientes a las historias de usuario desarrollador y cliente respectivamente, se han ido validando en las diferentes reuniones con los directores del proyecto al final de cada sprint. Para ello se han tenido en cuenta los criterios de aceptación incluidos en las propias historias de usuario. Estos criterios de aceptación fueron definidos lo antes posible con el fin de ayudar a entender mejor lo que se esperaba del proyecto y poder realizar estimaciones de forma más fácil y precisa. Además, estos criterios también sirvieron de guía para el desarrollo de pruebas unitarias, tal y como hemos visto en la sección anterior, para verificar el correcto funcionamiento de los test estadísticos.

A continuación se muestra la validación de las historias de usuario desarrollador (tabla 7.2):

H. Usuario	Título	Resultado	Comentario
HU-1	Acceder a los test	Hecho	Se proporcionan diferentes URIs.
HU-2	Gestionar ficheros	Hecho	Se permite la subida y consulta de datos.

Cuadro 7.2: Historias de usuario desarrollador.

H. Usuario	Título	Resultado	Comentario
HU-3	Devolver datos JSON	Hecho	Todos los servicios web manejan este formato.
HU-4	Analizar datos subidos	Hecho	Devolución de error en caso de formato incorrecto.
HU-5	Limitar ficheros subidos	Hecho	Almacenamiento en diccionario con límite de elementos.
HU-6	Visualizar información test	Hecho	Módulo STAC y API correctamente comentados.

Cuadro 7.2: Historias de usuario desarrollador.

Ahora pasamos a detallar la validación de las historias de usuario cliente (tabla 7.3):

H. Usuario	Título	Resultado	Comentario
HU-7	Realizar test de ANOVA	Hecho	En caso de ser estadísticamente significativo se proporcionan también los resultados de Bonferroni.
HU-8	Realizar test t-test	Hecho	La interfaz dispone de una sección donde aplicar el test.
HU-9	Realizar test de Wilcoxon	Hecho	Sin comentarios
HU-10	Realizar test de Friedman	Hecho	Sin comentarios
HU-11	Realizar test de Iman-Davenport	Hecho	Sin comentarios
HU-12	Realizar test de los Rangos Alineados de Friedman	Hecho	Sin comentarios
HU-13	Realizar test de Quade	Hecho	Sin comentarios
HU-14	Realizar test de Bonferroni-Dunn	Hecho	Tanto el test simple (método de control) como multitest.
HU-15	Realizar test de Holm	Hecho	Sin comentarios
HU-16	Realizar test de Finner	Hecho	Sin comentarios
HU-17	Realizar test de Hochberg	Hecho	Sin comentarios
HU-18	Realizar test de Li	Hecho	La interfaz dispone de una sección donde aplicar el test.
HU-19	Realizar test de Shaffer	Hecho	Sin comentarios
HU-20	Realizar test de normalidad	Hecho	Sección en la interfaz para el test de Shapiro-Wilk, D'Agostino-Pearson y Kolmogorov-Smirnov.
HU-21	Realizar test de Levene	Hecho	La interfaz dispone de una sección donde aplicar el test.

Cuadro 7.3: Historias de usuario cliente.

H. Usuario	Título	Resultado	Comentario
HU-22	Subir fichero de datos	Hecho	Se dispone de un botón en la barra de navegación superior.
HU-23	Consultar fichero de datos	Hecho	Se dispone de un botón en la barra de navegación superior y se visualiza automáticamente después de la subida.
HU-24	Visualizar resultados de los test	Hecho	Los resultados se muestran en forma de tabla.
HU-25	Exportar los resultados en formato csv	Hecho	Botón en la pantalla de visualización de resultados.
HU-26	Exportar los resultados en formato \LaTeX	Hecho	Botón en la pantalla de visualización de resultados.
HU-27	Seleccionar nivel de significancia	Hecho	Combobox con varios niveles.
HU-28	Seleccionar la función objetivo	Hecho	Combobox en la sección de test de ranking.
HU-29	Ver ayuda	Hecho	Se dispone de un botón en la barra de navegación superior y de varios enlaces a conceptos concretos.
HU-30	Recordar el test de ranking	Hecho	Mensaje que indica el test de ranking seleccionado.
HU-31	Avisar de las condiciones paramétricas	Hecho	Mensaje de alerta si no se comprueban las condiciones paramétricas al aplicar un test paramétrico.
HU-32	Ver una breve información de los test	Hecho	Se muestra una breve descripción de las hipótesis que se contrastan.
HU-33	Enlazar con ficheros de ejemplo	Hecho	Enlaces a ficheros de ejemplo.
HU-34	Aplicación en diferentes pestañas del navegador	Hecho	La plataforma puede operar independientemente en diferentes pestañas con diferentes archivos de datos.
HU-35	Diseño adaptable	Hecho	Interfaz adaptable a dispositivos de varias resolución.
HU-36	Idioma	Hecho	Interfaz en inglés.
HU-37	Flujo de trabajo	Hecho	Imagen en la página principal.

Cuadro 7.3: Historias de usuario cliente.

Capítulo 8

Valoraciones finales

Como hemos detallado en la presente memoria, una de las tareas más importantes que se deben llevar a cabo en el aprendizaje automático es la validación de resultados obtenidos por los algoritmos de aprendizaje. El método estándar más aceptado en la actualidad por los analistas de datos es el de la aplicación de test estadísticos sobre los experimentos, que, entre otras utilidades, apoyan la toma de decisiones, como por ejemplo la elección del algoritmo más adecuado.

La aplicación de test estadísticos se engloba en el ámbito de la Inferencia Estadística, que es la parte de la estadística que estudia cómo sacar conclusiones generales (sujetas a un determinado grado de fiabilidad o significancia) para toda la población a partir del estudio de una muestra. En este proyecto de fin de grado, se ha desarrollado una plataforma web con la que poder obtener conclusiones de los resultados obtenidos por diferentes algoritmos sobre distintos conjuntos de datos para determinar, por ejemplo, si los algoritmos podrían tener un rendimiento significativamente diferente, y por lo tanto no se podrían considerar iguales.

El objetivo de este Trabajo de Fin de Grado ha sido el desarrollo de una plataforma web para la validación de experimentación en aprendizaje automático y minería de datos. En concreto, el trabajo descrito en esta memoria se resume en los siguientes puntos:

- Se extendió una librería de test estadísticos: `nonparametric.py`, actualmente implementada en Python.
- Se crearon servicios web basados en REST para facilitar el uso de los test y para poder realizar la subida y consulta de datos (pertenecientes a la aplicación de distintos algoritmos sobre problemas o conjuntos de datos).
- Se desarrolló una interfaz web que hace uso de estos servicios y que muestra los resultados obtenidos.

El objetivo ha sido que el analista de datos pudiese introducir en la web los resultados obtenidos mediante experimentación, y seleccionase el test estadístico que desease utilizar para que, de forma automática, la plataforma le mostrase los resultados de la aplicación del test. Así, plataforma permite, de un modo fácil y centralizado, la validación de resultados mediante el uso de test estadísticos.

Como se ha ido viendo a lo largo de los distintos capítulos (en concreto en el capítulo 7 perteneciente a la validación y pruebas), todos los objetivos del proyecto se han realizado con éxito. La librería de test estadísticos consta de todos los test deseados (únicamente falta el test de comparaciones múltiples de Li, que fue sustituido en su lugar por el test de Finner, como se comentó en la sección 4.4 debido a la ocurrencia del riesgo **RPD-3**). Además, los servicios web proporcionan el acceso a los test, tal y como se había planeado. Por otra parte, interfaz web cumple también con las historias de usuario vistas a lo largo del capítulo 3.

8.1. Posibles mejoras

A continuación se indican algunas ampliaciones con las que se podría extender la aplicación desarrollada:

8.1.1. Test estadísticos para datos no apareados

Uno de los puntos a destacar sobre el presente trabajo es que los test estadísticos desarrollados en este caso sirven para ser aplicados sobre un tipo de datos en particular: datos apareados. Disponemos de este tipo de datos cuando los datos de las muestras pertenecen a los mismos individuos (p. ej. aplicamos dos algoritmos sobre los mismos N conjuntos de datos, obteniendo dos muestras de tamaño N apareadas). Se podría extender la funcionalidad de la plataforma STAC permitiendo la aplicación de test estadísticos para datos no apareados. Existen actualmente test para este propósito. Por ejemplo, la contraposición para datos no apareados del test de los Rangos Signados de Wilcoxon presente en este proyecto sería el test U de Mann-Whitney.

8.1.2. Datos de entrada

Otra de las posibles mejoras es poder permitir al usuario la subida de datos no sólo mediante ficheros, sino a través de entrada manual de datos. Además, sería interesante poder permitir al usuario establecer él mismo el formato mediante el cual quiera subir los datos (p. ej. establecer el separador de columnas, el separador de filas, etc.). También se podría disponer de modificación vía web de los datos, tanto si son subidos mediante fichero de entrada o mediante formulario.

8.1.3. Otras mejoras

Se podrían añadir cambios a la interfaz con el fin de hacerla más atractiva al usuario. Por ejemplo, se podría mostrar una barra de progreso en la subida de datos. También se podría dar la posibilidad de mostrar los resultados con un determinado número de decimales (establecido por el usuario).

Apéndice A

Manual técnico

En este apéndice se muestra el manual de despliegue de la aplicación, pensado para que los usuarios puedan configurar y poner en funcionamiento la plataforma en sus equipos ¹. Para ello, se tomará como base el Sistema Operativo Ubuntu 14.04, junto con Apache 2.4. Las razones de realizar el despliegue con Apache y no con el propio servidor de Bottle se detallan en la sección 5.3.4. En la figura A.1, podemos ver los sub directorios que forman parte del directorio del proyecto:



Figura A.1: Directorio del proyecto STAC.

- **api:** API REST (servicios web, archivo de la librería de Bottle y archivo app.wgi).
- **documentación:** archivos de la memoria y documentación de los test estadísticos con Sphinx.
- **stac:** módulo Python de los test estadísticos.
- **web:** archivos web (Framework Bootstrap, CSS, JavaScript, imágenes, ficheros de datos de entrada de ejemplo, etc.).

A.1. Despliegue de la aplicación en Apache

1. Instalar Apache, el módulo WSGI y la librería SciPy:

¹Recordemos que, en cualquier caso, la plataforma web está actualmente disponible en la URL:
<http://tec.citi.usc.es/stac/>

- sudo apt-get install apache2
- sudo apt-get install libapache2-mod-wsgi
- sudo apt-get install python-scipy

2. Crear un enlace simbólico para enlazar el directorio del proyecto al directorio raíz de Apache y cambiar los permisos del directorio raíz `/var/www`:

- sudo ln -s \$STAC /var/www/stac
- sudo chmod -R 755 /var/www

3. El siguiente paso es cambiar la configuración de Apache para que el módulo WSGI cargue la API REST. Para ello se crea el archivo `/etc/apache2/sites-available/stac.conf` con el siguiente contenido:

```
<VirtualHost *:80>
    ServerName stac
    ServerAlias stac

    DocumentRoot /var/www/stac/web

    <Directory /var/www/stac/web>
        Order deny,allow
        Allow from all
    </Directory>

    WSGIDaemonProcess stac user=www-data group=www-data processes=1 threads=1
    WSGIScriptAlias /api /var/www/stac/api/app.wsgi

    <Directory /var/www/stac/api>
        WSGIProcessGroup stac
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

4. El módulo WSGI buscará el archivo `/var/www/stac/api/app.wsgi` para cargar la aplicación en el servidor web. El contenido de `app.wsgi` es el siguiente:

```
import sys, os, bottle
from bottle import route, Bottle

sys.path = ['/var/www/api/'] + sys.path
os.chdir(os.path.dirname(__file__))

import servicios # Importa los servicios REST

application = bottle.default_app() # Carga la aplicacion por defecto con los servicios REST
root = Bottle()
root.mount('/api/', application) # Hacemos que todos los servicios escuchen en /api
```

5. Activar el sitio:

- sudo a2ensite stac

6. Desactivar el sitio por defecto si es necesario:

- sudo a2disite 000-default

7. Reiniciar servidor:

- sudo service apache2 restart
- O también: sudo /etc/init.d/apache2 restart

Apéndice B

Manual de usuario

A continuación se explicarán las funcionalidades de la plataforma, apoyándose en capturas de pantalla del sistema.

B.1. Página de inicio

En la figura B.1 se puede ver el aspecto que presenta la página de inicio de la plataforma. En la parte izquierda un menú desplegable permite navegar por los diferentes test que ofrece el sistema: evaluación de condiciones paramétricas, test paramétricos o no paramétricos.

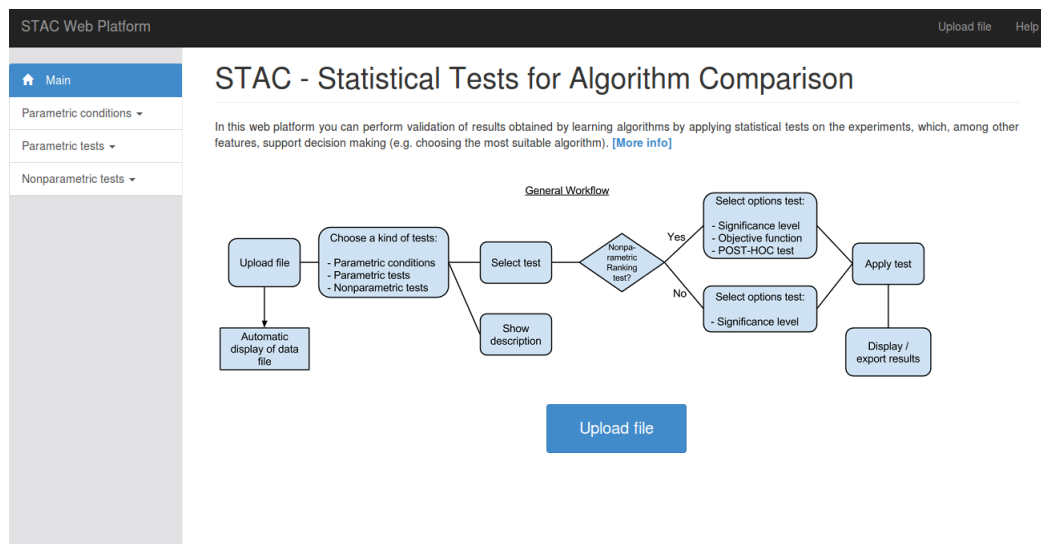


Figura B.1: Página de inicio de la plataforma.

Asimismo, y con la intención de que los usuarios no familiarizados con la plataforma se puedan

hacer una idea de los pasos a seguir en la validación de resultados mediante los test, se muestra en la parte central de la ventana el flujo de trabajo a seguir en una sesión de uso de STAC. En la parte inferior, un botón para subir un fichero de datos resalta claramente como primer paso a realizar en la validación. En la figura B.2 se puede ver la ventana emergente que se muestra para la subida de ficheros:

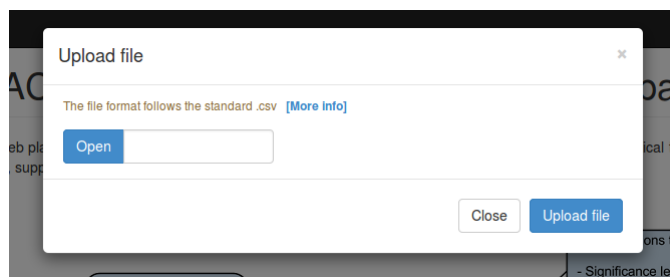


Figura B.2: Ventana emergente de subida de ficheros.

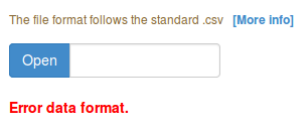


Figura B.3: Mensaje de error.

Para subir el fichero primero se debe hacer clic en “Open”, que abrirá la ventana de búsqueda del fichero. Para subir los datos, se debe hacer clic en “Upload file”. Asimismo, se puede cancelar la acción con “Close”, o haciendo clic en algún otro lugar de la plataforma distinto al de la ventana emergente para la subida de ficheros.

En la figura B.3, se muestra un mensaje de error típico cuando los datos no cumplen con el formato especificado. Por otra parte, una barra de navegación superior (común a todas las páginas de la plataforma excepto en la página de ayuda) proporciona un botón de subida de ficheros, consulta de ficheros (después de realizar una subida) y acceso a la ayuda, de forma que desde cualquier página de la plataforma se pueda tanto subir y visualizar ficheros como ir directamente a la ayuda.

B.2. Ayuda y consulta de datos

Las figuras B.4 y B.5 representan algunas secciones de la ayuda y una sección en particular respectivamente (la del formato del fichero de datos):

Help

1. Data file format
2. Basics
3. Available normality tests.
4. ANOVA test
5. T-test

Figura B.4: Listado secciones ayuda.

A esta ayuda se accede, como se ha comentado anteriormente, a través de la barra de navegación superior. Sin embargo, la aplicación cuenta también con enlaces que permiten acceder directamente a secciones particulares de la ayuda. Como se puede apreciar en la ventana emergente de subida de ficheros (Fig. B.2), el enlace **[more info]** permite realizar esta acción, que nos llevaría a la página representada en la figura B.5. En toda la aplicación existen enlaces de este tipo (la mayoría de los cuales están dedicados a los test de la plataforma.)

STAC Web Platform Help

1. Data file format

The data file follows the csv standard ("comma separated values", which is a plain text format where each element of the data set is written to a row whose elements are separated by commas. In this picture we can see an example:

```
1 datasets, A, B
2 1, 78, 78
3 2, 24, 24
4 3, 64, 62
5 4, 45, 48
6 5, 64, 68
7 6, 52, 56
8 7, 30, 25
9 8, 50, 44
10 9, 64, 56
11 10, 50, 40
12 11, 78, 68
13 12, 22, 36
14 13, 84, 68
15 14, 40, 20
16 15, 90, 58
17 16, 72, 32
```

It consists of the following elements:

- First row: datasets (here it could go either word before the first comma), algorithmName1, algorithmName2, ... , algorithmNameN
- Next rows: datasetName, algorithmResult1, algorithmResult2, ... , algorithmResultN

Examples:

[Sample 2 algorithms](#)

[Sample 4 algorithms](#)

Figura B.5: Sección del formato del fichero de datos.

Por otra parte, en la figura B.6 se muestra el aspecto de la pantalla de visualización del contenido del fichero. En esta pantalla, se resalta con un tono azul (similar al usado en los menús y botones del sistema) la línea de datos en la que está posicionado el ratón, con el fin de ayudar en la lectura.

STAC Web Platform					Show file	Upload file	Help
file content							
datasets	PDFC	NNEP	IS-CHC+INN	FH-GBML			
1	0.752	0.773	0.785	0.795			
2	0.727	0.748	0.724	0.713			
3	0.736	0.716	0.585	0.638			
4	0.994	0.861	0.88	0.791			
5	0.508	0.553	0.575	0.515			
6	0.535	0.536	0.513	0.471			
7	0.967	0.871	0.954	0.532			
8	0.831	0.807	0.819	0.768			
9	0.745	0.702	0.719	0.705			
10	0.709	0.572	0.669	0.607			
11	0.722	0.728	0.725	0.732			
12	0.967	0.947	0.953	0.96			
13	0.832	0.752	0.802	0.691			

Figura B.6: Pantalla de visualización del fichero de datos.

B.3. Selección de parámetros / opciones de test

En cada sección en el menú desplegable de la izquierda (condiciones paramétricas, test paramétricos o no paramétricos) aparece como primera opción una descripción general de los test. Por ejemplo en la figura B.7 se puede ver la descripción de las condiciones paramétricas:

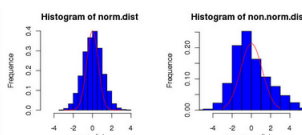
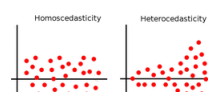
STAC Web Platform		Upload file	Help
Parametric Conditions			
Description			
To run parametric tests three conditions must be met:			
<ul style="list-style-type: none"> Normality: Samples or data obtained by the algorithms follow a normal distribution. 			
			
Source: http://commons.wikimedia.org			
<ul style="list-style-type: none"> Independence: Samples or data obtained by the algorithms do not depend on each other (assumed by users). 			
<ul style="list-style-type: none"> Homoscedasticity: It is the condition which says that the populations of input or data obtained by the algorithms come from populations with equal variances. The opposite would be heteroscedasticity. 			
			

Figura B.7: Descripción de las condiciones paramétricas.

Cuando se selecciona un tipo de test determinado, la pantalla que se muestra permite la

selección de un test en caso de que haya varios disponibles y sus opciones. Por ejemplo, en la figura B.8 podemos ver las opciones disponibles para el caso de la sección de test de normalidad. En la parte superior disponemos de las distintas opciones. En la parte inferior se dispone de las descripciones, para que no interfieran en el proceso de aplicación de los test:

STAC Web Platform

Upload file Help

Main

Parametric conditions

- Description
- Normality
- Homoscedasticity

Parametric tests

Nonparametric tests

Parametric Conditions

Normality - Select a Normality test:

Normality test: [\[More info\]](#)

Shapiro-Wilk

D'Agostino-Pearson

Kolmogorov-Smirnov

Significance level: [\[More info\]](#)

Apply

General information:

Shapiro-Wilk:

- Null hypothesis (H_0): The samples follow a normal distribution.
- Power: The best of the three methods, especially for samples with less than 30 elements.

D'Agostino-Pearson:

- Null hypothesis (H_0): The samples come from a normally distributed population.
- Power: Less than the Shapiro-Wilk test.

Kolmogorov-Smirnov:

Figura B.8: Selección de opciones.

Como se puede ver, para cada opción existe un enlace directo a la sección de la ayuda que lo explica. Hay que destacar que no se puede aplicar ningún test en caso de que no exista un fichero de datos subido en la plataforma. En la figura B.9 se muestra el mensaje de error que se genera al querer aplicar un test sin datos. Por otra parte, ciertos test, como el T-Test o el test de Wilcoxon requieren únicamente ficheros de datos en los que sólo haya datos para dos algoritmos. Si se intenta aplicar estos test sobre ficheros con datos para más de dos algoritmos se genera un error como el que se muestra en la figura B.10:

Parametric Tests

Upload a file! In the top right of the navigation bar you can select and upload a file by clicking "Upload file". Then, click "Show file" to watch its contents.

Back

Figura B.9: Error cuando no existe ningún fichero.

Parametric Tests

T-test result:

Require only two samples.

[Volver atrás](#)

Figura B.10: Error cuando el número de algoritmos es mayor a 2.

También se muestran mensajes de aviso en caso de que se aplique un test paramétrico sin haber comprobado previamente si los datos cumplen con las condiciones paramétricas.

B.4. Visualización de resultados

Después de aplicar algún test, se muestra la pantalla con los resultados obtenidos representados en forma de tabla. En la figura B.11 podemos ver la pantalla que se genera para mostrar los resultados. En este ejemplo en concreto se muestran los resultados obtenidos después de la aplicación de un test de ranking y un test POST-HOC, para los cuales se generan dos tablas:

STAC Web Platform Show file Upload file Help

[Main](#)
[Parametric conditions](#)
[Parametric tests](#)
[Nonparametric tests](#)
 • [Description](#)
 • [Wilcoxon](#)
 • **[Ranking tests](#)**

Nonparametric Tests

Friedman test result:

[Export to csv](#) [Export to LaTeX](#)

Ranking	Algorithms	Statistic	p-value	Result:
1.729	FH-GBML	16.225	0.001	H0 is rejected
2.521	IS-CHC+INN	-	-	-
2.521	NNEP	-	-	-
3.229	PDFC	-	-	-

[Go help.](#)

Bonferroni-Dunn test result:

[Export to csv](#) [Export to LaTeX](#)

Control Method	Adjusted alpha	Control Method VS	Statistic	p-value	Adjusted p-value	Result
FH-GBML	0.017	PDFC	-2.875	0.004	0.012	H0 is rejected
-	-	IS-CHC+INN	-1.517	0.129	0.388	H0 is accepted

Figura B.11: Página de visualización de resultados.

Asimismo, en cada tabla se muestra la opción de exportar los resultados (tanto a formato \LaTeX como a formato CSV). Si se pulsa en uno de estos botones se abre una ventana de diálogo similar a la mostrada en la figura B.12 para poder guardar el fichero generado:

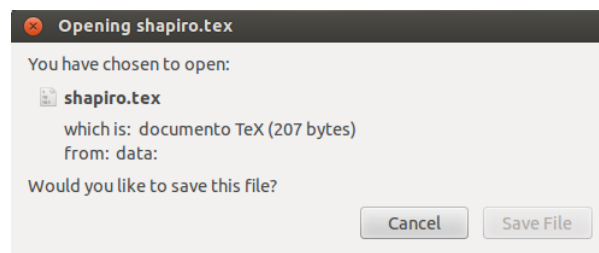


Figura B.12: Ventana de diálogo exportación resultados.

En las figuras B.13 y B.14 se muestra cómo sería el contenido de los resultados exportados (en \LaTeX y CSV respectivamente) para otro caso de ejemplo (resultados generados en la aplicación del test de normalidad de Shapiro-Wilk):

```
shapiro.tex ✕
\begin{tabular}{c|c|c|c}
Dataset&W Statistics&p-values&Results\\
\hline
1&0.860&0.003&H0 is rejected\\
2&0.923&0.068&H0 is accepted\\
3&0.918&0.053&H0 is accepted\\
4&0.943&0.188&H0 is accepted
\end{tabular}
```

Figura B.13: Formato datos \LaTeX .

```
shapiro.csv ✕
"Dataset","W Statistics","p-values","Results"
"1","0.860","0.003","H0 is rejected"
"2","0.923","0.068","H0 is accepted"
"3","0.918","0.053","H0 is accepted"
"4","0.943","0.188","H0 is accepted"
```

Figura B.14: Formato datos CSV.

Bibliografía

- [1] Tom M. Mitchell, *Machine Learning*, McGraw Hill, 1997.
- [2] Oded Maimon and Lior Rokach, *Data Mining and Knowledge Discovery Handbook*, Springer, New York, 2010.
- [3] F. Wilcoxon, Individual Comparisons by Ranking Methods Biometrics, *Biometrics* 1, 80-83, 1945.
- [4] William Navidi, *Estadística para ingenieros y científicos*, Colorado School of Mines, 370-372, 2006.
- [5] J. Derrac, S. García, D. Molina, F. Herrera, *A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms*, Swarm and Evolutionary Computation, 3-18, 2011.
- [6] J.H. Zar, *Biostatistical Analysis*, Prentice Hall, Englewood Cliffs, 1999.
- [7] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental Analysis of Power, *Information Sciences* 180, 2044–2064, 2010.
- [8] Kent Beck, *Extreme Programming Explained: Embrace Change*, 1999.
- [9] Ian Sommerville, *Ingeniería del software*, 2005.
- [10] Juan Palacio, *Flexibilidad con Scrum. Principios de diseño e implantación de campos de Scrum*, 2008.
- [11] Acunote. <http://www.acunote.com/>. Recuperado el 21 de julio de 2014
- [12] PayScale. <http://www.payscale.com/wizards/choose.aspx>. Recuperado del 21 de julio de 2014
- [13] SciPy. <http://www.scipy.org/>. Recuperado el 28 de julio de 2014.
- [14] Numpy and Scipy Documentation. <http://docs.scipy.org/>. Recuperado el 28 de julio de 2014.
- [15] Bottle: Python Web Framework. <http://http://bottlepy.org/>. Recuperado el 30 de julio de 2014.
- [16] Twitter Bootstrap. <http://getbootstrap.com/>. Recuperado el 1 de agosto de 2014.

- [17] *GitHub Bootstrap*. <https://github.com/twbs/bootstrap/>. Recuperado el 1 de agosto de 2014.
- [18] *GitLab CiTIUS*. <https://gitlab.citius.usc.es/>. Recuperado el 1 de agosto de 2014.
- [19] PPK - Resurrection, *SOFTWARE ENGINEERING - AS IT IS*, 1979.
- [20] *UCI Machine Learning Repository*. <https://archive.ics.uci.edu/ml/index.html>. Recuperado el 4 de agosto de 2014.
- [21] *Ejemplo de ANOVA*. http://www.hrc.es/bioest/Anova_4.html. Recuperado el 4 de agosto de 2014.
- [22] *KEEL-dataset Data set repository*. <http://www.keel.es/>. Recuperado el 4 de agosto de 2014.
- [23] JA Parejo, Jorge García, A Ruiz-Cortés, JC Riquelme, *STATService: Herramienta de análisis estadístico como soporte para la investigación con Metaheurísticas*, Actas del VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bio-inspirados, 2012.