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

Agradecimientos

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 Intelixentes 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

Índice de figuras

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, análisis matemático mediante la estadística 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 tests estadísticos sobre los experimentos, que, entre otras utilidades, apoyan la toma de decisiones (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 tests 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, el sistema muestre los resultados de la aplicación del test. Así, el sistema permitirá de un modo fácil y centralizado la validación de resultados mediante el uso de tests 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 aprendizaje automático y que 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 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.
2. Estudiar y conocer las definiciones de los tests estadísticos de uso habitual en aprendizaje automático.
3. Crear los servicios web en Python, basados en REST, que hagan disponible el acceso a los tests estadísticos vía web.
4. Desarrollar una interfaz web (HTML + JavaScript) para facilitar el uso de los tests sobre los datos introducidos por el analista de datos.

1.2. Organización del documento

Sección que incluirá la descripción de los distintos apartados del documento.

Capítulo 2

Contraste de hipótesis

El contraste de hipótesis, también conocido como tests 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 (muestra) para determinar, por ejemplo, si los algoritmos tienen un rendimiento significativamente diferente y por lo tanto no se pueden considerar iguales (población).

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, podría establecer una regla de decisión sobre la cual se basaría nuestra decisión final. Por ejemplo: si la media obtenida está próxima a la afirmada 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 afirmar que la evidencia desmiente la afirmación del ingeniero sobre el algoritmo en cuestión. Esto supone un problema y es el hecho de cuándo 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:

$$H_0 : \text{Hipótesis nula}$$

$$H_1 : \text{Hipótesis alternativa}$$

- 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, ya que 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 bien 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:

$$H_0 : \text{El acusado es inocente}$$

$$H_1 : \text{El acusado es culpable}$$

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.

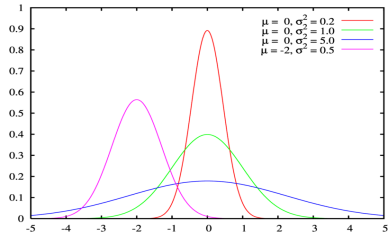
2.2. Estadístico de contraste

Los tests 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 tests 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 tests implementados harán uso de estadísticos que siguen distribuciones como:

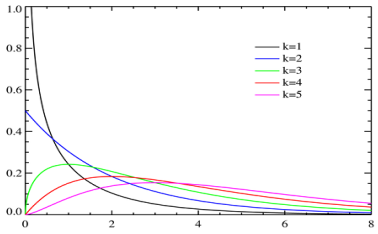
- Distribución normal (p. ej. test de Wilcoxon).
- Distribución chi-cuadrado χ^2 (p. ej. test de Friedman).
- Distribución f de Fisher-Snedecor (p. ej. test de Iman-Davenport).
- Distribución t de Student (p. ej. t-test).

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, que es la 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:

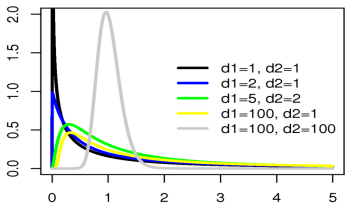
$$\text{Estadístico} \sim N(\mu, \sigma^2)$$



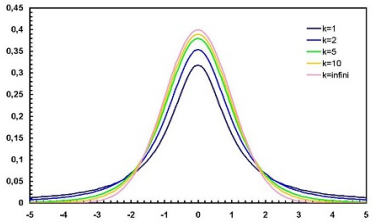
(a) Distribución normal con media μ y varianza σ^2.



(b) Distribución χ^2 con K grados de libertad.



(c) Distribución f con $d1$ y $d2$ grados de libertad.



(d) Distribución t con K grados de libertad.

Figura 2.1: Distribuciones de probabilidad.

En las distribuciones χ^2 y t de Student se habla del parámetro K o grados de libertad ($d1$ y $d2$ en la distribución 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 refiriendo 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 tests (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 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 puedan seguir un estadístico nos dan un valor diferente de probabilidad para cada valor diferente del estadístico. Este valor de probabilidad indica cómo de probable que 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 error. La figura 2.2, nos muestra las decisiones y los dos tipos de errores que se pueden cometer:

		Decisión	
		No se rechaza Ho	Se rechaza Ho
Realidad	Ho es verdadera	Decisión correcta	Error de tipo I
	Ho 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 cómo de importante considere 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 tests 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):

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu < \mu_0$$

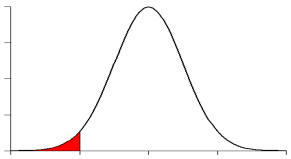
- Media mayor (test unilateral con cola a la derecha):

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu > \mu_0$$

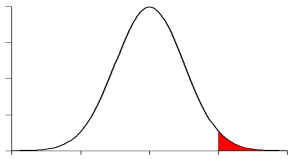
- Media distinta (test bilateral o de dos colas):

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu \neq \mu_0$$

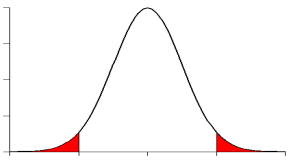
En la figura 2.3, podemos ver cómo quedarían establecidos los intervalos para el ejemplo. En rojo se muestra la región de rechazo:



(a) Test unilateral. Cola a la izquierda.



(b) Test unilateral. Cola a la derecha.



(c) Test bilateral o de dos colas.

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 constituye la región de rechazo. La región de aceptación tendrá en todos los casos probabilidad $1 - \alpha$.

2.5. Decisión final y Concepto de $p - valor$

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

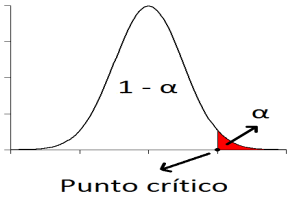


Figura 2.4: Punto crítico.

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.

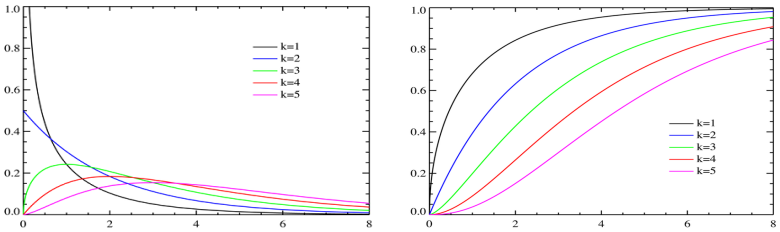
La decisión de rechazar o aceptar la hipótesis nula, se puede determinar también mediante el $p - valor$, que es el parámetro utilizado para realizar los tests estadísticos en este proyecto. El $p-valor$ proporciona un forma más eficiente de determinar si el contraste es o no estadísticamente significativo,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

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 (cómo de 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:

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



(a) Función de densidad de probabilidad.

(b) Función de distribución acumulada.

Figura 2.5: Comparativa FDP y FDA.

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 las 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 tests estadísticos en este proyecto. El $p-valor$ proporciona un 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 tests cambia de nivel de significación.

2.7. Tests paramétricos

Uno de los tipos más comunes de tests son los tests paramétricos. En general, estos tests son más robustos y tienen mayor potencia que los tests 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 inteligencia computacional 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. Cuando se comparan dos algoritmos de optimización normalmente suelen ser independientes. Cuando se comparan dos métodos de aprendizaje automático, depende de cómo sea la partición. La independencia es una característica que en este proyecto debe asumir el usuario de los tests, 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:

En este proyecto se pueden realizar los siguientes tests 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

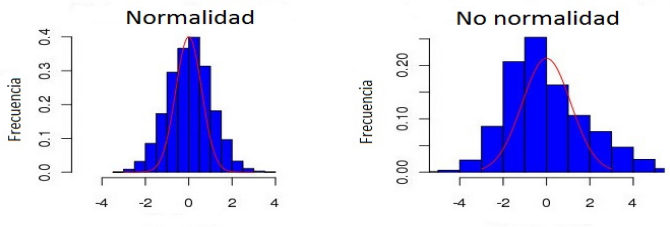


Figura 2.6: Normalidad VS No normalidad.

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 qué tan lejos se está de la distribución normal. Luego, calcula cómo de lejos cada uno de estos valores difiere de los valores 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 proceden de poblaciones con varianzas similares y aquellos que no:

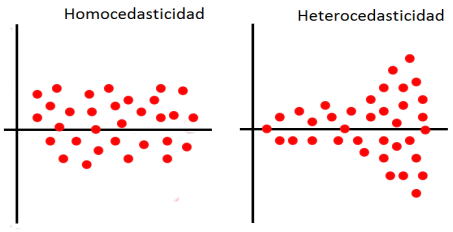


Figura 2.7: Homocedasticidad VS Heterocedasticidad.

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

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 existe 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 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.

2.7.3. T-test

El caso más simple de ANOVA donde intervienen únicamente 2 muestras o algoritmos es realizado por este test. 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, se distribuye como una distribución 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 podemos hablar de tests no paramétricos. Recientemente, los tests estadísticos no paramétricos han emergido como una metodología eficaz, robusta y asequible para la evaluación de nuevas propuestas de metaheurísticas y algoritmos evolutivos, alcanzando gran popularidad. 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 tests no paramétricos debido a su flexibilidad y a las pocas restricciones de uso que presentan (a diferencia de los tests 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 t de Student cuando no se puede suponer la normalidad de las muestras. Es, por tanto, menos potente que que la prueba 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: $Mediana_{diferencias} = 0$.
- Hipótesis alternativa H_1: $Mediana_{diferencias} \neq 0$.

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 de los algoritmos 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 excluye el 0).

- 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:

  - $T = \text{mín}[T(+), T(-)]$

- 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: $T <=$ 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 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 se distribuye de acuerdo a 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

- Similar 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, se distribuye de acuerdo a una distribución 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 quiera realizar una comparación entre conjuntos de datos.

Hipótesis

- Similar 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 se distribuye de acuerdo a una distribución χ^2 con $K - 1$ grados de libertad:

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

  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 considera, 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

- Similar 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 se distribuye como una distribución 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 tests 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 tests 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 tests 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 tests 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 del test aplicado. 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:

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

- **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.

- **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.

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

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

Los tests 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 cambiar esto por el parámetro m. Para los tests 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:

$$p - valor_i \leq \frac{\alpha}{t_i},$$

donde t_i puede ser obtenido mediante:

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

, 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 tests 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 [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 del capítulo 4. 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, puede prescindir de ellos.

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 tests estadísticos.
- **EP-2:** Permitir realizar tests estadísticos paramétricos.
- **EP-3:** Permitir realizar tests estadísticos no paramétricos.
- **EP-4:** Permitir realizar tests estadísticos no paramétricos para evaluar las condiciones paramétricas.
- **EP-5:** Permitir gestionar un fichero.
- **EP-6:** Visualizar resultados tests.
- **EP-7:** Permitir modificar opciones de los tests.
- **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 tests 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 tests estadísticos, se obtienen las siguientes historias de usuario:

HU-1 - Acceder a tests	
Como:	Desarrollador
Me gustaría:	acceder a los tests como recursos independientes con parámetros opcionales
Ya que:	esto facilita su utilización y los hace más flexibles
C. Aceptación:	- Los tests 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:	- 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 siempre el formato csv
Ya que:	es un formato abierto y sencillo para representar datos en forma de tabla y los tests 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 tests	
Como:	Desarrollador
Me gustaría:	obtener información acerca de los tests 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 tests 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 tests 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 la media de más de dos muestras son similares
C. Aceptación:	- El módulo de Python tiene implementado el test ANOVA - La plataforma muestra una sección para tests paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p-valor$ y el resultado. En caso de ser estadísticamente significativo, también aparecerán los resultados del test POST-HOC de Bonferroni ($p-valores$, $p-valores$ 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:	- El módulo de Python tiene implementado el test T-test - La plataforma muestra una sección para tests paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p-valor$ 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 tests 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:	- El módulo de Python tiene implementado el test de Wilcoxon - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve el estadístico, el $p - valor$,... 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:	- El módulo de Python tiene implementado el test de Friedman - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p - valor$, 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:	- El módulo de Python tiene implementado el test de Iman-Davenport - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p - valor$, 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 Quade
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:	- El módulo de Python tiene implementado el test de Rangos Alineados de Friedman - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p - valor$, 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:	- El módulo de Python tiene implementado el test de Quade - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los datos: estadístico, el $p - valor$, 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 tests de ranking obtengan diferencias significativas para detectar diferencias concretas entre pares de algoritmos
C. Aceptación:	- 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 tests no paramétricos donde se puede aplicar la prueba tanto simple como comparación múltiple - El test devuelve los resultados propios del test, como los $p - valores$ 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:	- 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 tests no paramétricos donde se puede aplicar la prueba tanto simple como comparación múltiple - El test devuelve los resultados propios del test, como los $p - valores$ 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:	- 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 tests no paramétricos donde se puede aplicar la prueba tanto simple como comparación múltiple - El test devuelve los resultados propios del test, como los $p - valores$ 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:	- 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 tests no paramétricos donde se puede aplicar la prueba tanto simple como comparación múltiple - El test devuelve los resultados propios del test, como los $p - valores$ 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:	- El módulo de Python tiene implementado el test de Li simple con método de control - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los resultados propios del test, como los $K - 1$ $p - valores$ 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:	- El módulo de Python tiene implementado el test de Shaffer de comparaciones múltiples - La plataforma muestra una sección para tests no paramétricos donde se puede aplicar la prueba - El test devuelve los resultados propios del test, como los $\frac{K(K-1)}{2}$ $p - valores$ ajustados, estadísticos, resultados, etc.
Prioridad:	alta

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

HU-20 - Realizar tests de normalidad	
Como:	Cliente
Me gustaría:	poder realizar distintos tests 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 tests paramétricos o si por el contrario lo más adecuado (si no existe normalidad) es aplicar un test no paramétrico
C. Aceptación:	- La plataforma muestra una sección para tests de condiciones paramétricas donde se pueden aplicar la 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, $p - valores$ 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:	- La plataforma muestra una sección para tests de condiciones paramétricas donde se puede aplicar la prueba de homocedasticidad de Levene - El test devuelve, el estadístico, el $p - valor$ 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 tests sobre ellos
C. Aceptación:	- 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:	- 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 resultados tests, 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:	- 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 resultados 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:	- 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 resultados 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 LaTeXsin necesidad de crear manualmente un tabla
C. Aceptación:	- 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 LaTeXlos resultados de éste
Prioridad:	media

Desglosando el Epic **EP-7:** Permitir modificar opciones de los tests, 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 tests
Ya que:	el nivel es un parámetro muy importante a la hora de calcular los resultados de los tests y conviene que sea un parámetro 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 función objetivo	
Como:	Cliente
Me gustaría:	poder elegir la función objetivo de los algoritmos con los que voy a aplicar los tests
Ya que:	esto modifica el orden del ranking en los tests 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 tests, y los conceptos relacionados con los resultados obtenidos, etc.
C. Aceptación:	- 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 tests, etc.) hay disponible un enlace al concepto relacionado en la ayuda
Prioridad:	media

HU-30 - Recordar 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 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:	- 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 breve información tests	
Como:	Cliente
Me gustaría:	debajo de las opciones de realizar los tests, 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 tests aparece una breve descripción de las hipótesis que se contrastan
Prioridad:	media

HU-33 - Enlazar 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 pequeños
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 los 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 tests 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:	- 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 explicita (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 tests 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 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

4.2.1. Metodologías ágiles

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 o 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.

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.

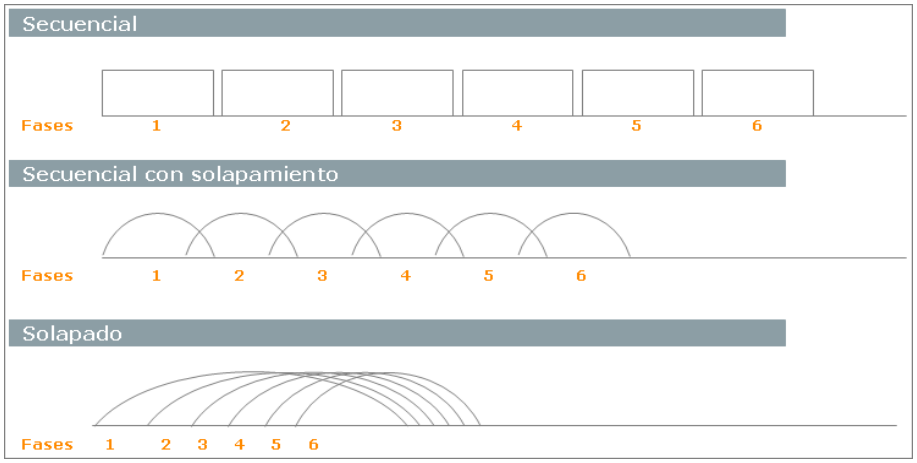


Figura 4.1: Ciclos de desarrollo.

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. Este modelo 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 60 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).
- **Facilitador (*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.
- **Facilitador (*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.
- **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.

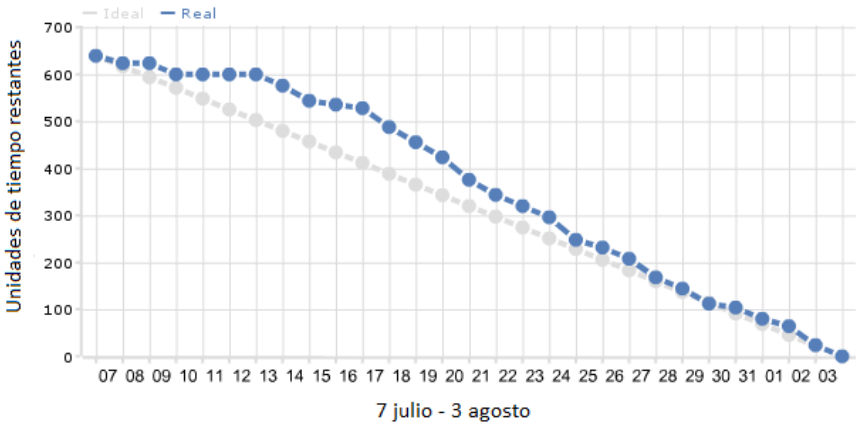


Figura 4.2: Ejemplo gráfico Burn-Down.

- **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 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.

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.

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 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 tests 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 tests 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

4.5. Estimación de costes

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] Wilcoxon, F. 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] Zar, J.H. *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